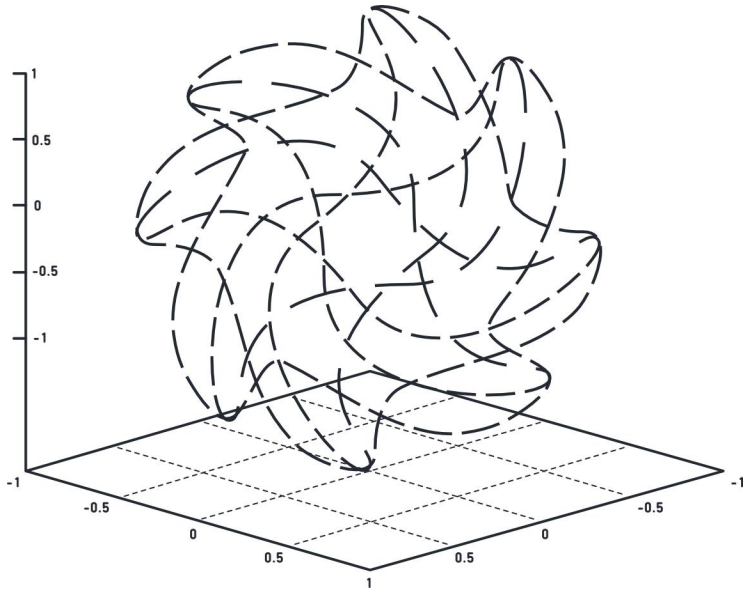


МАТЕРИАЛЫ ЗАДАНИЙ

командной инженерной олимпиады школьников «Олимпиада Кружкового движения Национальной технологической инициативы»

по профилю «Интеллектуальные робототехнические системы»

2019/20 учебный год

http://nti-contest.ru

Оглавление

Введение

Олимпиада Кружкового движения Национальной технологической инициативы (далее – Олимпиада КД НТИ) – это командная инженерная олимпиада школьников, завершающаяся разработкой действующего устройства, системы устройств или компьютерной программы. Олимпиада является проектом Агентства стратегических инициатив, элементом дорожной карты НТИ «Кружковое движение» и ключевым механизмом вовлечения школьников в образовательные программы высшего образования, ориентированные на рынки НТИ. Оператором Олимпиады КД НТИ является некоммерческая организация «Ассоциация участников технологических кружков».

Олимпиада КД НТИ является одним из этапов работы по реализации дорожной карты НТИ «Кружковое движение». Подготовка дорожной карты велась в фаблабах, ЦМИТах, детских технопарках, на базе активных школ и лицеев, центров дополнительного образования по всей России. Цели рабочей группы «Кружковое движение» НТИ связаны с развитием технологического сообщества, объединяющего школьников и студентов, ориентированных на инженерную деятельность на рынках НТИ, самодеятельных технических энтузиастов, лидеров технологических кружков, разработчиков педагогических технологий, технологических предпринимателей, популяризаторов науки и технологий. На сайте «Кружкового движения» НТИ собраны категоризированные подборки материалов для учащихся разных классов и запущен проект wiki-библиотеки образовательных материалов: https://kruzhok.org/.

В настоящее время во исполнение поручения Президента Российской Федерации В.В. Путина Правительством Российской Федерации совместно с Ассоциацией участников технологических кружков прорабатывается вопрос о создании сети кружков на базе общеобразовательных организаций по модели Кружкового движения НТИ: https://regnum.ru/news/economy/2918327.html.

Профили Олимпиады КД НТИ выбраны на основе приоритетов Национальной технологической инициативы: «Аэрокосмические системы», «Автоматизация бизнес-процессов», «Автономные транспортные системы», «Анализ космических снимков и геопространственных данных», «Беспилотные авиационные системы», «Большие данные и машинное обучение», «Водные робототехнические системы», «Интернет вещей», «Инженерные биологические системы: агробиотехнологии и геномное редактирование», «Интеллектуальные робототехнические системы», «Интеллектуальные энергетические системы», «Информационная безопасность», «Искусственный интеллект», «Композитные технологии», «Научно-инженерная коммуникация», «Наносистемы и наноинженерия», «Надводные робототизированные аппараты», «Нейротехнологии и когнитивные науки», «Передовые производственные технологии», «Программная инженерия финансовых технологий», «Разработка приложений виртуальной и дополненной реальности», «Разработка игр», «Спутниковые системы (Системы связи и дистанционного зондирования Земли)», «Технологии беспроводной связи», «Умный город», «Урбанистика».

Целевыми победителями Олимпиады КД НТИ являются школьники, способные реализовать сложные технические проекты в прорывных областях. Олимпиада выделяет команды участников с особыми характеристиками мышления, коммуникации и действия, необходимыми для решения задач НТИ. Победители и призеры Олим-

пиады КД НТИ должны показать высокие результаты в области применения предметных знаний в практической работе. Одновременно с этим, система подготовки Олимпиады КД НТИ должна предоставлять участникам инструменты для подготовки и получения недостающих знаний и практических навыков.

Цель Олимпиады КД НТИ – поддержка школьников в стремлении решать технологические вызовы XXI века через включение в решение технологических задач переднего края и, одновременно, через повышение социальной значимости такой работы благодаря льготам к поступлению. Эта цель лежит в рамках миссии Кружкового движения: формирование и подготовка команд, способных запускать глобальные технологические проекты, менять мир, создавая новые общественные практики.

Важной особенностью олимпиады является командный формат отборочного и заключительного этапов. Команды формируются на основе компетентностного принципа, различные компетенции участников в одной команде позволяют найти оригинальное нестандартное решение задачи. В командах участники планируют свою работу, обсуждают, ищут решения, распределяют роли – часто один участник выполняет несколько ролей. Комплексные инженерные задачи разработаны таким образом, что их можно декомпозировать на несколько подзадач, за решение которых берутся участники согласно своей роли в команде. Каждый участник несет ответственность за результат работы команды.

Организаторы Олимпиады КД НТИ соблюдают принцип равных возможностей и доступности участия школьников с ограниченными возможностями здоровья. В олимпиаде беспрепятственно могут участвовать дети с ОВЗ, способные выполнять инженерные работы и работать в команде, а также те, кто обучался по состоянию здоровья на дому. Организаторы также заинтересованы в дальнейшем сопровождении ее участников, а школьники – участники Олимпиады КД НТИ заинтересованы в дальнейшем сотрудничестве. В организации заключительного этапа Олимпиады КД НТИ 2019/20 учебного года в качестве волонтеров приняли участие победители и призеры Олимпиады КД НТИ прошлых лет, студенты первых курсов из различных регионов России.

В рамках интеграции выпускников и финалистов в проведение заключительных этапов были опробованы форматы взаимодействия амбассадоров и стажеров Кружкового движения с финалистами Олимпиады КД НТИ, а также выстроена система интеграции участников прошлых лет в сообщество Кружкового движения.

Олимпиада КД НТИ в 2015–2020 гг.

Олимпиада КД НТИ впервые состоялась в 2015/16 учебном году. За период с 2015 по 2020 год количество зарегистрированных участников выросло с нескольких тысяч до 58 тысяч, а количество школьников, приглашенных к участию в заключительном этапе – со 100 до 1262 человек. Количество профилей выросло в пять раз – с четырех до 30. В 2016/17 учебном году четыре профиля Олимпиады КД НТИ впервые вошли в Перечень олимпиад школьников, что позволило победителям и призерам воспользоваться льготами при поступлении в вузы России (в зависимости от правил приема конкретного вуза), в 2017/18 году число включенных в Перечень профилей выросло до девяти, в 2018/19 году – до 13, в 2019/20 учебном году – до 16 (семь – II уровня, девять – III уровня).

В 2017/18 учебном году заключительный этап Олимпиады КД НТИ стал рас-

пределенным и проходил в течение нескольких месяцев (с февраля по апрель) на площадках вузов по всей России: ОЦ «Сириус», МАИ, МИФИ, ТПУ, Университет Иннополис, СПбПУ, ДВФУ, УрФУ. В 2018/19 учебном году распределенный финал Олимпиады КД НТИ приняли МФТИ, МАИ, МИФИ, ТПУ, Университет Иннополис, СПбПУ, ДВФУ, НГУ, НовГУ, Московский Политех, ИГУ, ИрНИТУ и ряд других площадок. Также в 2018/19 учебном году впервые были проведены синхронные по времени распределенные финалы на площадках в разных городах в рамках одного профиля. Участники распределенных финалов решали одинаковые задания, имели одинаковые критерии оценивания и единый рейтинг участников.

В 2019/20 учебном году до 17 марта финалы проходили в очном формате в Новосибирске, Иркутске, Владивостоке, Сочи, Москве, Тюмени, Иннополисе, Томске, Санкт-Петербурге. После 17 марта в связи с эпидемиологической обстановкой и необходимостью принятия мер по предупреждению распространения новой коронавирусной инфекции Оргкомитет Олимпиады принял решение о проведении оставшихся заключительных этапов в формате, позволяющем участникам не покидать пределы города проживания и не принимать участия в массовых мероприятиях. Для этого была проведена комплексная оценка и разработаны меры по переводу всех оставшихся профилей в распределенный формат. Предметный тур заключительного этапа и апелляция проводились с применением системы прокторинга. Для организации командного тура использовались разные схемы доступа участников к лабораториям и оборудованию:

- работа с «аватарами»: лаборанты и технические специалисты выполняли роль удаленных операторов и действовали по инструкции от участников, участники видели оборудование через видеокамеры;
- удаленное управление компьютером, к которому подключено оборудование: участники подключались к компьютеру разработчиков и управляли оборудованием, наблюдая за ним с помощью видеокамер;
- разработка 3D-моделей полезной нагрузки и исполняемых программ для управления устройствами запускаемых на компьютерах разработчиков профилей или загружаемых в устройства;
- работа с виртуальными лабораториями и стендами;
- работа с общедоступным оборудованием и расходными материалами, пересылка решений разработчикам.

Всего в новом распределенном формате прошли 14 финалов в период с 17 марта по 31 апреля 2020 года. Для поддержки финалистов, проживающих в разных часовых поясах, и организации командной работы были привлечены модераторы, которые регулярно общались с участниками, контролируя их эмоциональное состояние, стимулируя вовлеченность, и давали обратную связь разработчикам профиля.

Для работы с участниками в новых обстоятельствах организаторы Олимпиады КД НТИ разработали рекомендации по управлению коммуникацией удаленных команд – материал опубликован на сайте Министерства просвещения РФ: `https://edu.gov.ru/press/2324/ministerstvo-prosvescheniya-rekomenduet-ispolzovat-principy-distancionnoy-raboty-komand-predlozhennye-kruzhkovym-dvizheniem-nti/`.

Структура отбора участников Олимпиады КД НТИ

Соревнование проходит в три этапа. Первый отборочный этап проводится в заочной форме для всех зарегистрированных школьников. Для проведения этапа используется интернет-платформа «Stepik» (http://stepik.org) интегрированная с личным кабинетом на сайте олимпиады. Участники решают олимпиадные задания по выбранным предметам (математика, информатика, физика, химия, биология, география, русский язык; таблица соответствия профилей и предметов: https://nti-contest.ru/materials/) согласно уровню: 8-9 классы и 10-11 классы. В зачет идет результат наилучшей из трех доступных попыток. Проверка решений производится на платформе автоматически. По итогам первого отборочного этапа устанавливаются проходные баллы по каждому профилю. Если у участника не хватило баллов на желаемый профиль, он может выбрать другой.

Второй отборочный этап также проводится в заочной форме на интернет-платформе «Stepik» и с использованием инженерных онлайн-симуляторов и виртуальных стендов на платформах технологических компаний или вузов – разработчиков профилей. Участники, прошедшие во второй отборочный этап, решают задачи по тематике профиля. В большинстве профилей на этом этапе требуется работа участников в командах. Этот этап является не только отборочным, но и обучающим мероприятием. Во время проведения второго этапа формируются знания и умения, связанные с выбранным профилем, а также сквозные компетенции: программирование на языках Python и C++, электроника, схемотехника и работа с микроконтроллерами, алгоритмы компьютерного зрения, 3D-моделирования и САПР.

Отборочные этапы сопровождаются подготовительными мероприятиями, разработанными профилями: дистанционными мероприятиями (вебинары); мероприятиями для самостоятельной подготовки (онлайн-курсы); мероприятиями, направленными на получение практических навыков (интенсивы). Для формирования команд создан специальный онлайн-сервис на платформе Олимпиады КД НТИ, а также проводятся мероприятия, направленные на командообразующую деятельность (специальные встречи, интенсивы, очные курсы на площадках по подготовке). Подготовительные мероприятия доступны для всех желающих и разработаны таким образом, чтобы их можно было провести на минимальном количестве оборудования. Часто такие мероприятия проводятся на площадках региональных партнеров со статусом «Методическая площадка» или «Площадка подготовки». Информация о партнерских площадках размещена в специальном разделе официального сайта олимпиады: http://nti-contest.ru/places_to_prepare/.

По итогам второго отборочного этапа список участников заключительного этапа определяется на основании рейтинга и количества мест в финале. Количество мест в финале каждого профиля определяется вместимостью площадки и количеством оборудования для проведения очных соревнований.

Заключительный этап олимпиады состоит из двух частей: индивидуальное решение предметных олимпиадных задач по предметам выбранным профилем и командная разработка инженерного решения с его последующим испытанием. Задание второй части (инженерная задача) заключительного этапа имеет свою специфику для каждого профиля.

Подготовка участников

Для вовлечения участников в олимпиаду были разработаны «Урок НТИ» и «Демо-этап», благодаря чему участники могли определиться с выбором профилей и попробовать свои силы.

«Урок НТИ» (http://nti-contest.ru/ntilessonteacher/) – это инициатива, созданная в сентябре 2018 года и направленная на распространение информации об НТИ среди школьников и привлечение их к Олимпиаде КД НТИ путем проведения уроков и занятий в школах и учреждениях дополнительного образования. Учебный материал для проведения «Урока НТИ» сформирован в виде конструктора, с помощью которого учителя могут собрать урок по теме НТИ. Для участия в проекте «Урок НТИ» зарегистрировалось 3260 педагогов и образовательных организаций.

«Демо-этап» Олимпиады КД НТИ (https://stepik.org/course/24389/) – это опубликованная подборка задач олимпиады. Демо-этап знакомит с профильными задачами, позволяет попробовать решать инженерные задачи. Перед регистрацией и выбором профиля потенциальные участники и их наставники могут познакомиться с задачами и выбрать наиболее интересный для себя профиль.

Сборники задач прошлых лет по всем профилям с 2015 по 2020 годы с решениями задач отборочных и финальных этапов и критериями оценки размещены на сайте Олимпиады КД НТИ: https://nti-contest.ru/problembooks/. Также команды разработчиков профилей подготовили видеоразборы задач 2 этапа, которые размещены в видеоканале олимпиады: https://www.youtube.com/channel/UCZV1CNpOrDNj7tuWuf35lgw/playlists.

На платформе «Stepik» собраны онлайн-курсы по решению олимпиадных заданий для подготовки к Олимпиаде КД НТИ. Эти курсы объединяют практически все задания отборочных этапов и часть заданий финала олимпиады за все время ее существования:

- https://stepik.org/course/1296/ – 2015/16 учебный год;
- https://stepik.org/course/3598/ – 2016/17 учебный год;
- https://stepik.org/course/15697/ – 2017/18 учебный год;
- https://stepik.org/course/55997/ – 2018/19 учебный год;
- https://stepik.org/course/71851/ – 2019/20 учебный год.
- https://stepik.org/course/24389/ – пробный этап для ознакомления с задачами Олимпиады КД НТИ по тематическим кластерам: «Информация», «Природа» «Производство», «Стратегия», «Техника» и «Человек».

Все указанные материалы находятся в свободном доступе и размещены на официальном сайте олимпиады, на страницах профилей, а также в разделе «Материалы для участников» с разбивкой по предметам и профилям https://nti-contest.ru/materials/.

Организаторы и партнеры Олимпиады КД НТИ

Оргкомитет Олимпиады представлен ректорами крупнейших политехнических и инженерных вузов России, руководителями технологических компаний и представителями государственных органов.

Вузы-соучредители олимпиады:

- ФГБОУ ВО «Московский политехнический университет»;
- ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»;
- ФГАОУ ВО «Национальный исследовательский Томский политехнический университет»;
- ФГАОУ ВО «Дальневосточный федеральный университет»;
- АНО ВО «Университет Иннополис».

Технологические партнеры

Олимпиада КД НТИ проводится при поддержке партнеров – технологических компаний, присутствующих на рынке инновационных технологий: ПАО «Аэрофлот», Благотворительный фонд Сбербанка «Вклад в будущее», ПАО «Ростелеком», фирма «1С», ФИОП РОСНАНО, ПАО «Газпром нефть», компания «Movavi», ПАО «ОАК», ПАО «Компания Сухой», госкорпорация «Роскосмос», компания «Bayer», Фонд новых форм развития образования, сеть детских технопарков «Кванториум», компания «Спутникс», компания «Полюс-НТ», компания «BiTronicsLab», компания «КРОК», компания «Инфосистемы Джет», компания «Лоретт», компания «COEX», Академия Высоких Технологий, компания «Образование будущего» и др. Полный список организаторов и партнеров олимпиады размещен в соответствующем разделе на официальном сайте: http://nti-contest.ru/about/.

Популяризация Олимпиады КД НТИ

В период с 15 августа 2019 г. по 01 апреля 2020 г. об Олимпиаде КД НТИ вышло 5174 публикации. 1571 публикация вышла в СМИ федерального уровня, 3548 – в СМИ на региональном уровне, 51 – в зарубежных СМИ. Охват из открытых источников: 450,4 млн.; МедиаИндекс – 27,805; количество реакций на публикации олимпиады в социальных сетях (likes и shares) – 9234. Об Олимпиаде КД НТИ вышло более 10 телевизионных сюжетов на федеральных и региональных каналах: «Россия - 1», «Россия – Культура», «Телеинформ», ГТРК «Амур», «Новгородское областное телевидение», «Аист-ТВ», «Телеканал ТюмГУ», «Удмуртия-24» и др.

Во время проведения отборочных этапов Олимпиада КД НТИ освещалась в федеральных, массовых, родительских, образовательных и других медиа («ИТАР-ТАСС», «РИА-Новости», «Интерфакс», «Такие Дела», Letidor, «Дети Mail.ru», «Индикатор», «Занимательная робототехника», Rusbase, «Учёба.Ру»), на официальных образовательных порталах и порталах органов государственности власти в регионах. На радио «MediaMetrics» регулярно выходила программа «Выше среднего», гостями выступали разработчики профилей олимпиады и ее партнеры. Кампания по привлечению шла также в научно-популярных группах и группах вузов и площадок партнеров.

Заключительный этап Олимпиады КД НТИ в 2019/20 учебном году проходил при поддержке федеральных и региональных СМИ, телевизионных каналов, популярных молодежных и социальных порталов. В период проведения финалов на очных площадках большинство представителей профилей подготовили ролики и фотографии, которые активно распространялись в социальных сетях.

Широкое освещение мероприятий заключительного этапа имеет целью распро-

странение информации среди потенциальных участников Олимпиады КД НТИ будущего года и привлечение талантливых школьников со всей России. Список лучших материалов об олимпиаде: http://nti-contest.ru/publications/.

Профиль «Интеллектуальные робототехнические системы»

Профиль "Интеллектуальные робототехнические системы", проводимый в рамках Олимпиады НТИ 2019-2020 учебного года, был посвящен изучению алгоритмов компьютерного зрения и межагентному взаимодействию. Необходимость высокого уровня подготовки участников для решения данных задач диктовала логику проведения отборочных этапов: необходимо было не только выявить школьников, заинтересованных в решении сложной финальной задачи, но и дать необходимые знания для ее решения.

Первый отборочный дистанционный этап (индивидуальный) определял общий уровень подготовки школьников по предметам математика и информатика. Решая задачи по программированию, школьники должны были продемонстрировать простейшие навыки составления и отладки программ, обрабатывающих массивы данных, и понимание таких тем, как комбинаторика, операции со строками, вычислительная геометрия, теория графов. Задачи по математике проверяли у участников знания по алгебре, комбинаторике, геометрии. Количество попыток сдачи решения задач не ограничивалось. Таким образом, задачи первого этапа выявляли наличие у участников знаний необходимых не только для решения задач следующего этапа, но и финальной задачи.

Задачи второго отборочного этапа были разработаны таким образом, что их было бы сложно решить индивидуальному участнику, поэтому школьники должны были объединиться в команды для успешного прохождения в финал. Задачи требовали от участников погружения в такие робототехнические темы, как навигация робототехнических устройств, планирование маршрутов перемещения, построение карты с использованием экстероцептивных сенсоров, компьютерное зрение, цифровая обработка сигналов. Для получения дополнительной информации, необходимой для решения задач второго этапа, командам были предложены образовательные материалы, разработанные в Университете Иннополис.

Для команд, прошедших в финал профиля, сначала дважды проводились дистанционные учебно-тренировочные сборы, когда у команд имелась возможность познакомиться со средой имитационного программирования, которая в дальнейшем использовалась на финал, и порешать в ней задачи, элементы которых были включены в финальную командную задачу.

Также финалисты приглашались на очные учебно-тренировочные сборы, где они могли познакомиться с аппаратными особенностями платформы, на которой предстояло решать задачу финала. В течение сборов команды оттачивали умение управлять наземным мобильным роботом, изучали специфику работы с цифровыми датчиками, реализовывали простейшие алгоритмы обработки изображения, захваченного с камеры робототототехнического устройства, а также организовывали передачу информации между несколькими роботами по беспроводному каналу данных.

Таким образом при решении финальной задачи в очном заключительном эта-

пе участники могли использовать все знания и наработки, которые они сделали во время участия во втором туре и учебно-тренировочных сборах. Несмотря на то, что задача финала была заранее неизвестна, ее элементы были рассмотрены на предварительных этапах, что значительно упрощало реализацию алгоритма управления робототехническим устройством в течение 3.5 соревновательных дней. Дополнительной частью заключительного этапа являлся индивидуальный тур, в ходе которого участники решали задачи по математике и информатике. Задачи по математике покрывали следующие области математики: оптимизация, комбинаторика, алгебра и геометрия. А темы задач по информатике перекликались с классическими темами всероссийской олимпиады школьников.

Первый отборочный этап

Первый отборочный тур проводится индивидуально в сети Интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Для каждой из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике общие для всех участников. Решение задач по информатике предполагало написание программ. Участники не были ограничены в выборе языка программирования для решения задач. На решение задач каждого предмета первого отборочного этапа участникам давалось 2 дня. У участников было два временных слота по 2 дня каждый, когда они могли решать задачи по предмету. Решение каждой задачи дает определенное количество баллов.

Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и информатика) — суммарно от 0 до 200 баллов.

Задачи первого этапа. Информатика

Первая попытка

Задача I.1.1.1. Фанатам стратегий (20 баллов)

Миссия простая: нужно либо накопить c кредитов (читайте: единиц вымышленной валюты), либо сокрушить врага в этом регионе. Миссия проходная и не интересная (даже никаких юнитов кроме пехоты), так что надо её закрыть побыстрее.

Изначально у нас нет ни одного кредита и ни одного отряда пехоты. Но есть база и гарантия того, что враг нас не обнаружит, пока мы на него не нападём.

На нашей базе можно хранить не более x кредитов. Чтобы хранить больше, нужно строить хранилища: каждое стоит y ($y \leqslant x$) кредитов и в каждом можно будет хранить дополнительно по максимум z кредитов. Строительство моментально (а вы что, реализма ожидали?).

Через каждые m минут нам привозят добычу на s кредитов. Добыча моментально конвертируется в кредиты и мы перераспределяем новые s кредитов и кредиты, имеющиеся на базе, по трём направлениям: строительство хранилищ, наём отрядов пехоты, хранение на базе. Количество кредитов, отданных на строительство хранилищ, должно быть кратно стоимости постройки одного хранилища. Количество кредитов, отданных на наём отрядов пехоты, должно быть кратно стоимости наёма одного отряда пехоты. Если мы отдаём на хранение больше кредитов, чем может храниться на базе (хранилища, на строительство которых мы только что отдали кредиты, тоже учитываются), то излишки кредитов исчезают.

Если после очередного описанного выше распределения на базе будет храниться хотя бы c кредитов, то считается, что мы накопили c кредитов, и миссия считается пройденной.

Один отряд пехоты стоит f ($f \leqslant x$) кредитов. Сразу после оплаты, которая тоже

моментальна, отряд будет ждать приказаний на нашей базе.

Проще и быстрее всего избавиться от врага в этом регионе – заставить его капитулировать. Для этого нужно привести к стенам вражеской базы больше отрядов пехоты, чем есть на вражеской базе.

По данным разведки, на вражеской базе e отрядов, а добраться до вражеской базы наши отряды пехоты смогут за t минут (все отряды, независимо от их количества, могут идти вместе).

За сколько в лучшем случае мы завершим миссию?

Формат входных данных

Единственная строка содержит 9 целых чисел $c, x, y, z, m, s, f, e, t$ ($1 \leqslant c, x, y, z, m, s, f, e, t \leqslant 10^9$; $f, y \leqslant x$).

Формат выходных данных

Выведите одно целое число – искомое количество минут.

Пояснения

Чтобы максимально быстро накопить 2700 кредитов, мы можем построить 2 хранилища через 3 минуты после начала (в этот момент у нас появятся первые кредиты) и далее просто складировать все кредиты на базе. Тогда через 9 минут после начала на базе накопится необходимая сумма.

Чтобы заставить врага сдаться, мы можем через 3 минуты после начала нанять 10 отрядов пехоты и сразу же пойти на вражескую базу. Тогда через 6 минут после начала враги сдадутся.

Примеры

Пример №1

Стандартный ввод
`2700 1000 150 1000 3 1000 100 7 3`
Стандартный вывод
`6`

Решение

В данной задаче нужно выбрать быстрейший путь: накопить c кредитов или сокрушить врага.

Рассмотрим накопление. В этом случае для минимизации времени стоит тратить кредиты только на постройку хранилищ. Так как стоимость одного хранилища не превышает изначального максимального количества кредитов, которое мы можем хранить на базе, то есть $y \leqslant x$, при поставке добычи мы всегда можем распределить кредиты так, что мы отдаём на хранение меньше кредитов, чем может храниться на базе.

Таким образом, если нужны хранилища, то есть $c > x$, достаточно построить $\left\lceil \frac{c-x}{z} \right\rceil$ хранилищ. Получается, нужно набрать в сумме $c + \left\lceil \frac{\max(0, c-x)}{z} \right\rceil \cdot y$ кредитов, что мы сделаем за $\left\lceil \frac{c+\left\lceil \frac{\max(0, c-x)}{z} \right\rceil \cdot y}{s} \right\rceil \cdot m$ минут.

Рассмотрим капитуляцию врага. В этом случае для минимизации времени стоит тратить кредиты только на наём отрядов пехоты. Так как стоимость одного отряда пехоты не превышает изначального максимального количества кредитов, которое мы можем хранить на базе, то есть $f \leqslant x$, при поставке добычи мы всегда можем распределить кредиты так, что мы отдаём на хранение меньше кредитов, чем может храниться на базе.

Чтобы заставить врага капитулировать, достаточно нанять $e + 1$ отрядов пехоты и привести все эти отряды к стенам вражеской базы. Для этого придётся потратить в сумме $(e + 1) \cdot f$ кредитов на пехоту, что мы сделаем за $\left\lceil \frac{(e+1) \cdot f}{s} \right\rceil \cdot m$ минут. Плюс t минут на то, чтобы добраться до врага. Итого $\left\lceil \frac{(e+1) \cdot f}{s} \right\rceil \cdot m + t$ минут.

Минимум из двух описанных значений и является ответом. Также стоит отметить, что 64-битных типов данных для полного решения будет недостаточно.

Пример программы-решения

Ниже представлено решение на языке Python3

```
c, x, y, z, m, s, f, e, t = [int(i) for i in input().split()]
print(min((c + (max(0, c - x) + z - 1) // z * y + s - 1) // s * m,
          ((e + 1) * f + s - 1) // s * m + t))
```

Задача I.1.1.2. Настроение профессора (20 баллов)

Все персонажи и описываемые события являются вымышленными. Любое совпадение с реальными людьми или событиями случайно.

Завтра студенты первого курса пойдут сдавать математический анализ. Экзамен будет принимать профессор Ильдар.

Экзамен будет проходить по старинке: студенты по одному подходят к профессору, отвечают на заданные им вопросы и получают свои оценки. Результат экзамена сильно зависит от настроения профессора Ильдара: если у него плохое настроение, то не важно, насколько хорошо вы подготовились, – он отправит вас на пересдачу.

Пусть настроение профессора в некоторый момент времени равно x. После ответов отличника настроение профессора повышается и становится равно $x + 1$. После ответов хорошиста настроение профессора не меняется. А если ответы явно не тянут на оценку 4, то профессор ставит 3 и его настроение падает до $x - 1$.

Но если завтра в какой-либо момент времени настроение профессора будет равно отрицательному числу, то после этого момента описанные выше закономерности перестают действовать и все студенты, что ещё не получили своих оценок, отправляются на пересдачу.

Сегодня вы (неожиданно) – староста группы и хотите, чтобы никто из ваших студентов не отправился на пересдачу. Порядок, в котором студенты будут подходить

к профессору, уже сформирован и его изменить нельзя, но вы знаете, насколько хорошо подготовился каждый из студентов, и знаете про профессора Ильдара ещё одну вещь – он любит шоколад.

Вы можете купить шоколадку (а лучше не одну) и подарить её профессору сегодня вечером. Каждая подаренная профессору шоколадка повышает его настроение на 1. Что профессор делает с шоколадками, никому не известно.

Какое минимальное количество шоколадок вам надо сегодня подарить профессору, чтобы завтра все студенты сдали экзамен?

Формат входных данных

В первой строке вводятся два целых числа n и k ($1 \leqslant n \leqslant 2 \cdot 10^5$, $-10^9 \leqslant k \leqslant 10^9$) – количество студентов в вашей группе и настроение профессора сегодня вечером (настроение профессора до начала экзамена может измениться только благодаря вам).

Во второй строке вводится строка из n символов a_i ($a_i \in \{A, B, C\}$). Эта строка описывает порядок, в котором студенты будут подходить к профессору. Каждый студент описывается одним символом. Символом A обозначается отличник, символом B – хорошист, символом C – троечник или неподготовившийся к экзамену студент.

Формат выходных данных

Выведите одно целое число – искомое минимальное количество шоколадок.

Примеры

Пример №1

Стандартный ввод
3 0 BCA
Стандартный вывод
1

Пример №2

Стандартный ввод
3 3 AAA
Стандартный вывод
0

Решение

Промоделируем сдачу экзамена без учёта отправки всех на пересдачу. Найдём минимальное настроение профессора в день экзамена (учитывая настроение до начала и не учитывая после окончания). Если оно отрицательно и равно x, то достаточно купить профессору $-x$ шоколадок перед экзаменом, чтобы никто не был отправлен на пересдачу. Если оно неотрицательно, то можно обойтись без подарков профессору.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, k = [int(i) for i in input().split()]
min_k = 10 ** 9 + 1
for c in input():
    min_k = min(k, min_k)
    k += ord('B') - ord(c)

print(max(0, -min_k))
```

Задача I.1.1.3. Комната ярости (20 баллов)

Гертруда имеет n тарелок. И хочет разбить их все. По одной. Но тарелки бьются очень звонко. Она опасается, что повредит слух.

Известно, что сила звона первой разбитой тарелки будет равна a_1. Сила звона каждой последующей разбитой тарелки будет в b раз больше силы звона предыдущей. То есть сила звона i-ой $(i > 1)$ разбитой тарелки будет равна $a_i = a_{i-1} \cdot b$.

Гертруда знает максимальное суммарное значение сил звона MAX, которое могут выдержать её уши, и желает максимально насладиться звуками бьющейся посуды.

Помогите Гертруде, найдите максимальное количество тарелок, которые она может разбить, не повредив слух. И побыстрее.

Формат входных данных

В первой строке вводятся четыре целых числа n, a_1, b, MAX $(1 \leqslant n, MAX \leqslant 10^{1000}, 1 \leqslant a_1, b \leqslant 10)$.

Формат выходных данных

Выведите максимальное количество тарелок.

Пояснения

Если Гертруда разобьёт одну тарелку, то суммарное значение сил звона будет равно $a_1 = 1$.

Если разобьёт две тарелки, то суммарное значение будет равно $a_1 + a_2 = 1 + 2 = 3$.

Если разобьёт три, то – $a_1 + a_2 + a_3 = 1 + 2 + 4 = 7$.

Примеры

Пример №1

Стандартный ввод
10 1 2 4

Стандартный вывод
2

Решение

В данной задаче для нахождения ответа можно было воспользоваться формулой суммы первых k элементов геометрической прогрессии и бинпоиском по ответу, но ограничения на входные данные устроены так, что можно было просто промоделировать разбивание тарелок по одной.

Для работы с длинными целыми числами стоило либо использовать соответствующий встроенный тип данных, либо писать свою реализацию длинной арифметики.

Первые подзадачи решались без длинной арифметики.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, ai, b, max = [int(i) for i in input().split()]

if b == 1:
    print(min(max // ai, n))
    exit(0)

# b > 1
l, s = 1, ai
while l <= n:
    if s > max:
        print(l - 1)
        exit(0)
    ai *= b
    s += ai
    l += 1

print(n)
```

Задача I.1.1.4. Подлизы (20 баллов)

Жила-была девочка Катя, и было у неё много-много денег. И подруг. Ну как подруг...

И собрались они как-то раз все вместе у Кати дома и обсуждали фильмы. Многие хвалили вкус Кати. Редко кто не соглашался с её мнением. О вкусах, конечно, не спорят, но Кате показалось это странным и она решила устроить проверку.

Катя записала m пар фильмов, которые девочки сравнивали, и для каждой такой пары пометила, какой из фильмов девочки посчитали однозначно лучше другого. А потом воспользовалась своим обаянием влиянием и убедила школьного психолога провести тестирование, в котором есть вопрос о трёх любимых фильмах. Вот так вот всё просто, когда ты – Катя.

Среди неиспорченных бланков тестирования (не спрашивайте, как она их достала) Катя нашла заполненные бланки n своих подруг. Скажите, согласовываются ли записи Кати с каждым из ответов на вопрос о трёх любимых фильмах в отдельности.

Формат входных данных

В первой строке заданы числа n и m ($1 \leqslant n, m \leqslant 10^5$).

В следующих m строках – пары фильмов, записанные у Кати. Первый фильм в паре считается лучше второго.

В следующих n строках – списки любимых фильмов девочек. Первый фильм в тройке считается лучше второго, а второй – лучше третьего.

Записи Кати непротиворечивы. Каждая пара фильмов в записях Кати встречается не более одного раза.

Так сложилось, что все фильмы, что встречаются в списках любимых фильмов девочек, встречаются и в записях Кати, а в каждом отдельно взятом списке все три фильма различны.

Для вашего же удобства названия фильмов во входных данных заменены на положительные натуральные числа, не превышающие 10^6.

Формат выходных данных

Выведите n строк, в i-ой из которых должно быть написано *honest*, если список любимых фильмов из i-го бланка не противоречит записям Кати, или *liar*, если противоречит.

Не выводите лишние пробелы в конце или начале строк - это будет считаться за ошибку.

Пояснения

Тройка фильмов 1 2 4 противоречит записям Кати, так как по записям Кати фильм 5 лучше фильма 4, но его нет в тройке.

Тройка фильмов 1 3 2 противоречит, так как по записям Кати фильм 2 лучше фильма 3, а в тройке фильм 3 стоит до фильма 2.

Тройка фильмов 5 4 8 противоречит, так как по записям Кати фильм 2 лучше фильма 4, но его нет в тройке.

Примеры

Пример №1

Стандартный ввод
5 8 1 3 1 2 2 3 2 4 4 8 5 4 5 6 7 6 1 2 3 1 2 4 1 3 2 5 4 8 5 7 6

Стандартный вывод
honest liar liar liar honest

Решение

Из условия следует, что тройка любимых фильмов подруги Кати может противоречить записям Кати одним из двух способов (или сразу обоими способами):

1. Существует фильм, который лучше одного из тройки, но в эту тройку не включён;
2. 2 фильма из тройки сравнивали и фильм, который хуже по записям Кати, оказался в тройке на месте выше, чем фильм, который лучше по записям Кати.

Также можно вывести определение того, какая тройка фильмов не противоречит записям Кати.

Такая тройка удовлетворяем всем следующим условиям:

1. Не существует фильма, который по записям Кати лучше первого фильма в тройке;
2. Фильмов, которые по записям Кати лучше второго фильма в тройке,
   либо не существует,
   либо такой фильм только один и это первый фильм в тройке;
3. Фильмов, которые по записям Кати лучше третьего фильма в тройке,
   либо не существует,
   либо такой фильм только один и это первый фильм в тройке,
   либо такой фильм только один и это второй фильм в тройке,
   либо таких фильм ровно два и это первый и второй фильмы в тройке.

Остаётся лишь аккуратно реализовать проверку каждой из троек фильмов.

Асимптотика: $O(m+n)$

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, m = [int(i) for i in input().split()]

better = [[] for i in range(int(1e6) + 1)]

for i in range(m):
    film1, film2 = [int(i) for i in input().split()]
    better[film2].append(film1)

for i in range(n):
    film1, film2, film3 = [int(i) for i in input().split()]
    ok = len(better[film1]) == 0
    for f in better[film2]:
        if not f == film1:
            ok = False
            break
    for f in better[film3]:
        if (not f == film1) and (not f == film2):
            ok = False
            break
    print("honest" if ok else "liar")
```

Задача I.1.1.5. Циклические сдвиги vs разворот строки (20 баллов)

На этой неделе на уроках информатики Васе рассказывают про строки.

Вчера Вася узнал, что такое циклический сдвиг:

k-й циклический сдвиг строки – это строка, полученная перестановкой первых k символов строки в её конец. В частности, 0-й циклический сдвиг строки – это сама строка.

И он написал программу, которая умеет перемещать первый символ строки в её конец k раз, получая таким образом k-й циклический сдвиг строки.

Сегодня Васе нужно реализовать разворот строки. Но у Васи тренировка. Ему некогда писать новые сложные программы. Поэтому он задался вопросом:

Можно ли циклическими сдвигами развернуть строку?

Благодаря своим друзьям Вася узнал, на какой строке (да, всего одной) будет тестировать его программу учитель, у которого нет времени рецензировать код каждого ученика. Поэтому вопрос упростился:

Можно ли циклическими сдвигами развернуть строку s?

Помогите ему ответить на этот вопрос.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 3 \cdot 10^5$) – длина строки s.

Во второй строке – сама строка s, на которой будет тестировать программу Васи учитель. Состоит строка s только из строчных латинских букв.

Формат выходных данных

Если строку нельзя развернуть циклическими сдвигами, то выведите число -1. В противном случае выведите такое целое число k ($0 \leqslant k < n$), что k-й циклический сдвиг строки s равен развёрнутой строке s. Если таких k несколько, выведите любое из них.

Пояснения

0-й циклический сдвиг строки s равен $abac$.

1-й циклический сдвиг строки s равен $baca$.

2-й циклический сдвиг строки s равен $acab$.

3-й циклический сдвиг строки s равен $caba$.

Развёрнутая строка s равна $caba$.

Единственное подходящее k равно трём.

Примеры

Пример №1

Стандартный ввод
4 abac
Стандартный вывод
3

Решение

Условие задачи можно свести к одному вопросу: можно ли циклическими сдвигами развернуть строку s?

Для нахождения ответа можно было искать развёрнутую строку s в строке $t = s + s$, так как все различные циклические сдвиги строки s являются подстроками строки t.

Наивная реализация этого поиска имеет асимптотику $O(n^2)$ по времени. За $O(n)$ по времени данную задачу можно было решить, например, с помощью алгоритма Кнута-Морриса-Пратта.

Другой способ решения: проверим, что строку s можно разрезать на два палиндрома. Быстро сделать это можно было, например, с помощью алгоритма Манакера.

Пример программы-решения

Ниже представлено решение на языке Python3

```python
def pi(s):
    n = len(s)
    pi = [0] * n
    j = 0
    for i in range(1, n):
        while j > 0 and s[i] != s[j]:
            j = pi[j - 1]
        if s[i] == s[j]:
            j += 1
        pi[i] = j
    return pi


n = int(input())
s = input()

pi = pi(s[::-1] + '#' + s + s)
for i in range(n):
    if pi[n * 2 + i] == n:
        print(i)
        break
else:
    print(-1)
```

Вторая попытка

Задача I.1.2.1. Фанатам стратегий 2 (16 баллов)

В текущей миссии, очевидно, необходимо укрепить базу, прежде чем идти в открытый бой.

Для обеспечения устойчивой обороны требуется построить n различных новых зданий. Но не всё так просто.

Для поддержания процессов, которые будут происходить в этих зданиях, необходимо электричество. А получать электроэнергию новые здания могут только от новых электростанций. Новых электростанций на базе нет, так что их тоже придётся построить.

Зная, сколько единиц электроэнергии в единицу времени производит одна новая электростанция и количество электроэнергии, потребляемое за единицу времени каждым из упомянутых выше n новых зданий, определите минимальное количество электростанций, которое необходимо для полного функционирования требуемых n зданий.

Примечание: Считается, что электростанции не потребляют электроэнергии и среди n различных новых зданий, которые требуется построить, нет электростанции.

Формат входных данных

В первой строке заданы числа n и e ($1 \leqslant n \leqslant 10^5, 1 \leqslant e \leqslant 10^9$) – количество требуемых зданий и количество единиц электроэнергии, которое производит одна новая электростанция.

Во второй строке даны n чисел – количество единиц электроэнергии, потребля-

емое за единицу времени каждым из зданий. Все числа во второй строке неотрицательны и не превышают 10^9.

Формат выходных данных

Выведите одно целое число – минимальное количество электростанций, которое необходимо для полного функционирования требуемых n зданий.

Пояснения к примеру

Двух электростанции явно мало. Электроэнергию трёх электростанций можно распределить по зданиям следующим образом: на первое здание идёт 5 единиц от первой электростанции, на второе – 8 единиц от второй электростанции и 4 единицы от третьей, а на третье – 3 единицы от первой электростанции и 4 единицы от третьей.

Примеры

Пример №1

Стандартный ввод
3 8 5 12 7
Стандартный вывод
3

Решение

Пусть sum – суммарное количество единиц электроэнергии, потребляемое за единицу времени описанными n зданиями. Тогда, чтобы эти здания работали, нужно построить $\left\lceil \frac{sum}{s} \right\rceil$ новых электростанций.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, s = [int(i) for i in input().split()]
sm = sum(int(i) for i in input().split())
print((sm + s - 1) // s)
```

Задача I.1.2.2. I Don't Like (20 баллов)

Кому-то не нравятся наши задачи. Наверно, из-за их сложности. Кто-то ругает нас за то, что программа не компилируется на компиляторах, имеющихся на Stepik, или просто не проходит наши тесты к задаче, хотя у кого-то на компьютере всё работает. Кто-то, не указывая на недочёты в задаче, хочет, чтобы ему или ей разжевали условие задачи, и после возмущается, прочитав, что мы не делаем пояснений и кратких пересказов условий, так как некорректностей найдено не было и мы хотим оставить всех участников олимпиады в равных условиях. А кто-то считает, что его

тесты к задаче не хуже тех, что создали мы, и его решение верно, так как на его тестах оно работает (да, и такие есть). А кто-то списывает.

Всем этим замечательным людям мы можем лишь пожелать здоровья и бесконечного количества нервных клеток. Смириться с правилами олимпиады тоже не помешает.

А маленькому Коле не нравится, когда числа в массиве не отсортированы по возрастанию (если быть точным, по неубыванию, но Коля таких слов не знает).

Вот кто придумал дарить детям неотсортированные массивы? Мы не знаем, но Коля сегодня получил именно такой подарок. Он даже решил посчитать число таких пар индексов массива (i, j), что $i < j$ и $a_i > a_j$, чтобы хоть как-то измерить силу своей ненависти к подаренному ему массиву a и тому человеку, который это сделал.

Коля устал злиться, но сумеет сделать ещё ровно одно действие – поменять два элемента массива a местами. Ручки у него короткие, так что Коля может поменять местами только соседние элементы массива a (то есть такие элементы, индексы которых различаются не более чем на 1).

Определите количество способов, которыми Коля может уменьшить описанное выше число пар индексов. Два способа считаются различными, если существует индекс, который встречается только в одной из двух пар индексов, описывающих эти два способа.

Формат входных данных

В первой строке задано число n ($1 \leqslant n \leqslant 10^5$) – количество элементов в массиве a.

Во второй строке даны n чисел a_i ($-10^9 \leqslant a_i \leqslant 10^9$) – элементы массива a.

Гарантируется, что числа в массиве a не упорядочены по неубыванию.

Формат выходных данных

Выведите одно целое число – количество способов, которыми Коля может уменьшить описанное выше число пар индексов.

Примеры

Пример №1

Стандартный ввод
3 1 3 2

Стандартный вывод
1

Решение

Как изменится описанное в условии число пар индексов, если мы поменяем местами элементы a_i и a_{i+1} ($1 \leqslant i < n$)? Если $a_i > a_{i+1}$, то оно уменьшится на единицу; если $a_i < a_{i+1}$, то – увеличится на единицу; если $a_i = a_{i+1}$, то – не изменится.

Таким образом, в данной задаче достаточно посчитать количество пар соседних элементов в массиве a, в которых $a_i > a_{i+1}$.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n = int(input())
a = [int(i) for i in input().split()]
print(sum(1 if a[i] > a[i + 1] else 0 for i in range(n - 1)))
```

Задача I.1.2.3. Мультимножества (20 баллов)

Дан набор из n чисел. Каждое число отнесли ровно к одному из 5-и мультимножеств: A, B, C, D или E.

По итогу такого распределения чисел получилось так, что все 5 мультимножеств непусты, суммы элементов мультимножеств равны и соблюдается следующее условие:

Для любых $a \in A$, $b \in B$, $c \in C$, $d \in D$ и $e \in E$ выполняется неравенство $a \leqslant b \leqslant c \leqslant d \leqslant e$.

Определите, правда ли, что такое могло произойти.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 10^5$) – размер набора чисел.

Вторая строка содержит n целых чисел a_i ($-10^9 \leqslant a_i \leqslant 10^9$) – сами числа набора.

Формат выходных данных

Выведите Yes, если возможно разбиение данных n чисел на мультимножества. Иначе выведите No.

Примеры

Пример №1

Стандартный ввод
19 2 1 1 2 2 0 2 3 11 3 3 4 3 4 0 6 5 1 2

Стандартный вывод
Yes

Решение

Пусть sum – сумма всех чисел данного набора, а $\max(X)$ и $\min(X)$ – наибольший и наименьший элементы множества X соответственно.

Тогда сумма элементов каждого из мультимножеств в отдельности равна $sum/5$ и описанное условие эквивалентно следующему:

$$\max(A) \leqslant \min(B) \leqslant \max(B) \leqslant \min(C) \leqslant \max(C) \leqslant \min(D) \leqslant \max(D) \leqslant \min(E)$$

Следовательно, для нахождения возможных мультимножеств стоит жадно добавлять наименьшие числа из набора в мультимножество A, пока сумма его элементов не достигнет $sum/5$, потом в мультимножество B, пока сумма его элементов не достигнет $sum/5$, и так далее. Если получилось сформировать мультимножества, удовлетворяющие всем условиям, то ответ `Yes`, иначе – `No`.

Асимптотика: $O(n \cdot \log(n))$

Пример программы-решения

Ниже представлено решение на языке Python3

```python
n = int(input())
a = [int(i) for i in input().split()]
a.sort()
sum_a = sum(a)

if sum_a % 5 != 0:
    print('No')
    exit(0)

cur, cnt = 0, 1
for i in a:
    cur += i
    if cur == sum_a // 5 * cnt:
        cnt += 1
        if cnt == 6:
            break

print('Yes' if cnt == 6 else 'No')
```

Задача I.1.2.4. Фанатам стратегий 3 (24 баллов)

Данная задача – логическое продолжение задачи "Фанатам стратегий 2". Рекомендуем перед решением данной задачи полностью решить задачу "Фанатам стратегий 2".

Вскоре стало понятно, что всё совсем не просто. Нельзя взять и построить здание. Их в этой игре ещё и открыть нужно.

Новое здание типа A можно построить, только если на нашей базе функционирует хотя бы по одному новому зданию из списка необходимых зданий здания типа A.

Сколько на самом деле нам придётся построить зданий (не считая электростанций)? Какие они? В каком порядке их строить? Ваша задача – найти ответы на эти вопросы.

Примеры

Гарантируется, что существует такая последовательность постройки зданий, что здания всех типов можно построить.

Формат входных данных

В первой строке записаны три целых числа n, m и t ($1 \leqslant m \leqslant n \leqslant 5 \cdot 10^4$; $1 \leqslant t \leqslant 2$) – количество различных типов новых зданий в игре, количество новых зданий, которые нужно построить, и номер формата выходных данных.

В следующей строке записаны m названий типов зданий, разделённых пробелами – требуемые для обеспечения устойчивой обороны здания. Гарантируется, что строка не содержит одинаковых типов зданий.

Далее идёт n блоков по 2 строки следующего вида:

В первой строке – название типа здания.

Во второй – длина списка необходимых зданий для здания данного типа и сам список необходимых зданий. Гарантируется, что список не содержит одинаковых типов зданий.

Сумма длин списков необходимых зданий не превышает $5 \cdot 10^4$.

Название каждого типа здания состоит только из латинских букв и имеет длину не более десяти символов.

Формат выходных данных

Если $t = 1$, то выведите одно число – минимальное количество зданий, которые нужно построить.

Если $t = 2$, то в первой строке выведите одно число – минимальное количество зданий, которое необходимо построить, а во второй – k названий зданий, которые нужно построить, в том порядке, в котором их нужно строить. Если существует несколько подходящих последовательностей – выведите любую из них.

Примеры

Пример №1

Стандартный ввод
13 5 2 refinery vehicle repair palace turret constryard 0 windtrap 1 constryard refinery 1 windtrap outpost 1 windtrap silo 2 refinery constryard vehicle 3 refinery windtrap outpost barracks 2 constryard outpost wall 1 outpost turret 1 outpost starport 2 silo refinery repair 1 vehicle hitech 3 vehicle wall outpost palace 1 starport

Стандартный вывод
10 constryard windtrap refinery outpost silo vehicle turret starport repair palace

Решение

Если представить, что здание типа A – это вершина орграфа, а список необходимых зданий здания типа A – список рёбер, направленных в вершину-здание типа A из других вершин-зданий, то задачу можно свести к задаче о нахождении топологической сортировки, с той лишь разницей, что вывести требуется не все вершины орграфа, а только те, из которых доступна хотя бы одна из m выделенных вершин-зданий.

Вершины-здания, которые нужно выводить, можно определить, например, с помощью серии обходов в глубину из m выделенных вершин-зданий, предварительно развернув в орграфе все рёбра. Порядок, в котором можно выводить эти вершины, может задать топологическая сортировка этих вершин. Топологическую сортировку можно найти, например, с помощь алгоритма Кана или алгоритма Тарьяна.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, m, tp = [int(i) for i in input().split()]
need = set(input().split())
lists = {}
rev_lists = {}

for i in range(n):
    s = input()
    lists.setdefault(s, [])
    rev_lists.setdefault(s, [])
    for t in input().split()[1:]:
        lists.setdefault(t, [])
        lists[t].append(s)
        rev_lists[s].append(t)

q = [""] * n
ql = qr = 0
for s in lists.keys():
    if len(rev_lists[s]) == 0:
        q[qr] = s
        qr += 1

cnt = {}
while ql < qr:
    s = q[ql]
    ql += 1
    for t in lists[s]:
        cnt.setdefault(t, 0)
        cnt[t] += 1
        if cnt[t] == len(rev_lists[t]):
            q[qr] = t
            qr += 1

for s in reversed(q):
    if s in need:
        for t in rev_lists[s]:
            need.add(t)

ans = []
for s in q:
    if s in need:
        ans.append(s)

print(len(ans))
if tp == 2:
    print(' '.join(map(str, ans)))
```

Задача I.1.2.5. Тир (20 баллов)

Давно были в тире? Мы недавно.

В нашем тире висят и стоят жестяные и алюминиевые банки из под различных напитков. Точнее, висели и стояли.

От наших выстрелов банки мотались из стороны в сторону на верёвке, срывались, звенели, мялись. Это вам не из пальцев стрелять.

Каждая из пуль либо прошла насквозь одной из банок, после чего поражённая банка упала на пол и откатилась в сторону так, что в неё было уже невозможно попасть; либо не попала ни в одну из банок. В любом случае, каждая из пуль застряла в стене, стоящей позади наших банок-мишеней.

Но тот день в прошлом. Осталась только стена с застрявшими в ней пулями и фотография. В попытке восстановить тот день и насладиться им снова мы собрали данные о положении каждой пули в стене, расположении банок и порядке выстрелов.

Помогите определить про каждую пулю, поразила ли она какую-то из банок, и если поразила, то какую именно.

Формат входных данных

В первой строке записаны два целых числа n и m $(1 \leqslant m,\ n \leqslant 10^5)$ – количество банок, которые были нашей мишенью в тот день, и количество совершённых в тот день выстрелов.

В i-ой из следующих n строк описывается положение i-ой банки. Положение задаётся координатами проекции банки на вертикальную плоскость. Проекция представляет из себя прямоугольник, стороны которого параллельны нанесённой на эту плоскость системы координат. Ось Y этой системы направлена вертикально вверх, а ось X – горизонтально. А прямоугольник задаётся парой точек – своей левой нижней и правой верхней вершинами.

Гарантируется, что ни одна пара этих прямоугольников не имеет ни одной общей точки.

В i-ой из следующих m строк описывается положение i-ой пули в стене. Пули заданы в том же порядке, в котором они выходили из наших дул. Сама стена строго вертикальна, поэтому мы можем считать, что положение задаётся координатами проекции пуль на вертикальную плоскость. Причём траектории движения пуль были строго перпендикулярна этой плоскости. Сами точки задаются парой координат в уже описанной выше системе координат.

Расстояние между банками и стеной по сравнению с расстоянием до стреляющих настолько мало, что мы им пренебрегаем.

Примечание: Значения координат по модулю не превышают 10^9.

Формат выходных данных

В первой и единственной строке выведите m чисел, i-ое из которых говорит, какую из банок i-я пуля прошла насквозь, если такая имеется. Если i не задела ни одну банку, то выведите -1, иначе выведите порядковый номер во входных данных банки, которую поразила i-я пуля.

Примеры

Пример №1

Стандартный ввод
4 10 0 0 1 1 2 3 3 8 15 15 20 20 10 12 12 13 2 2 0 -1 23 18 13 12 10 13 16 16 17 17 3 5 3 5 3 3

Стандартный вывод
-1 -1 -1 -1 4 3 -1 2 -1 -1

Решение

Один из способов решения:

Разобьём каждый прямоугольник на вертикальные отрезки: открывающий (с меньшей x координатой) и закрывающий (с большей x координатой).

Получившиеся отрезки отсортируем и будем обходить по неубыванию x координаты («запустим вертикальную сканирующую прямую от $-\infty$ до $+\infty$»). При равенстве x координаты сначала будем обрабатывать открывающие отрезки.

При обработке открывающего отрезка мы добавляем в некоторую структуру информацию о том, что сканирующая прямая на данный момент пересекает соответствующий отрезку прямоугольник. При обработке закрывающего отрезка мы удаляем из той же структуры информацию о том, что сканирующая прямая на данный момент пересекает соответствующий отрезку прямоугольник.

Так как прямоугольники не имеют общих точек, при любом положении сканирующей прямой открывающие отрезки, которые ещё «не закрыли», будут образовывать множество непересекающихся отрезков.

Следовательно можно множество открытых отрезков поддерживать, например, в set'е.

Если отсортировать точки-пули по неубыванию x координаты и параллельно обходить и их тоже, то мы сможем для каждой точки однозначно определить соответствующий набор открывающих отрезков; остаётся лишь найти ближайший к этой точке по y координате отрезок и проверить, лежит ли по y координате эта точка на нём. Для быстрого определения этого в set'е имеет смысл хранить открывающие отрезки в порядке, например, возрастания их «нижних» y координат и пользоваться функцией `lower_bound`.

Однако, так как мы изменили изначальный порядок точек-пуль, в случае если мы нашли для точки-пули прямоугольник, в который она попадает, это не значит, что данная пуля поразила соответствующую этому прямоугольнику банку. В проекцию банки могли попасть несколько проекций пуль, но только пуля, которая была выпущена раньше других, поразила банку.

Давайте для каждой проекции банки сохраним список проекций пуль, которые в него попадают. Тогда после окончания движения сканирующей прямой мы сможем определить для каждой банки, поразила ли её какая-либо пуля, и если поразила, то какая именно. А по этим данным мы можем восстановить ответ на задачу.

Асимптотика: $O(n \cdot \log(n))$

Пример программы-решения

Ниже представлено решение на языке C++

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define P pair
#define fi first
#define se second

#define V vector
typedef V<int> Vi;
typedef V<ll> Vll;

#define all(v) (v).begin(), (v).end()

#define forn(i, n) for (int (i) = 0; (i) < (n); (i)++)

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);

    int n, m;
    cin >> n >> m;
    Vll leftX(n), downY(n), rightX(n), upY(n);
    forn(i, n) {
        cin >> leftX[i] >> downY[i] >> rightX[i] >> upY[i];
    }
    Vll x(m), y(m);
    forn(i, m) {
        cin >> x[i] >> y[i];
    }

    V<P<ll, int>> s(n * 2 + m);
    forn(i, n) {
        s[i * 2] = {leftX[i], -i - 1};
        s[i * 2 + 1] = {rightX[i], m + i};
    }
    forn(i, m) {
        s[n * 2 + i] = {x[i], i};
```

```
    }
    sort(all(s));

    Vi killed_by(n);
    fill(all(killed_by), INT_MAX);

    map<ll, int> mp;

    forn(i, n * 2 + m) {
        if (s[i].se < 0) {
            mp[downY[-s[i].se - 1]] = -s[i].se - 1;
        } else if (s[i].se >= m) {
            mp.erase(downY[s[i].se - m]);
        } else if (!mp.empty() && mp.upper_bound(y[s[i].se]) != mp.begin() &&
                   y[s[i].se] <= upY[(--mp.upper_bound(y[s[i].se]))->se]) {
            killed_by[(--mp.upper_bound(y[s[i].se]))->se] =
                min(killed_by[(--mp.upper_bound(y[s[i].se]))->se], s[i].se);
        }
    }

    Vi ans(m);
    fill(all(ans), -1);
    forn(i, n) {
        if (killed_by[i] < INT_MAX) {
            ans[killed_by[i]] = i + 1;
        }
    }
    forn(i, m) {
        cout << ans[i] << ' ';
    }

    return 0;
}
```

Задачи первого этапа. Математика

Первая попытка. Задачи 8-9 класса

Задача I.2.1.1. (10 баллов)

Вася взял число 5689756193846349 и вычеркнул из него 8 цифр. В результате у него получилось максимально возможное число, которое таким образом можно получить из исходного. В ответ укажите это число.

Решение

Нужно вычеркнуть 8 цифр, чтобы число получилось максимально возможным. Для этого вы стараемся, использую оставшиеся зачеркивания, добиться того, чтобы на текущем месте стояла максимальная цифра, начиная с первого места.

Ход решения: у нас есть 8 зачеркивания и нужно, чтобы первая цифра была максимально возможной. Легко можно видеть, что используя 3 зачеркивания можно получить 9. Теперь у нас осталось 5 зачеркиваний и нужно, чтобы вторая цифра

была максимально возможной. Видно, что, используя 4 зачеркивания, можно и на втором месте получить 9. У нас осталось одно зачеркивание и нужно выбрать цифру на третьем месте. Используя одно зачеркивание мы можем получить 8. Зачеркивания кончились.

Ответ: 99846349.

Задача I.2.1.2. (10 баллов)

На парковке 5×4 метра электрику нужно установить фонари так, чтобы в любом квадрате 2×2 метра был как минимум один 1 фонарь. Сколько минимально фонарей понадобится электрику?

Решение

Рассмотрим квадрат 3×3. Если поставить в центр фонарь, то в какую сторону не направить квадрат 2×2, он всегда будет с 1 фонарем, а это значит что фонари нам нужно ставить на расстоянии в 1 метр. Поставим 1 фонарь на расстоянии в 1 метр от верхней и левой сторон. Ставим 2 фонарь на расстояние в 1 метр от левой стены и 1 фонаря. 3 фонарь ставим на расстояние в 1 метр от нижней и правой сторон. Ставим 4 фонарь на расстояние в 1 метр от правой стены и 3 фонаря.

Ответ: 4.

Задача I.2.1.3. (10 баллов)

Настя наугад приписала к числу 375 справа 3 цифры. Число, которое у нее получилось, оказалось наибольшим из всех чисел, которые начинаются на 375 и делятся без остатка на 5, 6 и 7. В ответ укажите это число.

Решение

Полученное число должно делиться на $5 \cdot 6 \cdot 7 = 210$. И должно быть максимально возможным. Возьмем самое большое число из возможных: 375999 и поделим его на 210. Тогда $375999/210 = 1790 +$ остаток. Возьмем целую часть и умножим на 210. $1790 \cdot 210 = 375900$. Это и есть ответ.

Ответ: 375900.

Задача I.2.1.4. (10 баллов)

Однажды Ване понадобилось посчитать сумму натуральных чисел, состоящих только из цифр 1, 2, 3, 4, 5 без повторов (число не обязательно должно состоять из всех этих цифр, например, число 5 тоже подходит). Какое число получилось у Вани, если он посчитал всё верно?

Решение

Всего таких чисел будет 5! = 125. При этом каждая цифра, в каждом разряде, будет стоять 25 раз (125/5). Тогда сумму можно представить как:

$$25 \cdot (10000 \cdot (1+2+3+4+5) + 1000 \cdot (1+2+3+4+5) + 100 \cdot (1+2+3+4+5) + \\ +10 \cdot (1+2+3+4+5) + (1+2+3+4+5)) = 4166625$$

Ответ: 4166625.

Задача I.2.1.5. (10 баллов)

Такси приедет к аэропорту не раньше чем в **15:25** и не позже чем в **15:50**. Для того, чтобы зарегистрироваться на самолёт и дойти до выхода на посадку, необходимо **20** минут. По расписанию выход закрывается в **15:50**, но если пассажир опаздывает, его ждут **10** минут, после чего закрывают выход и улетают. Найдите вероятность того, что человек, севший на это такси, успеет на самолёт. Ответ укажите в процентах.

Решение

Человек пройдёт регистрацию в 15:45 - 16:10 в зависимости от времени прибытия такси. С учётом времени ожидания самолёт вылетает в 16:00.

В промежутке от 15:45 до 16:00 – 15 минут, а в промежутке от 15:45 до 16:10 – 25 минут. Следовательно вероятностью того, что пассажир попадет на самолёт $15/25 = 0.6$.

Так как ответ нужно указать в процентах, ответ 60.

Ответ: 60.

Задача I.2.1.6. (10 баллов)

Предположим, что мы разматываем нить вдоль экватора Земли, а потом делаем то же самое, но в метре от экватора. Представим теперь, что мы разматываем один клубок вдоль поверхности Солнца и в метре от его поверхности, а второй – вдоль экватора Земли и в метре от его поверхности. К какому клубку нужно добавить больше ниток?

1. К тому, что мы разматываем в метре от Солнца
2. Одинаково
3. К тому, что мы разматываем в метре от Земли

Укажите ответ одним словом.

Решение

Длина окружности вычисляется по формуле $l = 2\pi R$, где R – радиус окружности. Если радиус увеличить на 1м, то длина дуги будет равна $L = 2\pi(R+1) = 2\pi R + 2\pi$,

следовательно, не зависимо от радиуса, длина дуги в обоих случаях увеличится на одну и ту же величину: 2π, то есть примерно на 6 м.

Ответ: Одинаково.

Задача I.2.1.7. (10 баллов)

Количество пользователей системы «Умный дом», выпущенной компанией «Технологии будущего», росло в течение всего года. На четыре разных квартала (в каком-то порядке) пришлись: наибольший абсолютный прирост, наименьший абсолютный прирост, наибольший относительный прирост и наименьший относительный прирост. (Абсолютный прирост – разность между новым и старым значением величины. Относительный прирост – это абсолютный прирост, делённый на старое значение.)

Известно, что наименьший относительный прирост был раньше, чем наибольший относительный. В каком квартале был наибольший абсолютный прирост? В ответ укажите номер квартала.

Решение

Способ 1. Докажем, что если количество пользователей растет и относительный прирост увеличивается, то увеличивается и абсолютный прирост. Пусть A и B – количество пользователей в какие-то моменты времени, причем $A < B$, а абсолютный прирост составляет x и y человек соответственно. Тогда относительный прирост равен соответственно $\frac{x}{A}$ и $\frac{y}{B}$. Если $\frac{x}{A} < \frac{y}{B}$, то $Bx < Ay < By$, следовательно, $x < y$.

Таким образом, наибольший относительный прирост не мог быть позже, чем наибольший абсолютный, и потому был не позже третьего квартала. Аналогично, наименьший относительный прирост не мог быть раньше, чем наименьший абсолютный прирост, и потому был не раньше второго квартала. Так как по условию задачи, наименьший относительный прирост был раньше, чем наибольший относительный, то они были во втором и третьем квартале соответственно. Следовательно, наибольший абсолютный прирост был позже, то есть в четвертом квартале.

Способ 2. Количество пользователей растет. Значит, если относительный прирост остается постоянным, то в следующем квартале он отсчитывается от большего значения, поэтому ему отвечает больший абсолютный прирост. Если же относительный прирост возрастает, то абсолютный прирост тем более возрастает.

Ответ: 4.

Задача I.2.1.8. (10 баллов)

Девять айтишников вошли в лифт на первом уровне одиннадцатиэтажного дворца техники. Если на одном из уровней вышли два человека, на другом – три, и еще на одном – четыре, то сколькими способами пассажиры могли выйти из лифта?

Решение

Распределить три группы на выход на трех уровнях из 10 можно $A_{10}^3 = \frac{10!}{(10-3)!}$ способами. Число перестановок из 9 человек с повторениями или число разбиений группы из девяти на три группы по 2,3 и 4 человека равно $N_9(2,3,4) = \frac{9!}{2!3!4!}$. Итого: $A_{10}^3 N_9(2,3,4) = \frac{10!}{4} = 907200$.

Ответ: 907200.

Задача I.2.1.9. (10 баллов)

В трех коробках лежат яблоки. Всего их 100. В первой коробке 50 яблок и все гнилые. Во второй коробке 30 яблок, десять из которых гнилые. А в третьей коробке 20 яблок, 5 из которых гнилые. Все яблоки из коробок переложили в пустой контейнер. Найдите вероятность того, что наугад выбранное яблоко из этого контейнера окажется не гнилым. Ответ укажите в виде десятичной дроби.

Решение

Длина окружности вычисляется по формуле $l = 2\pi R$, где R – радиус окружности. Если радиус увеличить на 1м, то длина дуги будет равна $L = 2\pi(R+1) = 2\pi R + 2\pi$, следовательно, не зависимо от радиуса, длина дуги в обоих случаях увеличится на одну и ту же величину: 2π, то есть примерно на 6 м.

Ответ: 0,35.

Задача I.2.1.10. (10 баллов)

Решите уравнение и запишите в качестве ответа разность наибольшего и наименьшего корня

$$x^2 + \frac{3x^2}{(x^2-2)} + 2 = 0.$$

Решение

Запомним, что модуль x не равен корню из двух. Домножим уравнение на $x^2 - 2$ и приведем подобные. Получим $x^4 + 3x^2 - 4 = 0$. Решим его как квадратное относительно x^2, получим $x^2 = 1$ и $x^2 = -4$. Так как корень числа не может быть отрицательным, остается только $x^2 = 1$. Возьмем корень справа и слева и получим, что $x = 1$ и $x = -1$. Вычтем $(1 - (-1) = 2)$.

Ответ: 2.

Первая попытка. Задачи 10-11 класса

Задача I.2.2.1. (10 баллов)

Предположим, что мы разматываем нить вдоль экватора Земли, а потом делаем то же самое, но в метре от экватора. Представим теперь, что мы разматываем один клубок вдоль поверхности Солнца и в метре от его поверхности, а второй – вдоль экватора Земли и в метре от его поверхности. К какому клубку нужно добавить больше ниток?

1. К тому, что мы разматываем в метре от Солнца
2. Одинаково
3. К тому, что мы разматываем в метре от Земли

Укажите ответ одним словом.

Решение

Длина окружности вычисляется по формуле $l = 2\pi R$, где R – радиус окружности. Если радиус увеличить на 1м, то длина дуги будет равна $L = 2\pi(R + 1) = 2\pi R + 2\pi$, следовательно, не зависимо от радиуса, длина дуги в обоих случаях увеличится на одну и ту же величину: 2π, то есть примерно на 6 м.

Ответ: Одинаково.

Задача I.2.2.2. (10 баллов)

Девять айтишников вошли в лифт на первом уровне одиннадцатиэтажного дворца техники. Если на одном из уровней вышли два человека, на другом – три, и еще на одном – четыре, то сколькими способами пассажиры могли выйти из лифта?

Решение

Распределить три группы на выход на трех уровнях из 10 можно $A_{10}^3 = \frac{10!}{(10-3)!}$ способами. Число перестановок из 9 человек с повторениями или число разбиений группы из девяти на три группы по 2,3 и 4 человека равно $N_9(2, 3, 4) = \frac{9!}{2!3!4!}$. Итого: $A_{10}^3 N_9(2, 3, 4) = \frac{10!}{4} = 907200$.

Ответ: 907200.

Задача I.2.2.3. (10 баллов)

Вычислите: $\cos 36^\circ - \sin 18^\circ$.

Решение

Разделим и умножим данное выражение на одно и то же, не равное нулю, число, затем воспользуемся формулами преобразования произведения тригонометрических функций в сумму:

$$\cos 36^\circ - \sin 18^\circ = \frac{\cos 36^\circ \cos 18^\circ - \sin 18^\circ \cos 18^\circ}{\cos 18^\circ} =$$
$$= \frac{0,5\cos 18^\circ + 0,5\cos 54^\circ - 0,5\sin 36^\circ}{\cos 18^\circ} = \frac{0,5\cos 18^\circ}{\cos 18^\circ} = 0,5$$

Ответ: 0,5.

Задача I.2.2.4. (10 баллов)

Компьютер последовательно решает несколько задач. Было замечено, что на решение каждой следующей задачи компьютер тратил в одно и то же число раз меньше времени, чем на решение предыдущей. Сколько было предложено задач, и сколько времени затрачено машиной на решение всех задач, если на решение всех задач, кроме первой, затрачено 63,5 мин.; на решение всех задач, кроме последней, затрачено 127 мин.; а на решение всех задач, кроме двух первых и двух последних, затрачено 30 мин?

Напишите ответ в формате: X задач(и), Y мин

Решение

Пусть b_i – время, затрачиваемое машиной на решение i-ой задачи – образуют геометрическую прогрессию со знаменателем $0 < q < 1$, то есть $b_i = b_1 q^{i-1}$. Тогда

$$b_2 + b_3 + b_4 + \cdots + b_n = 63,5,$$

$$b_1 + b_2 + b_3 + \cdots + b_{n-1} = 127,$$

$$b_3 + b_4 + \cdots + b_{n-2} = 30.$$

Анализируя цифры, видим, что q может быть только $\frac{1}{2}$. Рассмотрим третье равенство с разным количеством членов. Если в нем только один член $b_3 = 30$, тогда $b_1 q^2 = 30$, $b_1 = 120$, и мы имеем ряд $120+60+30+15+7,5$, не удовлетворяющий второму условию. Если в третьем уравнении два члена: $b_3+b_4 = 30$, то $b_1 q^2(1+q) = 30$, и подставляя $q = \frac{1}{2}$, находим $b_1 = 80$. В таком случае имеем ряд 80+40+20+10+5+2,5, который опять не удовлетворяет второму уравнению. Если в третьем уравнении три члена: $b_3 + b_4 + b_5 = 30$, то из уравнения $b_1 q^2(1 + q + q^2) = 30$ находим $b_1 = 30 \cdot 4 \cdot 4/7$ – не целое число, что не возможно. И наконец, если в третьем уравнении 4 члена: $b_3 + b_4 + b_5 + b_6 = 30$, то $b_1 = 64$, и получаем ряд $64 + 32 + 16 + 8 + 4 + 2 + 1 + 0,5$, который удовлетворяет всем условиям. Сумма этого ряда 127,5 мин.

Ответ: 8, 127,5.

Задача I.2.2.5. (10 баллов)

4 раза в сутки с интервалом 6 часов в порту с. Никольское (Камчатка) измеряют уровень воды. Со временем обнаружили, что колебания уровня воды приблизительно задаются уравнением

$$h = 1,4 - 0,8sin(0,3t)$$

Вопрос 1. Найдите высоту прилива через 15 часов с начала измерений.
Вопрос 2. В какое время ожидался самый высокий уровень воды в первые сутки с округлением до часов?

Решение

Вопрос 1. Для решения подставим значение $t = 15$ в формулу и проведем вычисления, считая значение синуса в радианах.

$$h = 1,4 - 0,8sin(0,3 \cdot 15)$$

$$h = 1,4 - 0,8 \cdot (-0,978)$$

$$h = 2,18m$$

Ответ: $h = 2,18m$.

Вопрос 2. Для поиска самого высокого уровня воды нужно вычислить экстремумы функции $h = 1,4 - 0,8sin(0,3t)$ для $t \in [0;24]$и определить точку максимума.

Известно, что в точках экстремума производная функции равна нулю, а при переходе через точку максимума знак производной меняется с положительного на отрицательный. Будем искать решение по шагам.

1. Найдем производную функции

$$h\prime = 1,4\prime - (0,8sin(0,3t))\prime$$

$$h\prime = -0,8(sin(0,3t))\prime$$

$$h\prime = -0,8cos(0,3t) \cdot (0,3t)\prime$$

$$h\prime = -0,8cos(0,3t) \cdot 0,3$$

$$h\prime = -2,4cos(0,3t)$$

2. Приравняем производную к нулю и решим полученное тригонометрическое уравнение

$$-2,4cos(0,3t) = 0$$

$$cos(0,3t) = 0$$

$$0,3t = \pm arccos(0) + 2\pi n$$

$$0,3t = \frac{\pi}{2} + \pi n$$

$$t = 1,67\pi + 3,33\pi n$$

Вычислим, при каких значениях n $t \in [0;24]$, получим два значения: $t_1 = 5,34$ и $t_2 = 15,79$

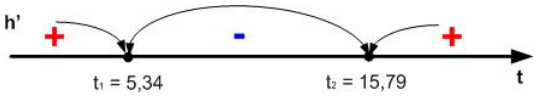


Рис. I.2.1: Знаки производной на промежутках

Отметим значения t_i на числовой прямой и проверим знаки производной на полученных промежутках (см. рис. I.2.1)

Ответ: 2,18, 16:00.

Задача I.2.2.6. (10 баллов)

К конгрессу математиков было принято решение оформить клумбу цветами разных видов так, как показано на рисунке I.2.2. Для заполнения треугольника ABC было использовано 72 цветка. Сколько цветков при той же плотности посадки понадобится для заполнения четырехугольника $BCDE$, если стороны AB и AC равны, соответственно, 9 и 8 метров, а диаметр круга равен 10 метрам. Отрезок AD проходит в точности через центр круга.

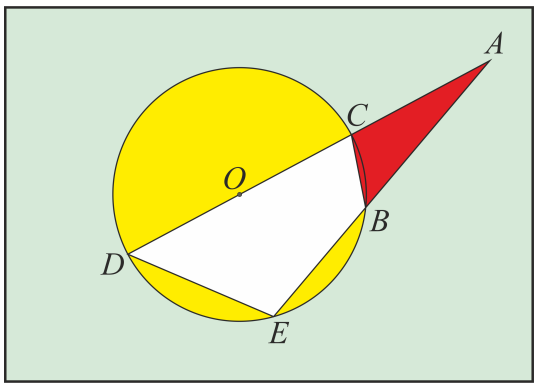


Рис. I.2.2: Чертёж клумбы

Решение

Произведения отрезков секущих, проведенных из общей точки, равны: $AC \cdot AD = AB \cdot AE$, а угол $\angle A$ является общим углом треугольников $\triangle ABC$ и $\triangle ADE$. Соответственно, данные треугольники подобны. Коэффициент подобия равен отношению сторон AD и AB: $k = \frac{AD}{AB} = \frac{(8+10)}{9} = 2$, т.к. отрезок DC совпадает с диаметром круга. Значит, площади треугольников $\triangle ADE$ и $\triangle ABC$ соотносятся как квадрат коэффициента подобия: $S_{\triangle ADE} = k^2 \cdot S_{\triangle ABC} = 4S_{\triangle ABC}$.

Далее имеем: $S_{BCDE} = S_{\triangle ADE} - S_{\triangle ABC} = 3S_{\triangle ABC}$

Так как плотность посадки цветов одинаковая, то для заполнения четырехугольника $BCDE$ необходимо $N = 3 \cdot 72 = 216$ цветков.

Ответ: 216 цветков.

Задача I.2.2.7. (10 баллов)

Дизайнер в известной компании придумал новый продукт для поддержания имиджа – термос-кружку с логотипом компании. Внутренняя поверхность термоса имеет форму прямого усеченного конуса с диаметрами нижнего и верхнего оснований 10 см и 8 см соответственно. В нижней части поверхность завершается шаровым сегментом высотой 2 см. Определите высоту термоса от макушки шарового сегмента до верхнего основания конуса, чтобы в него помещалась жидкость объемом 0,7 литра.

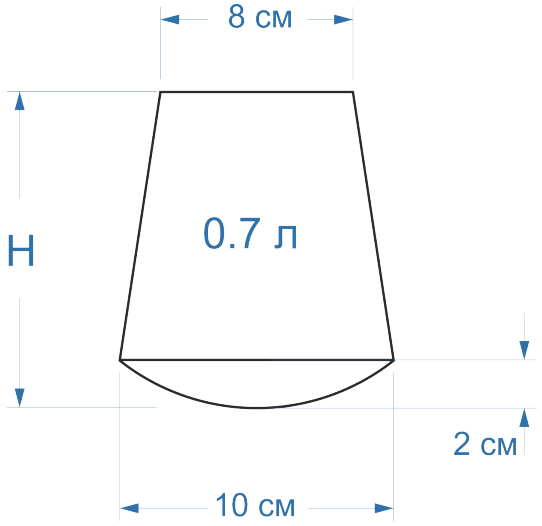


Рис. I.2.3: Схема внутренней поверхности термоса

Решение

Представим схему внутренней поверхности термоса, как показано на рисунке. H на рисунке I.2.3 - высота, которую нужно найти.

Для рассчета объема шарового сегмента все исходные данные в задаче имеются $V_1 = \frac{1}{6}\pi h_1(h_1^2 + 3r_{bot}^2) = \frac{1}{6}\pi 2(2^2 + 3 \cdot 5^2) = \frac{79}{3}\pi$. Объем усеченного конуса определяется выражением $V_2 = \frac{1}{3}h_2\pi(r_{bot}^2 + r_{bot} \cdot r_{top} + r_{top}^2) = \frac{1}{3}h_2\pi(5^2 + 5 \cdot 4 + 4^2) = \frac{61}{3}h_2\pi$. Общий объем термоса складывается из объема шарового сегмента и усеченного конуса $V = V_1 + V_2$. Подставим в эту формулу выражения для объемов шарового сегмента и конуса, а затем выразим высоту конуса через известный из условия задачи объем термоса, получим $h_2 = \frac{3(700 - \frac{79\pi}{3})}{61\pi} \approx 9.66$. Следовательно искомая высота термоса будет равна $H \approx 2 + 9.66 = 11.66$ см.

Ответ: 11.66 см.

Задача I.2.2.8. (10 баллов)

Известная строительная компания выпускает универсальные комплектующие для сооружения легких каркасных конструкций типа «пирамида». Отдельный элемент конструкции имеет форму равнобедренного треугольника с углом между боковыми сторонами 15 градусов. При монтаже конструкции треугольники соединяются попарно своими боковыми сторонами. Высота конструкции, составленной из 8 таких

элементов, равна 4 метра. Какова будет высота конструкции, если количество элементов уменьшить до 3?

Решение

Согласно условию задачи, конструкция представляет собой правильную пирамиду в основании, которой лежит восьмиугольник.

Рассмотрим прямоугольный треугольник, образованный высотой пирамиды и высотой ее боковой стороны. Высоты пирамиды H и боковой грани h являются соответственно катетом и гипотенузой данного треугольника. Второй его катет можно выразить через длину основания боковой грани пирамиды a, пользуясь тем, что основанием пирамиды является правильный восьмиугольник: $OO^{'} = a(\frac{1}{2} + \sqrt{2})$. По теореме Пифагора $OO^{'2} + H^2 = h^2$. Комбинируя два эти выражения, получим соотношение между высотами и длиной основания $a^2(\frac{1}{2} + \sqrt{2})^{'2} + H^2 = h^2$

Боковая грань пирамиды является равнобедренным треугольником. Его высоту можно выразить через длину основания и угол при боковых сторонах $h = \frac{a}{2\tan\frac{\phi}{2}}$. Подставляя данное выражение в предыдущее уравнение получим для длины основания $a = \frac{H}{\sqrt{\frac{1}{4\tan^2\frac{\phi}{2}} - (\frac{1}{2}+\sqrt{2})^2}}$, вычисляя которое, имеем $a \approx 1.22$ метра. Соответственно найдем $h \approx 4.63$ метра.

Теперь рассмотрим пирамиду, в основании которой лежит правильный треугольник. В прямоугольном треугольнике образованном высотами пирамиды H_1 и боковой грани h, катет в плоскости основания пирамиды равен $OO^{'} = \frac{a}{2\sqrt{3}}$. Соотношение между высотами и длиной основания боковой грани запишем в виде $\frac{a^2}{12} + H_1^2 = h^2$, из которого найдем $H_1 \approx 4.62$ метра.

Ответ: 4.21 метра.

Задача I.2.2.9. (10 баллов)

Количество пользователей системы «Умный дом», выпущенной компанией «Технологии будущего», росло в течение всего года. На четыре разных квартала (в каком-то порядке) пришлись: наибольший абсолютный прирост, наименьший абсолютный прирост, наибольший относительный прирост и наименьший относительный прирост. (Абсолютный прирост – разность между новым и старым значением величины. Относительный прирост – это абсолютный прирост, делённый на старое значение.)

Известно, что наименьший относительный прирост был раньше, чем наибольший относительный. В каком квартале был наибольший абсолютный прирост? В ответ укажите номер квартала.

Решение

Способ 1. Докажем, что если количество пользователей растет и относительный прирост увеличивается, то увеличивается и абсолютный прирост. Пусть A и B – количество пользователей в какие-то моменты времени, причем $A < B$, а абсолютный прирост составляет x и y человек соответственно. Тогда относительный прирост равен соответственно $\frac{x}{A}$ и $\frac{y}{B}$. Если $\frac{x}{A} < \frac{y}{B}$, то $Bx < Ay < By$, следовательно, $x < y$.

Таким образом, наибольший относительный прирост не мог быть позже, чем наибольший абсолютный, и потому был не позже третьего квартала. Аналогично, наименьший относительный прирост не мог быть раньше, чем наименьший абсолютный прирост, и потому был не раньше второго квартала. Так как по условию задачи, наименьший относительный прирост был раньше, чем наибольший относительный, то они были во втором и третьем квартале соответственно. Следовательно, наибольший абсолютный прирост был позже, то есть в четвертом квартале.

Способ 2. Количество пользователей растет. Значит, если относительный прирост остается постоянным, то в следующем квартале он отсчитывается от большего значения, поэтому ему отвечает больший абсолютный прирост. Если же относительный прирост возрастает, то абсолютный прирост тем более возрастает.

Ответ: 4.

Задача I.2.2.10. (10 баллов)

Имеется информационная сеть, состоящая из центров хранения информации. Некоторые пары центров соединены каналами связи. Обмен информацией между любыми двумя центрами выполняется либо непосредственно, либо через другие каналы и центры. Если каждая пара центров может обмениваться информацией, сеть является исправной.

Известно, что в сети всего $n = 12$ центров, каждый из центров непосредственно связан каналом с $k = 7$ другими центрами.

Какое наименьшее количество центров надо разрушить, чтобы сеть стала неисправной?

Решение

Информационной сети следует сопоставить граф G: верршины графа - это центры сети, ребра - каналы связи. Если сеть исправна, то ей соответствует связный граф. Разрушить сеть можно или удалением вершин (разрушением центров) или удалением ребер (повреждением каналов). В результате любого из выбранных способов должен получиться несвязный граф. В данной сети минимальная степень вершины $\delta(G) = 7$. Известно, что для графа, не являющегося полным и имеющего n вершин выполняется неравенство:

$$k(G) \geqslant 2\delta(G) - n + 2,$$

где $k(G)$ - связность графа G.

Подставим в формулу данные о нашей сети

$$k(G) \geqslant 2 \cdot 7 - 12 + 2$$

$$k(G) \geqslant 4$$

Из последнего неравенства следует, что если удалить из сети минимально возможное количество центров, равное 4, граф распадется на несвязные области.

Ответ: 4.

Вторая попытка. Задачи 8-9 класса

Задача I.2.3.1. (10 баллов)

Вася загадал какое-то трехзначное число. После этого Вася начал это число делить на 7 с остатком следующим образом: он поделил число на 7, записал его остаток и начал делить неполное частное от предыдущего деления.

Пример того, как Вася делил число 548:

$548/7 = 78$ остаток 2

$78/7 = 11$ остаток 1

$11/7 = 1$ остаток 4

В ответ напишите, какое число загадал Вася, если сумма его остатков наибольшая, а само число наименьшее из возможных для данной суммы.

Решение

Так как требуется минимальное трехзначное число, то найдем количество цифр 7, которые при перемножении получаем трехзначное число.

$$7^3 = 343$$

$$7^4 = 2401$$

Из этого следует, что необходимо сделать три перемножения для нахождения числа.

Максимальная сумма остатков является сумма максимальных остатков. Максимальный остаток - цифра 6, т.е. $6+6+6 = 18$. Для того, чтобы найти искомое число, необходимо к минимальному числу добавить остаток и умножить его на 7. Покажем пошагово:

$$7 + 6 = 13$$

$$13 \cdot 7 = 91$$

$$91 + 6 = 97$$

$$97 \cdot 7 = 679$$

$$679 + 6 = 685$$

Ответ: 685.

Задача I.2.3.2. (10 баллов)

Пете, чтобы добраться домой из деревни, вначале нужно сесть на автобус номер 35, затем на электричку, следующую по маршруту Васюки-Москва, а после этого сесть на автобус номер 465.

Автобус номер 35 едет до железнодорожной станции Васюки 45 минут и, в зависимости от пробок и настроек светофоров, может ехать на 9 минут дольше или на 5 минут быстрее. На поездку на электричке Петя тратит 70 ± 25 минут.

Петя сел на автобус номер 35 в **16:25**. Электрички по расписанию отправляются со станции Васюки с **7:15** каждые полчаса. А автобус 465 от железнодорожного вокзала в Москве с **8:30** каждые 45 минут.

Напишите время отправления автобуса 465 от вокзала в Москве, на который Петя сядет с наибольшей вероятностью. Пешие переходы между остановками автобусов и платформами электрички считать равными 0.

Ответ укажите в виде четырехзначного числа вида ччмм, где первые две цифры отвечают за часы, а оставшиеся — за минуты. Для времени **9:15** нужно написать 0915.

Решение

Рассмотрим варианты прибытия Пети до электрички на автобусе номер 35, время отбытия **16:25**. Автобус едет 45 минут, может ехать на 9 минут дольше или на 5 минут быстрее. Таком образом получаем три варианта:

А. 16:25 + 45 минут + 9 минут = 17:19

Б. 16:25 + 45 минут = 17:10

В. 16:25 + 40 минут = 17:05

Рассмотрим расписание электричек до Москвы: 7:15, 7:45, 8:15, 8:45, . . . , 17:15, 17:45 . . . Следовательно, при варианте А - Петя успевает на электричку в 17:45, при вариантах Б и В - Петя успевает на электричку в 17:15. Электричка едет 70 ± 25 минут, получаем:

А.1. 17:45 + (70 + 25) минут = 19:20

А.2. 17:45 + 70 минут = 18:55

А.1. 17:45 + (70 - 25) минут = 18:30

Б.1. 17:15 + (70 + 25) минут = 18:50

Б.2. 17:15 + 70 минут = 18:25

Б.3. 17:15 + (70 - 25) минут = 18:00

В.1. 17:15 + (70 + 25) минут = 18:50

В.2. 17:15 + 70 минут = 18:25

В.3. 17:15 + (70 - 25) минут = 18:00

Рассмотрим расписание автобуса 465: 8:30, 9:15, 10:00, 10:45, . . . , 17:30, 18:15, 19:00, 19:45 . . . Тогда:

А.1. 19:20 → 19:45

А.2. 18:55 → 19:00

А.1. 18:30 → 19:00

Б.1. 18:50 → 19:00

Б.2. 18:25 → 19:00

Б.3. 18:00 → 18:15

В.1. 18:50 → 19:00

В.2. 18:25 → 19:00

В.3. 18:00 → 18:15

С бóльшей вероятностью Петя успеет на атобус 465, который отправляется в

19:00.

Ответ: 1900.

Задача I.2.3.3. (10 баллов)

Решите уравнение: $\cos\sqrt{x} + \sqrt{\cos x} = 2$.

Решение

Так как $|\cos\sqrt{x}| \leqslant 1$ и $|\sqrt{\cos x}| \leqslant 1$, то уравнение $\cos\sqrt{x}+\sqrt{\cos x} = 2$ равносильно системе уравнений:

$$\begin{cases} \cos\sqrt{x} = 1, \\ \sqrt{\cos x} = 1. \end{cases}$$

Из певрого уравнения системы получим, что $x = 2\pi k$, где $k \in Z$, а из второго уравнения получим, что $\sqrt{x} = 2\pi n \Leftrightarrow x = 4\pi^2 n^2$, где $n \in Z$, $n \geqslant 0$. Найдем пересечение полученных множеств: $2\pi k = 4\pi^2 n^2 \Leftrightarrow k = 2\pi n^2$. Так как π - иррациональное число, то последнее равенство выполяется тогда, и только тогда, когда $n = k = 0$. Следовательно, единственным решением данного уравнения является $x = 0$.

Ответ: $x = 0$.

Задача I.2.3.4. (10 баллов)

На столе разложены 2020 спичечных коробков, в некоторых из них есть спички, а в некоторых — нет.

На первом коробке написано: «Все коробки пустые».

На втором — «По крайней мере 2019 коробков пустые».

На третьем — «По крайней мере 2018 коробков пустые».

На 2020-ом — «По крайней мере один коробок пустой».

Известно, что надписи на пустых коробках ложные, а на коробках со спичками — истинные. Определите, сколько коробков не пустые.

Решение

Рассмотрим коробок №1.

1. Если коробок пуст, то надпись на нем ложна.
2. Если коробок полон, то надпись на нем верна. Следовательно, все 2020 должны быть пусты, получаем противоречие.

Коробок №1 полон. Отсюда, надпись на коробке №2020 верна и он полон.

Рассмотрим коробок №2.

Мы уже знаем, что один коробок полон и один коробок пуст.

1. Если коробок пуст, то надпись на нем ложна.
2. Если коробок полон, то надпись на нем верна. Следовательно, 2019 коробков должны быть пусты, получаем противоречие, т.к. коробки №2 и №2020 полные.

Отсюда коробоки №1 и №2 пусты, коробки №2019 (надпись на коробке верна) и №2020 полны.

Рассмотрим коробок №3.

Мы уже знаем, что два коробка полоны и два коробка пусты.

1. Если коробок пуст, то надпись на нем ложна.
2. Если коробок полон, то надпись на нем верна. Следовательно, 2018 коробков должны быть пусты, получаем противоречие, т.к. коробки №3, №2019 и №2020 полные.

Отсюда коробоки №1, №2 и №3 пусты, коробки №2018 (надпись на коробке верна), №2019 и №2020 полны.

Продолжая рассуждения, понимаем, что ровно половина коробков пустые, половина коробков полные.

Ответ: 1010.

Задача I.2.3.5. (10 баллов)

Саша любит придумывать и записывать "волшебные" числа. Для него это пятизначные числа, на чётных разрядах которых находятся чётные цифры, а на нечётных — нечётные. Числа: 12301 и 36789 — "волшебные", а 78645 — нет.

Юноша занимается этим нелегким делом очень давно, и за это время он успел выписать все такие числа. Однажды он посмотрел на эти числа и решил сосчитать, сколько цифр "5" ему пришлось написать, для того, чтобы записать все "волшебные" числа по одному разу.

Сколько цифр "5" насчитал Саша, если известно, что он не ошибся?

Решение

"Волшебное" число выглядит так:

$$\textbf{н} \parallel \textbf{ч} \parallel \textbf{н} \parallel \textbf{ч} \parallel \textbf{н}$$

Где **н** - нечетная цифра, **ч** - четная цифра

Всего четных цифр - 5: 0, 2, 4, 6, 8.

Всего нечетный цифр - 5: 1, 3, 5, 7, 9.

На каждой позиции может стоять по 5 чисел, следовательно всего таких чисел:

$$5 \cdot 5 \cdot 5 \cdot 5 \cdot 5 = 3125$$

Есть несколько вариантов чисел, котрые включают цифру 5:

1.	5	**ч**	**н** - 5	**ч**	**н** - 5
2.	**н** - 5	**ч**	5	**ч**	**н** - 5
3.	**н** - 5	**ч**	**н** - 5	**ч**	5
4.	5	**ч**	5	**ч**	**н** - 5
5.	5	**ч**	**н** - 5	**ч**	5
6.	**н** - 5	**ч**	5	**ч**	5
7.	5	**ч**	5	**ч**	5

Где (**н** - 5) – нечетные числа, не включающие число 5: 1, 3, 7, 9.

Таким образом получаем, что чисел с тремя цифрами 5:

$$1 \cdot 5 \cdot 1 \cdot 5 \cdot 1 = 1200,$$

чисел с двумя цифрами 5:

$$(1 \cdot 5 \cdot 1 \cdot 5 \cdot 4) \cdot 3 = 300,$$

чисел с одной цифрой 5:

$$(1 \cdot 5 \cdot 4 \cdot 5 \cdot 4) \cdot 3 = 1200.$$

В задании спрашивается, сколько записей цифры 5 было, следовательно, необходимо полученные результаты умножить на количество 5 в записи чисел:

$$1200 \cdot 1 + 300 \cdot 2 + 25 \cdot 3 = 1875$$

Ответ: 1875.

Задача I.2.3.6. (10 баллов)

Двое друзей, каждый со своей позиции, ведут наблюдение через вертикальную щель в круглую комнату.

Определить величину щели, если вместе они контролируют только четвертую часть стены комнаты, и при этом угол зрения одного и второго равны соответственно °10 и °20 градусов. При решении считать, что каждый из них видит свой участок стены.

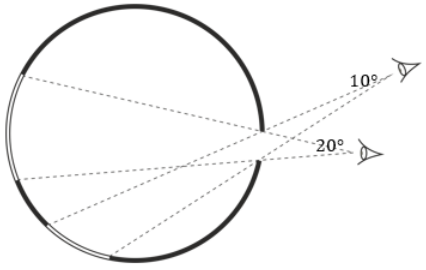


Рис. I.2.4: Схема задачи

Решение

Дополним схему задачи следующими обозначениями, как показано на рисунке. Общую дугу окружности - щель обозначим символом m, а дуги, показывающие сектора обзора, обозначим через n_1 и n_2. Углы обозначающие сектора обзора назовём $\angle A$ и $\angle B$ соответственно.

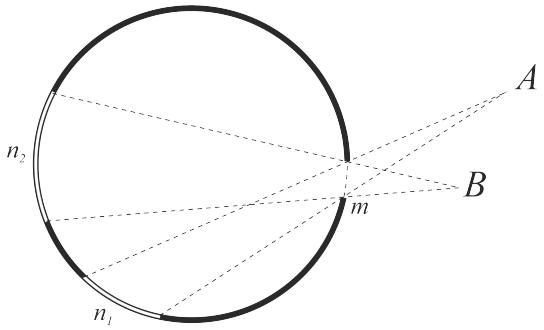


Рис. I.2.5: Схема задачи

Известно, что угол $\angle A$ между секущими, пересекающимися вне окружности, вычисляется по формуле: $\angle A = \frac{n_1 - m}{2}$. Аналогично, $\angle B = \frac{n_2 - m}{2}$. Также по условию задачи $n_1 + n_2 = 90°$. Тогда, складывая левые и правые части двух первых уравнений, получим выражение $\angle A + \angle B = \frac{n_1 + n_2}{2} - m$. Откуда можем найти искомую дугу $m = \frac{n_1 + n_2}{2} - \angle A - \angle B = \frac{90°}{2} - 10° - 20° = 15°$.

Ответ: °15.

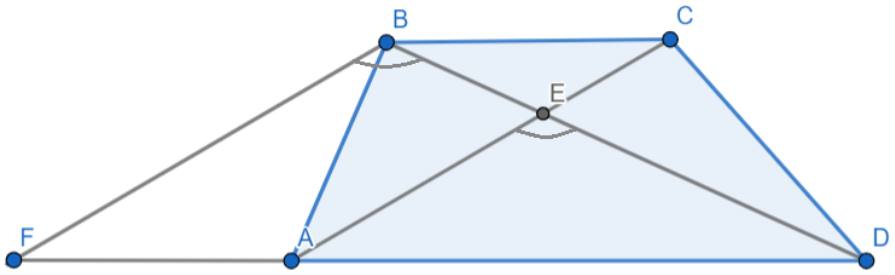
Задача I.2.3.7. (10 баллов)

В трапеции $ABCD$ с основаниями AD и BC диагонали пересекаются в точке E. $\angle AED = 60°$. $AC = BC + AD$. Найдите отношение $AB : CD$.

Введите ответ с точностью **не менее** 3 знаков после запятой (отличие от точного значения не должно превышать 0.0005).

Решение

Построим параллелограмм $AFCD$.



Тогда $\angle FBD = \angle AOD$, $AF = BC$ и

$$FB = AC = BC + AD = AF + AD = FD.$$

Значит, треугольник $\triangle FBD$ – равнобедренный ($FB = FD$). Поскольку углы при основании равнобедренного треугольника – острые, то угол $\angle FBD$ не может быть равен 120°. Следовательно, он равен 60°, и треугольник $\triangle FBD$ – равносторонний. Поэтому $BD = FB = AC$, то есть диагонали трапеции равны между собой. Отюда следует, что трапеция равнобедренная.

Таким образом $\frac{AB}{CD} = 1$.

Ответ: 1.

Задача I.2.3.8. (10 баллов)

Решите уравнение $-3x^5 + 375x^3 - 7500x = 0$.

В ответ запишите сумму наибольшего и наименьшего из неотрицательных корней.

Решение

Разделим уравнение на -3.

$$x^5 - 125x^3 + 2500x = 0$$

$$x(x^4 - 125x^2 + 2500) = 0$$

Разобьем слагаемое $-125x^2$ на два $-100x^2$ и $-25x^2$:

$$x(x^4 - 100x^2 - 25x^2 - 2500) = 0$$

$$x(x^2(x^2 - 100) - 25(x^2 - 100)) = 0$$

$$x(x^2 - 100)(x^2 - 25) = 0$$

Применим формулу разности квадратов:

$$x(x - 10)(x + 10)(x - 5)(x + 5) = 0$$

Корни уравнения: $x_1 = 0$, $x_2 = 5$, $x_3 = -5$, $x_4 = 10$, $x_5 = -10$.

Следовательно, минимальный корень - x_1 и максимальный x_4: $x_4 - x_1 = 10$.

Ответ: 10.

Задача I.2.3.9. (10 баллов)

Составляя расписание мастер-классов для финала олимпиалы НТИ по профилю "Разработка приложений виртуальной и дополненной реальности"организаторы столкнулись с затруднением. Мастер-классы должны проводится 1 раз в день с понедельника по четверг, но приглашенные специалисты имеют ряд требований к дню проведения мастер-класса. Предложите варианты расписания мастер-классов, если:

а) специалист по математическому моделированию может провести провести мастер-класс в понедельник, вторник или среду;

б) специалист по 3D моделированию может пообщаться с финалистами или в понедельник, или во вторник;

в) мастер-класс по физике может состояться во вторник или четверг;

г) мастер-класс по программированию интерактивного окружения - только во вторник или среду.

Сколько возможно вариантов составления расписания мастер-классов для проведения финала? Укажите их, перечисляя через точку с запятой мастер-классы в порядке их следования в течение финала. Например: Вариант 1. Математическое моделирование; 3D моделирование; программирование интерактивного окружения; физика.

Решение

Построим граф, вершины которого - возможные дни недели (1, 2, 3, 4) и мастер-классы (м, ф, п, 3d). Ребрами соединим мастер-классы с возможными днями проведения(рис. I.2.10).

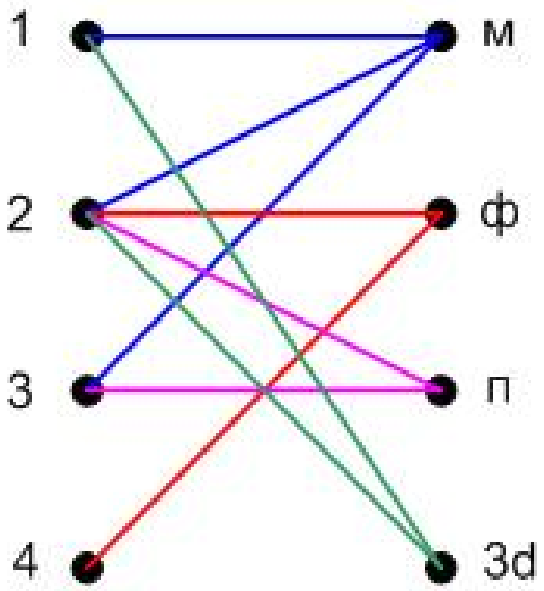


Рис. I.2.6: Граф вариантов расписания

Задача сводится к определению ребер, задающих взаимно однозначное соответствие между множеством дней и множеством мастер-классов.

Из рисунка можно увидеть, что всего количество таких соответствий $k = 3$

Приведем варианты расписания. Из графа видно, что в четверг может быть только физика. Все могут провести мастер-класс во вторник, а в понедельник и в среду согласны поработать вдое специалистов. Если математическое моделирование поставить в понедельник, то во вторник можно провести только 3D моделирование, в противном случае в среду мастер-класс поставить невозможно.

Ответ: 1234, 2134, 2314.

Задача I.2.3.10. (10 баллов)

Миша любит собирать карточки с супергероями.

Всего есть три типа карточек: обычные, редкие и легендарные.

Обычных карточек — 125 видов. Каждого вида было напечатано по 1000 штук.

Редких карточек — 25 видов. Каждого вида было напечатано по 100 штук.

Легендарных карточек — 5 видов. Каждого вида было напечатано по 10 штук.

Карточки запакованы таким образом, что невозможно узнать, какого они вида, пока не откроешь. Мише нравятся 20 видов обычных карточек, 5 видов редких карточек и все виды легендарных. Найдите вероятность того, что, если Михаил купит одну карточку, она окажется именно того вида, который ему нравится?

Решение

Всего карточек $125 \cdot 1000 + 25 \cdot 100 + 5 \cdot 10$.

Всего карточек, которые нравятся Мише: $20 \cdot 1000 + 5 \cdot 100 + 5 \cdot 10$.

Следовательно, вероятность, того, что Миша купит карточку, которая ему нравятся:

$$\frac{20 \cdot 1000 + 5 \cdot 100 + 5 \cdot 10}{125 \cdot 1000 + 25 \cdot 100 + 5 \cdot 10} = \frac{20550}{127550} = 0,1611132889063$$

Ответ: 0,1611132889063.

Вторая попытка. Задачи 10-11 класса

Задача I.2.4.1. (10 баллов)

IT-специалист в первую неделю отпуска израсходовал менее, чем 3/5 количества взятых с собой денег, во вторую неделю – 1/4 остатка и еще 3000 руб., в третью неделю – 2/5 нового остатка и еще 1200 руб. После чего осталось 6/35 от количества взятых денег. Известно также, что количество денег, оставшихся неизрасходованными к концу первой, второй и третьей недели убывало в арифметической прогрессии.

Сколько денег было израсходовано в три недели отпуска?

Ответ укажите в рублях.

Решение

Обозначим за x общее количество взятых с собой денег, за y – остаток первой недели, а за d – разность арифметической прогрессии. Заполним таблицу:

неделя	израсходовано	остаток	арифметическая прогрессия
1 нед	$< \frac{3}{5}x$	$y > \frac{2}{5}x$	$\frac{6}{35}x + 2d$
2 нед	$\frac{1}{4}y + 30$	$\frac{3}{4}y - 30$	$\frac{6}{35}x + d$
3 нед	$\frac{2}{5}(\frac{3}{4}y - 30) + 12$	$\frac{3}{4}y - 30 - \frac{3}{10}y$	$\frac{6}{35}x$

Приравняем второй и третий столбцы таблицы, и получим систему уравнений:

$$\begin{cases} y = \frac{6}{35}x + 2d, \\ \frac{3}{4}y - 30 = \frac{6}{35}x + d, \\ \frac{3}{4}y - 30 - \frac{3}{10}y = \frac{6}{35}x. \end{cases}$$

Решив систему, получаем $d = 180, y = 600, x = 1400$. Таким образом, за отпуск потрачено $x - \frac{6}{35}x = 1160$ тыс. рублей.

Ответ: 116000 руб.

Задача I.2.4.2. (10 баллов)

В некотором городе стоит дворец техники. На этажах со 2 по 8 включительно располагаются экспонаты выставок. Посетители заходят во дворец на 1 этаже, сдают одежду в гардероб и проходят в холл к лифтам.

В один из моментов на 1 этаже в лифт зашло 7 посетителей. Они поехали наверх к экспонатам. Какова вероятность, что хотя бы на одном уровне вышли по крайней мере два человека?

Решение

Рассмотрим обратную ситуацию: на каждом уровне вышло по одному человеку. Вероятность этого события $P(\bar{A}) = \frac{7!}{7^7} = 0,00612$. Следовательно, вероятность обратного события $P(A) = 1 - 0,00612 = 0,99388$.

Ответ: 0,994. (0,99388).

Задача I.2.4.3. (10 баллов)

Решите уравнение: $\cos\sqrt{x} + \sqrt{\cos x} = 2$.

Решение

Так как $|\cos\sqrt{x}| \leqslant 1$ и $|\sqrt{\cos x}| \leqslant 1$, то уравнение $\cos\sqrt{x} + \sqrt{\cos x} = 2$ равносильно системе уравнений:

$$\begin{cases} \cos\sqrt{x} = 1, \\ \sqrt{\cos x} = 1. \end{cases}$$

Из певрого уравнения системы получим, что $x = 2\pi k$, где $k \in Z$, а из второго уравнения получим, что $\sqrt{x} = 2\pi n \Leftrightarrow x = 4\pi^2 n^2$, где $n \in Z$, $n \geqslant 0$. Найдем пересечение полученных множеств: $2\pi k = 4\pi^2 n^2 \Leftrightarrow k = 2\pi n^2$. Так как π - иррациональное число, то последнее равенство выполняется тогда, и только тогда, когда $n = k = 0$. Следовательно, единственным решением данного уравнения является $x = 0$.

Ответ: $x = 0$.

Задача I.2.4.4. (10 баллов)

Две рамки квадратная и в форме равностороннего треугольника заполнены одинаковым количеством одинаковых шаров, касающихся друг друга и сторон рамок (рамки заполнены, как в бильярде).

Сколько шаров для этого потребуется, если к стороне треугольника примыкает на 14 шаров больше, чем к стороне квадрата?

Решение

Пусть a – количество шаров вдоль стороны квадрата, тогда в сосуде с квадратным дном a^2 шаров. И пусть n – количество шаров вдоль стороны треугольника, тогда в сосуде с треугольным дном будет $1+2+...+n$ шаров. По условию количество шаров в обоих сосудах одинаково, то есть $a^2 = 1+2+...+n$. Сумма арифметической прогрессии в правой части равна $\frac{(1+n)n}{2}$. В левой части воспользуемся условием, что $a = n-14$. Получим уравнение: $(n-14)^2 = \frac{(1+n)n}{2}$. Раскрыв скобки и приведя подобные, получим квадратное уравнение $n^2 - 57n + 392 = 0$. Оно имеет два корня: $n = 8$ и $n = 49$, первый из которых не подходит по условию $a = n - 14 > 0$. Для второго корня $a = n - 14 = 35$, следовательно всего шаров в каждом из сосудов $35^2 = 1225$.

Ответ: 2450.

Задача I.2.4.5. (10 баллов)

Какую геометрическую фигуру представляют собой множество точек комплексной плоскости, удовлетворяющих условию: $|z| = 1$?

Ответом служит одно слово в именительном падеже.

Решение

Модуль комплексного числа $z = x + yi$ равен $|z| = \sqrt{x^2 + y^2}$. По условию он равен 1: $\sqrt{x^2 + y^2} = 1$, следовательно, $x^2 + y^2 = 1$. Это окружность в комплексной плоскости радиуса 1 с центром в начале координат.

Ответ: Окружность.

Задача I.2.4.6. (10 баллов)

Двое друзей, каждый со своей позиции, ведут наблюдение через вертикальную щель в круглую комнату.

Определить величину щели, если вместе они контролируют только четвертую часть стены комнаты, и при этом угол зрения одного и второго равны соответственно °10 и °20 градусов. При решении считать, что каждый из них видит свой участок стены.

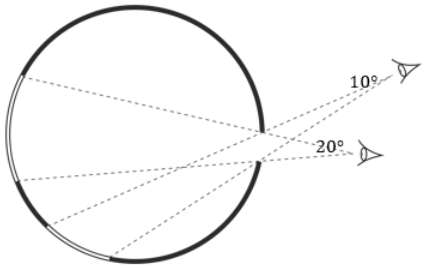


Рис. I.2.7: Схема задачи

Решение

Дополним схему задачи следующими обозначениями, как показано на рисунке. Общую дугу окружности - щель обозначим символом m, а дуги, показывающие сектора обзора, обозначим через n_1 и n_2. Углы обозначающие сектора обзора назовём $\angle A$ и $\angle B$ соответственно.

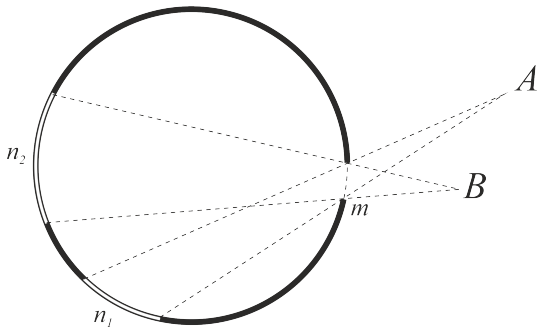


Рис. I.2.8: Схема задачи

Известно, что угол $\angle A$ между секущими, пересекающимися вне окружности, вычисляется по формуле: $\angle A = \frac{n_1 - m}{2}$. Аналогично, $\angle B = \frac{n_2 - m}{2}$. Также по условию задачи $n_1 + n_2 = 90°$. Тогда, складывая левые и правые части двух первых уравнений, получим выражение $\angle A + \angle B = \frac{n_1 + n_2}{2} - m$. Откуда можем найти искомую дугу $m = \frac{n_1 + n_2}{2} - \angle A - \angle B = \frac{90°}{2} - 10° - 20° = 15°$.

Ответ: °15.

Задача I.2.4.7. (10 баллов)

Каркасный бассейн в форме цилиндра высотой 1.4 метра и внутренним диаметром 3 метра стоит в помещении площадью 24 квадратных метра. Рассчитайте высоту порога при входе в помещение (в см), который необходимо соорудить строителям, чтобы в случае порыва стенок бассейна вода не перетекла через него в смежные помещения.

При расчетах учитывайте, что бассейн наполняется водой до уровня 20 см от верхней кромки.

Решение

Первым шагом рассчитаем объем воды, который находится в бассейне.

$V = \pi R^2(h - \triangle h) = \pi(\frac{3}{2})^2(1.4 - 0.2) \approx 8.48$ куб. метров

Чтобы найти уровень воды L, который получится в помещении в случае порыва бассейна, необходимо разделить полученный объем на площадь помещения $L = V/S \approx 8.48/24 \approx 0.35$ метра, т.е. строителям необходимо оборудовать порог не менее 35 см высотой.

Ответ: 35,3429173529.

Задача I.2.4.8. (10 баллов)

Студия дизайна решила сделать оригинальный сувенир – вазу в виде шара с отсеченным сверху шаровым сегментом. Внутри полость вазы имеет форму перевернутого вверх основанием прямого конуса объемом 1 литр.

Найдите диаметр горлышка вазы, если по задумке дизайнеров объем полости вазы должен был быть максимально возможным.

Введите ответ в дециметрах.

Решение

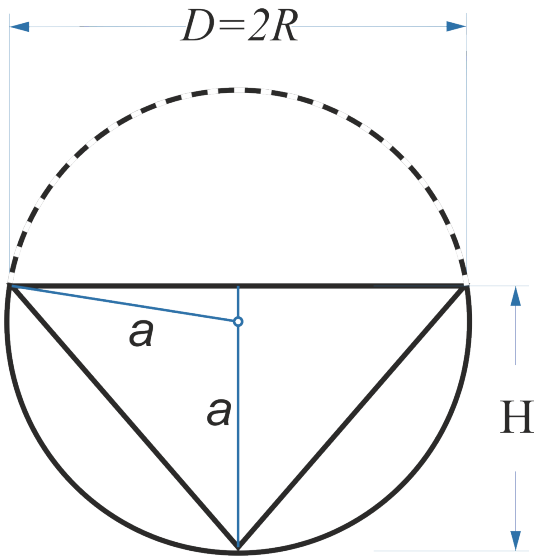


Рис. I.2.9: Проекция вазы на осевую плоскость

Обозначим радиус сферы буквой a, а радиус основания и высоту конуса R и H соответственно. Из рисунка найдём, что радиус горлышка зависит от высоты конуса и радиуса шара. Эта зависимость выражается уравнением $R^2 = a^2 - (H - a)^2$. Тогда выражение для объема конуса в зависимости от высоты и радиуса шара будет иметь вид: $V = \frac{1}{3}\pi(a^2 - (H - a)^2)H$.

Из условия задачи следует, что объем конуса выбран максимальным из тех, которые могут быть вписаны в шар заданного радиуса. Условием максимума объема

будет равенство нулю производной от объема по высоте конуса:

$$V(H)' = (\frac{1}{3}\pi(a^2 - (H-a)^2)H)' = 0$$

$$a^2 - (H-a)^2 - 2H(H-a) = 0$$

Откуда получим соотношение между высотой конуса и радиусом шара $H = \frac{4}{3}a$. Подставим полученные соотношения для высоты и радиуса основания конуса в формулу для объема и найдем радиус шара $a = \frac{81}{32}\sqrt[3]{\frac{V}{\pi}}$.

Для диаметра горлышка тогда имеем следующее выражение

$$D = 2R = \frac{4\sqrt{2}}{3}a = \frac{27\sqrt{2}}{8}\sqrt[3]{\frac{V}{\pi}} \approx 32.6$$

Ответ: 1,75461615531.

Задача I.2.4.9. (10 баллов)

Составляя расписание мастер-классов для финала олимпиалы НТИ по профилю "Разработка приложений виртуальной и дополненной реальности"организаторы столкнулись с затруднением. Мастер-классы должны проводится 1 раз в день с понедельника по четверг, но приглашенные специалисты имеют ряд требований к дню проведения мастер-класса. Предложите варианты расписания мастер-классов, если:

а) специалист по математическому моделированию может провести провести мастер-класс в понедельник, вторник или среду;

б) специалист по 3D моделированию может пообщаться с финалистами или в понедельник, или во вторник;

в) мастер-класс по физике может состояться во вторник или четверг;

г) мастер-класс по программированию интерактивного окружения - только во вторник или среду.

Сколько возможно вариантов составления расписания мастер-классов для проведения финала? Укажите их, перечисляя через точку с запятой мастер-классы в порядке их следования в течение финала. Например: Вариант 1. Математическое моделирование; 3D моделирование; программирование интерактивного окружения; физика.

Решение

Построим граф, вершины которого - возможные дни недели (1, 2, 3, 4) и мастер-классы (м, ф, п, 3d). Ребрами соединим мастер-классы с возможными днями проведения(рис. I.2.10).

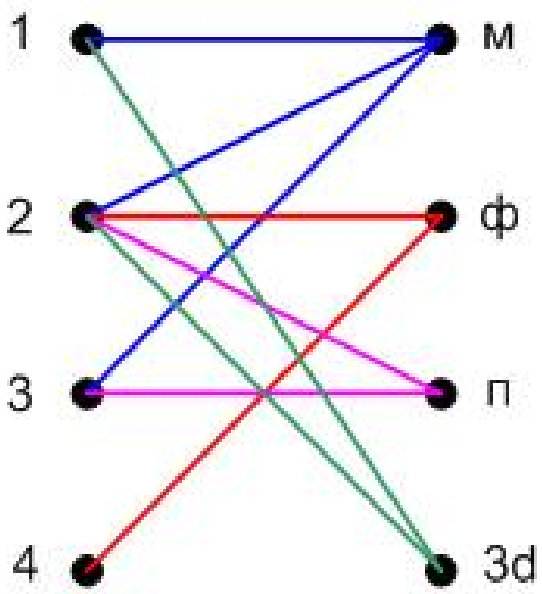


Рис. I.2.10: Граф вариантов расписания

Задача сводится к определению ребер, задающих взаимно однозначное соответствие между множеством дней и множеством мастер-классов.

Из рисунка можно увидеть, что всего количество таких соответствий $k = 3$

Приведем варианты расписания. Из графа видно, что в четверг может быть только физика. Все могут провести мастер-класс во вторник, а в понедельник и в среду согласны поработать вдое специалистов. Если математическое моделирование поставить в понедельник, то во вторник можно провести только 3D моделирование, в противном случае в среду мастер-класс поставить невозможно.

Ответ: 1234, 2134, 2314.

Задача I.2.4.10. (10 баллов)

При каких значениях параметра a фунцция

$g(x) = \frac{1}{3}x^3 + (a+2)x^2 + (a^2+4a-12)x - 24$ имеет экстремальные точки, принадлежащие промежутку $[-2, 9]$?

Найдите минимальное допустимое значение параметра a, при котором a соответствует условию.

Решение

$g'(x) = x^2 + 2(a+2)x + a^2 + 4a - 12.$

Находим критические точки: $g'(x) = 0$, $x^2 + 2(a+2)x + a^2 + 4a - 12 = 0$

$D_1 = (a+2)^2 - (a^2+4a-12) = 16$, $\sqrt{D_1} = 4$.

$x_1 = -a - 6$; $x_2 = -a + 2$; x_1, x_2- точки экстремума.

По условию

$$\begin{cases} -a-6 \geqslant -2, \\ -a+2 \leqslant 9; \end{cases}$$

$$\begin{cases} a \leqslant -4, \\ a \geqslant -7. \end{cases}$$

Таким образом, $-7 \leqslant a \leqslant -4$.

Ответ: -7.

Второй отборочный этап

Второй отборочный этап проводится в командном формате в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Продолжительность второго этапа составляет 52 дня. Задачи по информатике носят междисциплинарный характер и помогают отработать те навыки, которые потребуются для решения командной задачи заключительного этапа.

Участники не были ограничены в выборе языка программирования для решения задач.

Объем и сложность задач этого этапа подобраны таким образом, чтобы решение всех задач одним человеком было маловероятно. Это призвано обеспечить включение командной работы и распределения обязанностей. Решение каждой задачи дает определенное количество баллов. Баллы зачисляются в полном объеме за правильное решение задачи. Также существуют задачи, где допускается частичное решение. В данном этапе можно получить суммарно от 0 до 55 баллов.

Задачи по программированию выкладывались тремя партиями: в начале второго этапа, через две недели после начала и через пять недель после начала. Команды могут выполнять задачи в любом порядке. Задачи допускают неограниченное число попыток сдать решение.

Задачи второго этапа

Задачи по информатике

Задача II.1.1.1. Цилиндры (5 баллов)

Робототехническое устройство, собранное по дифференциальной схеме и с заданным диаметром колес, движется по прямой с постоянной скоростью. На устройстве установлен ультразвуковой датчик расстояния, направленный влево перпендикулярно направлению движения.

Слева от робототехнического устройства установлены в ряд цилиндры разного диаметра d_i. Цилиндры расположены таким образом, что все их центры находятся на одной прямой. Высота цилиндров много выше высоты, на которой установлен датчик расстояния. Цилиндры могут располагаться как вплотную друг к другу, так и с зазором.

Во время движения датчик расстояния, устроенный таким образом, что он имеет конус направленности $\alpha = 1''$ и возвращает минимальное значение из заданного диапазона или максимально возможное в случае, если отсутствуют какие-либо объекты в непосредственной видимости датчика. Известно, что показания ультразвукового **датчика идеальны**, в них всегда присутствует высокая доля помех. Частота опроса датчика — 10 Гц.

Необходимо найти цилиндр максимального диаметра. Известно, что радиус колес

равен 0.09 м.

При движении робота не гарантируется, что он движется параллельно прямой, на которой установлены цилиндры. Гарантируется, что робот движется так, что во время движения фиксирует датчиком все установленные цилиндры таким образом, что было сделано не менее 3 изменений для каждого цилиндра.

Формат входных данных

Первая строка входных данных содержит одно целое число — N, где:

- N — количество измерений ($3 \leqslant N \leqslant 1000$).

Далее идут N строк, содержащие 2 вещественных числа через пробел — Enc, S, где:

- Enc — среднее арифметическое показаний энкодеров левого и правого колеса в градусах ($0 \leqslant Enc \leqslant 10^6$);
- S — показание датчика расстояния в мм ($10 \leqslant S \leqslant 3700$).

Формат выходных данных

Одна строка, содержащая одно целое число — номер цилиндра максимального диаметра считая по ходу движения.

Примеры

Примеры входных данных и ответов к ним можно найти по данной ссылке.

Решение

Декомпозиция решения задачи:

1. Определить количество цилиндров в наборе данных и принадлежащие им точки на окружности
2. Вычислить диаметр каждого цилиндра по трем точкам на окружности
3. Определить номер цилиндра с максимальным диаметром

Для начала получим угол обзора камеры в десятичном формате из угловых значений ($d^\circ m' s''$) в десятичный формат (градусы) по формуле (II.1.1):

$$degrees = d + \frac{m}{60} + \frac{s}{3600} \tag{II.1.1}$$

В нашем случае получается $1'' = 0.0002778$ градусов, т.е. угол обзора незначительный и можно его принять за “прямую”, направленную перпендикулярно влево по ходу движения робота.

Для примера возьмем наборы данных N1 и N2. Посмотрим на данные в виде графиков (рисунки II.1.1 и II.1.2):

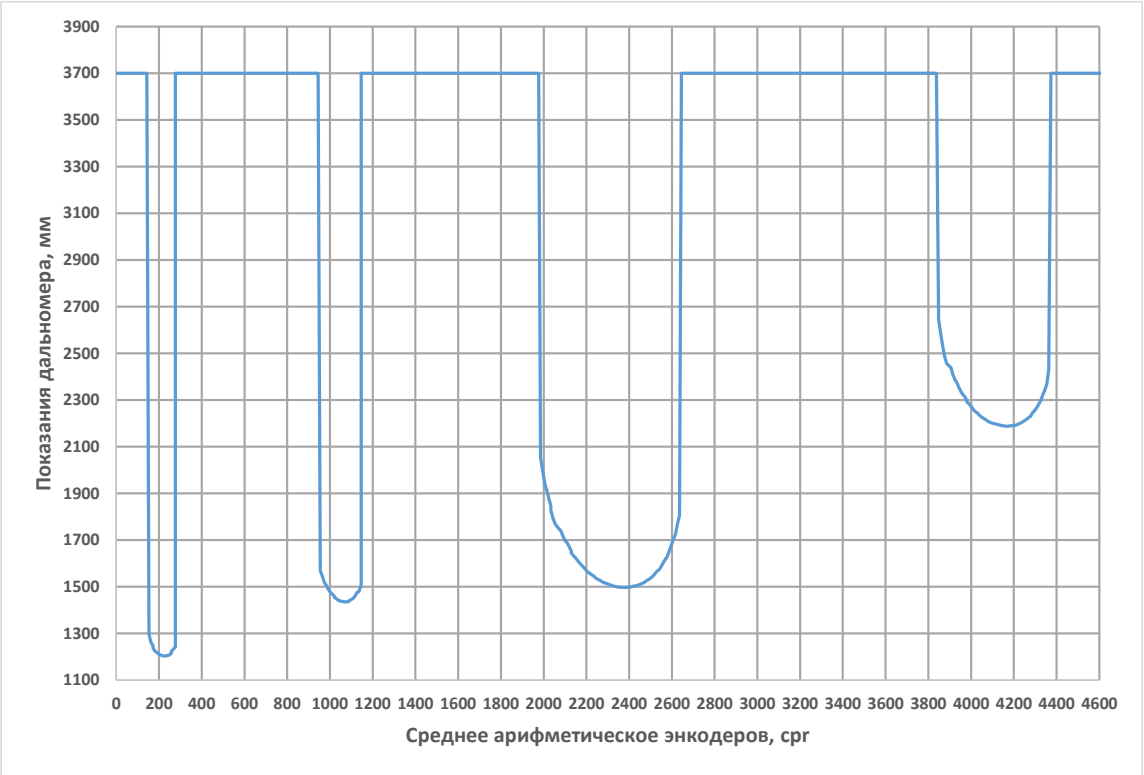


Рис. II.1.1: Набор данных N1

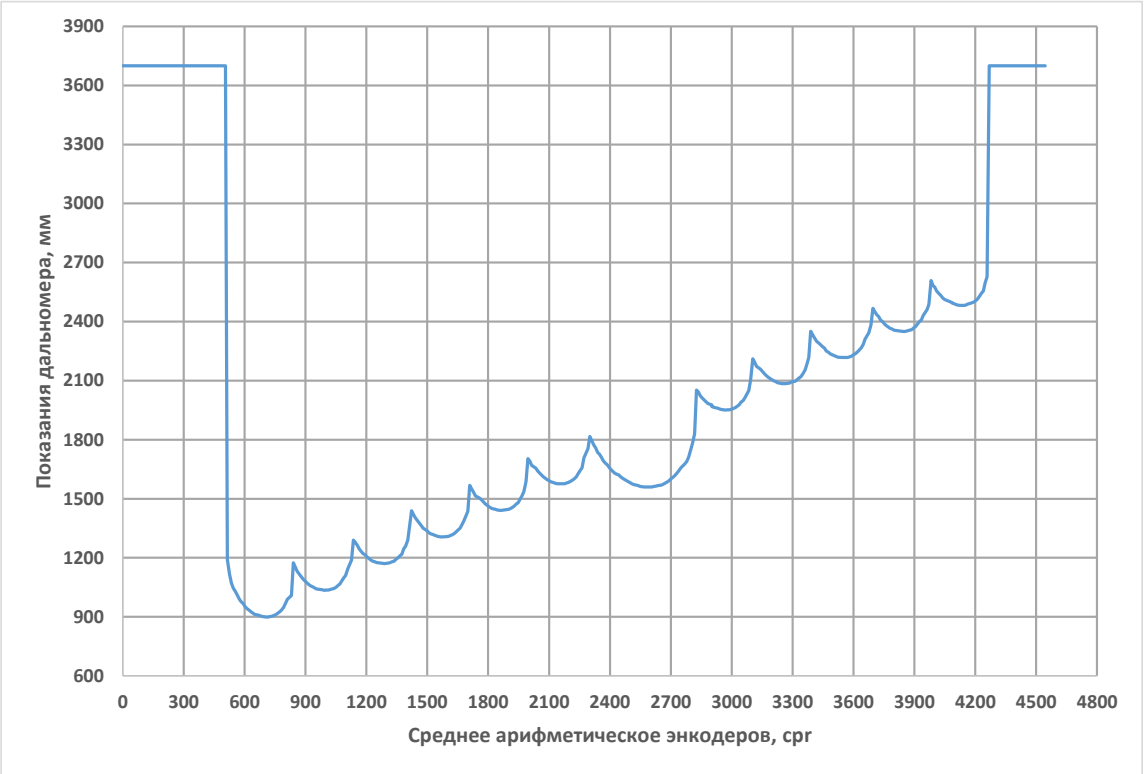


Рис. II.1.2: Набор данных N2

По графикам видно:

1. Искомые цилиндры могут располагаться очень близко друг к другу или между ними может быть какое-либо расстояние.
2. Максимальное расстояние в наборе данных, возвращаемое датчиком расстояния - 3700 мм, т.е. в этом случае отсутствуют какие-либо объекты.
3. Робот может двигаться как параллельно цилиндрам, так и по диагонали.

Декомпозируем задачу определения каждого цилиндра:

1. Определить начальную точку цилиндра (M_1).
2. Определить минимальное показание датчика расстояния относительно начальной точки цилиндра (M_2).
3. Определить конечную точку цилиндра.
4. Присвоить цилиндру номер (M_3).

Результаты определения точек каждого цилиндра для набора данных N1 показан на рисунке II.1.3:

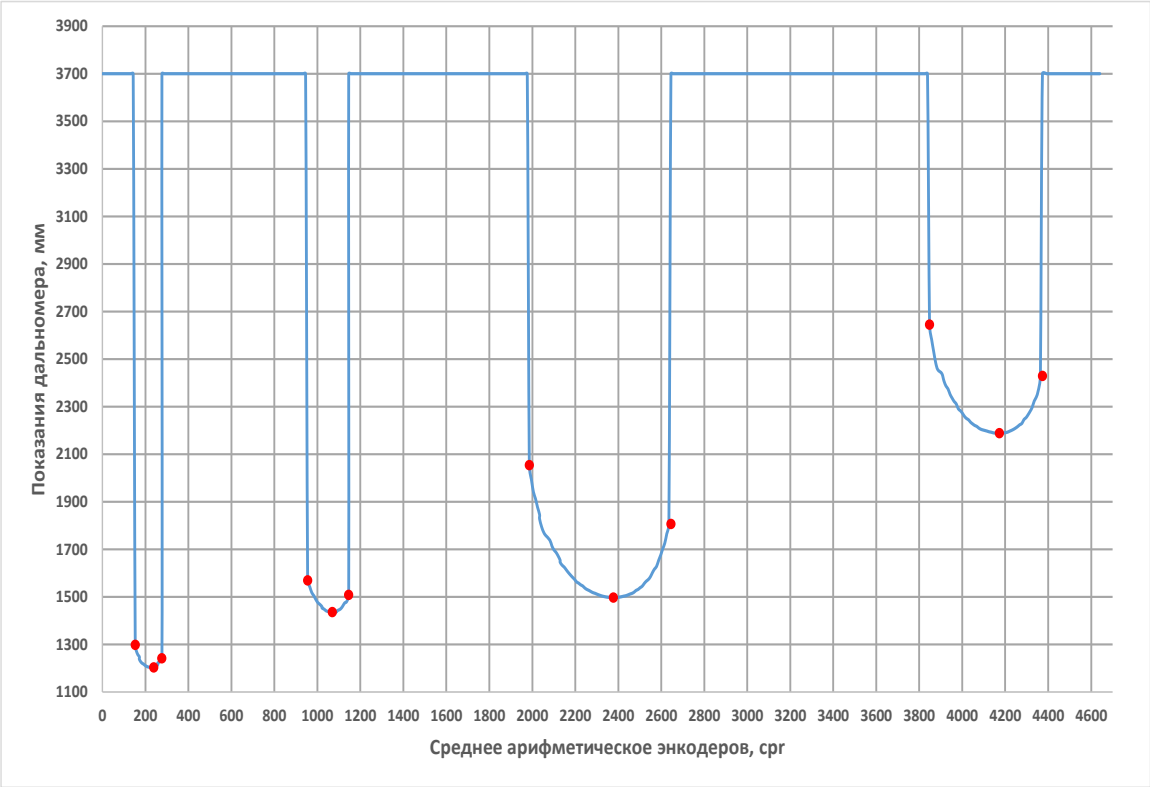


Рис. II.1.3: Крайние точки цилиндров из набора данных N1

Координаты точек указаны в таблице II.1.1:

Далее, по полученным 3 точкам каждого цилиндра (M_1, M_2, M_3) определяем их диаметры. Для этого сначала находим центры цилиндров. Схема нахождения центра цилиндра по трем известным точкам показана на рис. В примере использованы точки цилиндра N2 из набора данных N1.

Цилиндр	Точка	X	Y
1	M_1	152,634042	1298,516512
	M_2	229,025706	1203,757882
	M_3	276,769403	1242,196202
2	M_1	954,744156	1569,562435
	M_2	1069,330793	1435,830474
	M_3	1145,721379	1508,241296
3	M_1	1986,027389	2053,845644
	M_2	2377,531502	1497,469187
	M_3	2644,901633	1806,757331
4	M_1	3848,065298	2644,686222
	M_2	4172,728161	2188,261747
	M_3	4373,256726	2429,352999

Таблица II.1.1: Координаты точек цилиндров из набора данных N1

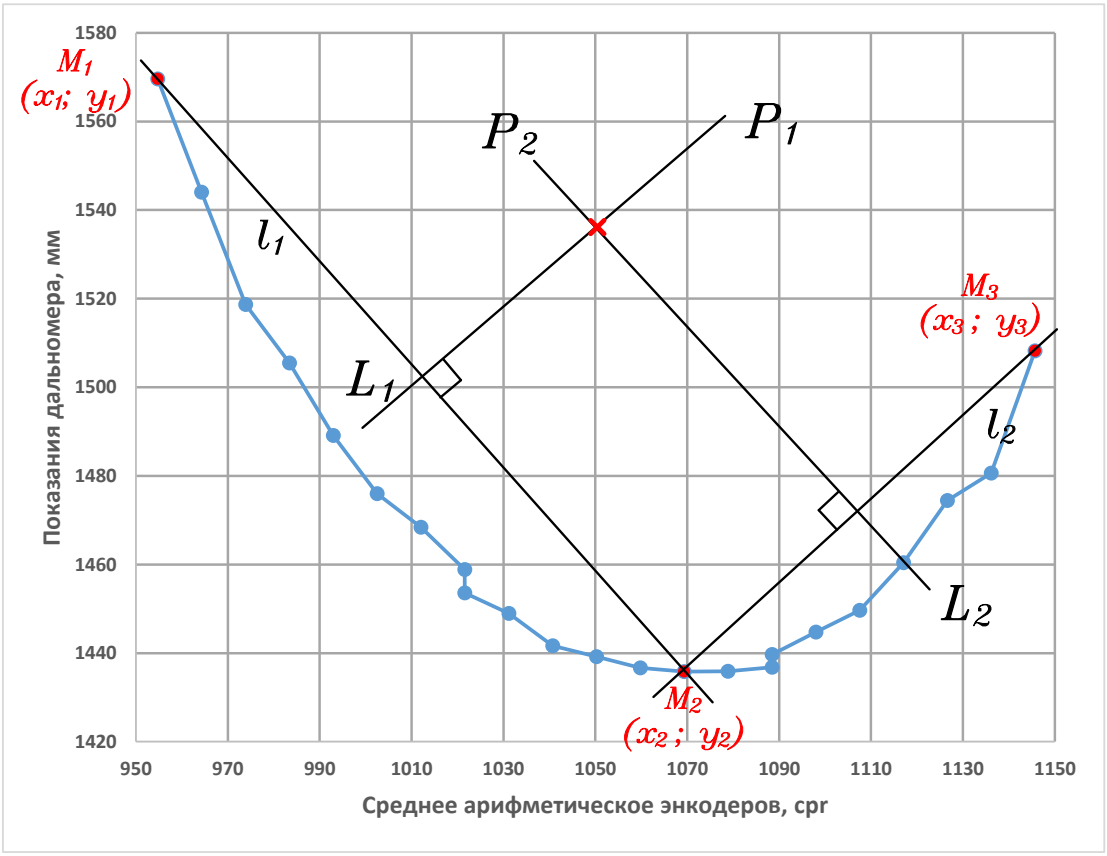


Рис. II.1.4: Схема нахождения центра цилиндра по 3 известным точкам

Уравнение прямой, проходящей через две точки, имеет вид:

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

В этом случае угловой коэффициент k определятся по формуле:

$$k = \frac{y_2 - y_1}{x_2 - x_1};$$

Найдем коэффициенты k и b для прямых l_1 и l_2 по формулам (II.1.2) и (II.1.2), (II.1.2) и (II.1.2):

$$\begin{aligned} k_{l_1} &= \frac{y_2 - y_1}{x_2 - x_1} \\ b_{l_1} &= y_2 - k_{l_1} \cdot x_2 \\ k_{l_2} &= \frac{y_3 - y_2}{x_3 - x_2} \\ b_{l_2} &= y_3 - k_{l_2} \cdot x_3 \end{aligned} \tag{II.1.2}$$

Центр цилиндра находится на пересечении двух перпендикулярных прямых P_1 и P_2, проходящих через середины отрезков M_1M_2 и M_2M_3.

Найдем середины отрезков M_1M_2 и M_2M_3 по формулам (II.1.3) и (II.1.3).

$$\begin{aligned} L_1 &= \left(\frac{x_1 + x_2}{2}; \frac{y_1 + y_2}{2} \right) \\ L_2 &= \left(\frac{x_2 + x_3}{2}; \frac{y_2 + y_3}{2} \right) \end{aligned} \tag{II.1.3}$$

Прямая, перпендикулярная к линии с коэффициентом наклона k имеет коэффициент наклона: $-1/k$, значит уравнения прямых P_1 и P_2, перпендикулярных l_1 и l_2, соответственно, запишем следующим образом: для P_1 – по формулам (II.1.4) и (II.1.4), для P_2 - по формулам (II.1.4) и (II.1.4):

$$\begin{aligned} k_{P_1} &= -\frac{1}{k_{l_1}} \quad \Leftrightarrow \quad k_{P_1} = -\frac{x_2 - x_1}{y_2 - y_1} = \frac{x_1 - x_2}{y_2 - y_1} \\ b_{P_1} &= y_{L_1} - k_{P_1} \cdot x_{L_1} \\ k_{P_2} &= -\frac{1}{k_{l_2}} \quad \Leftrightarrow \quad k_{P_2} = -\frac{x_3 - x_2}{y_3 - y_2} = \frac{x_2 - x_3}{y_3 - y_2} \\ b_{P_2} &= y_{L_2} - k_{P_2} \cdot x_{L_2} \end{aligned} \tag{II.1.4}$$

Сейчас мы уже можем найти координаты точки центра цилиндра (или точку пересечения прямых P_1 и P_2) по формулам (II.1.5) и (II.1.5):

$$\begin{aligned} x &= \frac{b_{P_1} - b_{P_2}}{k_{P_2} - k_{P_1}} \\ y &= k_{P_1} \cdot x + b_{P_1} \end{aligned} \tag{II.1.5}$$

Для нашего примера (цилиндр N2 набора данных N1) координаты центра цилиндра равны: $M_0(1051.406; 1528.247)$

Диаметр цилиндра вычислим по среднему расстоянию от центра цилиндра до каждой из известных точек M_i. Расстояние между двумя точками находим по формуле:

$$r = \sqrt{(M_{ix} - M_{0x})^2 + (M_{iy} - M_{0y})^2}$$

В качестве ответа выводим номер цилиндра с наибольшим диаметром.

Результаты вычислений по наборам данных N1 и N2 показаны на рис. II.1.5 и II.1.6

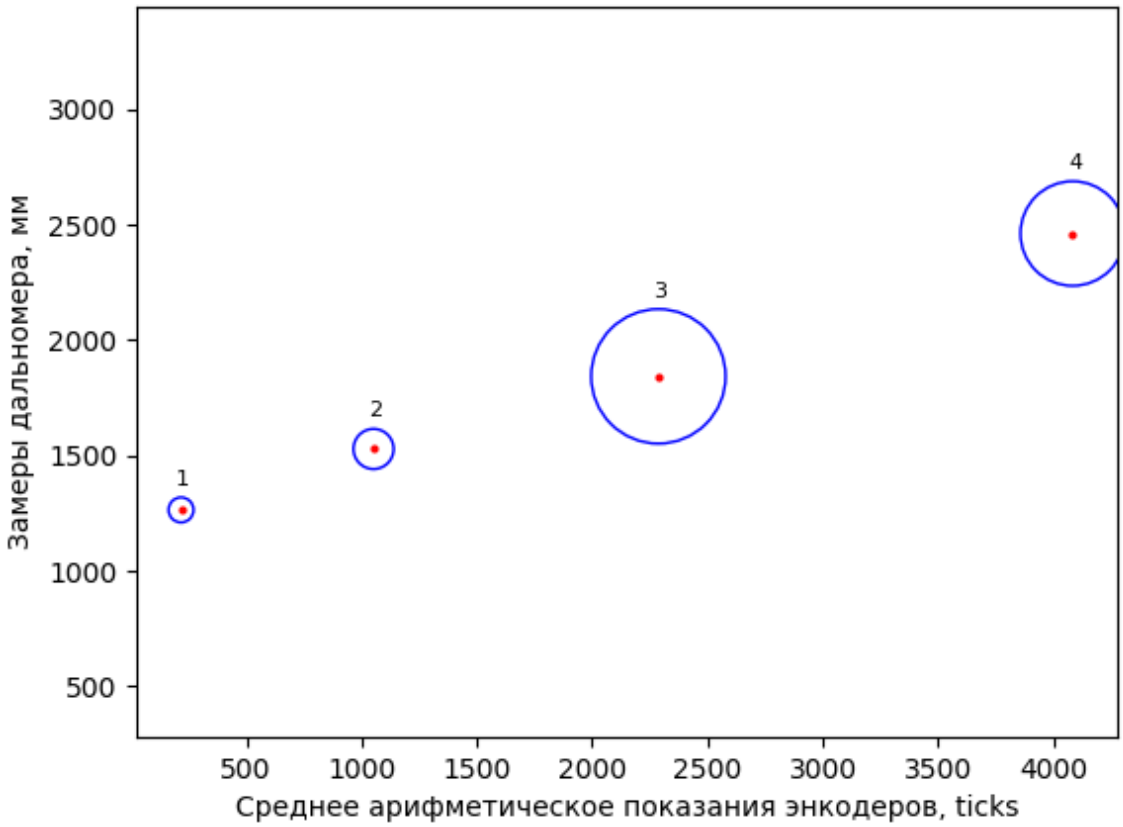


Рис. II.1.5: Цилиндры из набора данных N1

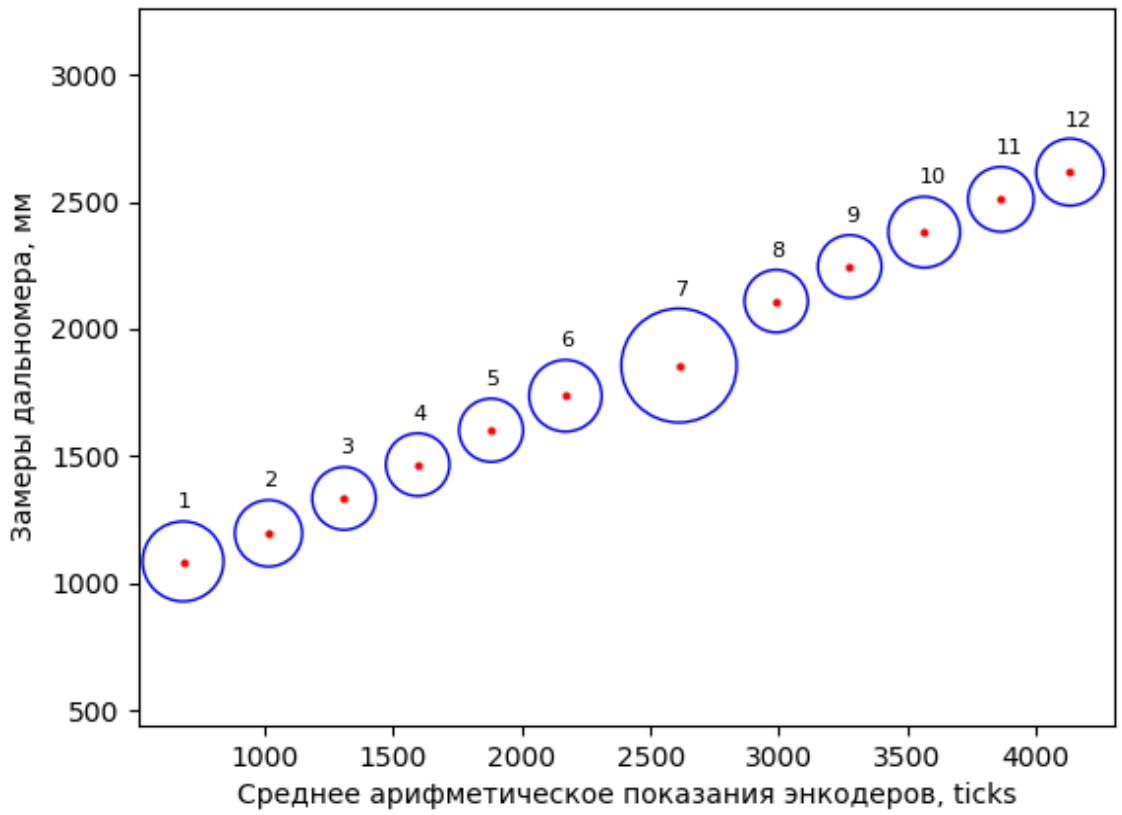


Рис. II.1.6: Цилиндры из набора данных N2

Пример программы-решения

Ниже представлено решение на языке Python 3

```
# -*- coding: utf-8 -*-
import math

def get_center(data):
  def get_kb(xy1, xy2, l):
    k = (xy1[0] - xy2[0]) / (xy2[1] - xy1[1])
    b = l[1] - k * l[0]
    return k, b

  def get_l(xy1, xy2):
    return (xy1[0] + xy2[0]) / 2, (xy1[1] + xy2[1]) / 2

  l1 = get_l(data[0], data[1])
  l2 = get_l(data[1], data[2])

  k1, b1 = get_kb(data[0], data[1], l1)
  k2, b2 = get_kb(data[1], data[2], l2)

  x = (b1 - b2) / (k2 - k1)
  y = k1 * x + b1

  return x, y
```

```
23
24  def calculate(lst):
25    dist_prev = 0
26    cylinders = [[]]
27    y = []
28    grow = False
29    for d in range(len(lst)):
30      encoder, distance = lst[d]
31
32      if distance == 3700:
33        if len(cylinders[-1]) == 2 and dist_prev != 3700:
34          cylinders[-1].append((encoder, y[-1]))
35          cylinders.append([])
36        dist_prev = distance
37        continue
38
39      dist_prev = distance
40      if len(y) > 0:
41        grow = distance <= y[-1]
42
43      if grow and len(cylinders[-1]) == 0:
44        cylinders[-1].append((encoder, y[-1]))
45
46      if not grow and len(cylinders[-1]) == 1:
47        cylinders[-1].append((encoder, y[-1]))
48
49      if grow and len(cylinders[-1]) == 2:
50        cylinders[-1].append((encoder, y[-1]))
51        cylinders.append([])
52
53      y.append(distance)
54
55    if len(cylinders[-1]) == 0:
56      cylinders.pop()
57
58    diams = []
59    for c in range(len(cylinders)):
60      center = get_center(cylinders[c])
61      radius = sorted([abs(cylinders[c][x][0] - center[0]) for x in range(3)])[1]
62      diams.append(radius * 2)
63
64    return diams.index(max(diams)) + 1
65
66  raw_data = []
67  for i in range(int(input())):
68    raw_data.append(list(map(float, input().split())))
69  print(calculate(raw_data))
```

Задача II.1.1.2. Цилиндры (5 баллов)

Робототехническое устройство, собранное по дифференциальной схеме и с заданным диаметром колес, движется по прямой с постоянной скоростью. На устройстве установлен ультразвуковой датчик расстояния, направленный влево перпендикулярно направлению движения.

Слева от робототехнического устройства установлены в ряд цилиндры разного диаметра d_i. Цилиндры расположены таким образом, что все их центры находятся на одной прямой. Высота цилиндров много выше высоты, на которой установлен

датчик расстояния. Цилиндры могут располагаться как вплотную друг к другу, так и с зазором.

Во время движения датчик расстояния, устроенный таким образом, что он имеет конус направленности $\alpha = 1^{''}$ и возвращает минимальное значение из заданного диапазона или максимально возможное в случае, если отсутствуют какие-либо объекты в непосредственной видимости датчика. Известно, что показания ультразвукового **датчика не идеальны**, в них всегда присутствует высокая доля помех. Частота опроса датчика — 10 Гц.

Необходимо найти цилиндр максимального диаметра. Известно, что радиус колес равен 0.09 м.

При движении робота не гарантируется, что он движется параллельно прямой, на которой установлены цилиндры. Гарантируется, что робот движется так, что во время движения фиксирует датчиком все установленные цилиндры таким образом, что было сделано не менее 3 изменений для каждого цилиндра.

Формат входных данных

Первая строка входных данных содержит одно целое число — N, где:

- N — количество измерений ($3 \leqslant N \leqslant 1000$).

Далее идут N строк, содержащие 2 вещественных числа через пробел — Enc, S, где:

- Enc — среднее арифметическое показаний энкодеров левого и правого колеса в градусах ($0 \leqslant Enc \leqslant 10^6$);
- S — показание датчика расстояния в мм ($10 \leqslant S \leqslant 3700$).

Формат выходных данных

Одна строка, содержащая одно целое число — номер цилиндра максимального диаметра считая по ходу движения.

Примеры

Примеры входных данных и ответов к ним можно найти по данной ссылке.

Решение

Решение задачи аналогично решению от предыдущей задачи "Цилиндры". Но с одним дополнением. Согласно условиям задачи, показания датчика расстояния **не идеальны**, т.е. имеют некий "разброс" значений. В реальных условиях это может быть вызвано несколькими факторами: влажностью, температурой, особенностями изготовления датчика и другими внешними и внутренними факторами.

Основная задача состоит в фильтрации значений.

Подробно тема фильтрации данных описана в материалах Университета Иннополис в разделе "Программирование интеллектуальных робототехнических систем" по адресу: Подраздел “Фильтрация значений датчиков”

Рассмотрим фильтрацию на примере набора данных N8. На рис. II.1.7 показаны исходные данные в виде графика.

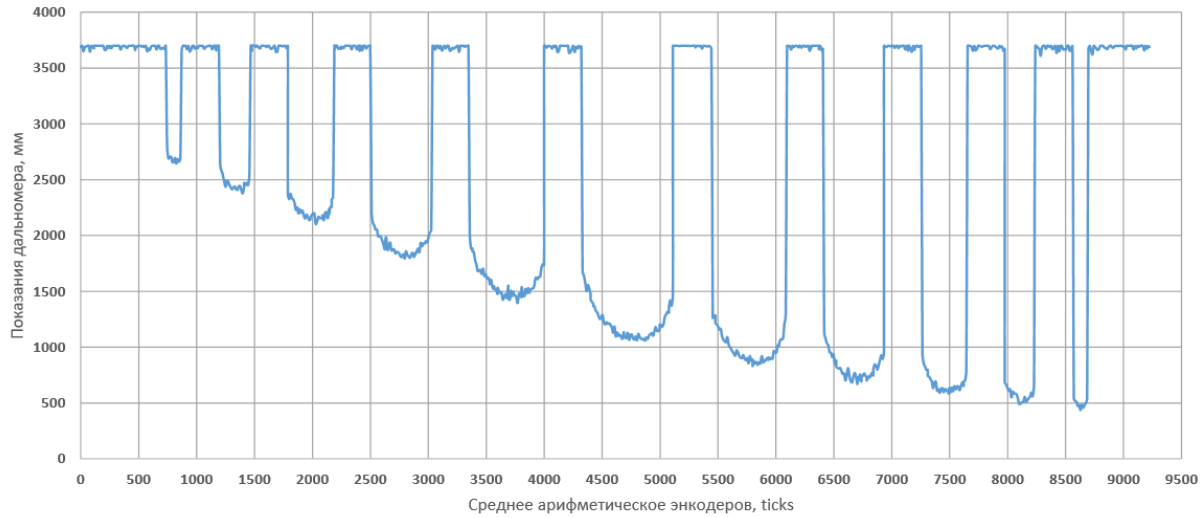


Рис. II.1.7: Данные из набора N8

Как заметно из графика, данные достаточно "шумные", что не позволяет однозначно определить начало и конец каждого цилиндра.

Пробуем провести фильтрацию данных. Сначала по алгоритму "Взвешенное скользящее среднее" с окном = 10, затем "простое скользящее среднее" с окном = 8. Но предварительно очистим "пустоты" между цилиндрами. Результат фильтрации показан на рис.

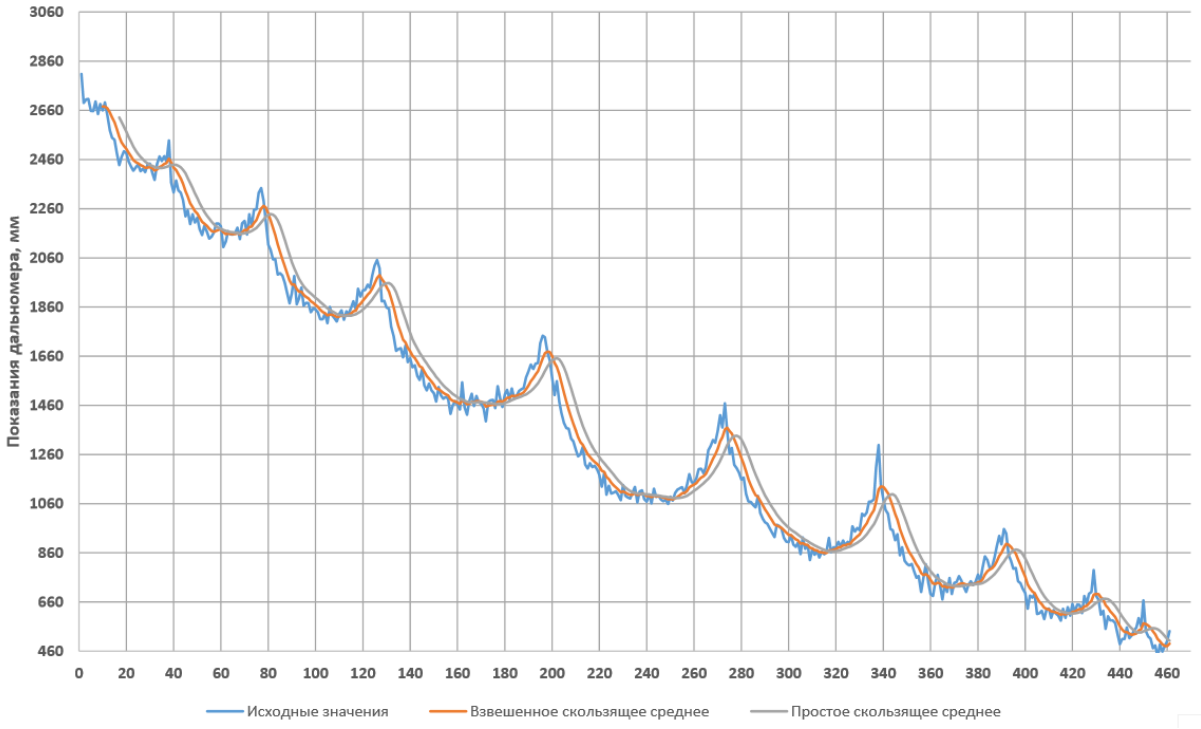


Рис. II.1.8: Данные из набора N8 после фильтрации

Теперь из полученных данных можно определить диаметры цилиндров по алгоритму из предыдущей задачи "Цилиндры".

Пример программы-решения

Ниже представлено решение на языке Python 3

```
# -*- coding: utf-8 -*-

def filters(data):
  def sma(lst, size=5):
    # Simple Moving Average
    return [sum(lst[i - size:i]) / size for i in range(size, len(lst))]

  def wma(lst, size=10):
    # Weighted moving average
    summ_w = sum(range(size + 1))
    res = []
    for x in range(len(lst) - size):
      dt = lst[x:size + x]
      res.append(sum([(i + 1) * dt[i] for i in range(len(dt))]) / summ_w)
    return res

  def median(lst, size=3):
    odd = size % 2 != 0
    res = []
    for i in range(size, len(lst)):
      l = sorted(lst[i - size:i])
      if odd:
        res.append(l[size // 2])
      else:
        res.append((l[size // 2] + l[size // 2 - 1]) / 2)
    return res

  def ema(lst, k_e=0.85):
    # Exponential Moving Average
    res = [lst[0]]
    for i in range(1, len(lst)):
      res.append(k_e * res[-1] + (1 - k_e) * lst[i])
    return res

  data = [i for i in data if 3600 > i > 10]
  return sma(wma(data, 10), 8)

def get_center(data):
  def get_kb(xy1, xy2, l):
    k = (xy1[0] - xy2[0]) / (xy2[1] - xy1[1])
    b = l[1] - k * l[0]
    return k, b

  def get_l(xy1, xy2):
    return (xy1[0] + xy2[0]) / 2, (xy1[1] + xy2[1]) / 2

  l1 = get_l(data[0], data[1])
  l2 = get_l(data[1], data[2])

  k1, b1 = get_kb(data[0], data[1], l1)
  k2, b2 = get_kb(data[1], data[2], l2)

  x = (b1 - b2) / (k2 - k1)
  y = k1 * x + b1

  return x, y
```

```
57
58  def calculate(lst_raw):
59    lst = filters(lst_raw)
60
61    cylinders, y = [[]], []
62    grow = False
63
64    for d in range(len(lst)):
65      distance = lst[d]
66
67      if len(y) > 0:
68        grow = distance <= y[-1]
69
70      if grow and len(cylinders[-1]) == 0:
71        cylinders[-1].append((d, y[-1]))
72
73      if not grow and len(cylinders[-1]) == 1:
74        cylinders[-1].append((d, y[-1]))
75
76      if grow and len(cylinders[-1]) == 2:
77        cylinders[-1].append((d, y[-1]))
78        cylinders.append([])
79
80      y.append(distance)
81
82    if len(cylinders[-1]) != 3:
83      cylinders.pop()
84
85    diams = []
86    for c in range(len(cylinders)):
87      center = get_center(cylinders[c])
88      radius = sorted([abs(cylinders[c][x][0] - center[0]) for x in range(3)])[1]
89      diams.append(radius * 2)
90
91    lengths = [
92        cylinders[i][1][0] - cylinders[i][0][0] for i in range(len(cylinders))
93    ]
94    return lengths.index(max(lengths)) + 1
95
96  data_raw = []
97  for i in range(int(input())):
98    data_raw.append(list(map(float, input().split()))[1])
99  print(calculate(data_raw))
```

Задача II.1.1.3. Цветные препятствия (10 баллов)

Даны два дифференциальных робота: Алиса и Боб. У Боба есть камера, а у Алисы — датчики препятствия. Данные роботы находятся в квадратном лабиринте с перегородками. Пол и ограждения — белые, а перегородки внутри лабиринта цветные. Перегородки или их линии образуют квадратные секции размером 250×250 мм, стороны которых параллельны стенкам.

Известно, что перегородки могут пересекаться и движения данных роботов ограничены следующими командами:

- F — проехать из центра одно сектора в центр следующего по ходу движения сектора;
- L — повернуться на месте налево, относительно текущего направления робота;

- R — повернуться на месте направо, относительно текущего направления робота.

Алиса начинает движение из неизвестной точки и останавливается где-то в лабиринте. Известны показания датчиков робота во время перемещения и его действия. Гарантируется, что по данным показаниям можно однозначно идентифицировать маршрут Алисы.

Перечислите последовательность в которой сменяются цвета препятствий перед роботом Бобом после каждой выполненной команды при его перемещении по оптимальному пути из точки старта в точку финиша. Гарантируется, что Боб сможет доехать до точки финиша. Путь считается оптимальным, если было сделано наименьшее количество действий(команд).

Характеристики роботов:

Камера Боба имеет угол обзора α и направленна по ходу движения робота. Высота и угол установки камеры таковы, что если бы перед роботом не было препятствий, то линия горизонта проходила бы горизонтально ровно по центру кадра. Высота перегородок в лабиринте значительно выше высоты установленной камеры.

Датчики препятствия, установленные на Алисе, расположены таким образом, что они позволяют определить есть ли слева, спереди или справа проезд в соседний сектор. Робот Боб и Алиса имеют вид цилиндров диаметром 250 мм однотонного чёрного цвета и занимает практически весь сектор. В случае если поле зрения датчиков Алисы попадет Боб, то он будет распознан как препятствие.

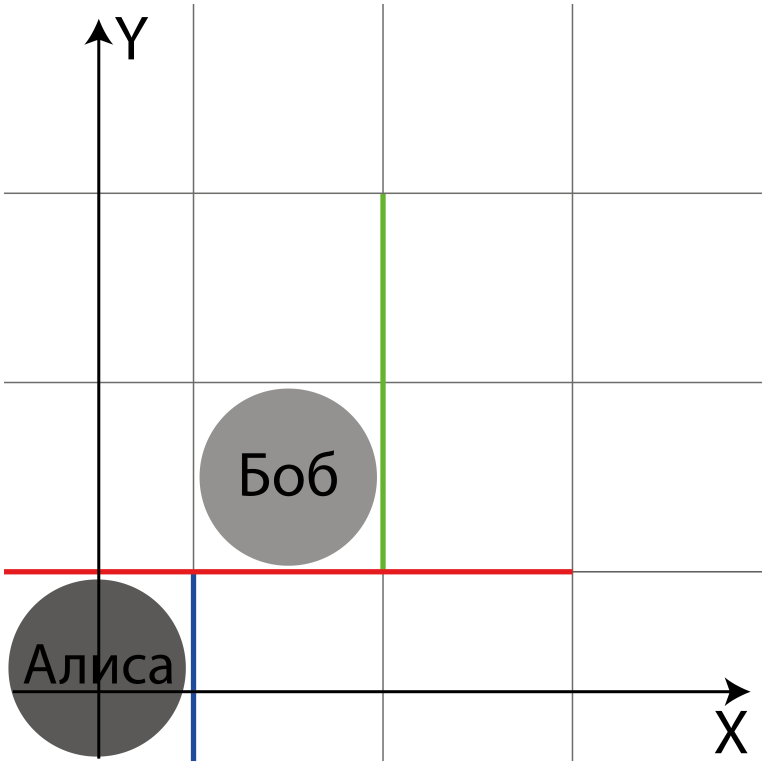


Рис. II.1.9: Пример начального расположения Боба и Алисы

Формат входных данных

Первая строчка содержит четыре целых числа через пробел – H, N, K, α где:

- H — сторона поля в мм ($10^3 \leqslant H \leqslant 10^6$);
- N — количество цветных перегородок ($1 \leqslant N \leqslant 100$);
- K — количество движений Алисы ($1 \leqslant K \leqslant 30$);
- α — угол обзора камеры ($5° \leqslant \alpha \leqslant 25°$).

Вторая строчка содержит одну букву и 4 целых числа через пробел — D, X_s, Y_s, X_e, Y_e, где:

- D — начальная ориентация Боба ($D \in \{U, D, L, R\}$, где U - по направлению оси Y, D - против направления оси Y, L - против направления оси X, R - по направлению оси X)
- X_s — начальная координата X_s Боба в мм ($0 \leqslant X_s \leqslant H$, $X_s = 125+250a$, $a \in \mathbb{N}$);
- Y_s — начальная координата Y_s Боба в мм ($0 \leqslant Y_s \leqslant H$, $Y_s = 125 + 250a$, $a \in \mathbb{N}$);
- X_e — конечная координата X_e Боба в мм ($0 \leqslant X_e \leqslant H$, $X_e = 125+250a$, $a \in \mathbb{N}$);
- Y_e — конечная координата Y_e Боба в мм ($0 \leqslant X_e \leqslant H$, $Y_e = 125 + 250a$, $a \in \mathbb{N}$);

Далее идут N строк, содержащие 4 целых числа и код цвета через пробел — $X_{i,1}$, $Y_{i,1}$, $X_{i,2}$, $Y_{i,2}$, Col_i, где:

- $X_{i,1}$ — координата $X_{i,1}$ начальной точки перегородки в мм ($0 \leqslant X_{i,1} \leqslant H$, $X_{i,1} = 250a$, $a \in \mathbb{N}$);
- $Y_{i,1}$ — координата $Y_{i,1}$ начальной точки перегородки в мм ($0 \leqslant Y_{i,1} \leqslant H$, $Y_{i,1} = 250a$, $a \in \mathbb{N}$);
- $X_{i,2}$ — координата $X_{i,2}$ конечной точки перегородки в мм ($0 \leqslant X_{i,2} \leqslant H$, $X_{i,2} = 250a$, $a \in \mathbb{N}$);
- $Y_{i,2}$ — координата $Y_{i,2}$ конечной точки перегородки в мм ($0 \leqslant Y_{i,2} \leqslant H$, $Y_{i,2} = 250a$, $a \in \mathbb{N}$);
- Col_i — код цвета данной перегородки в шестнадцатеричной системе счисления ($000000 \leqslant Col_i < FFFFFF$);

Далее идут K строк, содержащие 3 целых числа и одну букву через пробел — S_l, S_f, S_r, Act, где:

- S_l — показания датчика обнаружения препятствия, направленного влево ($S_l \in \{0, 1\}$, где 0 — препятствие не обнаружено);
- S_f — показания датчика обнаружения препятствия, направленного вперёд ($S_f \in \{0, 1\}$, где 0 — препятствие не обнаружено);
- S_r — показания датчика обнаружения препятствия, направленного вправо ($S_r \in \{0, 1\}$, где 0 — препятствие не обнаружено);
- Act — команда выполненная роботом после получения данных с датчиков ($Act \in \{F, L, R\}$).

Формат выходных данных

Одна строка, содержащая последовательность кодов цветов (исключая белый), которые будут менять друг друга в камере Боба после завершения каждого из действия, выполненного во время его перемещения. Разделитель между кодами цветов — пробел.

Примеры

Пример №1

Стандартный ввод
2000 3 11 15 U 875 875 125 375 250 1250 250 250 FF0000 250 250 1000 250 0000FF 1500 500 1500 1250 FFFF00 0 0 1 F 0 0 1 F 0 0 1 F 0 0 0 F 0 1 0 R 1 0 0 F 1 0 0 F 1 0 0 F 1 0 1 F 1 0 1 F 1 0 1 F
Стандартный вывод
000000

Пример №2

Стандартный ввод
1500 4 7 15 D 125 875 375 125 0 750 500 750 FF0000 250 0 250 250 FFFF00 250 250 1000 250 0000FF 750 500 750 750 00FF00 1 0 0 R 0 0 0 F 0 0 1 F 0 1 1 L 0 0 1 F 0 0 1 F 1 1 1 L
Стандартный вывод
FF0000 FFFF00

Комментарии

Оптимальным считается маршрут, длина которого, состоящая из возможных команд, минимальна.

Решение

Декомпозиция решения задачи:

1. Составить карту поля с учетом цветов перегородок
2. Локализоваться роботом Алиса (определить конечный сектор после выполнения указанных команд перемещения)
3. Определить оптимальный маршрут перемещения робота Боб в заданный конечный сектор
4. Во время перемещения робота Боба в конечный сектор отслеживать цвета перегородок, встречающие на пути

В данной задаче поле представлено в виде лабиринта. Координаты установки перегородок и их цвета определяются согласно условиям задачи.

Для представления карты местности, в т.ч. и для лабиринта, можно использовать графы, точнее матрицу смежности. Более подробно про варианты представления местности в цифровом виде описано в материалах Университета Иннополис в разделе "Программирование интеллектуальных робототехнических систем" по адресу: Подраздел “Построение карты. Локализация.”

После составления карты начинаем локализоваться роботом Алиса. По условиям задачи начальные координаты секции старта Алисы не заданы, но указаны состояния датчиков расстояния (с трех сторон) и команды перемещения.

Как вариант, можно "вставать" в каждую секцию лабиринта в каждом из четырех направлений (север, восток, юг, запад) и сравнивать состояние датчиков из задания с матрицей смежности лабиринта, если они совпадают, то “выполняем” указанное действие и переходим к сравнению следующих состояний. Если не совпадают - меняем направление и/или секцию. В задании точно есть одна секция в определенном направлении, где все совпадает. После того, как определим секцию старта робота Алисы, мы должны еще вычислить ее конечную секцию, т.к. робот Алиса для робота Боб является препятствием черного цвета. Для этого мы должны учитывать смену координат и направления робота при выполнении указанных действий для Алисы.

После локализации Алисы второй робот - Боб вычисляет оптимальный маршрут перемещения от секции старта до конечной секции. Более подробно про расчеты оптимального пути написано в материалах Университета Иннополис в разделе "Программирование интеллектуальных робототехнических систем" по адресу: Подраздел “Планирование и построение маршрута”

После составления пути, начинаем по нему перемещаться роботом “Боб” и отслеживать цвета стенок, которые встречаются перед ним.

В качестве результата нужно вывести цвета стенок, которые видел робот Боб во время передвижения.

Рассмотрим пример на наборе данных N8.

На рис. II.1.10 показан лабиринт с расставленными стенками и их цветами, на-

чальный сектор старта робота Боб и начальный сектор робота Алиса (уже после локализации)

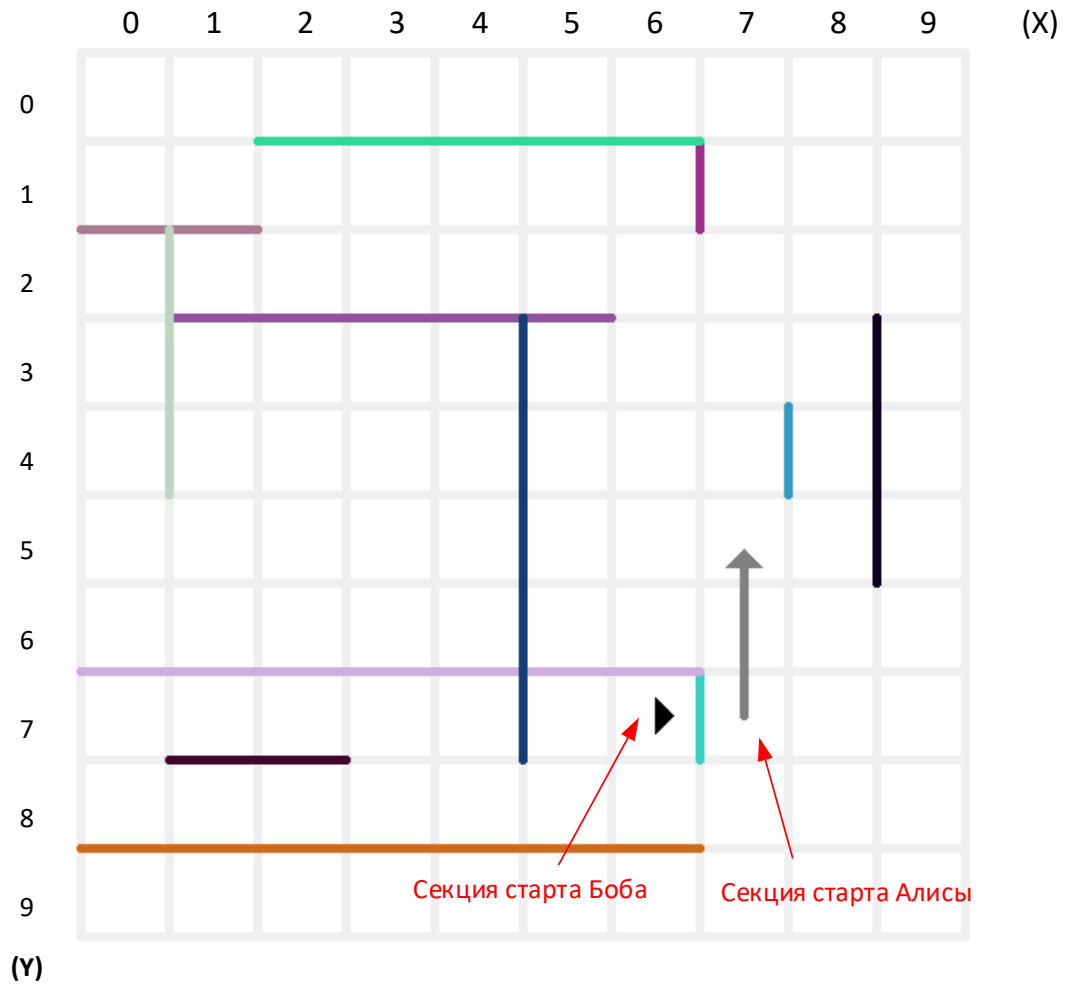


Рис. II.1.10: Расположение перегородок в лабиринте и положение роботов

На рис. II.1.11 показаны пути перемещения роботов Алиса и Боб и пунктирными кругами показаны перегородки, встречающие на пути робота Боб и их порядок.

Ответ в данной задаче на наборе данных N8: $3ACFC4\ \ CB6B1B$ 000000 $BDD5C1$

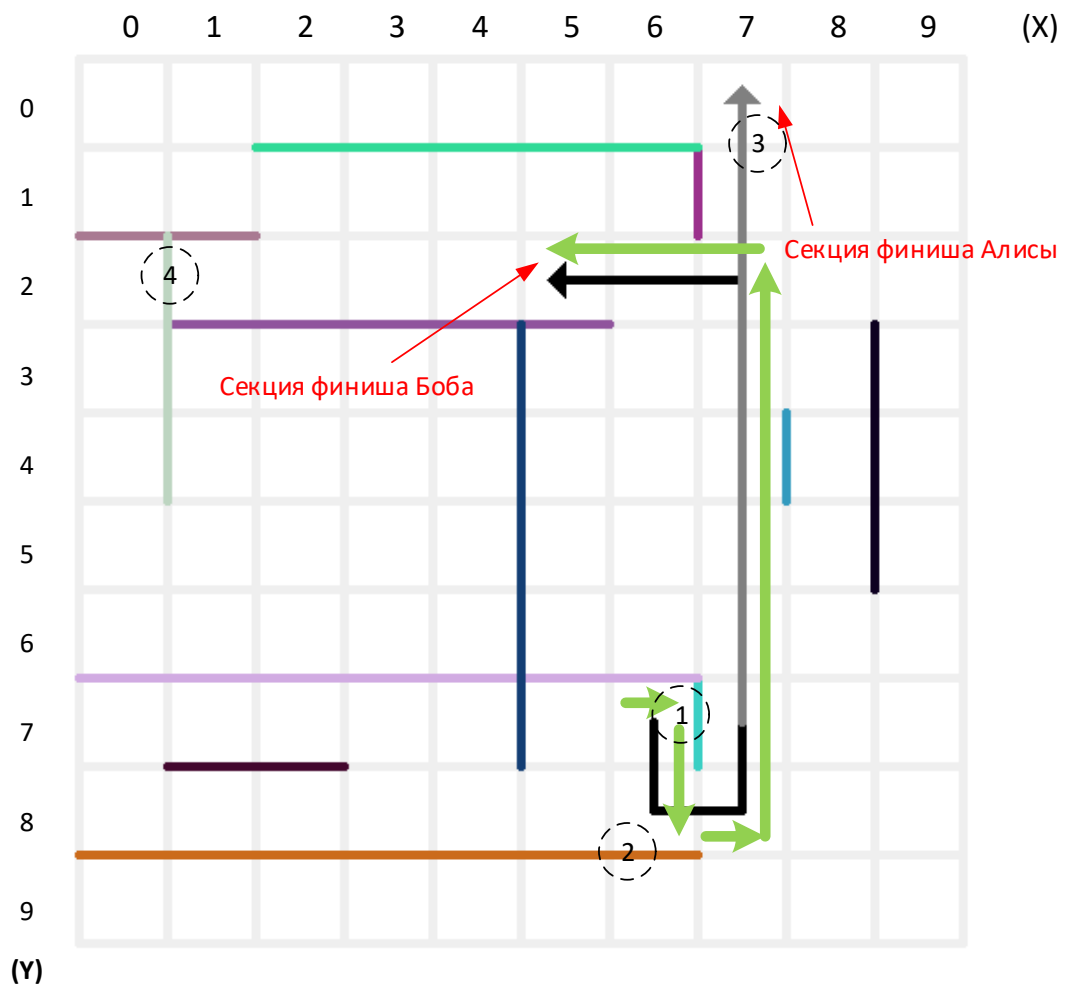


Рис. II.1.11: Положения роботов после выполнения заданий

Пример программы-решения

Ниже представлено решение на языке Python 3

```
# -*- coding: utf-8 -*-

import sys


def set_walls(datas):
  # расставляем перегородки
  global MAP

  for _ in range(len(datas)):
    Xs, Ys, Xe, Ye, clr = datas.pop(0).split()
    Xs, Ys, Xe, Ye = map(lambda x: int(x) // cell_size, (Xs, Ys, Xe, Ye))

    min_y, max_y = min(Ys, Ye), max(Ys, Ye)
    min_x, max_x = min(Xs, Xe), max(Xs, Xe)

    if Xs == Xe:  # вертикальная стена
      for y in range(min_y, max_y):
        cell1 = cells_by_task[xy_to_cell(y, Xs)]
```

```
20          cell2 = cells_by_task[xy_to_cell(y, Xs - 1)]
21          map_update(cell1, cell2, clr)
22
23      if Ys == Ye:  # горизонтальная стена
24        for x in range(min_x, max_x):
25          cell1 = cells_by_task[xy_to_cell(Ys, x)]
26          cell2 = cells_by_task[xy_to_cell(Ys - 1, x)]
27          map_update(cell1, cell2, clr)
28
29
30  def xy_to_cell(yy, xx):
31    return yy * map_size + xx
32
33
34  def cell_to_xy(cell):
35    yy = cell // map_size
36    xx = cell - yy * map_size
37    return yy, xx
38
39
40  def astar(start, end):
41    def hh(cur):
42      # расчет расстояния от текущей секции до конечной
43      xy_cur = cell_to_xy(cur)
44      xy_end = cell_to_xy(end)
45      return (abs(xy_cur[0] - xy_end[0]) + abs(xy_cur[1] - xy_end[1])) * 10
46
47    def gg(cur):
48      xy_start = cell_to_xy(start)
49      xy_cur = cell_to_xy(cur)
50      return (abs(xy_cur[0] - xy_start[0]) + abs(xy_cur[1] - xy_start[1])) * 10
51
52    visited = [False] * map_len
53    G = [float('inf')] * map_len  # расст от начала до текущей
54    H = [10**3] * map_len  # расст от текущей до начала
55    F = [10**3] * map_len
56
57    G[start] = 0
58    H[start] = hh(start)
59    F[start] = G[start] + H[start]
60
61    lst = [start]
62    pairs = []
63
64    done = False
65    step = 0
66    while len(lst):
67      minn = 10**3
68      # выбираем вершину из списка lst с меньшим весом
69
70      if step == 0:
71        cells = cells_around[start]
72        cell = cells[bob_dir]
73        if cell in lst:
74          for c in range(len(cells)):
75            if cells[c] in lst:
76              #print (c, bob_dir,cells[c], (bob_dir - c) % 3 )
77              dop = (bob_dir - c) % 4
78              dop == 1 if dop == 3 else dop
79              #G[cells[c]] = dop
```

```
80              #F[cells[c]] = dop
81              F[cells[c]] = G[cells[c]] + H[cells[c]]  #* 0.2 + dop * 0.8
82          G[lst[lst.index(cell)]] = 0
83          F[lst[lst.index(cell)]] = 0
84
85      for v in range(len(lst)):
86        if F[lst[v]] < minn:
87          current = lst[v]
88          minn = F[lst[v]]
89
90      for p in range(map_len):
91        if MAP[current][p] is True and not visited[p]:
92          if G[p] > G[current] + 10:  #MAP[current][p]:
93            G[p] = G[current] + 10  #MAP[current][p]
94            H[p] = hh(p)
95            F[p] = G[p] + H[p]
96            pairs.append([p, current])
97            if p == end:
98              #print ('end:',p, end)
99              done = True
100             break
101
102         if p not in lst:
103           lst.append(p)
104     step += 1
105     if done:
106       break
107
108     visited[current] = True
109     lst.remove(current)
110
111   if len(pairs) < 2:
112     return []
113
114   prev = pairs[-1][1]
115   path_ = [pairs[-1][0], prev]
116
117   for p in range(len(pairs)):
118     for w in range(len(pairs)):
119       if pairs[w][0] == prev:
120         prev = pairs[w][1]
121         path_.append(prev)
122         break
123
124   return path_[::-1]
125
126
127 def bfs(start, end):
128   visited = [False for i in range(map_len)]
129
130   path = []
131   queue = [start]
132   step = 0
133   while len(queue) > 0:
134     p = queue.pop(0)
135
136     if step == 0:
137       cells = cells_around[p]
138       cell = cells[bob_dir]
139       if cell in lst:
```

```
        for c in range(len(cells)):
          if cells[c] in lst:
            dop = (bob_dir - c) % 4
            dop == 1 if dop == 3 else dop
            F[cells[c]] = G[cells[c]] + H[cells[c]] * 0.2 + dop * 0.8

    if not visited[p]:
      visited[p] = True
      path.append(p)

      for i in range(len(MAP)):
        if not visited[i] and MAP[p][i] is True:
          queue.append(i)
    if (p == end):
      break
  path.reverse()
  back = [path[0]]
  for p in path[1:]:
    if MAP[back[-1]][p] is True:
      back.append(p)
  back.reverse()
  return back


def map_update(cell1, cell2, clr):
  if cell1 is None or cell2 is None:
    return

  global MAP
  if MAP[cell1][cell2] in ('FFFFFF', True, False, '000000'):
    MAP[cell1][cell2] = clr
    MAP[cell2][cell1] = clr


def cell_update_walls(cell1, clr):
  cells = cells_around[cell1]
  for cell2 in cells:
    if cell2 is not None and cell2 != alice_cell_end:
      map_update(cell1, cell2, clr)


def check_add_clr(cell1, dir):
  global looked
  cell2 = cell1
  while True:
    cell2 = cells_around[cell2][dir]

    if cell2 == cell1 or cell2 is None:
      break

    if MAP[cell1][cell2] not in (True, False, 'FFFFFF'):
      looked.append(MAP[cell1][cell2])
      if len(looked) > 1 and looked[-1] == looked[-2]:
        looked.pop()
      break

    cell1 = cell2


def solution(data):
```

```
200     global map_size, map_len, X0, Y0, MAP, walls, looked, bob_dir, \
201         alice_states, alice_moves, alice_cell_end, cells_by_task, cells_around, heads
202
203     looked = []
204
205     # parsing data START
206     H, N, K, alpha = map(int, data.pop(0).split())
207
208     map_size = H // cell_size
209     map_len = map_size * map_size
210
211     MAP = [[False for _ in range(map_len)] for _ in range(map_len)]
212
213     # нумерация секторов "как в задаче"
214     cells_by_task = sum(
215         sorted([[y * map_size + x for x in range(map_size)]
216                 for y in range(map_size)],
217                reverse=True), [])
218
219     BOBs = data.pop(0).strip().split()
220     bob_dir = 'URDL'.find(BOBs.pop(0))
221     bob_Xs, bob_Ys, bob_Xe, bob_Ye = map(lambda n: int(n) // 250, BOBs)
222     alice_cell_end = -1
223
224     # перегородки
225     set_walls(data[:N])
226
227     # Alice data
228     alice_states, alice_moves = [], []
229     for i in range(K):
230       states = data[i + N].strip().split()
231       alice_states.append(list(map(int, states[:3])))
232       alice_moves.append(states[3])
233
234     cells_around = []  # cells neighbours
235     for cell in range(map_len):
236       yy, xx = cell_to_xy(cell)
237       cells = [None] * 4
238       if yy - 1 >= 0: cells[0] = xy_to_cell(yy - 1, xx)
239       if xx + 1 < map_size: cells[1] = xy_to_cell(yy, xx + 1)
240       if yy + 1 < map_size: cells[2] = xy_to_cell(yy + 1, xx)
241       if xx - 1 >= 0: cells[3] = xy_to_cell(yy, xx - 1)
242       cells_around.append(cells)
243
244       for cell2 in cells:
245         map_update(cell, cell2, True)
246
247     # parsing data END
248
249     # start
250     bob_cell_start = cells_by_task[xy_to_cell(bob_Ys, bob_Xs)]
251     bob_cell_end = cells_by_task[xy_to_cell(bob_Ye, bob_Xe)]
252     #print('BOB start, end, dir:',bob_cell_start, bob_cell_end, bob_dir)
253
254     # сектор старта Боба (открытые стенки -> черный цвет)
255     cell_update_walls(bob_cell_start, '000000')
256
257     # Alice localize
258     sensors_global = []
259     for cell in range(map_len):
```

```
260       s = [1, 1, 1, 1]  # 1 - wall, 0 - free
261       cells = cells_around[cell]
262
263       for i in range(4):
264         if cells[i] is not None and MAP[cell][cells[i]] is True:
265           s[i] = 0
266       s = [s[3]] + s[:3]
267
268       sensors_global.append([])
269       for _ in range(4):
270         sensors_global[-1].append(s[:3])
271         s.append(s.pop(0))
272
273     hypoteses = []
274     for cell in range(len(sensors_global)):
275       if alice_states[0] in sensors_global[cell] and cell != bob_cell_start:
276         for d in range(4):
277           if alice_states[0] == sensors_global[cell][d]:
278             hypoteses.append((cell, d, alice_states[0]))
279     #print (hypoteses)
280     alice_states.append([])
281
282     for h in hypoteses:
283       cell0, dir0, state0 = h
284       cell, dir = cell0, dir0
285       fired = False
286       moves = [-map_size, 1, map_size, -1]
287
288       for m in range(len(alice_moves)):
289         move = alice_moves[m]
290         if move == 'R':
291           dir = (dir + 1) % 4
292         elif move == 'L':
293           dir = (dir + 3) % 4
294         elif move == 'F':
295           cell += moves[dir]
296
297         if sensors_global[cell][dir] != alice_states[m + 1]:
298           fired = True
299           break
300
301       if not fired or m == len(alice_moves) - 1:
302         break
303
304     #print('Alice start -> end:',cell0, dir0, state0, '->', cell, dir,
       ↪  sensors_global[cell][dir])
305     alice_cell_end = cell
306
307     cell_update_walls(cell, '000000')  # "walls" around Alice's cell
308     cell_update_walls(bob_cell_start, True)  # remove Bob 'black' walls
309
310     bob_path = astar(bob_cell_start, bob_cell_end)
311     bob_cell = bob_cell_start
312
313     for step in bob_path[1:]:
314       way = step - bob_cell
315       way_dir = [-map_size, 1, map_size, -1].index(way)
316
317       if way_dir != bob_dir:
318         b_w = str(bob_dir) + str(way_dir)
```

```
319       turn_cw = False if b_w in ('03', '10', '21', '32') else True
320
321       for _ in range(3):
322         check_add_clr(bob_cell, bob_dir)
323         if turn_cw:
324           bob_dir = (bob_dir + 1) % 4
325         else:
326           bob_dir = (bob_dir + 3) % 4
327
328         if way_dir == bob_dir: break
329
330     check_add_clr(bob_cell, bob_dir)
331     bob_cell = step
332
333   return looked
334
335
336 # not work: 1, 6, 7?, 11?
337
338 data = sys.stdin.readlines()
339 print(solution(data))
```

Задача II.1.1.4. Цветные препятствия 2 (10 баллов)

Дан дифференциальный робот: Боб. У Боба есть камера, имеющая угол обзора α, направленная по ходу движения робота и находящаяся в центре робота на его оси вращения. Высота и угол установки камеры таковы, что если бы перед роботом не было препятствий, то линия горизонта проходила бы горизонтально ровно по центру кадра. Высота перегородок в лабиринте значительно выше высоты установленной камеры.

Боб находится в квадратном лабиринте с перегородками. Пол и ограждения — белые, а перегородки внутри лабиринта цветные. Перегородки или их линии образуют квадратные секции размером 250×250 мм, стороны которых параллельны стенкам.

Перечислите последовательность в которой сменяются цвета препятствий попавших в камеру робота при его вращении на угол β, относительно начального положения. Известно, что препятствие считается попавшим если оно присутствует на 0.1 части изображения. В случае, если в камере присутствует несколько перегородок, то их цвета следует выводить в направлении вращения робота.

Формат входных данных

Первая строчка содержит два целых числа и одно вещественное число через пробел – H, N, α, где:

- H — сторона поля в мм ($10^3 \leqslant H \leqslant 10^6$);
- N — количество цветных перегородок ($1 \leqslant N \leqslant 100$);
- α — угол обзора камеры в рад ($0.1 \leqslant \alpha \leqslant 0.5$).

Вторая строчка содержит одну букву, два целых числа и одно вещественное число через пробел — D, X_s, Y_s, β, где:

- D — начальная ориентация Боба ($D \in \{U, D, L, R\}$, где U - по направлению оси Y, D - против направления оси Y, L - против направления оси X, R - по

направлению оси X)

- X_s — начальная координата X_s Боба в мм ($0 \leqslant X_s \leqslant H$, $X_s = 125+250a$, $a \in \mathbb{N}$);
- Y_s — начальная координата Y_s Боба в мм ($0 \leqslant Y_s \leqslant H$, $Y_s = 125 + 250a$, $a \in \mathbb{N}$);
- β — угол на который необходимо повернуться Бобу, в рад ($-2\pi \leqslant \beta \leqslant 2\pi$, при $\beta > 0$ – вращение происходит против часовой стрелки).

Далее идут N строк, содержащие 4 целых числа и код цвета через пробел — $X_{i,1}$, $Y_{i,1}$, $X_{i,2}$, $Y_{i,2}$, Col_i, где:

- $X_{i,1}$ — координата $X_{i,1}$ начальной точки перегородки в мм ($0 \leqslant X_{i,1} \leqslant H$, $X_{i,1} = 250a$, $a \in \mathbb{N}$);
- $Y_{i,1}$ — координата $Y_{i,1}$ начальной точки перегородки в мм ($0 \leqslant Y_{i,1} \leqslant H$, $Y_{i,1} = 250a$, $a \in \mathbb{N}$);
- $X_{i,2}$ — координата $X_{i,2}$ конечной точки перегородки в мм ($0 \leqslant X_{i,2} \leqslant H$, $X_{i,2} = 250a$, $a \in \mathbb{N}$);
- $Y_{i,2}$ — координата $Y_{i,2}$ конечной точки перегородки в мм ($0 \leqslant Y_{i,2} \leqslant H$, $Y_{i,2} = 250a$, $a \in \mathbb{N}$);
- Col_i — код цвета данной перегородки в шестнадцатеричной системе счисления ($000000 \leqslant Col_i < FFFFFF$);

Формат выходных данных

Одна строка, содержащая последовательность кодов цветов (исключая белый), которые будут менять друг друга в камере Боба во время его вращения, через пробел.

Примеры

Примеры входных данных и ответов к ним можно найти по данной ссылке.

Решение

Начнем решение задачи с формирования модели лабиринта, которая будет включать в себя координаты перегородок, стенок и их цвета (по условиям задачи все стены и пол - белого цвета). По сути, это отрезки с известными координатами.

Нам также известны:

1. Координаты установки робота и его направление
2. Угол, на который должен повернуться робот и направление вращения
3. Угол обзора камеры и

Для примера возьмем набор данных N1. Робот должен повернуться против часовой стрелки на 1.22 радиана (почти 70°). Угол обзора камеры - 0.01 радиана. С некоторой дискретностью, например, в 3° начинаем “вращаться” - проверять пересечение отрезков (“луча камеры” робота и стенок лабиринта). Не учитываем повторяющие друг за другом цвета и белый цвет. См. рис. II.1.12.

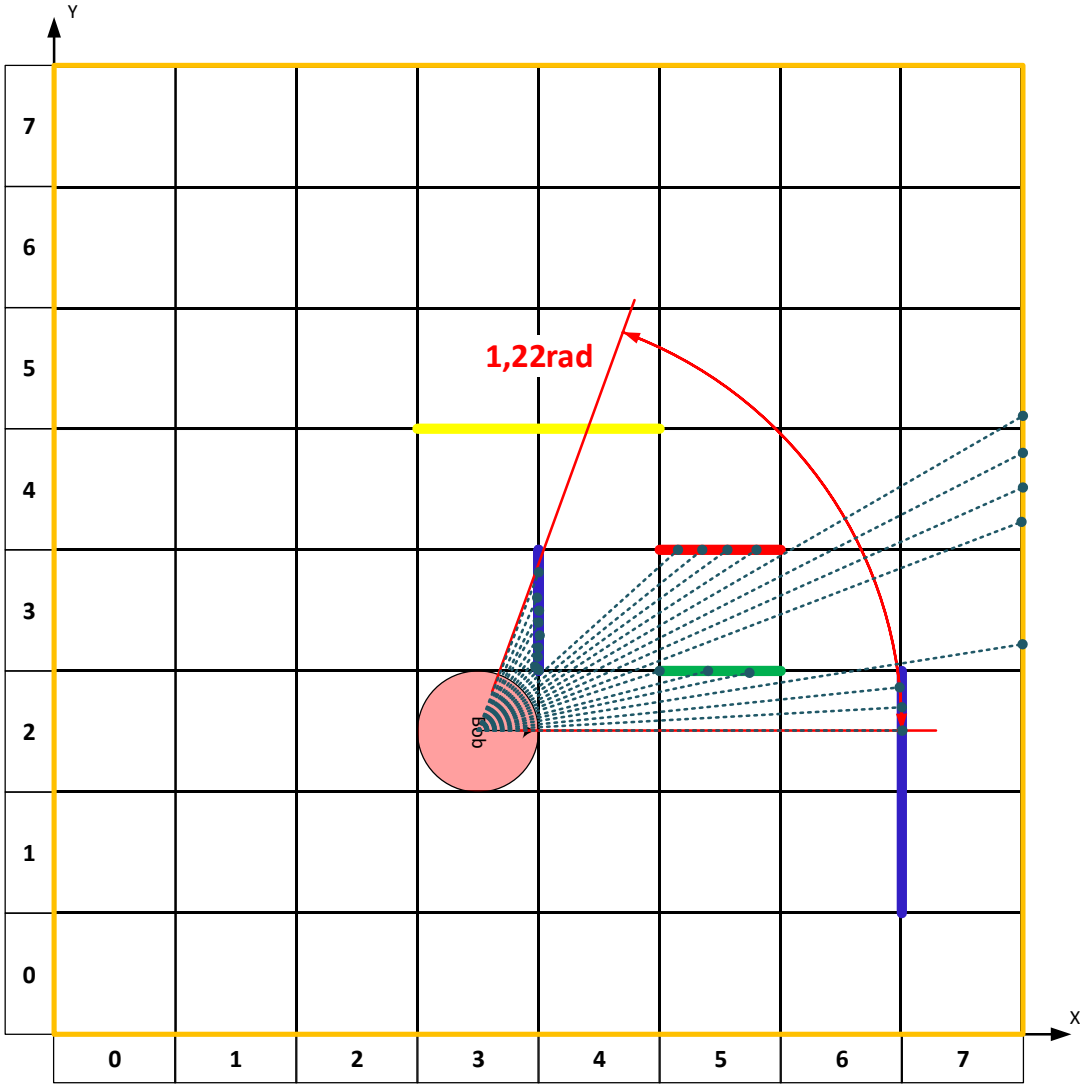


Рис. II.1.12: Пример сканирования стенок при повороте робота против часовой стрелки

В качестве ответа выводим список “замеченных” камерой цветов через пробел. В нашем примере ответ получается: $0000FF$ $00FF00$ $FF0000$ $0000FF$ (синий, зеленый, красный, синий).

С более широким углом камеры проверяем пересечение луча с ближайшими стенками внутри зоны видимости камеры, при этом не забываем “поворачивать” камеру в заданном направлении до заданного угла (рис. II.1.13)

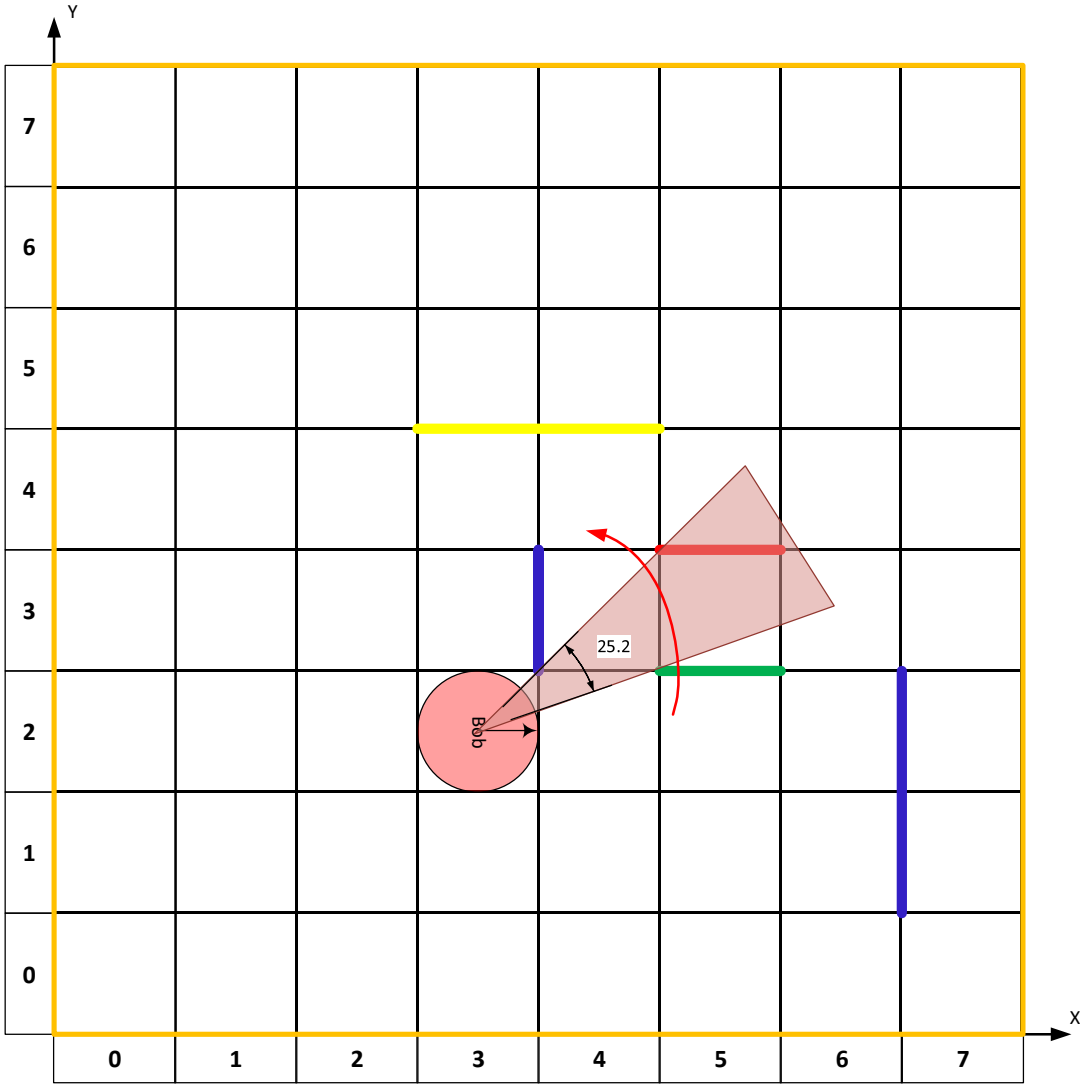


Рис. II.1.13: Пример зоны видимости камеры с углом обзора 25.2° (0.43 радиан)

Ниже представлено решение на языке Python 3

```
import numpy as np
from enum import IntEnum
import math


class Dir(IntEnum):
  LEFT = 0
  UP = 1
  RIGHT = 2
  DOWN = 3


def det(a, b):
  return a[0] * b[1] - a[1] * b[0]


```

```
17  def line_intersection(line1, line2):
18    xdiff = (line1[0][0] - line1[1][0], line2[0][0] - line2[1][0])
19    ydiff = (line1[0][1] - line1[1][1], line2[0][1] - line2[1][1])
20    div = det(xdiff, ydiff)
21    if div == 0:
22      return False
23    ua = ((line2[1][0] - line2[0][0]) * (line1[0][1] - line2[0][1]) -
24          (line2[1][1] - line2[0][1]) * (line1[0][0] - line2[0][0])) / div
25    ub = ((line1[1][0] - line1[0][0]) * (line1[0][1] - line2[0][1]) -
26          (line1[1][1] - line1[0][1]) * (line1[0][0] - line2[0][0])) / div
27
28    if ua < 0 or ua > 1 or ub < 0 or ub > 1:
29      return False
30
31    d = (det(*line1), det(*line2))
32    x = det(d, xdiff) / div
33    y = det(d, ydiff) / div
34    return x, y
35
36
37  def raycast(point1, point2, lines, skip=False):
38    line1 = (point1, point2)
39    intersections = []
40    for curline in range(len(lines)):
41      point3 = (lines[curline][0], lines[curline][1])
42      point4 = (lines[curline][2], lines[curline][3])
43      line2 = (point3, point4)
44      intersectresult = line_intersection(line1, line2)
45      if intersectresult:
46        intersections.append((intersectresult, lines[curline][4], curline))
47    if len(intersections) > 1:
48      intersections.sort(key=lambda x: np.sqrt((point1[0] - x[0][0]) * (point1[
49          0] - x[0][0]) + (point1[1] - x[0][1]) * (point1[1] - x[0][1])))
50    if skip:
51      if len(intersections) > 1:
52        return intersections[1]
53      else:
54        return False
55    else:
56      if len(intersections):
57        return intersections[0]
58      else:
59        return False
60
61
62  def coord_line(dim: int, image, point1: tuple, point2: tuple, color: tuple,
63                 thickness: int):
64    return cv2.line(
65        image, (int(point1[0] * 50) + 25, dim * 50 - 25 - int(point1[1] * 50)),
66        (int(point2[0] * 50) + 25, dim * 50 - 25 - int(point2[1] * 50)), color,
67        thickness)
68
69
70  def hextorgb(hex):
71    return tuple(int(hex[i:i + 2], 16) for i in (0, 2, 4))[::-1]
72
73
74  def vectorlen(vector):
75    return np.sqrt(vector[0] * vector[0] + vector[1] * vector[1])
76
```

```
77
78  def localvector(basevec, globalvec):
79    return [globalvec[0] - basevec[0], globalvec[1] - basevec[1]]
80
81
82  def globalvector(basevec, localvec):
83    return [basevec[0] + localvec[0], basevec[1] + localvec[1]]
84
85
86  def normvector(vector):
87    return [vector[0] / vectorlen(vector), vector[1] / vectorlen(vector)]
88
89
90  def rotate(x, y, xo, yo, theta):  # rotate x,y around xo,yo by theta (rad)
91    xr = math.cos(theta) * (x - xo) - math.sin(theta) * (y - yo) + xo
92    yr = math.sin(theta) * (x - xo) + math.cos(theta) * (y - yo) + yo
93    return [xr, yr]
94
95
96  def anglesort(vector, angle):
97    newvec = rotate(vector[0], vector[1], 0, 0, angle)
98    atn = math.atan2(newvec[1], newvec[0])
99    if atn < 0:
100     atn = 2 * math.pi - abs(atn)
101   return atn
102
103
104 def angbetween(start, end, mid):
105   end = end - start + math.pi * 2 if end - start < 0 else end - start
106   mid = mid - start + math.pi * 2 if mid - start < 0 else mid - start
107   return mid < end
108
109
110 def main():
111   inputdata = list(map(float, input().split()))
112   H = int(inputdata[0])
113   N = int(inputdata[1])
114   ALPHA = inputdata[2]
115   H //= 250
116   dirdict = {"L": Dir.LEFT, "U": Dir.UP, "R": Dir.RIGHT, "D": Dir.DOWN}
117
118   inputdata = input().split()
119   D = dirdict[inputdata[0]]
120   Xs = int(inputdata[1]) / 250 - 0.5
121   Ys = int(inputdata[2]) / 250 - 0.5
122   BETA = float(inputdata[3])
123   del inputdata
124
125   borders = []
126   for i in range(N):
127     data = input().split()
128     data[:4] = list(map(lambda x: int(x) / 250 - 0.5, data[:4]))
129     data[0], data[1], data[2], data[3] = min(data[0], data[2]), min(
130         data[1], data[3]), max(data[0], data[2]), max(data[1], data[3])
131     borders.append(data)
132
133   visiblepoints = []
134   for border in borders:
135     border_xlen = border[2] - border[0]
136     border_ylen = border[3] - border[1]
```

```
    borderpointc = [(border[0] + border[2]) / 2, (border[1] + border[3]) / 2]
    borderpoint1 = [border.copy()[0], border.copy()[1]]
    borderpoint2 = [border.copy()[2], border.copy()[3]]
    borderpoint3 = [border.copy()[0], border.copy()[1]]
    borderpoint4 = [border.copy()[2], border.copy()[3]]
    if border_xlen:
      borderpoint1[0] += 0.001
      borderpoint2[0] -= 0.001
      borderpoint3[0] -= 0.001
      borderpoint4[0] += 0.001
    if border_ylen:
      borderpoint1[1] += 0.001
      borderpoint2[1] -= 0.001
      borderpoint3[1] -= 0.001
      borderpoint4[1] += 0.001
    locvec1 = localvector((Xs, Ys), borderpoint3)
    locvec1[0] *= 100
    locvec1[1] *= 100
    raycastvec1 = globalvector((Xs, Ys), locvec1)
    locvec2 = localvector((Xs, Ys), borderpoint4)
    locvec2[0] *= 100
    locvec2[1] *= 100
    raycastvec2 = globalvector((Xs, Ys), locvec2)

    res = raycast((Xs, Ys), borderpoint1, borders)
    if res:
      image = coord_line(H, image, (Xs, Ys), (res[0][0], res[0][1]),
                         hextorgb(res[1]), 1)
      print(res)
      visiblepoints.append(res)

    res = raycast((Xs, Ys), borderpoint2, borders)
    if res:
      image = coord_line(H, image, (Xs, Ys), (res[0][0], res[0][1]),
                         hextorgb(res[1]), 1)
      print(res)
      visiblepoints.append(res)

    res = raycast((Xs, Ys), raycastvec1, borders)
    if res:
      image = coord_line(H, image, (Xs, Ys), (res[0][0], res[0][1]),
                         hextorgb(res[1]), 1)
      print(res)
      visiblepoints.append(res)

    res = raycast((Xs, Ys), raycastvec2, borders)
    if res:
      image = coord_line(H, image, (Xs, Ys), (res[0][0], res[0][1]),
                         hextorgb(res[1]), 1)
      print(res)
      visiblepoints.append(res)

    res = raycast((Xs, Ys), borderpointc, borders)

  dirarr = [math.pi, math.pi / 2, 0, -math.pi / 2]

  leftbound = ALPHA / 2
  rightbound = -ALPHA / 2
  if D == Dir.LEFT:
    leftbound += math.pi
```

```
197       rightbound += math.pi
198     elif D == Dir.UP:
199       leftbound += math.pi / 2
200       rightbound += math.pi / 2
201     elif D == Dir.RIGHT:
202       leftbound += 0
203       rightbound += 0
204     elif D == Dir.DOWN:
205       leftbound += math.pi / 2 * 3
206       rightbound += math.pi / 2 * 3
207 
208     if BETA > 0:
209       leftbound += BETA
210     else:
211       rightbound -= BETA * -1
212 
213     if leftbound > rightbound:
214       pass
215     else:
216       rightbound, leftbound = leftbound, rightbound
217     endray = []
218     if BETA > 0:
219       endray = raycast(
220           (Xs, Ys),
221           globalvector((Xs, Ys), (math.cos(leftbound - 0.001) * 10000,
222                                   math.sin(leftbound - 0.001) * 10000)), borders)
223     elif BETA < 0:
224       endray = raycast((Xs, Ys),
225                        globalvector((Xs, Ys),
226                                     (math.cos(rightbound + 0.001) * 10000,
227                                      math.sin(rightbound + 0.001) * 10000)),
228                        borders)
229     if endray:
230       pass
231       endray = list(endray)
232       endray.append("E")
233       visiblepoints.append(endray)
234     else:
235       if BETA > 0:
236         angi = leftbound - 0.001
237         while angi > rightbound + 0.001:
238           endray = raycast(
239               (Xs, Ys),
240               globalvector((Xs, Ys),
241                            (math.cos(angi) * 10000, math.sin(angi) * 10000)),
242               borders)
243           if endray:
244             endray = list(endray)
245             endray.append("E")
246             visiblepoints.append(endray)
247             break
248           else:
249             angi -= 0.001
250       elif BETA < 0:
251         angi = rightbound + 0.001
252         while angi < leftbound - 0.001:
253           endray = raycast(
254               (Xs, Ys),
255               globalvector((Xs, Ys),
256                            (math.cos(angi) * 10000, math.sin(angi) * 10000)),
```

```
257             borders)
258         if endray:
259           endray = list(endray)
260           endray.append("E")
261           visiblepoints.append(endray)
262           break
263         else:
264           angi += 0.001
265 
266   visiblepoints.sort(key=lambda x: anglesort(localvector(
267       (Xs, Ys), x[0]), -dirarr[D] + ALPHA / 2))
268   if BETA < 0:
269     visiblepoints.reverse()
270   startray = []
271   if BETA > 0:
272     startray = raycast((Xs, Ys),
273                        globalvector((Xs, Ys),
274                                     (math.cos(rightbound + 0.001) * 10000,
275                                      math.sin(rightbound + 0.001) * 10000)),
276                        borders)
277   elif BETA < 0:
278     startray = raycast(
279         (Xs, Ys),
280         globalvector((Xs, Ys), (math.cos(leftbound - 0.001) * 10000,
281                                 math.sin(leftbound - 0.001) * 10000)), borders)
282   if startray:
283     pass
284     visiblepoints.insert(0, startray)
285   i = 0
286   vecs = []
287 
288   markremoval = []
289   i = 0
290   for i in range(len(visiblepoints)):
291     if visiblepoints[i][-1] == "E":
292       locvec1 = ()
293       if BETA > 0:
294         locvec1 = (math.cos(rightbound), math.sin(rightbound))
295       else:
296         locvec1 = (math.cos(leftbound), math.sin(leftbound))
297       locvec2 = localvector((Xs, Ys), visiblepoints[i][0])
298       normvec1 = normvector(locvec1)
299       normvec2 = normvector(locvec2)
300       anglebetween = np.arccos(np.dot(normvec1, normvec2))
301       if abs(
302           BETA) <= math.pi or abs(BETA) > math.pi and anglebetween > ALPHA * 2:
303         visiblepoints = visiblepoints[:i + 1]
304       break
305   i = 0
306   while i < len(visiblepoints) - 1:
307     locvec1 = localvector((Xs, Ys), visiblepoints[i][0])
308     locvec2 = localvector((Xs, Ys), visiblepoints[i + 1][0])
309     normvec1 = normvector(locvec1)
310     normvec2 = normvector(locvec2)
311     anglebetween = np.arccos(np.dot(normvec1, normvec2))
312     angle1 = math.atan2(normvec1[1], normvec1[0])
313     angle2 = math.atan2(normvec2[1], normvec2[0])
314     locvec3 = None
315     normvec3 = None
316     angle3 = None
```

```
317       if i < len(visiblepoints) - 2:
318         locvec3 = localvector((Xs, Ys), visiblepoints[i + 2][0])
319         normvec3 = normvector(locvec3)
320         angle3 = math.atan2(normvec3[1], normvec3[0])
321       if i < len(visiblepoints) - 2 and visiblepoints[i][1] == visiblepoints[
322           i + 1][1] and visiblepoints[i][1] == visiblepoints[i + 2][1] and (
323               visiblepoints[i][2] == visiblepoints[i + 1][2]
324               or abs(angle2 - angle1) < 0.005) and (
325                   visiblepoints[i + 1][2] == visiblepoints[i + 2][2]
326                   or abs(angle3 - angle2) < 0.005):
327         markremoval.append(i + 1)
328         i += 1
329       else:
330         i += 1
331     for i in markremoval:
332       visiblepoints[i] = 0
333     i = 0
334     while i < len(visiblepoints):
335       if visiblepoints[i] == 0:
336         visiblepoints.pop(i)
337       else:
338         i += 1
339     for i in range(len(visiblepoints) - 1):
340       if visiblepoints[i][1] == visiblepoints[i + 1][1]:
341         locvec1 = localvector((Xs, Ys), visiblepoints[i][0])
342         locvec2 = localvector((Xs, Ys), visiblepoints[i + 1][0])
343         normvec1 = normvector(locvec1)
344         normvec2 = normvector(locvec2)
345         anglebetween = np.arccos(np.dot(normvec1, normvec2))
346         angle1 = math.atan2(normvec1[1], normvec1[0])
347         angle2 = math.atan2(normvec2[1], normvec2[0])
348         color = visiblepoints[i][1]
349         if anglebetween > ALPHA / 10:
350           vecs.append([angle1, angle2, color, anglebetween])
351 
352     colors = []
353     for i in vecs:
354       if not colors or colors[-1] != i[2]:
355         colors.append(i[2])
356     if len(colors) > 1:
357       print(' '.join(colors))
358     else:
359       if len(colors):
360         print(colors[0])
361 
362 
363 if __name__ == "__main__":
364     main()
```

Задача II.1.1.5. Калибровка датчика (10 баллов)

На робототехническом устройстве имеется датчик, который необходимо откалибровать.

Калибровка происходит следующим образом:

1. Имеется N точных значений данной величины y_i;
2. Имеется N измерений той же величины датчиком $\hat{y}_i$ в такой же последовательности;

3. $i \in [1, N]$.

Необходимо откалибровать датчик так, чтобы он давал удовлетворительное (в пределах ± 5) относительное расхождение между истинным значением измеряемой величины и показаниями калиброванного датчика в M промежуточных точках.

Для калибровки можно использовать различные методы интерполяции (сплайны).

Рассмотрим две из них:

1. Кусочно-линейная интерполяция — на каждом отрезке функция приближается линейной, других условий не требуется;
2. Гладкая кусочно-кубическая интерполяция — на каждом отрезке функция приближатся кубическим многочленом, дополнительно требуется непрерывность первой и второй производных функции на всём отрезке.

Кусочно-линейная интерполяция:

$$U(x) = s_i(x); \tag{II.1.6}$$

$$x \in [x_{i-1}, x_i]; \tag{II.1.7}$$

$$s_i(x) = u(x_{i-1})\frac{x_i - x}{x_i - x_{i-1}} + u(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}} \tag{II.1.8}$$

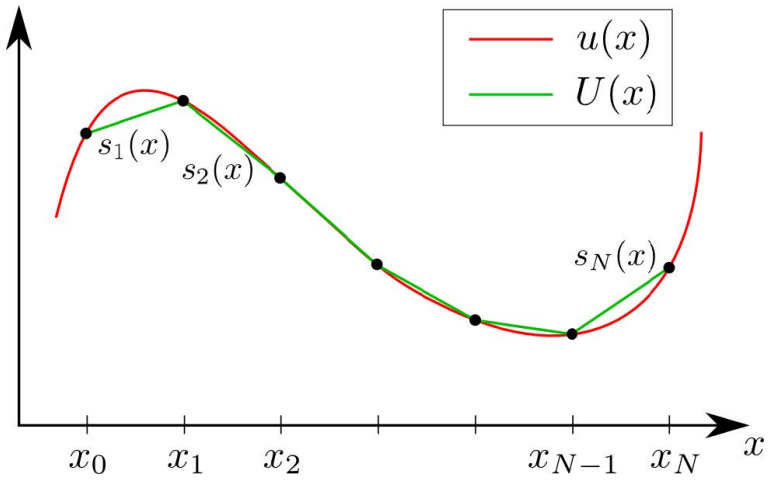


Рис. II.1.14: Кусочно-линейная интерполяция

Гладкая кусочно-кубическая интерполяция:

$$U(x) = s_i(x); \tag{II.1.9}$$

$$x \in [x_{i-1}, x_i]; \tag{II.1.10}$$

$$s_{i-1}(x_{i-1}) = s_i(xi - 1); \tag{II.1.11}$$

$$s_i(x_i) = s_{i+1}(xi); \tag{II.1.12}$$

$$s'_{i-1}(x_{i-1}) = s'_i(xi - 1); \tag{II.1.13}$$

$$s'_i(x_i) = s'_{i+1}(xi); \tag{II.1.14}$$

$$s''_{i-1}(x_{i-1}) = s''_i(xi - 1); \tag{II.1.15}$$

$$s''_i(x_i) = s''_{i+1}(xi); \tag{II.1.16}$$

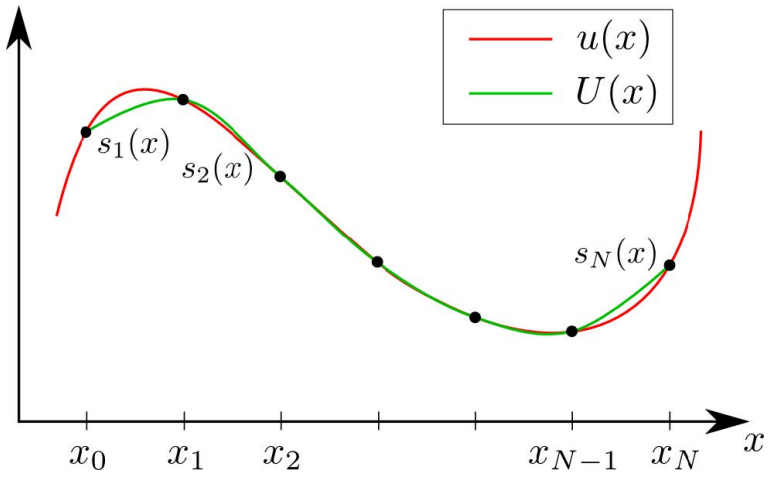


Рис. II.1.15: Кусочно-линейная интерполяция

Формат входных данных

Первая строчка содержит два целых числа через пробел – N, M, где:

- N — количество проведенных тестовых измерений ($1 \leqslant N \leqslant 100$);
- M — количество промежуточных точек ($1 \leqslant M < N$).

Вторая строчка содержит N вещественных чисел через пробел — точные значения y_i измеряемой величины ($-10000 \leqslant y_i \leqslant 10000$).

Третья строчка содержит N вещественных чисел через пробел — калибровочные измерения проведенные датчиком $\hat{y}_i$ ($-10000 \leqslant \hat{y}_i \leqslant 10000$).

Последняя строчка содержит M вещественных чисел через пробел — измерения проведенные датчиком $\hat{x_j}$ ($min(\hat{y}_i) \leqslant \hat{x_j} \leqslant max(\hat{y}_i)$).

Формат выходных данных

Одна строка, содержащая M вещественных чисел через пробел — откалиброванные измерения, проведенные датчиком x_i.

Примеры

Примеры входных данных и ответов к ним можно найти по данной ссылке.

Решение

Рассмотрим пример решения задачи методом кусочно-линейной интерполяции (далее - КЛИ).

Интерполяция - метод нахождения приближенной или точной величины по известным отдельным значениям этой величины (или можно сказать - восстановление функции по нескольким известным величинам).

Кусочно-линейная интерполяция относится к локальному методу интерполяции (когда на каждом интервале $[x_{i-1}, x_i]$ строим свою 'локальную' функцию).

По условиям задачи нам дается три набора данных:

1. Точные значения данной величины (*precision*)
2. Калибровочные измерения, проведенные датчиком (*calibrate*)
3. Измерения, проведенные датчиком (*values*)

На каждом отрезке заменяем функцию линией $F_i(x) = k_i x + b_i$. Для этого нам надо найти коэффициенты k_i и b_i. Условия кусочно-линейной интерполяции: $F_i(x_{i-1}) = f_{i-1}$ и $F_i(x_i) = f_i$, где f - локальные функции, где $k_i x_{i-1} + b_i = f_{i-1}$ и $k_i x_i + b_i = f_i$. Исходя из этого, находим коэффициенты по формулам II.1.17:

$$k_i = \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \qquad b_i = f_i - k_i x_i \tag{II.1.17}$$

Далее находим, в какой интервал (x_{i-1}, x_i) попадает искомая точка x. Вычисляем коэффициенты k_i и b_i и находим значение $F(x)$

Для примера возьмем набор данных N3 и найдем $F(x)$ для значения 8.289400292 $(VALUE)$, это второй элемент в массиве *values*.

Сначала находим значения по оси X в интервале которых находится значение $VALUE$. На графике II.1.16 видно, что значение $VALUE$ находится между 7.917250.. и 8.326198... Для точного результата нужно проверять значения по всем данным (на примере график показан по первым 20 значениям).

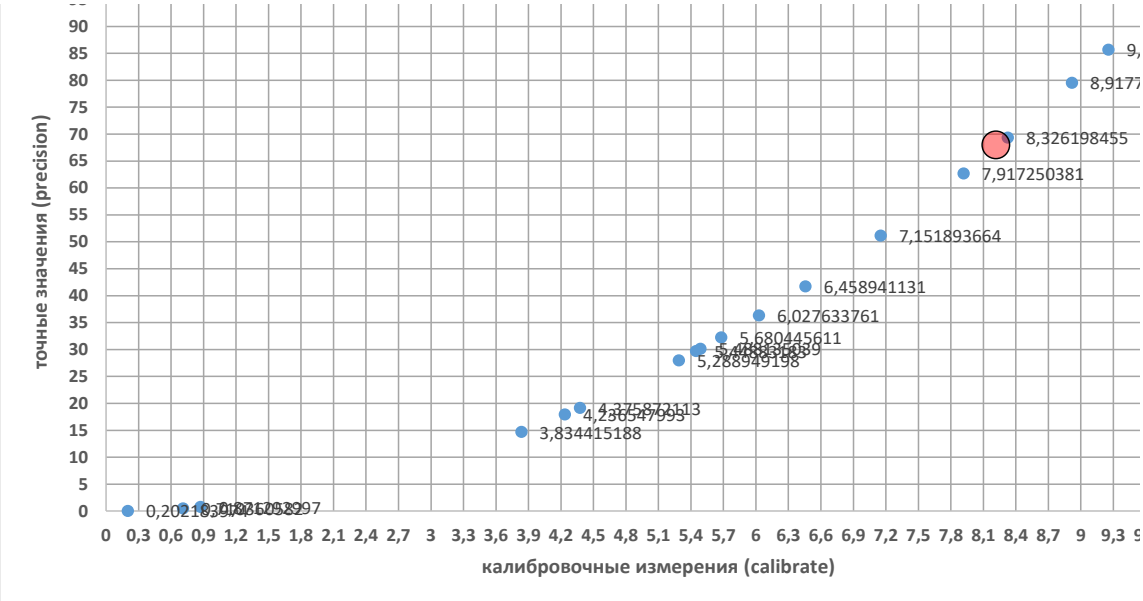


Рис. II.1.16: График по первым 20 значениям (пример)

Во время определения ближайших чисел необходимо запоминать их порядковые индексы в общем массиве *calibrate*. Далее по этим индексам мы значения из массива точных значений *precision*.

В нашем случае, это индексы [10] и [17] и, соответственно, значения из массива *precision* - 62.68285359 и 69.32558072 (см. рис. II.1.17)

idx	precision	calibrate	values
0	30,119626	5,488135	**8,2894003**
1	51,149583	7,1518937	0,0469548
2	36,332369	6,0276338	6,7781654
3	29,689768	5,4488318	2,7000797
4	17,948339	4,236548	7,3519402
5	41,717921	6,4589411	9,6218855
6	19,148257	4,3758721	2,4875314
7	79,525908	8,91773	5,7615733
8	92,864592	9,6366276	5,9204193
9	14,70274	3,8344152	5,7225191
10	**62,682854**	**7,9172504**	2,2308163
11	27,972984	5,2889492	9,5274901
12	32,267462	5,6804456	4,4712538
13	85,672914	9,2559664	8,4640867
14	0,5046122	0,7103606	6,9947928
15	0,7591515	0,871293	2,9743695
16	0,0408784	0,202184	8,1379782
17	**69,325581**	**8,3261985**	3,9650574
18	60,552793	7,7815675	8,811032
19	75,692114	8,7001215	5,8127287

Рис. II.1.17: Значения массивов данных с индексами

Теперь мы можем вычислить значение y для нашего $VALUE$:

$x_1 = 7.91725, y_1 = 62.68285 \qquad x_2 = 8.32619, y_2 = 69.32558$

$$k = \frac{y_2 - y_1}{x_2 - x_1} = \frac{69.32558 - 62.68285}{8.32619 - 7.91725} = 16.24$$

$$b = y_1 - x_1\dot{k} = 62.68285 - 7.91725 \cdot 16.24 = 62.9206$$

$$y = kx + b = 16.24 \cdot 8.289400292 + 62.9206 = 68.699$$

Результат вычисления показан на рис. II.1.18:

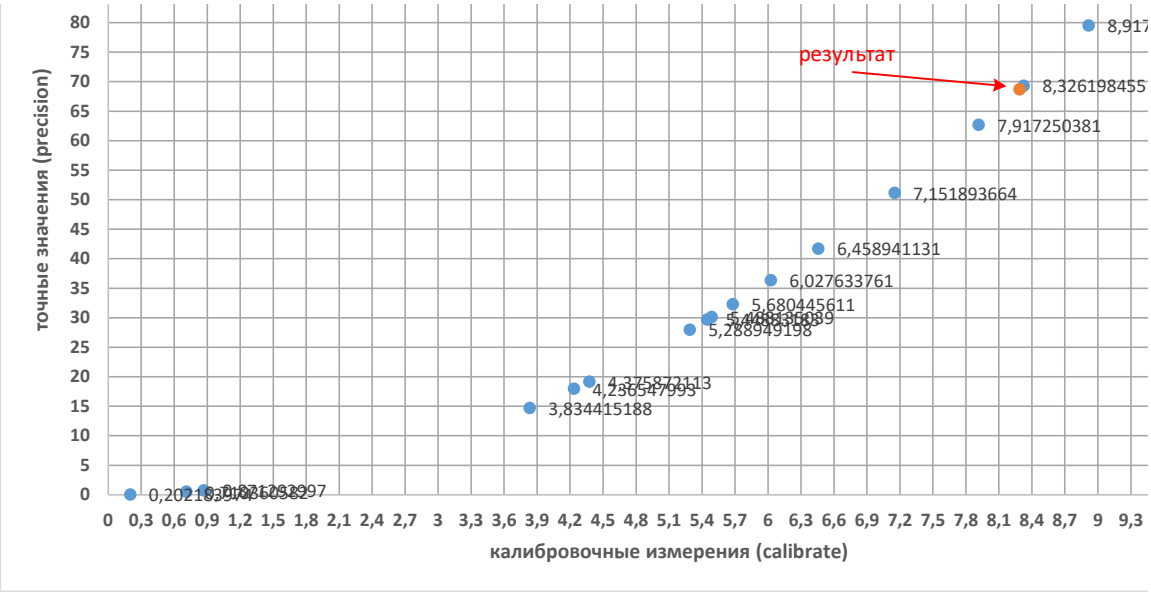


Рис. II.1.18: Результат вычисления интерполяции F(8.2894)

Аналогичным образом обрабатываем все значения из массива *values* и в качестве результата выводим полученные значения через пробел.

Пример программы-решения

Ниже представлено решение на языке Python 3

```
# -*- coding: utf-8 -*-
import sys


def solution(data):
  def getIdx(num):
    for i in range(1, len(dictX)):
      x1, x2 = dictX[i - 1], dictX[i]
      if x2[1] >= num >= x1[1]:
        return int(x1[0]), int(x2[0])

  n, m = map(int, data[0].strip().split())
  precision = list(map(float, data[1].strip().split()))
  calibrate = list(map(float, data[2].strip().split()))
  values = list(map(float, data[3].strip().split()))

  res = []

  for X in values:
    idx1, idx2 = -1, -1
    for i in range(len(calibrate)):
      if idx1 == -1 or idx2 == -1 or abs(calibrate[idx1] -
                                         X) > abs(calibrate[i] - X):
        idx2 = idx1
        idx1 = i

#  print (X, idx1, idx2)
```

```
28      x1, x2 = calibrate[idx2], calibrate[idx1]
29      y1, y2 = precision[idx2], precision[idx1]
30      a = (y2 - y1) / (x2 - x1)
31      b = y1 - x1 * a
32      y = a * X + b
33      res.append(y)
34    return res
35
36  data = sys.stdin.readlines()
37  print(*solution(data))
```

Задача II.1.1.6. Измерение расстояний (5 баллов)

Перед запуском робот, оснащенный двумя инфракрасными датчиками расстояния, установлен около стены так, что она находится слева. Один датчик расстояния направлен на стену и смонтирован так, что край левого колеса и передняя кромка датчика находятся на одной прямой. Второй датчик расстояния направлен по ходу движения робота, передняя кромка датчика не выходит за передний край корпуса робота.

В стене есть впадины разной глубины. Напишите программу для поиска самой глубокой впадины и заезда в нее. Гарантируется, что существует единственное решение. После заезда во впадину, робот должен остановиться и вывести на экран слово «finish».

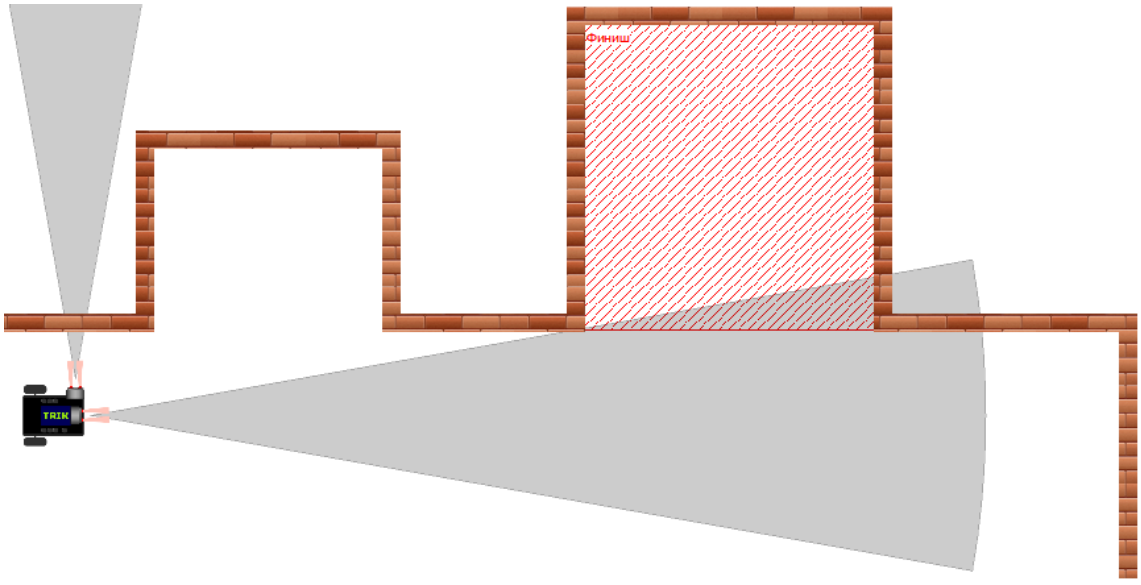


Рис. II.1.19: Внешний вид одного из вариантов поля

Конфигурация робота

Подключение моторов:

- Левый мотор — порт M3
- Правый мотор — порт M4

Подключение датчиков:

- Датчик расстояния, направленный вперед - порт D1

- Датчик расстояния, направленный влево - порт D2

Комментарии

Количество впадин может изменяться от 2 до 8. Расстояние между двумя соседними впадинами не будет меньше 525 мм. Ширина впадины - от 700 до 1050 мм. Глубина впадин может изменяться от 350 до 1000 мм, относительно передней кромки стены. От дальнего (относительно движения робота) края самой последней впадины на расстоянии 700 мм установлено препятствие.

На старте робот установлен так, что его датчик не направлен на впадину, и все впадины находятся по ходу движения робота.

Первоначальное расстояние от робота до стены может быть в диапазоне от 175 до 525 мм.

Будет считаться, что робот заехал во впадину, если самая крайняя часть робота не выступает за кромку стены.

Необходимо загружать js файл с программой, написанной в TRIK Studio 2019.7.

Примеры

Примеры тестовых полигонов представленны по данной ссылке.

Решение

Декомпозиция решения задачи:

1. Двигаться прямолинейно до стены спереди и считывать данные:
   1.1. Глубину каждой впадины
   1.2. Ширину каждой впадины
2. Определить впадину с максимальной глубиной
3. Развернуться и доехать до середины выбранной впадины
4. Повернуть и заехать в зону впадины

Прямолинейное движение и точные повороты роботов на дифференциальной платформе являются базовыми алгоритмами и данном решении не будут подробно описываться.

В случае решения задачи в симуляторе TRIK Studio для прямолинейного движения и точных поворотов лучше использовать пропорциональный регулятор с управлением по гироскопическому датчику.

Для примера возьмем карту N11 из набора данных.

Перед началом движения робот запоминает расстояние до стены слева. Это будет являться для него “0-уровнем” отметки ($wall$). После начала движения ждем увеличения расстояния слева на бОльшее, чем $wall$. При увеличении расстояния запоминаем значения энкодеров в данной точке (V_{ix0}). При считывании глубины впадины нужно учитывать расстояние до стены:

$$depth_i = sensorDistance - wall$$

где **sensorDistance** - показания датчика расстояния, а **wall** - расстояние до стены при старте

Робот продолжает движение и ждет (слева по ходу движения) расстояния, равное *wall* с погрешностью 1..2 см. После чего снова запоминаем значение энкодеров в данной точке (V_{ix1}).

Повторяем операции для каждой впадины, пока робот не доедет до стены спереди (когда расстояние датчика спереди не будет меньше, чем 15 см).

Когда робот доезжает до стены спереди по ходу движения - останавливаем робота.

На рис. II.1.20 показаны данные, которые у нас есть на момент остановки робота.

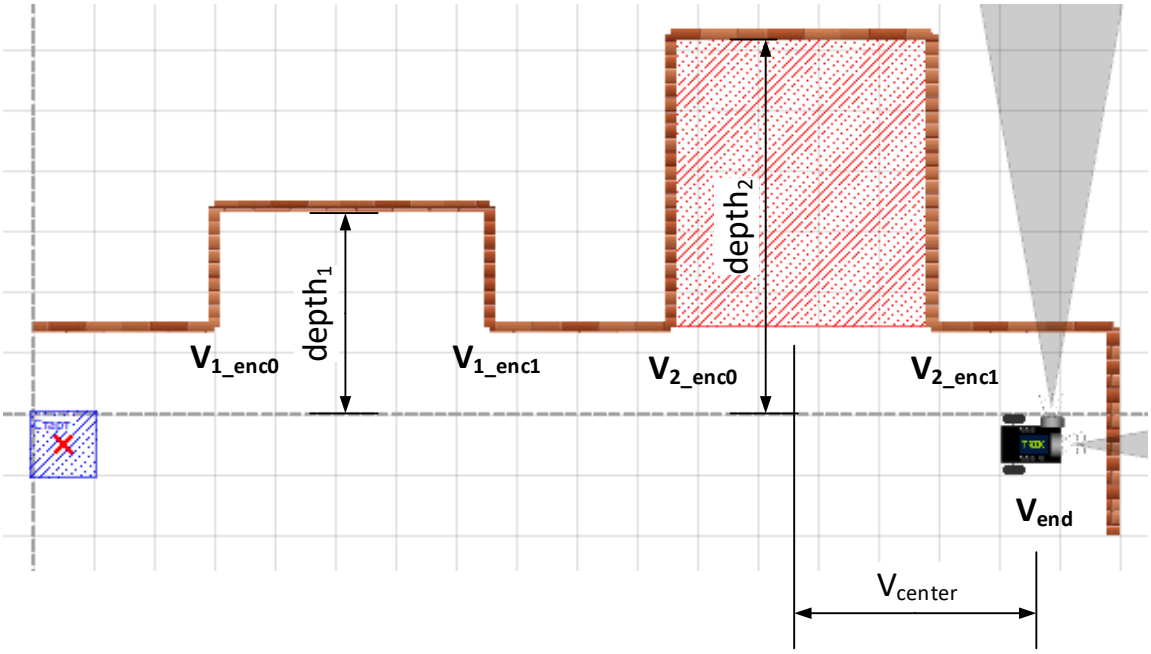


Рис. II.1.20: Полученные данные на момент остановки робота

Определяем впадину с максимальной глубиной, поочередно сравнивая все значения $depth_i$.

Определяем расстояние до центра выбранной впадины в “считываниях энкодера (тиках)” (V_{center}) по формуле:

$$V_{center} = V_{end} - \frac{V_{ix0} + V_{ix1}}{2}$$

После этого робот поворачивается на 180° и движется прямо на расстояние, равное V_{center} тиков энкодера относительно текущих значений энкодеров.

Робот доезжает до середины впадины. Разворачивается вправо на 90°. Едет прямо пока расстояние спереди не станет меньше 15 см. Выводит на экран надпись **“finish”**.

Пример программы-решения

Ниже представлено решение на языке JavaScript

```
function motors(vL, vR){
        vL = vL || 90, vR = vR || vL
        brick.motor('M4').setPower(vL)
        brick.motor('M3').setPower(vR)
}


function turnGyro(angle){
        angle *= 1000
        motors(40, -40)
        while (angle - yaw() > 2000) script.wait(20)
        motors(0)
}


function goGyroCpr(goal, gyro0){
        gyro0 *= 1000
        goal += eL()
        while (eL() < goal){
                var y = yaw()
                u = ((gyro0 - abs(y)) * sign(y))/1000 * 0.7
                motors(80 + u, 80 - u)
                script.wait(20)
        }
        motors(0)
}


function goFront(gyro0, getData){
        gyro0 *= 1000
        while (sFront() > 20){
                if (getData){
                        raw.push(sLeft())
                        encs.push(eL())
                }
                u = (gyro0 - yaw())/1000 * 0.7
                motors(80 + u, 80 - u)
                script.wait(20)
        }
}

//
function printDebug(info,txt){txt = txt|| ''; if (debug) print (txt, info)}
function yaw(){ return brick.gyroscope().read()[6]}
function sign(num){ return num > 0 ? 1 : -1}
function inRange(num, range){return num >= range[0] && num <= range[1]}

sFront = brick.sensor('D1').read
sLeft  = brick.sensor('D2').read
eL = brick.encoder('E4').read
eR = brick.encoder('E3').read
wait = script.wait
abs = Math.abs

brick.gyroscope().calibrate(2000)
wait(2100)
```

```
57
58  raw = [], encs = []
59  wall = sLeft()
60
61  goFront(0, true)
62
63  max = Math.max.apply(null, raw)
64  min = Math.min.apply(null, raw)
65
66
67  maxData = [-1, 0, 0]
68  for (var i=0, next=false, start=0, number=0; i < raw.length; i++){
69          if (raw[i] > wall + 5 && !next) next = true, number++, start = encs[i]
70          if (raw[i] == max) maxData[0] = number, maxData[1] = start
71          if (raw[i] < wall + 5){
72                  if (maxData[0] == number && next) maxData[2] = encs[i]
73                  next = false, start = 0
74          }
75  }
76
77
78  goal = eL() - ((maxData[1] + maxData[2])/2)-275
79  turnGyro(180)
80  goGyroCpr(goal, 180)
81
82  turnGyro(-90)
83  goFront(-90)
84
85  brick.display().addLabel('finish', 1, 1)
86  brick.display().redraw()
```

Задача II.1.1.7. Перемещение в лабиринте (10 баллов)

Робот, собранный по дифференциальной схеме оснащен тремя инфракрасными датчиками расстояния. Один из датчиков расстояния установлен так, что показывает расстояние до препятствий прямо по курсу робота. Второй датчик расстояния направлен влево. Третий датчик расстояния направлен вправо.

Необходимо в заранее неизвестном лабиринте размером 8×8 секторов переместиться в сектор финиша. Сектор — квадрат шириной 700 мм. Координаты сектора финиша передаются в виде двух изображений ArTag маркеров через входной файл.

Началом координат служит верхний левый сектор лабиринта. Положительным направлением оси X считается направление по горизонтали вправо. Положительным направлением оси считается направление по вертикали вниз. Расположение осей координат представлено на рис. II.1.21. Гарантируется что при старте робот направлен в сторону положительного направления оси X.

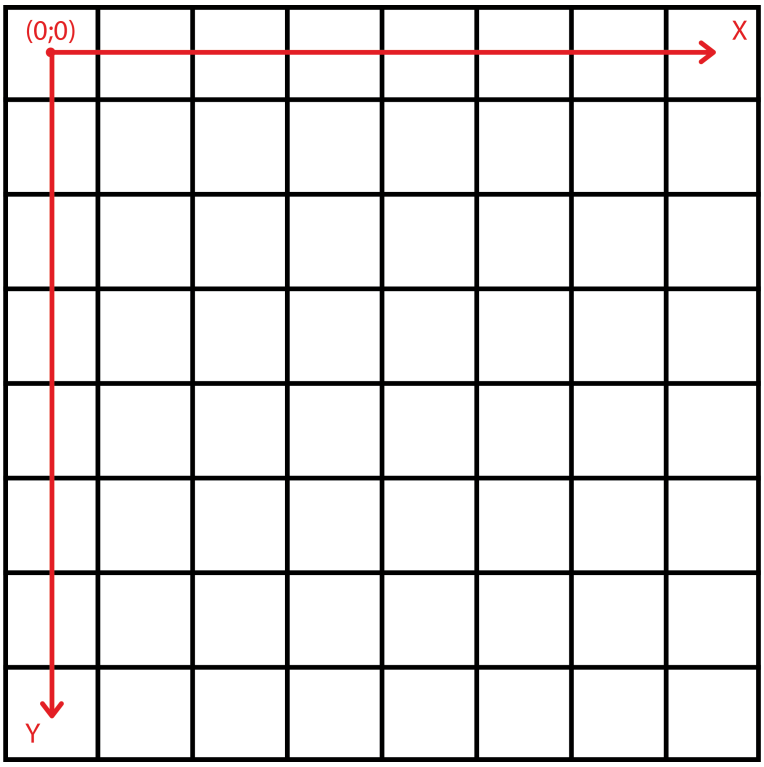


Рис. II.1.21: Расположение осей координат

Изображение поверхности с ArTag маркером приходит в управляющую программу в виде набора 160 × 120 точек, где каждая точка закодирована в RGB-формате.

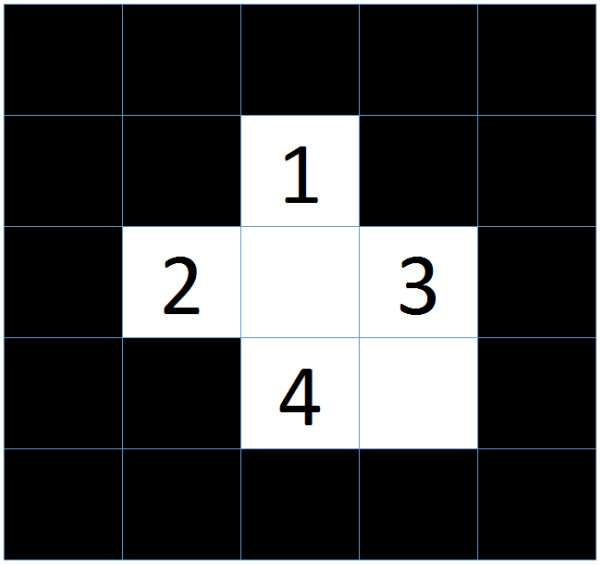


Рис. II.1.22: Нумерация битов в маркера

Элементы ARTag маркеры (https://inside.mines.edu/~whoff/courses/EENG512/lectures/other/ARTag.pdf), расположенные по его границе — всегда черные. Четыре элемента, находящиеся в углах внутреннего 3 × 3 квадрата определяют ориентацию маркера таким образом, что только элемент в нижнем правом углу квадрата — белый. Центральный элемент квадрата используется для проверки четности (parity check): если количество единичных бит в двоичной записи закодированного в маркере числа нечетное, то он черный. Оставшиеся 4 элемента маркера кодируют число

по следующему правилу: если элемент черный, то он обозначает 1, если белый, то 0 при этом самый первый элемент — старший бит закодированного числа. Элементы пронумерованы сверху вниз, слева направо (см. рис. II.1.22).

В случае если число меньше 8, то данные маркер кодирует координату по оси X. В противном случае следует вычесть 8 из полученного значения и получим координату по оси Y.

Например, на маркере с рис. II.1.23 закодировано число 0011_2, что эквивалентно 3_{10}. Следовательно, это X составляющая равная 3.

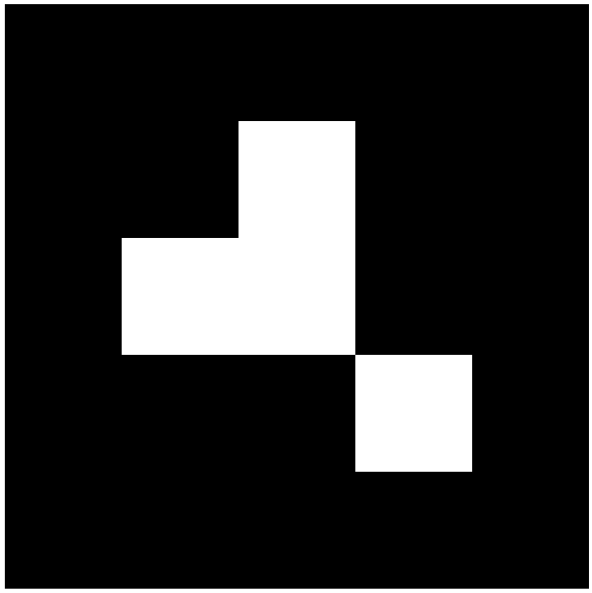


Рис. II.1.23: Маркер с закодированным значением - 0011_2

ris:arTagExample

Поскольку изображение было получено в процесе движения, то изображения ArTag маркера, получаемые с камеры, получаются в виде неправильного выпуклого четырехугольника, а непостоянные условия освещенности изменяют фокус и тон на изображении. Ориентация маркеров также заранее неизвестна, но его изображение таково, что оно по каждой из осей X, Y, Z относительно оптической оси камеры не превышает 25 градусов.

Формат входных данных

Входной файл содержит 2 строки, на каждой расположено изображение размером 160×120 в виде шестнадцатеричных чисел слева направо сверху вниз. Данные числа имеют следующий вид: $RRGGBB$, где RR - 16тиричное число R составляющей данного элемента матрицы, GG и BB — эт0 16ти-ричные числа G и B составляющих соответственно.

Все числа являются целыми.

Формат выходных данных

Доехать до сектора финиш и вывести на экран слово «finish».

Конфигурация робота

Подключение моторов:

- Левый мотор — порт M3
- Правый мотор — порт M4

Подключение датчиков:

- Датчик расстояния, направленный вперед - порт A1
- Датчик расстояния, направленный вправо - порт A2
- Датчик расстояния, направленный влево - порт A3

Примеры

Примеры тестовых полигонов представленны по данной ссылке.

Решение

Декомпозиция решения задачи:

1. Декодирование метки дополненной реальности ArTag
   1.1. Считать данные из файлов
   1.2. Распознать метки (получить десятичное число)
   1.3. Определить координаты X и Y сектора финиша
2. Исследовать неизвестный лабиринт и составить карту
3. Определить собственные координаты и направление (локализоваться)
4. Переместиться в сектор финиша

Распознавание меток ArTag подробно описано в материалах Университета Иннополис в разделе "Программирование интеллектуальных робототехнических систем" по адресу: Подраздел “Алгоритмические основы технического зрения”

Прямолинейное движение и точные повороты роботов на дифференциальной платформе являются базовыми алгоритмами и данном решении не будут подробно описываться.

Для точной навигации удобнее использовать П-регуляторы с управлением по энкодерам и гироскопу. Некоторые участники еще используют стенки лабиринта для выравнивания робота.

Согласно условиям задачи, робот стартует в заранее неизвестном секторе в направлении “восток (направо от начала координат)”.

Так как нам известен размер лабиринта - 8x8 секторов, но не известен стартовый сектор, представим весь лабиринт в размере 15x15 секторов. Предположим, что робот стартует в самом центре, в секторе с координатами $X : 7, Y : 7$. Двигаясь в любом направлении из этого сектора, робот сможет проехать максимум 8 секторов.

Возможность перемещения по секторам робот определяет с помощью датчиков расстояния. Перемещаясь между секторами робот составляет карту лабиринта в виде матрицы смежности, которая содержит информацию о возможности или не возможности перемещения в смежные сектора из текущего сектора.

Можно учитывать “тип” секторов - секторы в которых робот проезжал и секторы, в которые есть возможность проезда, но робот их не посещал.

Планирование и навигацию между секторами, особенно в случае, когда робот в месте, где с трех сторон стенки или все смежные сектора уже посещены, можно осуществлять с помощью алгоритма BFS (поиск в графе в ширину), составляя пути перемещения робота.

После исследования всей карты лабиринта определяем минимальные координаты реального лабиринта по осям X и Y ($deltaX$ и $deltaY$) и далее используем их для вычисления координат секторов для выполнения задания.

Расчет пути движения робота в сектор финиша выполнять через алгоритмы обхода графов: BFS, Дейкстры или A-star. Использование алгоритмов "правой руки" или "левой руки" в данном случае может не помочь, т.к. лабиринты могут быть "зациклены".

Пример программы-решения

Ниже представлено решение на языке JavaScript

```
function sign(num) {return num >= 0 ? 1 : -1}
function getYaw() {return brick.gyroscope().read()[6]}
function cm2cpr(cm) {return (cm / (Math.PI * robot.wheelD)) * robot.cpr}
function rad2deg(rad) {return rad * 180 / Math.PI }
function deg2rad(deg) {return deg * Math.PI / 180 }
function inRange(value, range){return (value >= range[0] && value <= range[1]) }
function printObj(obj){for (key in obj) print(key, ': ', obj[key]) }


function arrSum(arr){
        for(var i=0, summ=0, maxI = arr.length; i < maxI; i++)
                summ += arr[i].reduce(function(a, b) {return (a + b)})
        return summ
}

function arr1arr2(arr, cols, rows){
        var arrOut = []
        for (var i=0; i < rows; i++){
                arrOut.push([])
                for (var j=0; j < cols; j++)
                        // arrOut[i].push(arr[i * cols +
                        ↪  j])                                  // HEX
                        arrOut[i].push(parseInt(arr[i * cols + j],
                        ↪  16))                 // DEC
        }
        return arrOut
}

function printArr(arr, delim, file){
        delim = delim == undefined ? ',': delim

        if (file != undefined)
                script.removeFile(file)

        for (var i=0; i< arr.length; i++){
                var line = arr[i].join(delim)
                print (line)
```

```
36
37                  if (file != undefined){
38                          script.writeToFile(file, line + '\n')
39  }}}
40
41  function artag(arr, size){
42          function parityCheck(numBin, center){
43                  var binP = picBW[center[0]][center[1]]
44                  for (var i = 0, summ=0; i < numBin.length; i++) summ += numBin[i]
45                  return summ % 2 == binP
46          }
47
48          function makeBin(arr, key){
49                  var variants = [[2,1,3,0], [3,2,0,1], [0,3,1,2], [1,0,2,3]]
50                  var arrOut = []
51                  for (var i=0; i < 4; i++) arrOut.push(arr[variants[key][i]])
52                  return arrOut
53          }
54
55          //
56          size = size || [160, 120]
57          picW = size[0], picH = size[1]
58          var picRaw = arr1arr2(arr, picW, picH)
59          picBW  = pic2bw(pic2grey(picRaw))
60
61          var ABCD1=[], ABCD2 = [], ABCD
62          ABCD1 = findCorners_1(picBW)
63          ABCD2 = findCorners_2(picBW)
64          //print (ABCD1)
65          //print ('ABCD2: ',ABCD2)
66
67          ABCD = ABCD1
68
69          if (gauss(ABCD2) > gauss(ABCD1))
70                  ABCD = ABCD2
71
72          var center = findCross(ABCD)
73          var dotsDiag = []
74          var dots = []
75
76          for (var i = 0; i < 4; i++)
77                  dotsDiag.push(lineToSegmets(center, ABCD[i], 5))
78
79          for (var i = 0; i < 4; i++)
80                  dots.push(lineToSegmets(dotsDiag[i][2], dotsDiag[(i+1)%4][2], 2))
81
82          var KEY = -1
83          for (var i=0; i < 4; i++){
84                  if (picBW[dotsDiag[i][2][0]][dotsDiag[i][2][1]] == 0){
85                          KEY = i
86                          break
87                  }
88          }
89
90          for (var i=0, nums=[]; i < 4; i++)
91                  nums.push(picBW[dots[i][1][0]][dots[i][1][1]])
92
93          var bin = makeBin(nums, KEY)
94
95          if (parityCheck(bin, center)){
```

```
                var num10 = parseInt(bin.join(''),2)
        } else {
                return -3
        }

        return num10
}

function findCorners_1(arr, delta) {
        delta = delta || 4
        var A, B, C, D, x, y, l
        var roiH = picH-1-delta, roiW = picW-1-delta

        // top-left corner
        x = y = l = delta
        while (arr[y][x] != 1) {
                if (y >= roiH)
                        l += 1, y = delta, x = l

                if (x <= delta)
                        l += 1, y = delta, x = l
                else
                        y += 1, x -= 1
        }
        A = [y, x]

        //top-right corner
        x = roiW, y = l = delta
        while (arr[y][x] != 1) {
                if (y >= picH - 1 - delta) {
                        l += 1, y = delta, x = roiW - l
                        continue
                }
                if (x >= picW - 1 - delta)
                        l += 1, y = delta, x = roiW - l
                else
                        y += 1, x += 1
        }
        B = [y, x]

        //bottom-down corner
        x = roiW, y = roiH, l = delta
        while (arr[y][x] != 1) {
                if (y <= 1 + delta) {
                        l += 1, y = roiH, x = roiW - l
                        continue
                }
                if (x >= picW - 1 - delta)
                        l += 1, y = roiH, x = roiW - l
                else
                        y -= 1, x += 1
        }
        C = [y, x]

        //bottom-left corner
        x = l = delta, y = roiH
        while (arr[y][x] != 1) {
                if (y <= 1 + delta) {
                        l += 1, y = roiH, x = l
                        continue
```

```
156                 }
157                 if (x <= delta)
158                         l += 1, y = roiH, x = l
159                 else
160                         y -= 1, x -= 1
161         }
162         D = [y, x]
163 
164         return [A, B, C, D]
165 }
166 
167 function findCorners_2(arr, delta){                         // scanning horizontal,
    ↪  vertical
168         delta = delta || 4
169         var A, B, C, D, i, j
170         var roiW = arr[0].length - delta,
171                 roiH = arr.length - delta
172 
173         top:
174         for (i = delta; i < roiH; i++){
175                 for (j = delta; j < roiW; j++){
176                         if (arr[i][j] == 1){
177                                 A = [i, j];         break top
178         }}}
179 
180         bottom:
181         for (i = roiH - 1; i > delta-1; i--){
182                 for (j = delta; j < roiW; j++){
183                         if (arr[i][j] == 1){
184                                 C = [i, j]; break bottom
185         }}}
186 
187         right:
188         for (i = roiW - 1; i > delta-1; i--){
189                 for (j = delta; j < roiH; j++){
190                         if (arr[j][i] == 1){
191                                 B = [j, i]; break right
192         }}}
193 
194         left:
195         for (i = delta; i < roiW; i++){
196                 for (j = delta; j < roiH; j++){
197                         if (arr[j][i] == 1){
198                                 D = [j, i]; break left
199         }}}
200 
201         return [A, B, C, D]
202 }
203 
204 
205 function gauss(arr){
206 
207         arr.push(arr[0])
208 
209         var S = 0
210 
211 
212 
213         for (var i = 0; i < arr.length-1; i++){
214 
```

```
215                 S += arr[i][0] * arr[i+1][1]
216
217                 S -= arr[i][1] * arr[i+1][0]
218
219         }
220
221
222
223         S = Math.abs(S) * 0.5
224
225         return S
226
227 }
228
229
230 function lineToSegmets(xy1, xy2, segments){
231         var points = [xy1]
232         var dX = (xy2[1]-xy1[1])/segments
233         var dY = (xy2[0]-xy1[0])/segments
234
235         for (var i = 1, x, y; i < segments; i++){
236                 y = Math.floor(xy1[0] + dY * i)
237                 x = Math.floor(xy1[1] + dX * i)
238                 points.push([y, x])
239         }
240         points.push(xy2)
241         return points
242
243 }
244
245 function findCross(abcd){
246         var x1,x2,x3,x4,y1,y2,y3,y4,x,y
247         x1 = abcd[0][0], y1 = abcd[0][1]
248         x2 = abcd[2][0], y2 = abcd[2][1]
249
250         x3 = abcd[1][0], y3 = abcd[1][1]
251         x4 = abcd[3][0], y4 = abcd[3][1]
252
253         x =
        ↪  -(((x1*y2-x2*y1)*(x4-x3)-(x3*y4-x4*y3)*(x2-x1))/((y1-y2)*(x4-x3)-(y3-y4)*(x2-x1)))
254         y = ((y3-y4)*(-x)-(x3*y4-x4*y3))/(x4-x3)
255
256         return [Math.floor(x), Math.floor(y)]
257 }
258
259 function clr2rgb(clr){
260         var R, G, B
261         R = (clr & 0xFF0000) >> 16
262         G = (clr & 0x00FF00) >> 8
263         B = clr & 0x0000FF
264
265         return ([R, G, B])
266 }
267
268 function rgb2clr(R, G, B){
269         if (G == undefined && B == undefined) {
270                 G = R
271                 B = R
272         }
273         return R * Math.pow(256, 2) + G * 256 + B
```

```
}

function pic2bw(arr){
        var arrOut = []
        var threshold =  arrSum(arr) / (arr.length * arr[0].length)
        for (var i=0; i<arr.length; i++){
                arrOut.push([])
                for (var j=0; j < arr[0].length; j++)
                        arrOut[i].push(arr[i][j] > threshold ? 0 : 1)
        }
        return arrOut
}

function pic2grey(arr){
        var arrOut = [], grey, avg
        for (var i=0, maxI = arr.length; i < maxI; i++){
                arrOut.push([])
                for (var j=0, maxJ = arr[0].length; j < maxJ; j++){
                        grey = clr2rgb(arr[i][j])
                        avg = floor((grey[0] + grey[1] + grey[2])/3)
                        //arrOut[i].push(rgb2clr(avg))

                        arrOut[i].push(avg)
                }
        }
        return arrOut
}

function cell2coord(cell, cols){
        cols = cols || map.size
        var row = Math.floor(cell/cols)
        var col = cell - row * cols
        return [row, col]
}

function coord2cell(row, col, cols){
        cols = cols || map.cols
        return row * cols + col
}

function gotoCell(goalCell){ doCommands(makeCommands(bfs(robot.cell, goalCell))) }
function bfs(start, end, mtrx){
        mtrx = mtrx || map.matrix
        var queue = [start]
        var visited = []
        var path = []
        for (var i=0; i < mtrx.length; i++) visited.push(0)
        while (queue.length > 0){
                var p = queue.shift()
                if (visited[p] == 0){
                        visited[p] = 1
                        path.push(p)
                        for (var i=0; i<mtrx.length; i++){
                                if (mtrx[p][i] == 1 && visited[i] == 0)
                                        queue.push(i)
                        }
                }
                if (p == end) break
        }
        var pathBack = [Number(path.slice(-1))]
```

```
334         path.reverse()
335         for (var i = 1; i < path.length; i++){
336                 if (mtrx[pathBack.slice(-1)][path[i]] == 1)
337                         pathBack.push(path[i])
338         }
339         pathBack.reverse()
340         return pathBack
341 }
342
343 function makeCommands(cells, dir){
344
345         dir = dir || robot.dir
346         var commands = []
347         var sides = [-map.size, 1, map.size, -1]
348         var actions = [         ['F', 'L', 'LL', 'R'],          // -map.size
349                                 ['R', 'F', 'L', 'LL'],          // 1
350                                 ['LL', 'R', 'F', 'L'],          // map.size
351                                 ['L', 'LL', 'R', 'F']           // -1
352                               ]
353         for (var i = 1; i < cells.length; i++){
354                 var nextDir = sides.indexOf(cells[i] - cells[i-1])
355                 var act = actions[nextDir][dir]
356                 dir = nextDir
357                 commands.push(act)
358                 if (act != 'F') commands.push('F')
359         }
360         return commands.join('').split('')
361 }
362
363 function doCommands(commands){
364         for (var j=0; j < commands.length; j++){
365                 if(commands[j]=="F")                    moveGyro(map.cellLength)
366                 else if(commands[j]=="R")       turnGyro(1)
367                 else if(commands[j]=="L")       turnGyro(-1)
368                 else if(commands[j]=="B")       turnGyro(2)
369         }
370 }
371
372 function motors(mL, mR){
373         if (mL == 0 && mR == undefined)
374                 mL = 0, mR = 0
375         else{
376                 mL = mL || robot.v
377                 mR = mR == undefined ? mL : mR
378         }
379         brick.motor('M4').setPower(mL)
380         brick.motor('M3').setPower(mR)
381 }
382
383 function moveGyro(path, gyro0){
384         gyro0 = gyro0 || gyroAngles[robot.dir]
385         gyro0 *= 1000
386         path = cm2cpr(path) + encoderL()
387
388         while (encoderL() < path){
389                 var y = getYaw()
390                 u = (gyro0 - y) / 1000 * 0.9
391                 if (gyro0 == 180000) u = ((gyro0 - abs(y)) * sign(y))/1000 * 0.7
392
393                 motors(robot.v + u, robot.v - u)
```

```
394                 script.wait(30)
395         }
396         motors(0)
397
398
399         switch (robot.dir){
400                 case 0: robot.y--; break
401                 case 1: robot.x++; break
402                 case 2: robot.y++; break
403                 case 3: robot.x--; break
404         }
405         robot.updateCell()
406 }
407
408 function forward(dlina, turn){
409         var L = dlina / (Math.PI * robot.wheelD)
410         var sgn = sign(dlina)
411         var err, u, vL, vR, encL, encR
412
413         L *= robot.cpr, L += encoderL()
414         encL = encoderL(), encR = encoderR()
415
416         if (sgn == 1){
417                 while (encoderL() <= L){
418                         err = (encoderL() - encL) - (encoderR() - encR)
419                         u = err * 2
420                         vL = (60 - u) * sgn
421                         vR = (60 + u) * sgn
422                         motors(vL, vR)
423                         script.wait(30)
424                 }
425         } else {
426                 while (encoderL() >= L){
427                         err = Math.abs((encoderL() - encL)) - Math.abs((encoderR() -
                        ↪  encR))
428                         u = err * 2
429                         vL = (60 - u) * sgn
430                         vR = (60 + u) * sgn
431                         motors(vL, vR)
432                         script.wait(30)
433                 }
434         }
435         motors(0)
436 }
437
438 function turnGyro(side, v){
439         //  side: -1 -> left;  1 -> right; 2 - turn back
440         ///
441         v = v || robot.v/2
442         forward(17.5 * 0.5, true)
443         var deltaAngle =
        ↪  gyroAngles[robot.dir]                                                    //
        ↪  current direction
444         robot.dir = (robot.dir + (side < 0 ? 3 : side)) % 4                          //
        ↪  new direction
445         var angle =
        ↪  gyroAngles[robot.dir]
446
447         deltaAngle -= angle
448         angle *= 1000
```

```
449
450         var sgn = sign(deltaAngle) * -1
451         if (abs(deltaAngle) > 180) sgn *= -1
452
453         motors(v * sgn, -v * sgn)
454         var dGyro = 1500
455         while (true){
456                 var yaw = getYaw()
457                 if (angle === 180000){ if (abs(yaw) - dGyro < angle && abs(yaw) +
                    ↪  dGyro > angle) break }
458                 else {if (yaw - dGyro < angle && yaw + dGyro > angle) break }
459                 wait(10)
460         }
461         motors(0)
462         forward(17.5 * -0.5, true)
463 }
464
465 function readSensors(){
466         var s = brick.sensor
467         return [
468                 s('A3').read() > map.cellLength ? 1 : 0,
469                 s('A1').read() > map.cellLength ? 1 : 0,
470                 s('A2').read() > map.cellLength ? 1 : 0,
471         ].join('')
472 }
473
474 function mapping(dir, mapH, mapW){
475         function cellsAround(){
476                 var cells = [robot.cell - map.size, robot.cell + 1, robot.cell +
                    ↪  map.size, robot.cell - 1]
477                 var dirs = [(robot.dir + 3) % 4,
478                                         robot.dir,
479                                         (robot.dir + 1) % 4]
480
481                 return [cells[dirs[0]], cells[dirs[1]], cells[dirs[2]]]
482         }
483
484         function updateMatrix(sensors, cells){
485                 for (var i = 0; i < 3; i++){
486                         if (cells[i] < 0 || cells[i] > map.visited.length-1)
487                                 continue
488
489                         map.matrix[robot.cell][cells[i]] = sensors[i]
490                         map.matrix[cells[i]][robot.cell] = sensors[i]
491
492                         if (map.visited[cells[i]] == 0 && sensors[i] == 1)
493                                 map.visited[cells[i]] = 2
494                         else if (map.visited[cells[i]] == 0)
495                                 map.visited[cells[i]] = sensors[i]
496                 }
497                 minXY(robot.cell)
498         }
499
500         function minXY(cell){
501                 var xy = cell2coord(cell)
502                 if (xy[0] < map.dY) map.dY = xy[0]
503                 if (xy[1] < map.dX) map.dX = xy[1]
504                 map.dCell = coord2cell(map.dY, map.dX)
505         }
506         //
```

```
507         gyroAngles = setsAngles[dir]
508         mapW = mapW || mapH
509 
510         var mapSize = Math.max(mapH, mapW)
511 
512         robot.y = mapSize - 1
513         robot.x = mapSize - 1
514         robot.dir = dir
515 
516         map.size = mapSize * 2 - 1
517         map.cols = map.size, map.rows = map.size
518         robot.cell = coord2cell(robot.y, robot.x)
519 
520         for (var i = 0; i < map.size * map.size; i++){
521                 map.visited.push(0)
522                 map.matrix.push([])
523                 for (var j = 0; j < map.size * map.size; j++)
524                         map.matrix[i].push(-1)
525         }
526 
527 
528 
529 // ***************************** BEGIN ************************
530         map.visited[robot.cell] = 1
531         updateMatrix(readSensors(), cellsAround())
532 
533         if (brick.sensor('A3').read() > map.cellLength)                // left
534                 turnGyro(-1)
535         else
536                 turnGyro(1)
537 
538         updateMatrix(readSensors(), cellsAround())
539 
540         while (true){
541                 var looked = []
542                 // looked but not visited cells
543                 for (var i=0; i < map.visited.length; i++)
544                         if (map.visited[i] == 2) looked.push(i)
545 
546                 if (looked.length == 0) break
547 
548                 var min_len = Infinity, min_path = 0, min_cell = -1
549                 for (var i = 0, p; i < looked.length; i++){
550                         p = bfs(robot.cell, looked[i])
551                         if (p.length < min_len){
552                                 min_len = p.length
553                                 min_path = p
554                                 min_cell = looked[i]
555                         }
556                 }
557                 doCommands(makeCommands(min_path, robot.dir))
558                 map.visited[robot.cell] = 1
559                 robot.updateXY()
560 
561                 updateMatrix(readSensors(), cellsAround())
562                 wait(30)
563         }
564 }
565 
566 //
```

```
//

robot = {
        wheelD: 5.6,
        track: 17.5,
        cpr: 360,
        x: 0, y: 0, cell:0, dir: 1,
        v: 90,
        updateXY: function(){robot.y = Math.floor(robot.cell/map.size); robot.x =
        ↪  robot.cell % map.size},
        updateCell: function(){robot.cell = coord2cell(robot.y, robot.x)}
}

map = {rows: 8, cols: 8, matrix: [], dX: Infinity, dY: Infinity, dCell: 0, visited:
↪  [], size: 8, cellLength: 17.5*4}

robot.cell = coord2cell(robot.y, robot.x)

setsAngles = [[0, 90, 180, -90], [-90, 0, 90, 180], [180, -90, 0, 90],[90, 180, -90,
↪  0]]
gyroAngles = setsAngles[robot.dir]

pi = Math.PI
wait = script.wait
floor = Math.floor
abs = Math.abs

encoderL = brick.encoder('E4').read
encoderR = brick.encoder('E3').read

brick.encoder('E3').reset()
brick.encoder('E4').reset()

pics_raw = script.readAll('input.txt')
artag1 = artag(pics_raw[0].trim().split(' '))
artag2 = artag(pics_raw[1].trim().split(' '))

goalX = artag1, goalY = artag2 % 8
if (Math.max(artag1, artag2) == artag1){
        goalY = artag1 % 8
        goalX = artag2
}
//print ('Y: ',goalY, '   X: ',goalX)

brick.gyroscope().calibrate(2000)
script.wait(2100)

mapping(1, 8)
gotoCell(coord2cell(goalY + map.dY, goalX + map.dX))
brick.display().addLabel('finish',1,1)
brick.display().redraw()
```

Заключительный этап

Заключительный этап олимпиады состоит из двух частей: индивидуальное решение задач по предметам (математика, информатика) и командное решение задачи на создание прототипа программного обеспечения.

Индивидульный предметный тур

На индивидуальное решение задач дается по 2 часа на один предмет. Для каждой из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике - общие для всех участников.

Решение каждой задачи по математике дает определенное количество баллов (см. критерии оценки). При этом некоторые задачи делятся на подзадачи. За каждую подзадачу можно получить от 0 до указанного количества баллов.

Решение задач по информатике предполагало написание программ. Ограничения по используемым языкам программирования не было. Проверочные тесты для каждой задачи по информатике делились на несколько групп. Прохождение всех тестов в группе тестов дает определенное количество баллов за решение задачи.

Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и информатика) — суммарно от 0 до 200 баллов:

- **Математика 9 класс** количество набранных баллов (от 0 до 100);
- **Математика 10-11 класс** количество набранных баллов (от 0 до 100);
- **Информатика** количество набранных баллов (от 0 до 300) делится на коэффициент 3.

Математика. 8-9 класс

Задача III.1.1.1. (20 баллов)

(a) Приведите пример натурального числа, которое можно записать как в виде произведения трех натуральных чисел, так и в виде суммы тех же натуральных чисел.

(b) Докажите, что любое составное число можно записать как в виде произведения нескольких (хотя бы двух) натуральных чисел, так и в виде суммы тех же натуральных чисел.

Решение

(a) Например, $6 = 1 \cdot 2 \cdot 3 = 1 + 2 + 3$.

(b) Представим составное число n в виде ab, где a, $b > 1$. Без ограничения общности будем считать, что $a \leqslant b$, тогда $a + b \leqslant 2b \leqslant ab = n$. Значит, можно

записать n в виде

$$n = a + b + \underbrace{1 + \ldots + 1}_{n-a-b} = a \cdot b \cdot \underbrace{1 \cdot \ldots \cdot 1}_{n-a-b},$$

что и требовалось.

Дополнительные критерии оценки

За пункт **(a)** — 10 баллов, за пункт **(b)** — 10 баллов.

Задача III.1.1.2. (30 баллов)

С парой вещественных чисел a, b, такой, что $ab \neq 1$, проделывается следующая операция:

$$a \odot b = \frac{a + b - 2ab}{1 - ab}.$$

(a) Докажите, что если числа a и b лежат в интервале $(0; 1)$, то и число $a \odot b$ лежит в нем.

(b) Докажите, что операция $\odot$ ассоциативна, т.е. что $(a \odot b) \odot c = a \odot (b \odot c)$.

Решение

(a) Сначала докажем, что $a \odot b > 0$. Ясно, что $1 - ab > 0$. Пусть $a \leqslant b$. Тогда $a + b - 2ab \geqslant 2a - 2ab = 2a(1 - b) > 0$. Значит, $\frac{a+b-2ab}{1-ab} > 0$.

Теперь докажем, что $a \odot b < 1$. Учитывая, что знамнатель дроби $a \odot b$ положителен, домножим на него наше неравенство и полчим равносильное неравенство $a + b - 2ab < 1 - ab$, которое преобразуется в верное неравенство $(1 - a)(1 - b) > 0$. Значит, $a \odot b < 1$, и наше утверждение доказано.

(b) Утверждение этого пункта можно доказать непосредственными вычислениями, однако можно поступить хитрее. Рассмотрим функцию $f(x) = \frac{1}{1-x} - 1$. Докажем, что $f(a \odot b) = f(a) + f(b)$. В самом деле,

$$f(a \odot b) = \frac{1}{1 - \frac{a+b-2ab}{1-ab}} - 1 = \frac{1 - ab}{1 - a - b + ab} - 1 = \frac{1 - ab}{(1 - a)(1 - b)} - 1 =$$
$$= \left(\frac{1}{1 - a} - 1\right) + \left(\frac{1}{1 - b} - 1\right) = f(a) + f(b).$$

Таким образом,

$$f((a \odot b) \odot c) = f(a \odot b) + f(c) = f(a) + f(b) + f(c) = f(a) + f(b \odot c) = f(a \odot (b \odot c)).$$

Т.к. функция f инъективна, отсюда следует, что $(a \odot b) \odot c = a \odot (b \odot c)$, что и требовалось доказать.

Дополнительные критерии оценки

За пункт **(a)** — 10 баллов, за пункт **(b)** — 20 баллов.

Задача III.1.1.3. (50 баллов)

В колоде $2^n - 1$ карт ($n > 1$). Ее тасуют следующим образом. Колода делится на две стопки: в первой берутся верхние 2^{n-1} карт, а во второй — нижние $2^{n-1} - 1$. Далее карты из двух стопок смешиваются следующим образом: i-я карта второй стопки кладется между i-й и $(i + 1)$-й картами первой стопки (т.е. первая карта второй стопки кладется мжду первой и второй картами первой стопки, вторая карта — между второй и третьей, и т.д.; см. рис. **??**).

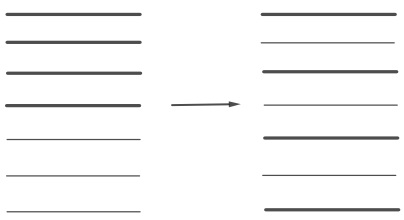


Рис. III.1.1: Перемешивание карт в колоде

(a) Докажите, что через несколько таких операций колода придёт в первоначальное положение.

(b) Найти наименьшее количество операций, при совершении которых колода придёт в первоначальное положение.

Решение

(a) Будем тасовать карты, применяя нашу операцию. Заметим, что количество возможных положений карт конечно, поэтому найдется такое положение карт, которое мы получим дважды (и тем самым возникнет цикл). Но каждое новое положение карт одноначно определяется исходным положением, поэтому мы не сможем получить из двух разных положений карт одно новое. Значит, наш цикл начинается с исхоного положения карт, что и требовалось доказать.

(b) Занумеруем позиции всех карт в изначальной расстановке, начиная с 0: 0, 1, 2, …, $2^n - 1$. Заметим, что карта с номером $k \leqslant 2^{n-1} - 1$ при совершении нашей операции перейдет в карту с номером $2k$, а карта с номером $\ell \geqslant 2^{n-1}$ перейдет в карту с номером $2\ell - (2^n - 1)$. Дальше можно рассуждать по-разному.

Способ 1. Запишем номер карты в двоичной системе счисления с помощью n цифр (при необходимости добавив в начало нули): $\overline{a_n a_{n-1} \dots a_1 a_0}$. Докажем, что наше операция меняет двоичный номер карты следующим образом:

$$\overline{a_n a_{n-1} \dots a_1 a_0} \to \overline{a_{n-1} \dots a_1 a_0 a_n}$$

(т.е. первяа цифра переставляется в конец). В самом деле, если $a_n = 0$, то число умножается на 2, т.е. справа к нему приписывается $0 = a_n$:

$$\overline{a_n a_{n-1} \dots a_1 a_0} = \overline{a_{n-1} \dots a_1 a_0} \to \overline{a_{n-1} a_{n-2} \dots a_1 a_0 0} = \overline{a_{n-1} a_{n-2} \dots a_1 a_0 a_n}.$$

Если же $a_n = 1$, то двоичная запись изеняется следующим образом:

$$\overline{a_n a_{n-1} \dots a_1 a_0} = \overline{1 a_{n-1} \dots a_1 a_0} \to \overline{1 a_{n-1} \dots a_1 a_0 0} - \overline{10 \dots 00} + 1 = \\ = \overline{a_{n-1} \dots a_1 a_0 1} = \overline{a_{n-1} a_{n-2} \dots a_1 a_0 a_n}.$$

И в том, и в другом случае первая цифра в двоичной записи смещается в конец числа.

Таким образом, ясно, что для того, чтобы номер каждой карты стал прежним, достаточно n операций (поскольку двоичная запись состоит из n цифр). С другой стороны, число $\overline{1 \dots 00}$ невозможно сделать прежним за меньшее число операций (т.к. единица должна послеовательно пройти по всем позициям, которых ровно n).

Способ 2. Обозначим через $f(k)$ номер, на который перейдет карта с номером k. Тогда $f(k) \equiv 2k \pmod{2^n - 1}$. Значит, применяя операцию f s раз, карта, стоящая на номере k, будет стоять на номере $f^{(s)}(k) \equiv 2^s k \pmod{2^n - 1}$. Пусть s — наименьшее число, такое, что все карты вернулись на исходные места. Тогда $2^s k \equiv k \pmod{2^n - 1}$ для всех $k = 0, 1, \dots, 2^n - 2$.

Заметим, что при $s = n$ это сравнение верно, и все карточки вернутся на свои мста. С другой стороны, при $s < n$ это сравнение неверно для числа $k = 1$, т.к. $2^s - 1 < 2^n - 1$. Поэтому минимальное количество операций равно n.

Дополнительные критерии оценки

За пункт **(a)** — 20 баллов, за пункт **(b)** — 30 баллов.

В пункте **(a)** доказано, что возникает цикл, но н доказано отсутствие предпериода — 10 баллов.

В пункте **(b)** верно выписаны номера всех карт после перетасовки — 10 баллов.

В пункте **(b)** показано, что существует карта, которая возвращается на исходную позицию не менее, чем за n ходов — 15 баллов.

Математика. 10-11 класс

Задача III.1.2.1. (20 баллов)

(a) Приведите пример натурального числа, которое можно записать как в виде произведения трех натуральных чисел, так и в виде суммы тех же натуральных чисел.

(b) Найдите все натуральные числа, которые можно записать как в виде произведения нескольких (хотя бы двух) натуральных чисел, так и в виде суммы тех же натуральных чисел.

Решение

(a) Например, $6 = 1 \cdot 2 \cdot 3 = 1 + 2 + 3$.

(b) Заметим, что $n = 1$ и простые n не подходят: случай $n = 1$ очевиден, а для простого n существуют представления в виде проиедения натуральных только вида $n \cdot 1 \ldots \cdot 1$. Но тогда сумма сомножителей не менее $n + 1$, что больше n.

Теперь докажем, что составное n удовлетворяет словию. Представим составное число n в виде ab, где a, $b > 1$. Без ограничения общности будем считать, что $a \leqslant b$, тогда $a + b \leqslant 2b \leqslant ab = n$. Значит, можно записать n в виде

$$n = a + b + \underbrace{1 + \ldots + 1}_{n-a-b} = a \cdot b \cdot \underbrace{1 \cdot \ldots \cdot 1}_{n-a-b},$$

что и требовалось.

Дополнительные критерии оценки

За пункт **(a)** — 10 баллов, за пункт **(b)** — 10 баллов.

Задача III.1.2.2. (30 баллов)

На сторонах треугольника ABC отмечены точки A_1, B_1, C_1 соответственно. Пусть ω_A, ω_B и ω_C — описанные окружности треугольников AB_1C_1, BC_1A_1, CA_1B_1 соответственно.

(a) Докажите, что окружности ω_A, ω_B и ω_C проходят через общую точку P.

(b) Пусть прямая AP пересекает сторону BC в точке X. Докажите, что описанная окружность треугольника PA_1X касается окружности ω_A.

Решение

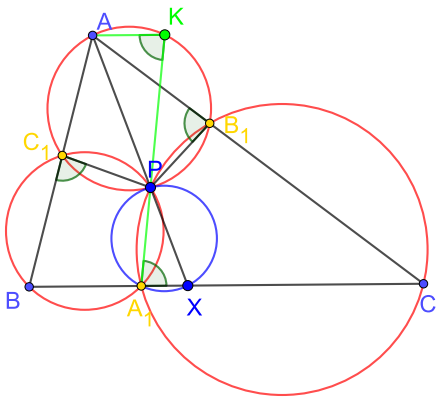


Рис. III.1.2: Пояснение к решению

(a) Используем критерий вписанного четырехугольника: четырехугольник является вписанным если и только если его внешний угол равен внутреннему не смежному с ним. Проведем окружности ω_A и ω_B, которые пересекаются в точке P, отличной от C_1. Докажем, что четырехугольник A_1CB_1P является вписанным. В самом деле, $\angle CA_1P = \angle BC_1P = \angle AB_1P$, что и требовалось.

(b) Проведем прямую A_1P, вторично пересекающую окружность ω_A в точке K. Заметим, что $\angle AKP = \angle AB_1P = \angle CA_1P$, поэтому $AK \parallel BC$. Значит, треугольники AKP и A_1PX гомотетичны с центром в P, а потому их описанные окружности касаются в точке P, что и требовалось доказать.

Дополнительные критерии оценки

За пункт **(a)** — 10 баллов, за пункт **(b)** — 20 баллов.

Задача III.1.2.3. (50 баллов)

Найти все $n \geqslant 3$, для которых правильный n-угольник можно разрезать непересекающимися диагоналями на равнобдренные треугольники.

Решение

Рассмотрим некоторое натуральное n, удовлетворяющее условию. Пусть n четно. Заметим, что тогда каждая сторона исходного правильного n-угольника должна быть боковой стороной какого-то равнобедренного треугольника разбиения, поскольку противном случае должна существовать вершина, находящаяся на серединном перпендикуляре к этой стороне, что невозможно при четном n. Поэтому при четном n вершины n-угольника необходимо соединять диагоналями через одну. тем-самым у нас останется правильный $n/2$-угольник. Продолжая этот процесс, мы в конце концов либо придем к квадрату, который легко триангируется (что дает нам ответ $n = 2^k$), либо к нечетному n.

Будем теперь исследовать случай нечетного n. В таком случае существует сторона, которая является основанием равнобдренного треугольника разбиения. Этот треугольник делит наш n-угольник на две одинаковые части, состоящие из $\frac{n-1}{2}$ коротких сторон и одной длинной стороны (бывшей диагонали). Эти части также должны триангулироваться на равнобедренные треугольники. Ясно, что самая длинная сторона должна быть основанием равнобдренного треугольника, и этот треугольник снова делит многоугольник на две одинаковые части. При этом количество сторон у частей меняется следующим образом: $n \to \frac{n+1}{2}$. Заметим, что после каждого шага число $\frac{n+1}{2}$ также должно быть нечетным (иначе не надется равнобедренного треугольника с основанием на длинной стороне), поэтому в кончном счете мы придем просто к треугольникам. Обратный процесс увеличения сторон выглядит так: $n \to 2n - 1$. Стартуя с $n = 3 = 2^1 + 1$, мы получам $n = 2^2 + 1$, $2^3 + 1$, $\ldots$, $2^\ell + 1$.

Таким образом, окончательный ответ выглядит так: $n = 2^k(2^\ell + 1)$, где k, $\ell \in \mathbb{Z}_{\geqslant 0}$ и k, l не равны 0 одновременно.

Ответ: $n = 2^k(2^\ell + 1)$, где k, $\ell \in \mathbb{Z}_{\geqslant 0}$ и k, l не равны 0 одновременно.

Дополнительные критерии оценки

Показано, что если n четно, то задача сводится к числу $\frac{n}{2}$ — 5 баллов.

С помощью перехода от n к $\frac{n}{2}$ получен ответ $2 = 2^k$ и указано, что осталось разобрать случай нечетного n — 15 баллов.

Для нечетного n доказано наличие треугольника, делящего исходный n-угольник на две одинаковые части — 5 баллов.

Задача верно сведена от нечетного n к числу $\frac{n+1}{2}$ и обоснована нечетность этого числа — 15 баллов.

Информатика. 8-11 класс

Задача III.1.3.1. Комната ярости 2 (100 баллов)

Нашего героя зовут Коля, и мы застали его не в лучшем расположении духа. Благо его добрая подруга Гертруда была неподалёку и убедила Колю пойти выпустить пар в ближайшую "Комнату ярости".

Со слухом у Николая всё в порядке, поэтому предметам в "Комнате" пощады ждать не придётся. Вернее, в "Комнатах", ведь новые времена требуют новых решений и "Комната ярости" разрослась практически до "тоннеля", по которому Николай собирается пронестись мощнейшим ураганом.

В ближайшей "Комнате ярости" n комнат, которые последовательно соединены коридорами. То есть первая комната, с которой и начнёт Николай, соединена коридором только со второй комнатой, i-я ($1 < i < n$) комната соединена с $i-1$-й и $i+1$-й, а n-я соединена только с $n-1$-й комнатой. Каждый из коридоров Николай преодолевает за единицу времени, абсолютно не тратя на это сил. Также в каждой комнате есть дверь, которая позволяет покинуть "Тоннель ярости", но наш герой не планирует выходить из "Тоннеля" ни из какой комнаты, кроме n-й, а из n-й выйдет только после того, как уничтожит всё, что можно уничтожить во всех n комнатах.

Единственное, что может остановить Николая в те моменты, когда хочется крушить всё, что попадается на глаза, – это усталость. Изначально у Коли есть e единиц энергии и ни в какой момент времени не может быть больше e единиц энергии. А при нуле энергии Коля теряет способность крушить предметы и максимум может ходить по комнатам и коридорам, как призрак, пока у него не появятся новые силы.

Новые времена – это ещё и новые технологии. Вот и в "Комнатах ярости" появились роботизированные мини-бары по одному на комнату. В них можно найти практически всё, что душе угодно.

Да, силы Николая не безграничны. Но он здоров как бык. И потому он давно для себя выбрал самый доступный, дешёвый и крайне вредный способ пополнения сил – заправляться энергетиками.

В мини-барах есть энергетики, но всего лишь одной марки. Изначально в мини-баре i-й комнаты есть k_i банок энергетика, каждая из которых может прибавить одну единицу энергии Коле.

После каждого подхода Николая к мини-бару соответствующий мини-бар обновляет себя: убирает все банки энергетика, что в нём были, и достаёт новые. В общем случае, при j-м ($j \geqslant 1$) подходе Николая к мини-бару в i-й комнате в мини-баре будет $\max(0, k_i + 1 - j)$ банок энергетика, каждая из которых может прибавить j единиц энергии Николаю.

Николай уже выбрал стратегию, которой он будет следовать в "Тоннеле". Он будет разрушать предметы последовательно: сначала все предметы в первой комнате,

потом все предметы во второй комнате и т. д., пока не уничтожит всё. Переносить предметы из одной комнаты в другую он не будет. На уничтожение одного предмета он тратит единицу своей энергии и единицу времени. Делать паузы он будет только в моменты, когда его энергия иссякнет, а в комнате, где он находится, ещё не все предметы разрушены.

В такой ситуации Николай сначала пойдёт к мини-бару той комнаты, в которой он сейчас находится, и выпьет все энергетики из этого мини-бара, даже если некоторые из них не прибавят ему энергии. Более формально, если текущий запас энергии Коли равен x единиц энергии и он собирается выпить банку энергетика, которая может прибавить y единиц энергии, то после выпивания этой банки запас энергии Коли будет равен $\min(x + y, e)$ единиц энергии.

Если энергия по-прежнему на нуле, то он выбирает одну из комнат, в которых он уже разрушил все предметы, и выпивает все энергетики из мини-бара этой комнаты. Из всех таких комнат он выбирает ту, энергетики в мини-баре которой прибавят Николаю больше всего энергии. Если и таких комнат несколько – ближайшую из них.

Для последнего описанного действия Николаю приходится напрячь мозги, чего ему совсем не хочется и еле-еле можется. Если Николаю придётся включать голову более 1000 (тысячи) раз за время нахождения в “Комнате”, то на 1001-й он потеряет ориентацию в пространстве и не сможет довести дело до конца.

Если после опустошения одного из мини-баров энергия Николая оказывается больше нуля, то он сразу идёт продолжать крушить предметы.

На перемещение внутри комнаты Николай не тратит ни времени, ни сил, а на выпивание одной банки энергетика тратит только единицу времени.

Давайте рассчитаем, на какое время Гертруда точно изолировала нашего героя от общества.

Формат входных данных

В первой строке даны два целых числа n ($1 \leqslant n \leqslant 100\,000$) и e ($0 \leqslant e \leqslant 10^{18}$) – количество комнат в “Комнате ярости” и максимальный размер запаса энергии Коли.

Во второй строке n целых чисел a_i ($0 \leqslant a_i \leqslant 10^{18}$) – изначальное количество предметов в i-й комнате.

Гарантируется, что суммарное количество предметов в комнатах не превышает 10^{18}.

В третьей строке n целых чисел k_i ($0 \leqslant k_i \leqslant 10^6$) – изначальное количество энергетиков в мини-баре i-й комнаты.

Формат выходных данных

Выведите -1, если наш герой не выберется из комнат самостоятельно. В противном случае выведите минимальное количество единиц времени, которое Николай проведёт в “Комнате ярости”.

Примеры

Пример №1

Стандартный ввод
3 2 0 0 3 1 1 0
Стандартный вывод
8

Пояснение к примеру

В первом тесте Коля спокойно пройдёт до последней комнаты за 2-е единицы времени, потом разрушит два предмета за 2-е единицы времени и пойдёт во вторую комнату, чтобы выпить энергетик, вернуться, уничтожить последний предмет и уйти. Таким образом, ответ – это $2 + 2 + 1 + 1 + 1 + 1 = 8$.

Пример №2

Стандартный ввод
5 5 1 2 3 4 5 3 2 3 2 2
Стандартный вывод
33

Способ оценки работы

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи
0	0	Тесты из условия	–
1	10	$k_i \leqslant 1$	–
2	10	$n \leqslant 1000$, $k_i \leqslant 100$	0
3	15	$k_i \leqslant 100$	0, 1, 2
4	15	$n \leqslant 1000$	0, 2
5	20	$k_i^2 \leqslant e$	–
6	15	Николаю не придётся включать мозг	–
7	15	Нет дополнительных ограничений	0, 1, 2, 3, 4, 5, 6

Баллы за подзадачу будут начислены в случае прохождения всех тестов необходимых подзадач и всех тестов самой подзадачи.

Для проверки результата использовался следующий код на языке C++

```cpp
#include "testlib.h"
#include <sstream>

using namespace std;

int main(int argc, char * argv[]) {
  setName("compare ordered sequences of signed int%d numbers",
    8 * int(sizeof(long long)));

  registerTestlibCmd(argc, argv);

  int n = 0;
  string firstElems;

  while (!ans.seekEof() && !ouf.seekEof()) {
    n++;
    long long j = ans.readLong();
    long long p = ouf.readLong();
    if (j != p)
      quitf(_wa, "%d%s numbers differ - expected: '%s', found: '%s'", n,
        englishEnding(n).c_str(), vtos(j).c_str(),
        vtos(p).c_str());
    else if (n <= 5) {
      if (firstElems.length() > 0)
        firstElems += " ";
      firstElems += vtos(j);
    }
  }

  int extraInAnsCount = 0;

  while (!ans.seekEof()) {
    ans.readLong();
    extraInAnsCount++;
  }

  int extraInOufCount = 0;

  while (!ouf.seekEof()) {
    ouf.readLong();
    extraInOufCount++;
  }

  if (extraInAnsCount > 0)
    quitf(_wa,
      "Answer contains longer sequence [length = %d], but output contains %d
      ↪  elements",
      n + extraInAnsCount, n);

  if (extraInOufCount > 0)
    quitf(_wa,
      "Output contains longer sequence [length = %d], but answer contains %d
      ↪  elements",
      n + extraInOufCount, n);

  if (n <= 5)
    quitf(_ok, "%d number(s): \"%s\"", n, compress(firstElems).c_str());
  else
    quitf(_ok, "%d numbers", n);
}
```

Решение

Данная задача – задача на моделирование с дальнейшей оптимизацией. Для начала важно внимательно прочесть условие и правильно смоделировать описанные события. Так получается решение с асимптотикой $O\Big(\sum\limits_{i=1}^{n}(k_i) + b \cdot n\Big)$, где $b = 1000$.

При $k_i \leqslant 1$ в мини-баре каждой комнаты либо есть одна банка энергетика, либо нет ни одной и никогда больше не появится. Поэтому для быстрого определения комнаты в случаях, когда Николаю приходится включать мозг, выгодно воспользоваться стеком.

При $k_i > 1$ стек уже не подойдёт. Нам нужна структура данных, которая умеет быстро добавлять в себя элемент и удалять из себя «лучший» элемент. Для этого подойдёт, например, очередь с приоритетом.

Для полного решения остаётся лишь научиться быстро определять, сколько раз Николай подойдёт к мини-бару в i-й комнате, когда будет разрушать предметы i-й комнаты.

За j-й подход к мини-бару i-й комнаты Николай может получить

$$j \cdot (k_i + 1 - j) = j \cdot k_i - j \cdot (j - 1)$$

единиц энергии.

Так как достаточно рассматривать лишь такие j, что $1 \leqslant j \leqslant k_i \leqslant 10^6$, значения $\sum\limits_{m=0}^{j}(m)$ и $\sum\limits_{m=0}^{j}(m \cdot (m-1))$ можно предподсчитать.

Пусть

$$sum_j = \sum_{m=0}^{j}(m) = \tfrac{(1+j)\cdot j}{2}\ ,$$

$$diff_j = \sum_{m=0}^{j}(m \cdot (m-1)) = \begin{cases} 0, & \text{если } j = 0; \\ diff_{j-1} + j \cdot (j-1), & \text{если } 1 \leqslant j \leqslant k_i; \end{cases}.$$

Тогда за первые j $(1 \leqslant j \leqslant k_i)$ подходов к мини-бару в i-й комнате Николай получит не более $sum_j \cdot k_i - diff_j$ единиц энергии и потратит на эти подходы ровно $\frac{(k_i \cdot 2 + 1 - j)\cdot j}{2}$ единиц времени. Сколько именно энергии получит Николай за первые j $(1 \leqslant j \leqslant k_i)$ подходов, зависит от e.

Не сложно заметить, что график функции $f(j) = j \cdot (k_i + 1 - j)$ при фиксированном k_i является параболой с ветвями вниз. Следовательно, существует конкретное наибольшее количество энергии, которое может получить Николай за один подход к мини-бару в i-й комнате. Так как мы работаем в целых числах, это значение достигается при $j = \left\lfloor \frac{k_i+1}{2} \right\rfloor$ и равно $\max_{k_i} = \left\lfloor \frac{k_i+1}{2} \right\rfloor \cdot \left(k_i + 1 - \left\lfloor \frac{k_i+1}{2} \right\rfloor\right) = \left\lfloor \frac{k_i+1}{2} \right\rfloor \cdot \left\lfloor \frac{k_i+2}{2} \right\rfloor$.

Если $\max_{k_i} \leqslant e$, то количество раз, которое Николай подойдёт к мини-бару в i-й комнате, когда будет разрушать предметы i-й комнаты, можно найти с помощью

бинарного поиска, так как за первые j подходов Николай получит в сумме ровно $sum_j \cdot k_i - diff_j$ единиц энергии (разумеется с учётом того, что менее чем за j подходов Николай не уничтожит все предметы в i-й комнате и $1 \leqslant j \leqslant k_i$).

Если $\max_{k_i} > e$, то бинарный поиск также можно применить, но с дополнительной модификацией.

Пусть s_j – это суммарное количество энергии, которое получит Николай за первые j подходов к мини-бару в i-й комнате. Тогда:

$$s_j = \begin{cases} sum_j \cdot k_i - diff_j, & \text{если } j \leqslant c_i; \\ sum_{c_i} \cdot k_i - diff_{c_i} + e \cdot (j - c_i), & \text{если } c_i < j \leqslant k_i - c_i; \\ (sum_{c_i} \cdot k_i - diff_{c_i}) \cdot 2 + e \cdot (k_i - 2 \cdot c_i) - (sum_{k_i - j} \cdot k_i - diff_{k_i - j}), & \text{если } k_i - c_i < j \leqslant k_i. \end{cases}$$

Значение c_i для фиксированных k_i и e можно определить, например, также бинарным поиском.

Пример программы-решения

Ниже представлено решение на языке Python 3

```python
from heapq import heappop, heappush

# считываем данные
n, e = [int(i) for i in input().split()]
a = [int(i) for i in input().split()]
k = [int(i) for i in input().split()]
j = [1] * n

# текущее время, энергия и количество включений мозга
t, cur_e, brain_on_cnt = -1, e, 0
# очередь с приоритетом
heap = []

# предподсчёт
max_k = max(k)
sum_k = [0] * (max_k + 1)
diff = [0] * (max_k + 1)
for i in range(1, max_k + 1):
    sum_k[i] = sum_k[i - 1] + i
    diff[i] = diff[i - 1] + i * (i - 1)

for i in range(n):
    # входим в i-ю комнату
    t += 1

    # уничтожаем все предметы, что можем
    m = min(cur_e, a[i])
    t += m
    cur_e -= m
    a[i] -= m

    # заходим в мини-бар i-й комнаты столько раз, сколько нужно
    if a[i] > 0:
        # если энергия от энергетиков «не пропадёт»
```

```
        if (k[i] // 2 + 1) * ((k[i] + 1) // 2) <= e:
            # определяем, сколько раз Николай подойдёт к мини-бару в текущей комнате
            l, r = 0, k[i] + 1
            while l + 1 < r:
                m = (l + r) // 2
                if sum_k[m] * k[i] - diff[m] < a[i]:
                    l = m
                else:
                    r = m
            r = min(k[i], r)  # ровно r раз

            t += (k[i] * 2 - r + 1) * r // 2
            cur_e = sum_k[r] * k[i] - diff[r]
            j[i] = r + 1

            # снова уничтожаем всё, что можем
            m = min(cur_e, a[i])
            t += m
            cur_e -= m
            a[i] -= m

        # если энергия от энергетиков «пропадёт»
        else:
            # определяем, сколько первых подходов произойдут без потери энергии
            l, r = 0, (k[i] + 1) // 2
            while l + 1 < r:
                m = (l + r) // 2
                if m * (k[i] + 1 - m) > e:
                    r = m
                else:
                    l = m
            cl = l  # ровно cl подходов

            # определяем, сколько раз Николай подойдёт к мини-бару в текущей комнате
            l, r = 0, k[i] + 1
            while l + 1 < r:
                m = (l + r) // 2

                if m <= cl:
                    cur_sum = sum_k[m] * k[i] - diff[m]
                elif m <= cl + k[i] - 2 * cl:
                    cur_sum = sum_k[cl] * k[i] - diff[cl] + e * (m - cl)
                else:
                    cur_sum = (sum_k[cl] * k[i] - diff[cl]) * 2 + e * (k[i] - 2 * cl)
                    ↪ \
                              - sum_k[k[i] - m] * k[i] + diff[k[i] - m]

                if cur_sum < a[i]:
                    l = m
                else:
                    r = m
            r = min(k[i], r)  # ровно r раз

            t += (k[i] * 2 - r + 1) * r // 2
            if r <= cl:
                cur_e = sum_k[r] * k[i] - diff[r]
            elif r <= cl + k[i] - 2 * cl:
                cur_e = sum_k[cl] * k[i] - diff[cl] + e * (r - cl)
            else:
                cur_e = (sum_k[cl] * k[i] - diff[cl]) * 2 + e * (k[i] - 2 * cl) \
```

```
                        - sum_k[k[i] - r] * k[i] + diff[k[i] - r]
            j[i] = r + 1

            # снова уничтожаем всё, что можем
            m = min(cur_e, a[i])
            t += m
            cur_e -= m
            a[i] -= m

    # идём в другие комнаты за энергией, если можно и нужно
    while a[i] > 0 and brain_on_cnt < 1000 and len(heap) > 0:
        brain_on_cnt += 1
        cur_e = -heap[0][0]
        id = -heappop(heap)[1]
        t += (i - id) * 2 + k[id] + 1 - j[id]
        j[id] += 1
        if j[id] <= k[id]:
            heappush(heap, [-min(e, j[id] * (k[id] + 1 - j[id])), -id])

        # снова уничтожаем всё, что можем
        m = min(cur_e, a[i])
        t += m
        cur_e -= m
        a[i] -= m


    # если в комнате остались предметы
    if a[i] > 0:
        print(-1)  # то Николай потерялся
        exit(0)

    # иначе осталось добавить в очередь с приоритетом
    # информацию о комнате, если в ней есть энергетики
    if j[i] <= k[i]:
        heappush(heap, [-min(e, j[i] * (k[i] + 1 - j[i])), -i])

# и вывод ответа в конце программы
print(t)
```

Задача III.1.3.2. Дрон Ло (100 баллов)

Это интерактивная задача.

После “Туннеля ярости” наш Николай вернулся в реальную жизнь и приехал домой к своей Вике – вот кто по-настоящему развлекался всё это время. Оказывается, недалеко от нас, на соседних планетах живут Зентради и Атария. И Вика помогает Зентради доставить подарок Атарии.

– Что, кто, кому, как? Стоп. Ничего не понимаю. Давай с начала.

– С начала. Вчера Зентради от чистого сердца подарил мне инопланетного дрона. Причин он не назвал – кто этих инопланетян разберёт.

– Таак.

– Описание у дрона непонятное. Ну, на инопланетном языке. Но я потыкалась и немного поняла, как он работает. Я даже дала дрону имя – я зову его Дрон Ло.

– Прелестно.

– Подожди. Сегодня Зентради позвонил и попросил помочь ему. Он забыл в Дрон Ло подарок для Атарии и просит, чтобы Дрон Ло сбросил его в определённую точку планеты Атарии. Сказал, что Дрон Ло знает координаты. Я, конечно же, согласилась и ещё раз поблагодарила за замечательный подарок. Но с функциями полёта я совсем запуталась. Пыталась перезвонить Зентради, но он не берёт трубку.

И на данный момент без инопланетян, которые куда-то подевались и не отвечают на звонки, Вика мало что может понять. Но пару функций она освоила:

- Можно сделать запрос о любой точке на планете Атарии. В качестве ответа на запрос придёт расстояние от этой точки до точки, куда нужно сбросить подарок;
- Можно отправить Дрон Ло сбросить подарок в заданную нами точку.

Все точки планеты Атарии проверять не имеет смысла, так как Вика определила регион, в который нужно сбрасывать подарок и этот регион можно представить в виде двумерной плоскости.

Помогите Вике определить, куда нужно отправить Дрон Ло, и отправить его. Постарайтесь при этом сделать не очень много запросов.

Протокол взаимодействия

В данной задаче вам предстоит работать не со стандартным вводом-выводом, а со специальной программой – интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

Каждый запрос вашей программы должен состоять из одной строки следующего вида: `? x y` , где x и y ($-10^{18} \leqslant x, y \leqslant 10^{18}$) – дробные числа, обозначающие абсциссу и ординату точки, ответ на запрос о которой вы хотите узнать. Ответом на запрос будет расстояние от этой точки до точки, куда нужно сбросить подарок.

Когда вы решите отправить Дрон Ло сбросить подарок, выведите `! x y`, где x и y ($-10^{18} \leqslant x, y \leqslant 10^{18}$) – дробные числа, обозначающие абсциссу и ординату точки, куда вы решили отправить Дрон Ло. После этого вывода ваша программа должна завершиться и ничего больше не выводить в стандартный поток вывода.

Подарок будет считаться доставленным, если расстояние между точкой, куда вы отправили Дрон Ло, и точкой, куда отправить его было нужно, не превышает 1 (единицы).

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждой выведенной команды. На C++ для printf надо использовать функцию `fflush(stdout)`, для cout – `cout « flush` или `cout.flush()`; на Java для System.out вызов `System.out.flush()`, для PrintWriter pw – `pw.flush()`; на Pascal – `flush(output)`; на Python для sys.stdout – sys.stdout.flush(), для print – print(..., flush=True).

Примеры

Пример №1

Стандартный ввод
1.812474491323 1.234868311003 0.4356714247987 0.355991865445 0.1204635151743
Стандартный вывод
? 0.53264 0.48924 ? 0.87407 1.27174 ? 1.86586 1.02896 ? 1.77805 1.54036 ? 2.20679 1.33373 ! 2.20679 1.33373

Пояснения к примеру

В примере подарок нужно было сбросить в точку $(2.1, 1.39)$.

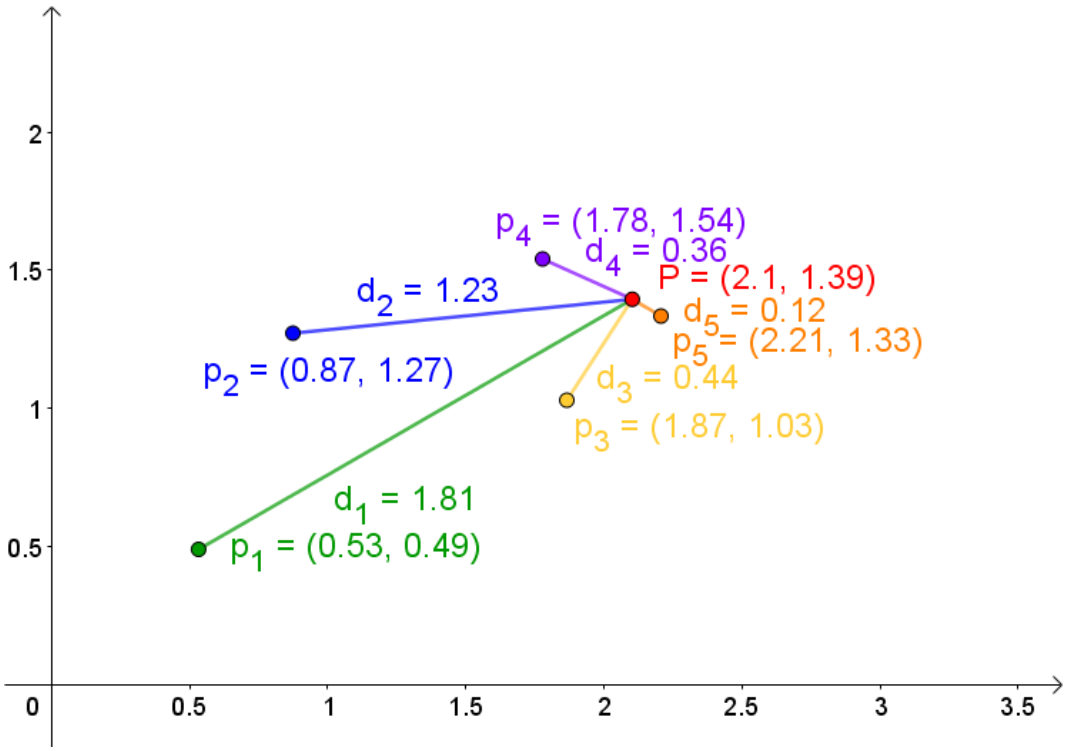


Рис. III.1.3: Рисунок к примеру 1

Способ оценки работы

В таблице ниже k – это максимальное количество запросов, которое вы можете сделать в тесте, R – расстояние от точки с координатами $(0, 0)$ до точки, куда нужно сбросить подарок, x и y – координаты точки, куда требуется сбросить подарок.

Если решение совершит более k запросов, то решение получит вердикт `Wrong Answer`.

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи
0	0	Тест из условия; $k = 100\,000$	–
1	20	$R \leqslant 100, k = 100\,000$	0
2	20	$R \leqslant 1000, k = 10\,000$	0, 1
3	15	$R \leqslant 10\,000, k = 1000$	0, 1, 2
4	15	$R \leqslant 10^6, k = 500$	0, 1, 2, 3
5	10	$\lvert x\rvert, \lvert y\rvert \leqslant 10^8, k = 100$	0, 1, 2, 3, 4
6	20	$\lvert x\rvert, \lvert y\rvert \leqslant 10^9, m \leqslant k \leqslant 100$	0, 1, 2, 3, 4, 5

Баллы за подзадачу будут начислены в случае прохождения всех тестов необходимых подзадач и всех тестов самой подзадачи.

Но поскольку и тут не обошлось без НЛО и теперь только инопланетяне знают число m, в последней подзадаче объявляется **потестовая оценка**. То есть баллы за последнюю подзадачу (с учётом того, что решение прошло все остальные подзадачи) будут начисляться пропорционально количеству пройденных тестов.

Для проверки результата использовался следующий код на языке C++

```
#include <bits/stdc++.h>

#include "testlib.h"

#
define forn(i, n) for (int i = 0; i < int(n); i++)

  using namespace std;

long double
dist(long double x1, long double y1, long double x2, long double y2) {
  return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
}

int
main(int argc, char * argv[]) {
  setName("Interactor for 'Collective password' problem.");
  registerInteraction(argc, argv);

  long double x = inf.readDouble(), y = inf.readDouble();
  int max_queries = inf.readInt();

  int queries = 0;
  while (true) {
    queries++;

    string q = ouf.readToken();

    if (q != "!" && q != "?")
      quitf(_wa, "String '%s' from stdin is incorrect", q.c_str());
```

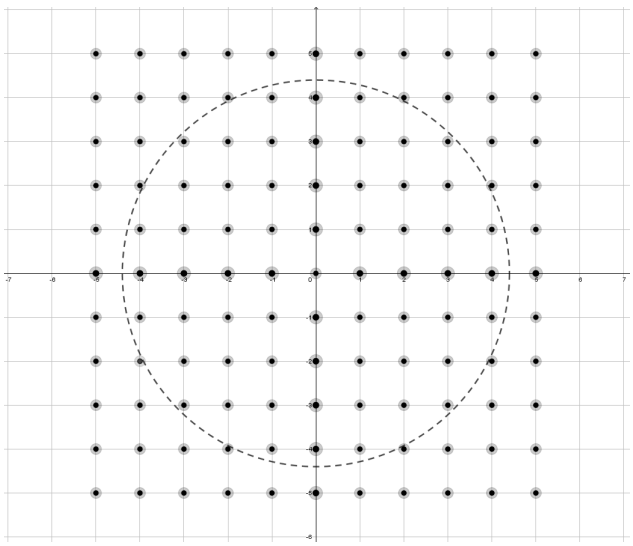
```
31
32      if (q == "?" && queries > max_queries)
33        quitf(_wa, "So mach queries");
34
35      if (q == "!") {
36        long double px = ouf.readDouble(-1e18, 1e18, "px"), py =
37          ouf.readDouble(-1e18, 1e18, "py");
38        long double pd = dist(x, y, px, py);
39
40        if (pd > 1.0)
41          quitf(_wa, "Wrong guest [%Lf ? %Lf, %Lf ? %Lf, %Lf]", px, x, py,
42            y, pd);
43
44        tout << fixed << setprecision(20) << queries << " " << px << " " <<
45          py << " " << pd << endl;
46        quitf(_ok, "%Lf %Lf is guessed (%Lf %Lf %Lf)", x, y, px, py, pd);
47      }
48
49      long double px = ouf.readDouble(-1e18, 1e18, "px"), py =
50        ouf.readDouble(-1e18, 1e18, "py");
51      cout << fixed << setprecision(20) << dist(x, y, px,
52        py) << endl << flush;
53    }
54  }
```

Решение

Подзадача 1

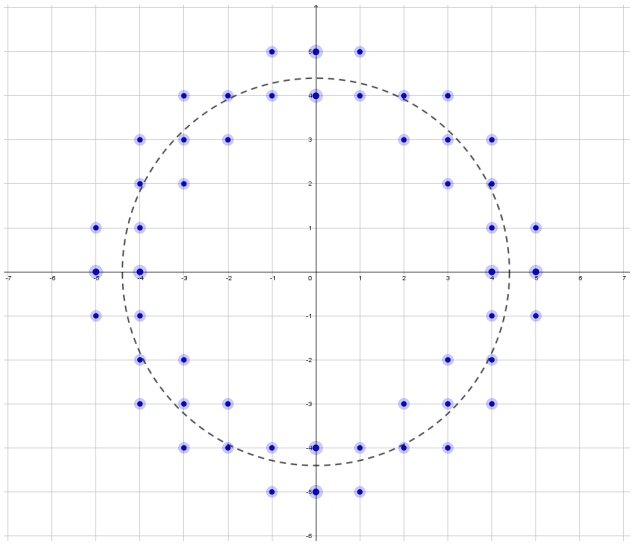
Заметим, что так как сбросить подарок можно в любую точку, расстояние от которой до загаданной не превышает 1, то существует подходящая точка с целочисленными координатами.

«Пробьём» во все такие точки следующего вида:

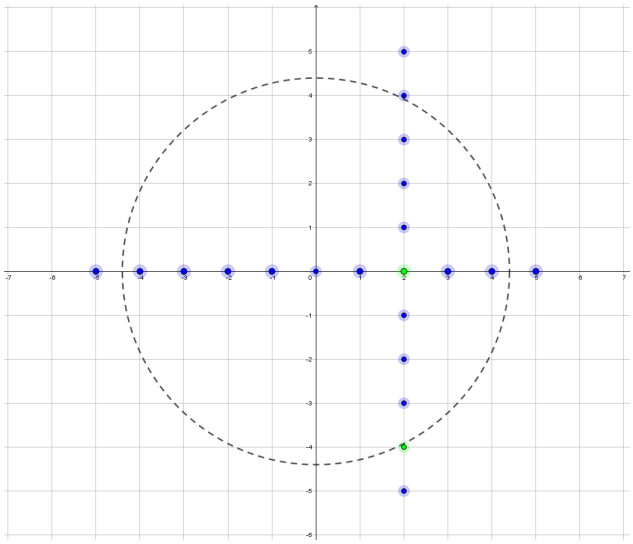


Подзадача 2

Заметим, что достаточно запрашивать про точки, которые находятся близко к окружности.

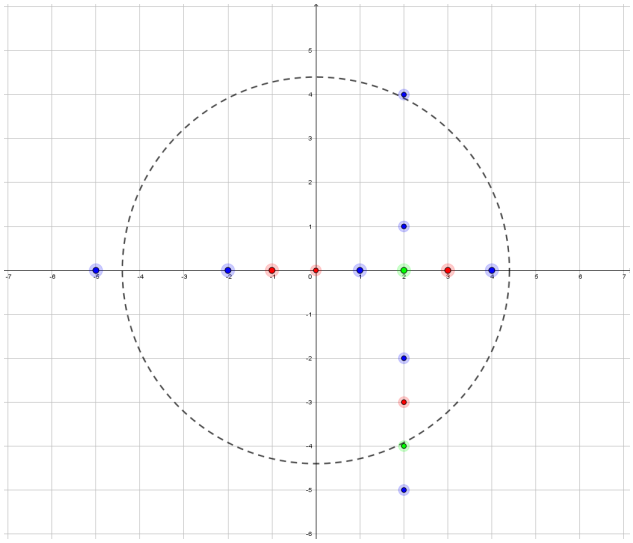


Либо можно найти ближайшую точку на оси X, а потом «пройтись» параллельно оси Y.



Подзадача 3

То же самое можно сделать перебором с шагом.



Подзадачи 4 и 5

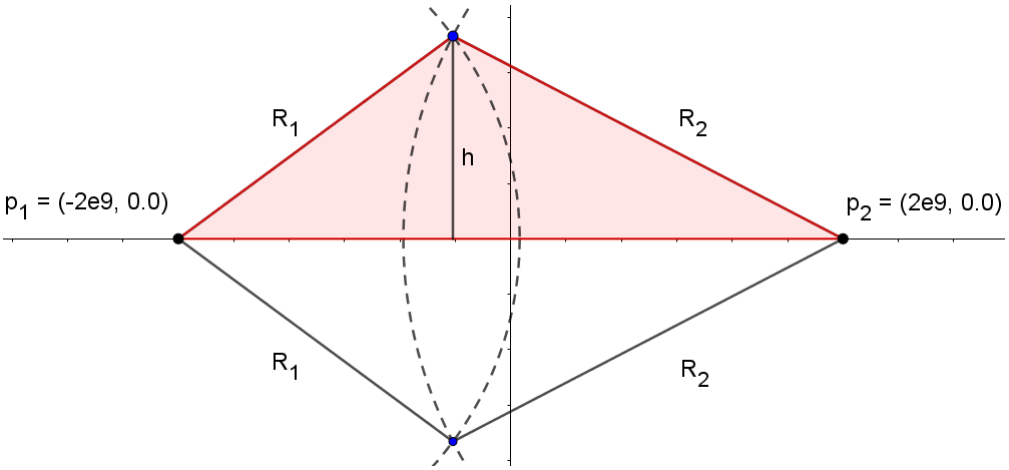
Заметим, что:

1. Найти ближайшую точку на оси X можно тернарным поиском.
2. Расстояние от ближайшей точки на оси X до загаданной точки примерно равно модулю координаты загаданной точки по оси Y
3. Чтобы не запрашивать о некоторых точках несколько раз, можно, например, сохранять результаты запросов в словарь

Полное решение

Можно сделать два запроса и найти точки пересечения двух окружностей (таким образом, $m = 3$).

Для этого можно, например, воспользоваться формулой Герона. Правда при такой версии решения требуется точность вычислений больше, чем могут предложить стандартные типы данных для работы с дробными числами.



$p = \frac{R_1+R_2+4\cdot 10^9}{2}$

$\frac{h\cdot 4\cdot 10^9}{2} = h \cdot 2 \cdot 10^9 = \sqrt{p \cdot (p-R_1) \cdot (p-R_2) \cdot (p-4\cdot 10^9)}$

$|y| = h = \sqrt{p \cdot (p-R_1) \cdot (p-R_2) \cdot (p-4\cdot 10^9)}/(2 \cdot 10^9)$

$x = \sqrt{R_1^2 - h^2} - 2 \cdot 10^9$

Пример программы-решения

Ниже представлено решение на языке Python 3

```
from decimal import Decimal

maxDist = Decimal(2e9)
print('?', -maxDist, 0, flush=True)
distLeft = Decimal(input())

print('?', maxDist, 0, flush=True)
distRight = Decimal(input())

p = (distLeft + distRight + Decimal(2.0) * maxDist) / Decimal(2.0)
S = (p * (p - distLeft) * (p - distRight) * (p - Decimal(2.0) * maxDist)).sqrt()
```

```
h = S / maxDist
x = (distLeft * distLeft - h * h).sqrt() - maxDist

print('?', x, h, flush=True)
dist = Decimal(input())

print('!', x, (h if dist < Decimal(1.0) else -h), flush=True)
```

Задача III.1.3.3. Нестандартное лечение (100 баллов)

Никто не хочет стать пациентом психиатрических больниц. Но иногда получается так, что другого выхода нет – человека нужно лечить. Психиатрическая больница имени Гаценко – это пока единственное место, где применяют новый метод с нейровирусом. Собственно, на досуге врачи этой больницы и изобрели этот полезный вирус и метод его применения.

Особенность этого нейровируса в том, что он временно "поглощает" память. При введении лучшего штамма этого нейровируса вся имеющаяся память больного сводится к отдельным фрагментам воспоминаний, имеющим ассоциации с другими фрагментами. Информация о каждом из фрагментов оказывается в отдельном нейроне мозга (то есть один такой нейрон хранит информацию только об одном фрагменте), а существование ассоциации между двумя фрагментами обеспечивают связи, своего рода мосты, между соответствующими нейронами. Главное, что если представить, что эти нейроны (фрагменты воспоминаний) – вершины графа, а связи (ассоциации) – это рёбра, то получается дерево. Врачи называют его деревом твёрдых ассоциаций.

Чтобы окончательно свести нашего больного с ума (временно, в чём и заключается метод врачей) достаточно сделать так, чтобы не существовало фрагмента, из которого по ассоциациям можно пройтись **цепочкой** более чем по k другим фрагментам, не "заходя" в один и тот же фрагмент более одного раза.

Как только эта цель достигается, оставшиеся нейроны с фрагментами воспоминаний отмирают, человек окончательно теряет связь с реальностью и начинается процесс восстановления памяти, которая, как показала практика, становится лучше прежнего. При восстановлении верно определяется, является воспоминание реальным или оно является плодом фантазии больного, а граница разделения реального и иллюзорного становится чётче. Именно поэтому этот метод рекомендуется людям с болезнями, вызванными неправильной работой памяти и мышления.

Для того, чтобы разрушить память до конца, у врачей есть специальный вирус-киллер, один экземпляр которого может добраться до любого, но только до одного заранее заданного нейрона и избавиться от него и связей, соединяющих его с другими нейронами. Однако врачи не разбираются в теории графов и тратят много вирусов-киллеров почём зря.

Помогите врачам минимизировать количество экземпляров вируса-киллера, которое они потратят на больных.

Формат входных данных

В каждом тесте описывается состояние памяти после применения нейровируса ровно одного больного.

В первой строке находятся числа n и k ($1 \leqslant n, k \leqslant 100\,000$) — число фрагментов воспоминаний и число k, описание которого ищите на предыдущей странице. Сами фрагменты воспоминаний пронумерованы от 1 до n.

Каждая из следующих $n - 1$ строк содержит по два целых числа u и v ($1 \leqslant u, v \leqslant n, u \neq v$), говорящих о существовании ассоциации между двумя фрагментами.

Гарантируется, что эти данные образуют дерево твёрдых ассоциаций.

Формат выходных данных

В первой строке выведите минимальное количество экземпляров вируса-киллера, которое нужно потратить на больного.

Во второй строке выведите номера нейронов, к которым нужно послать вирусов-киллеров.

Если подходящих списков номеров нейронов несколько – выведите любой.

Примеры

Пример №1

Стандартный ввод
4 2 1 2 2 3 3 4

Стандартный вывод
1 3

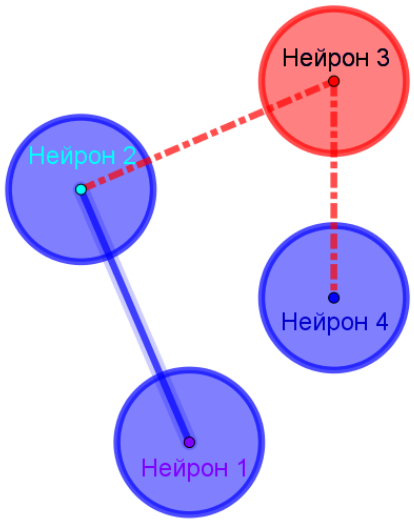


Рис. III.1.4: Рисунок к примеру 1

Пример №2

Стандартный ввод
6 3 1 2 2 3 2 4 2 5 4 6

Стандартный вывод
1 6

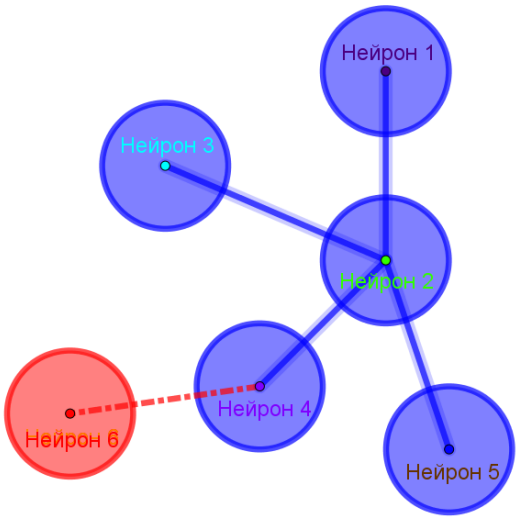


Рис. III.1.5: Рисунок к примеру 2

Пример №3

Стандартный ввод
6 2 1 2 2 3 2 4 2 5 4 6

Стандартный вывод
1 2

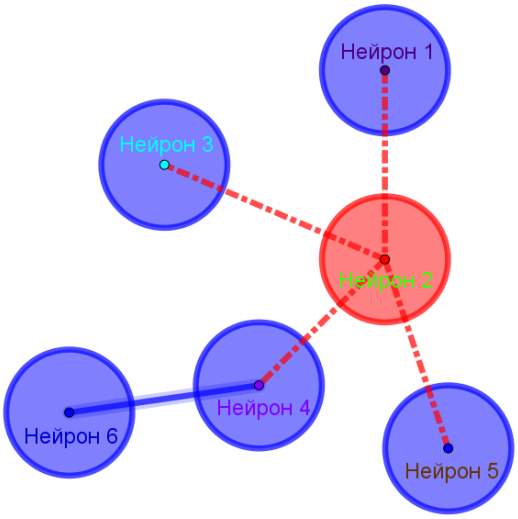


Рис. III.1.6: Рисунок к примеру 3

Пояснение к примерам

Ответом в первом примере мог быть и второй нейрон, но не первый или четвёртый, так как в этих случаях остаётся цепочка из трёх нейронов.

Ответ в третьем примере также является ответом и для второго, но ответ второго теста не является ответом для третьего, так как при отмирании 6-го нейрона в оставшемся дереве есть цепочка из трёх нейронов (например, $3 \leftrightarrow 2 \leftrightarrow 5$).

Способ оценки работы

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи
0	0	Тесты из условия	–
1	20	$n \leqslant 15$	0
2	15	Каждый нейрон имеет не более двух связей	–
3	15	$k = 1$	–
4	15	$k = 2$	3
5	10	$n \leqslant 100$	0, 1
6	10	$n \leqslant 1000$	0, 1, 5
7	15	Нет дополнительных ограничений	0, 1, 2, 3, 4, 5, 6

Баллы за подзадачу будут начислены в случае прохождения всех тестов необходимых подзадач и всех тестов самой подзадачи.

Для проверки результата использовался следующий код на языке C++

```cpp
#include "testlib.h"

#include <bits/stdc++.h>

using namespace std;

int main(int argc, char * argv[]) {
  registerTestlibCmd(argc, argv);

  int n = inf.readInt(1, 100000, "n");
  inf.readSpace();
  int k = inf.readInt(1, 100000, "k");
  inf.readEoln();
  vector < vector < int > > edges(n);
  for (int i = 0; i < n - 1; i++) {
    int u = inf.readInt(1, n, "u") - 1;
    inf.readSpace();
    int v = inf.readInt(1, n, "v") - 1;
    inf.readEoln();
    edges[u].push_back(v);
    edges[v].push_back(u);
  }
  inf.readEof();

  long long ja = ans.readLong();
  ans.readEoln();
  vector < bool > used(n, 0);
  for (int i = 0; i < ja; i++) {
    int v = ans.readInt(1, n, "next v") - 1;
    if (used[v])
      quitf(_fail, "wrong sequence (double %d)", v + 1);
    used[v] = 1;
  }

  vector < int > cnt(n, 0), len(n, 0);
  queue < int > q;

  for (int i = 0; i < n; i++) {
    if (used[i]) continue;
    for (int j: edges[i]) {
      if (!used[j]) {
        cnt[i]++;
      }
    }
  }
  for (int i = 0; i < n; i++) {
    if (!used[i] && cnt[i] <= 1) {
      used[i] = true;
      q.push(i);
    }
  }

  int answer = 0;
  while (!q.empty()) {
    int v = q.front();
    q.pop();
    int max1 = 0, max2 = 0;
    for (int u: edges[v]) {
      if (len[u] > 0) {
        if (len[u] > max1) {
```

```
          max2 = max1;
          max1 = len[u];
        } else if (len[u] > max2) {
          max2 = len[u];
        }
      } else if (!used[u] && --cnt[u] == 1) {
        used[u] = true;
        q.push(u);
      }
    }
    answer = max(answer, max1 + 1 + max2);
    len[v] = max1 + 1;
  }

  if (answer > k)
    quitf(_fail, "wrong sequense (%d > %d)", answer, k);

  long long pa = ouf.readLong();
  ouf.readEoln();
  if (ja < pa)
    quitf(_wa, "wrong answer");
  fill(used.begin(), used.end(), 0);
  for (int i = 0; i < pa; i++) {
    int v = ouf.readInt(1, n, "next v") - 1;
    if (used[v])
      quitf(_wa, "wrong sequence (double %d)", v + 1);
    used[v] = 1;
  }

  fill(cnt.begin(), cnt.end(), 0);
  fill(len.begin(), len.end(), 0);

  for (int i = 0; i < n; i++) {
    if (used[i]) continue;
    for (int j: edges[i]) {
      if (!used[j]) {
        cnt[i]++;
      }
    }
  }
  for (int i = 0; i < n; i++) {
    if (!used[i] && cnt[i] <= 1) {
      used[i] = true;
      q.push(i);
    }
  }

  answer = 0;
  while (!q.empty()) {
    int v = q.front();
    q.pop();
    int max1 = 0, max2 = 0;
    for (int u: edges[v]) {
      if (len[u] > 0) {
        if (len[u] > max1) {
          max2 = max1;
          max1 = len[u];
        } else if (len[u] > max2) {
          max2 = len[u];
        }
```

```
121         } else if (!used[u] && --cnt[u] == 1) {
122           used[u] = true;
123           q.push(u);
124         }
125       }
126       answer = max(answer, max1 + 1 + max2);
127       len[v] = max1 + 1;
128     }
129
130     if (answer > k)
131       quitf(_wa, "wrong sequense (%d > %d)", answer, k);
132
133     if (ja > pa)
134       quitf(_fail, "participant has better answer");
135
136     quitf(_ok, "%d numbers", n);
137
138 }
```

Решение

Подзадача 1

Переберём все возможные множества удалённых вершин и для каждого проверим, есть ли в полученном лесе (то есть графе, каждая компонента связности которого является деревом) путь длины более чем k

Подзадача 2

Когда с каждой вершиной дерева связано не более двух других вершин, дерево превращается в «цепочку».

Одним из решений для такого графа является удаление каждой $k+1$-й вершины на пути от одного из листьев.

Иллюстрация для $k = 2$:

Подзадача 3

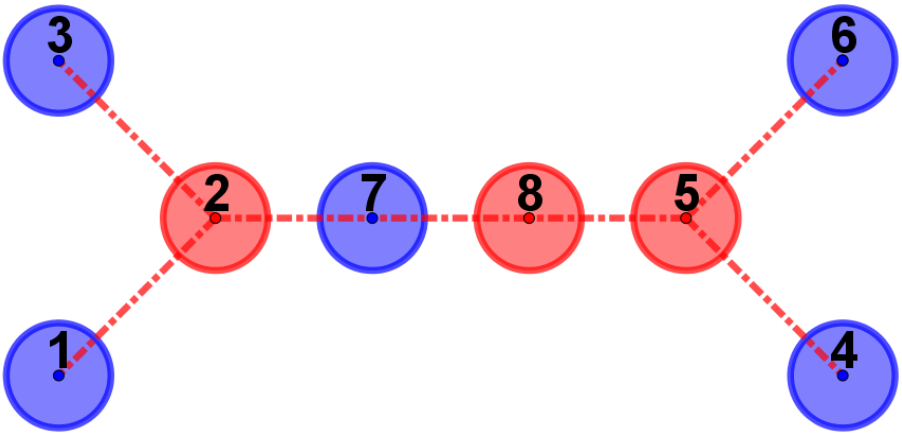
При $k = 1$ в итоговом графе каждая компонента связности будет состоять из одной вершины.

Таким образом, задача сводится к задаче о независимом множестве вершин дерева.

Для решения этой задачи можно применить жадный алгоритм, который кратко можно изложить следующим образом:

Пока в графе есть вершины, будем по одному удалять листья и связанные с ними вершины.

Вершины, которые удалялись не из-за того, что являются листьями, являются одним из оптимальных ответов для нашей задачи.



Вершины пронумерованы в порядке удаления.

Полное решение

Задачу в терминах теории графов можно сформулировать так:

Дано дерево из n вершин и число k. Нужно удалить как можно меньше вершин, чтобы в оставшемся графе не существовало пути длиной больше чем k.

Давайте решим задачу для вершины v и ее поддерева.

Для начала нужно выполнить условие для поддеревьев детей вершины v, то есть в поддеревьях детей вершины v нет пути длиной больше чем k.

Теперь предположим, что мы нашли ответ такой, что удалено минимальное количество вершин и последняя удаленная вершина выбрана так, чтобы минимизировать длину остаточного пути из соответствующего поддерева.

Теперь, когда мы решили задачу для поддеревьев детей вершины v, у нас есть какие-то остаточные пути из этих поддеревьев. Пусть два самых длинных из них имеют длины mx_1 и mx_2 соответственно (если какого-то пути не существует, то будем считать, что его длина равна нулю). Если мы оставим вершину v в графе, то останется и путь длиной $mx_1 + mx_2 + 1$. Если $mx_1 + mx_2 + 1 > k$, то мы обязаны удалить вершину v; иначе – можем ее оставить. Если мы удаляем вершину v, то длина остаточного пути, идущего из вершины v в её поддерево, будет считаться равной нулю. Иначе она равна $mx_1 + 1$.

Запустив данное решение для любой вершины в качестве корня дерева, можно решить задачу для всего дерева (удобно использовать обход в глубину).

Докажем оптимальность такого решения методом математической индукции.

Рассмотрим поддерево из одной вершины. Очевидно, что алгоритм отработает оптимально: ничего не удалит и оставит минимальный по длине путь длины 1.

Рассмотрим поддерево вершины v. Для каждого поддерева детей вершины v мы решили задачу оптимально нашим алгоритмом. Докажем, что наш алгоритм найдёт оптимальное решение для всего поддерева вершины v.

Если длина самого длинного пути в поддереве вершины v превышает k, то мы обязаны удалить как минимум одну вершину. Наш алгоритм удаляет ровно одну вершину и это вершина v.

Предположим, что мы можем удалить какую-то вершину в поддереве вершины v так, чтобы длина самого длинного пути в поддереве вершины v не превышала k, и это не вершина v. Если мы удалим её вместо вершины v, то останется еще какой-то ненулевой остаточный путь из вершины v в её поддерево. А при равенстве количества удаленных вершин мы должны минимизировать длину остаточного пути. Отказ от минимизации длины остаточного пути, очевидно, может привести к нахождение неоптимального ответа.

Если предположить, что мы можем не удалять ещё одну вершину, а заменить удаление одной из включённых в ответ вершин на удаление вершины v, то мы приходим к противоречию: если мы можем отменить удаление одной из вершин в поддереве одного из детей вершины v, то найденный нами ответ для поддерева этого ребёнка вершины v неоптимален, так как мы можем уменьшить количество удалённых вершин в поддереве этого ребёнка вершины v.

Если длина самого длинного пути в поддереве вершины v не превышает k, то мы ничего не удаляем. Количество удаленных вершин оптимально. Несложно доказать, что длина остаточного пути вершины v тоже будет оптимальной (она равна $mx_1 + 1$ и так как длина пути mx_1 была оптимальной, то оставшийся путь тоже оптимален).

Пример программы-решения

Ниже представлено решение на языке Python 3

```
n, k = map(int, input().split())
graph = [[] for i in range(n)]
for i in range(n - 1):
    v, u = map(int, input().split())
    v -= 1
    u -= 1
    graph[v].append(u)
    graph[u].append(v)

deleted = [False] * n
used = [False] * n
cnt = [len(graph[i]) for i in range(n)]
len = [0] * n
q = [0] * n
ql, qr = 0, 0

for i in range(n):
    if cnt[i] < 2:
        used[i] = True
        q[qr] = i
        qr += 1

answer = 0
while ql < qr:
    v = q[ql]
    ql += 1
    max1, max2 = 0, 0
    for u in graph[v]:
        if len[u] > 0:
            if len[u] > max1:
                max1, max2 = len[u], max1
            elif len[u] > max2:
                max2 = len[u]
```

```
34          elif not used[u]:
35              cnt[u] -= 1
36              if cnt[u] == 1:
37                  used[u] = True
38                  q[qr] = u
39                  qr += 1
40
41      if max1 + 1 + max2 > k:
42          answer += 1
43          deleted[v] = True
44      else:
45          len[v] = max1 + 1
46
47  print(answer)
48  for i in range(n):
49      if deleted[i]:
50          print(i + 1, end=' ')
```

Командный практический тур

В командной части заключительного этапа участники должны спроектировать автономную систему управления групой роботов-погрузчиков для решения задач навигации на модели логистического центра с использованием алгоритмов компьютерного зрения.

Командная часть заключительного этапа проходит в течении 3,5 дней (всего 26 астрономических часов), которые включают работу по оснащению роботов необходимыми датчиками, программированию, пробные заезды на макете логистического центра, зачетные попытки.

Легенда

Недалёкое будущее, в автоматических логистических центрах практически нет людей, всю работу выполняют роботы-погрузчики, которыми управляет интеллектуальное ПО по распределению задач.

Несмотря на отсутствие человеческого фактора, не следует думать, что в таких центрах никогда не будет проблем. Форс-мажор может произойти в любой момент и система из роботов и ПО должна уметь справляться с такими ситуациями.

Поэтому для финальной задачи профиля «Интеллектуальные робототехнические системы» предлагается рассмотреть следующий эпизод: в логистическом центре произошла перезагрузка всех систем, что привело к сбросу информации о местоположениях работающих в данный момент роботов-погрузчиков, а также карты данной части логистического центра.

В начале выполнения задания считается, что робототехнические устройства активируются в каких-то секторах (каждый в своём) логистического центра. Один из роботов выполняет считанные с arTag маркера команды, а затем ожидает прибытия второго робота в смежный сектор. Первый робот должен найти второго, доехав до смежного с ним сектора, а после вернуться в один из секторов старта. Структура логистического центра не известна заранее. При перемещении роботы не должны

сталкиваться, а также повреждать логистический центр.

Задача участников Олимпиады — разработать программу управления несколькими робототехническими устройствами для выполнения задания описанного выше.

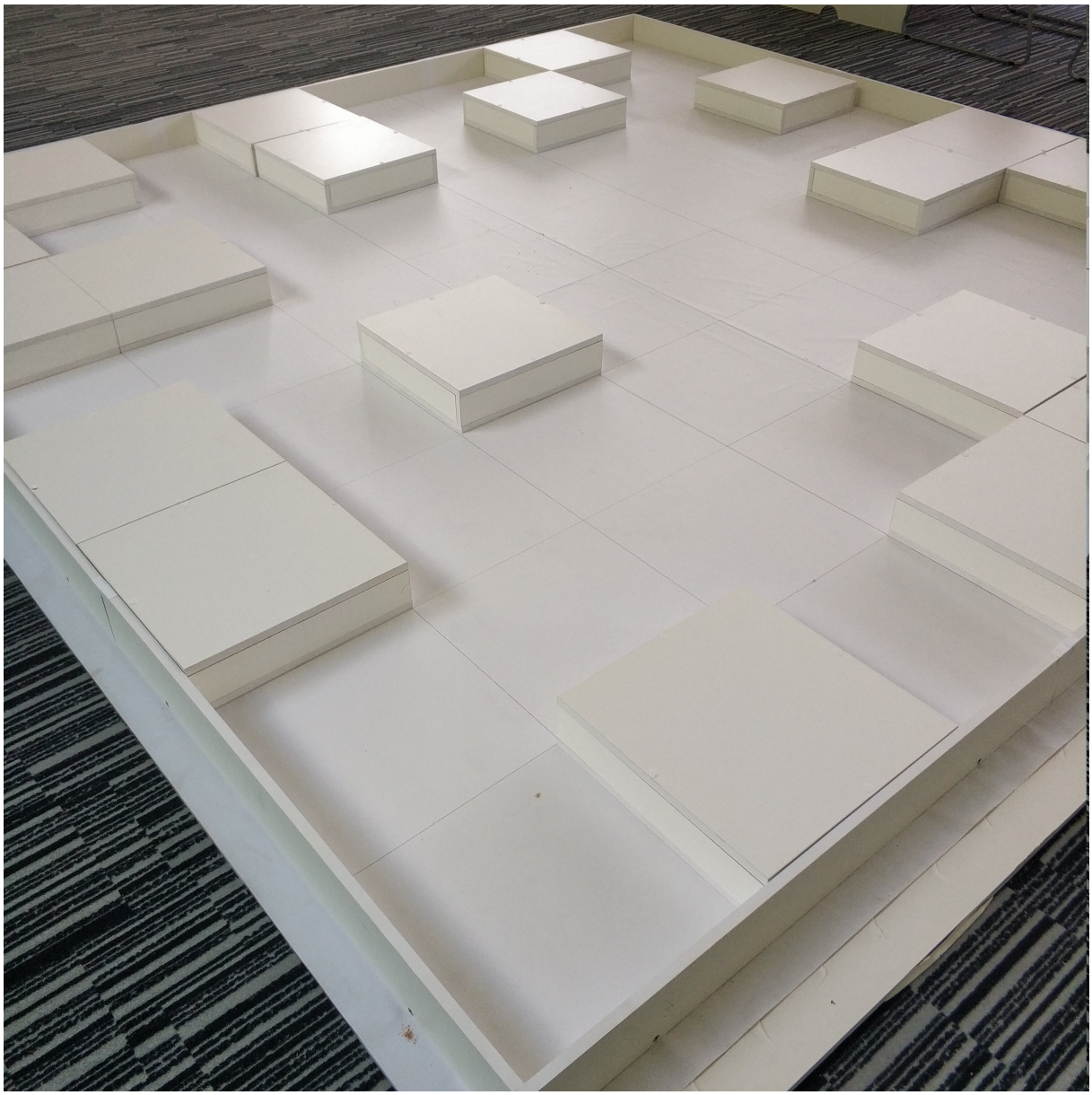


Рис. III.2.1: Полигон для запуска робототехнических устройств на финале Олимпиады НТИ

Набор заданий

Решение командной задачи разбито на четыре этапа. Первые три этапа итеративно подводят участников к решению полной финальной задачи, осуществляемому во время последнего четвёртого этапа. На каждом этапе в проверку решения заданий данного этапа входят:

- способность проверить гипотезу о работоспособности алгоритма через демонстрацию решения в симуляторе;

- полнота решения задания конкретного этапа;
- воспроизводимость результатов — робототехническое устройство участников должно неоднократно выполнить требуемые действия.

Первый этап

Задача: робототехническое устройство располагается в модели логистического центра. Ему необходимо проехать по маршруту в точку окончания работы. Структура логистического центра заранее неизвестна.

Включая содержательные задачи:

- Нахождение порогового значения для датчиков расстояния;
- Реализация алгоритмов перемещения в неизвестной среде.
- Реализация алгоритмов сегментации действий.

Второй этап

Задача: два робототехнических устройства должны определить своё местоположение в модели логистического центра (структура которого заранее неизвестна) из случайных, заранее неизвестных точек старта.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов локализации для группы робототехнических устройств;
- Реализация алгоритмов перемещения в неизвестной среде.
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

Третий этап

Задача: робототехническое устройство должно проехать по модели логистического центра заданные команды. Команды робот должен определить самостоятельно по ARTag метке, расположенной на стеллаже, прилегающему к сектору старта.

Включая содержательные задачи:

- Калибровка камеры робототехнического устройства;
- Реализация алгоритмов компьютерного зрения: считывание ARTag меток без использования дополнительных библиотек;
- Декодирование бинарного кода с использованием кода Хэмминга.

Четвёртый этап

Задача: два роботехнических устройства располагаются в неизвестной среде. Первый робот должен обнаружить второго робота и после вернуться в одну из начальных позиций. Второй робот должен выполнить заданные команды, переданные через ArTag маркер, расположенный в секторе старта.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов составления карты неизвестной местности и локализация на данной местности;
- Реализация алгоритмов компьютерного зрения: считывание ARTag меток без использования дополнительных библиотек;
- Декодирование бинарного кода, с использованием кода Хэмминга;
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

Описание модели логистического центра

Полигон - квадратное поле 3200 × 3200 мм., разделенное на квадратные сектора 400 × 400 мм. Некоторые сектора недоступны для посещения робототехническим устройством и представляют из себя модель стеллажа высотой 100 мм.

Полигон окружен бортом высотой 100 мм. Конфигурация полигона изменяется и определяется непосредственно перед каждым запуском роботов.

На стеллаже, прилегающего одной из четырех сторон к сектору активации (старта), на высоте от 100-150 мм от уровня поверхности поля закреплен ARTag маркер (https://goo.gl/WaTFMB), определяющий серию команд, предназначенную для выполнения роботом. Размер маркера - 40 × 40 мм. Маркер обращен лицевой стороной внутрь сектора старта данного робота. Конкретная высота расположения маркеров определяется в первый день финала и остается постоянной на все дни финального этапа. При этом допустимая погрешность установки маркеров ±5 мм. Пример расположения маркера на стелаже представлен на рисунке III.2.2

Рис. III.2.2: Стеллаж с установленным ARTag маркером

Маркер состоит из 8 × 8 элементов одинакового размера. Элементы маркера, расположенные по его границе — всегда черные. Четыре элемента, находящиеся в углах внутреннего 6 × 6 квадрата определяют ориентацию маркера таким образом, что только один из них — белый. Оставшиеся 32 элемента маркера кодируют число

по следующему правилу: если элемент черный, то он обозначает 1, если белый, то 0 при этом самый первый элемент — старший бит закодированного числа. Нумерация элементов относительно ориентационных элементов обозначена на рисунке III.2.3.

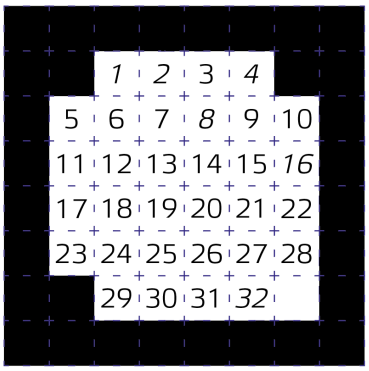


Рис. III.2.3: Нумерация элементов маркера относительно ориентационных элементов

Закодированное на маркере двоичное число использует код Хэмминга (https://habr.com/ru/post/140611/), в котором 26 информационных битов и 6 контрольных:

1 Первый контрольный бит;
2 Второй контрольный бит;
3 Старший бит первой команды $K_1 (0 \leqslant K_1 \leqslant 3)$;
4 Третий контрольный бит;
5 Младший бит первой команды $K_1 (0 \leqslant K_1 \leqslant 3)$;
6 Старший бит второй команды $K_2 (0 \leqslant K_2 \leqslant 3)$;
7 Младший бит второй команды $K_2 (0 \leqslant K_2 \leqslant 3)$;
8 Четвёртый контрольный бит;
9 Старший бит третьей команды $K_3 (0 \leqslant K_3 \leqslant 3)$;
10 Младший бит третьей команды $K_3 (0 \leqslant K_3 \leqslant 3)$;
11 Старший бит четвёртой команды $K_4 (0 \leqslant K_4 \leqslant 3)$;
12 Младший бит четвёртой команды $K_4 (0 \leqslant K_4 \leqslant 3)$;
13 Старший бит пятой команды $K_5 (0 \leqslant K_5 \leqslant 3)$;
14 Младший бит пятой команды $K_5 (0 \leqslant K_5 \leqslant 3)$;
15 Старший бит шестой команды $K_6 (0 \leqslant K_6 \leqslant 3)$;
16 Пятый контрольный бит;
17 Младший бит шестой команды $K_6 (0 \leqslant K_6 \leqslant 3)$;
18 Старший бит седьмой команды $K_7 (0 \leqslant K_7 \leqslant 3)$;
19 Младший бит седьмой команды $K_7 (0 \leqslant K_7 \leqslant 3)$;
20 Старший бит восьмой команды $K_8 (0 \leqslant K_8 \leqslant 3)$;
21 Младший бит восьмой команды $K_8 (0 \leqslant K_8 \leqslant 3)$;
22 Старший бит девятой команды $K_9 (0 \leqslant K_9 \leqslant 3)$;
23 Младший бит девятой команды $K_9 (0 \leqslant K_9 \leqslant 3)$;
24 Старший бит десятой команды $K_{10} (0 \leqslant K_{10} \leqslant 3)$;
25 Младший бит десятой команды $K_{10} (0 \leqslant K_{10} \leqslant 3)$;
26 Старший бит одиннадцатой команды $K_{11} (0 \leqslant K_{11} \leqslant 3)$;
27 Младший бит одиннадцатой команды $K_{11} (0 \leqslant K_{11} \leqslant 3)$;

28 Старший бит двенадцатой команды $K_{12}(0 \leqslant K_{12} \leqslant 3)$;
29 Младший бит двенадцатой команды $K_{12}(0 \leqslant K_{12} \leqslant 3)$;
30 Старший бит тринадцатой команды $K_{13}(0 \leqslant K_{13} \leqslant 3)$;
31 Младший бит тринадцатой команды $K_{13}(0 \leqslant K_{13} \leqslant 3)$;
32 Шестой (последний) контрольный бит;

Данные команды задают действия, которые должен выполнить робот-погрузчик:

0 Действие не требуется (“N”);
1 Роботу необходимо повернуть налево на 90° внутри данного сектора (“L”);
2 Роботу необходимо повернуть направо на 90° внутри данного сектора (“R”);
3 Роботу необходимо проехать в следующий по направлению движения сектор (“F”);

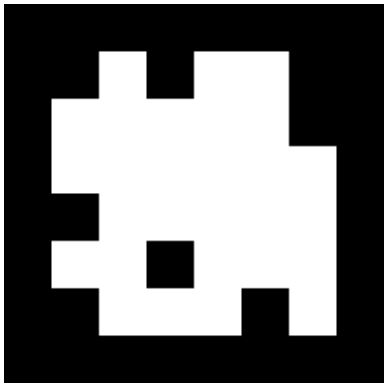


Рис. III.2.4: Маркер с закодированным значением - $10111111101111110111111101111110_2$

Маркер на рисунке III.2.4 кодирует число $10111111101111110111111101111110_2$ — $K_1 = 3$, $K_2 = 3$, $K_3 = 2$, $K_4 = 3$, $K_5 = 3$, $K_6 = 2$, $K_7 = 3$, $K_8 = 3$, $K_9 = 3$, $K_{10} = 2$, $K_{11} = 3$, $K_{12} = 3$, $K_{13} = 3$.

Гарантируется, что робот начинает свое движение в секторе, к сторонам которого прилегает как минимум один стеллаж.

Сектора активаций роботов (стартов) никак не обозначаются на поле и определяются непосредственно перед каждым заездом робота.

В некоторых задачах используются координаты, в таком случае, сектор $(0, 0)$ находится в левом верхнем углу, ось X направлена вправо, ось Y – вниз.

Описание конструктора

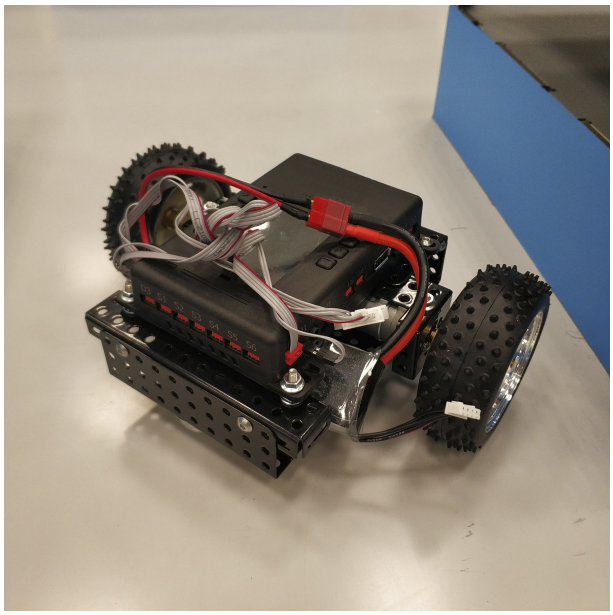


Рис. III.2.5: Мобильная платформа TRIK в сборе

В первый день финального тура каждой команде выдаются два комплекта:

- Инструкция по сборке наземной платформы на базе конструктора TRIK. Мобильная платформа построена по принципу дифференциального управления. Физические размеры платформы позволяют совершать все маневры внутри одного сектора модели логистического центра без касания со стенками стеллажей или бортов.
- Комплект деталей для сборки мобильной наземной платформы на базе конструктора TRIK (блок управления TRIK, аккумулятор, два мотора с энкодерами на датчиках Холла, колеса).
- Комплект дополнительных деталей из конструктора TRIK и набор датчиков: 2 инфракрасных датчика дальности, 2 ультразвуковых датчика расстояния и 1 VGA-камера;
- комплект дополнительных деталей из конструктора TRIK;
- Ноутбуки с установленной TRIK Studio, каждой команде не более двух ноутбуков. При этом участники могут пользоваться своими ноутбуками.

Условия проведения

1. Из полученного набора датчиков команды могут выбирать те, с помощью которых, по мнению участников, можно решить задачу наиболее эффективным способом.
2. Команды могут вносить любые изменения в мобильные наземные платформы.
3. Участники во время командного этапа финального тура могут использовать интернет и заранее подготовленные библиотеки для решения задачи.

4. Участники не могут использовать помощь тренера, сопровождающего лица или привлекать третьих лиц для решения задачи.
5. Финальная задача формулируется участникам в первый день финального тура, но участники выполняют решение задачи поэтапно. Критерии прохождения каждого этапа формулируются для каждого дня финального тура. За подзадачи, решенные в конкретном этапе начисляются баллы. Баллы за подзадачи можно получить только в день, закреплённый за конкретным этапом.
6. Некоторые подзадачи строго требуют выполнение каких-то предыдущих подзадач. Выполнение данных подзадач без выполнения предыдущих допускается, однако данная попытка будет оценена в 0 баллов.
7. Во время рабочего времени команды могут проводить испытания на полигоне. Количество подходов, которое может сделать команда может быть ограничено в зависимости ограничений, накладываемых расписанием финального этапа Олимпиады.
8. Испытания на полигоне должны осуществляться так, чтобы не мешать другим командам, проводящим в это время свои испытания на полигоне. Для этого всем командам может быть назначено ограничение по времени, которое они могут тратить на одно испытание. После истечения этого времени, команда должна дать возможность проводить испытания следующей команде.
9. Часть подзадач необходимо будет решить в симуляторе TRIK Studio: команда получает 3 тестовых виртуальных полигона с соответствующими наборами входные данных для подготовки решения, в то время как приемка решения происходит на расширенном наборе полигонов для проверки универсальности управляющей программы. Начисление баллов за подзадачи может происходить только в тот этап, в котором данные подзадачи сформулированы.
10. У команды есть неограниченное количество попыток решения подзадач в симуляторе:
    10.1. Решения принимаются на проверку до истечения первых 6,5 часов работы в соответствующий соревновательный день. Время может меняться в зависимости от дня.
    10.2. До истечения 6,5 часов, команда должна загрузить свое решение на gitlab.com в соотвествующий репозиторий дня, внутри своей группы, доступ к которой участники получат в начале олимпиады.
    10.3. До истечения указанного времени команды могут изменять файл с решением сколько угодно раз. Проверяться будет всегда только последняя доступная версия.
    10.4. Зафиксировать последнее решение необходимо, создав Merge Request (MR).
    10.5. При создании Merge Request в качестве ответственного (assignee) укажите того, кто будет отвечать за приемку результатов.
11. Часть подзадач для реальных роботов может быть запрещена к приемке без успешного прохождения 60% всех тестов, предназначенных для проверки решения соответствующей подзадачи в симуляторе.
12. Каждый день финального тура за 2 часа (может варьироваться в зависимости от расписания) до конца выделенного рабочего времени команды должны сдать роботов в зону карантина. Время сдачи роботов в карантин может изменяться и зависит от количества команд и сложности подзадач, принимаемых в конкретный этап.
13. Перед сдачей робота в карантин команды должны загрузить на роботов управ-

ляющие программы, подготовленные для демонстрации решения задачи, а также ее копию в репозиторий, доступ к которому участники получат в начале соревновательного дня. Без программы, загруженной в репозиторий, команды не допускаются до проверки решения на реальном роботе.

14. Зафиксировать загрузку файлов, содержащих решения задачи на реальном роботе, необходимо при помощи MR.
15. Программа должна успешно собираться в pdf.
16. После момента, когда все роботы сданы в карантин, судьи по одной вызывают команды для приемки решения подзадач, закрепленных за этапом конкретного дня финального тура.
17. Может быть предусмотрено до двух попыток сдачи решения одной и той же подзадачи на реальных роботах. Конкретное количество попыток определяется в конкретных подзадачах.
18. После прохождения приемочных запусков, баллы набранные командой заносятся судьями в протокол. Один из участников команды расписывается за набранный результат, подтверждая согласие команды с оценкой проведенных запусков.
19. Роботы должны выполнять задание полностью автономно. Удаленное управление не допускается. Касание какого-либо робота участником команды после его старта во время приемочных запусков не допускается. Алгоритм, реализующий систему управления группой роботов, должен планировать свое выполнение, полагаясь только на информацию с датчиков.
20. Введение данных в программу до старта устройства (например, координат робота в начале работы) разрешается только для тех задач, где это явно прописано. Во всех других случаях введение данных в программу роботов перед запуском запрещено.
21. Для всех роботов программа должна быть одинаковой, допускается отличие лишь в разрешенных входных данных.
22. Если какая-то подзадача подразумевает считывание информации с элементов, расположенных на полигоне, запрещается при запуске роботов вводить информацию о положении этих элементов или значениях, которые данные элементы определяют.
23. Если во время приемочных запусков у судьи возникли сомнения о том, что задачи подэтапа решены корректно (роботы не выполняют задачу полностью автономно, участник вводит значения в каких-либо роботов перед запуском), то он вправе провести инспекцию кода. По результатам инспекции, судья вправе снять с команды баллы, набранные за данный этап.
24. Если во время приемочных запусков у судьи возникает ситуация, когда он не может однозначно решить выполняются ли критерии решения подзадачи, он вправе принять решение не в пользу команды.
25. В случае если возникает техническая проблема, независящая от участников, то по решению судей может быть предоставлена возможность перезапуска
26. Команда вправе обсуждать с судьей результаты приемочных запусков до вызова следующей команды, но финальное решение остается о начислении баллов остается за судьей.

Процедура проведения приемочных запусков и критерии оценки

Первый этап

1. Командам необходимо подготовить две задачи для симулятора:
   1.1. В качестве первой задачи участникам необходимо выполнить следующее:
      1.1.1. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Задача робота выполнить все команды переданные через входной файл. Робот выполнил все команды и вывел `finish` на всех проверочных полигонах.
      Входные данные: через файл `input.txt` управляющей программе передаются:
         A. В первой строке через пробел — команды которые необходимо выполнить, где:
            0 Действие не требуется (“N”);
            1 Роботу необходимо повернуть налево на 90° внутри данного сектора (“L”);
            2 Роботу необходимо повернуть направо на 90° внутри данного сектора (“R”);
            3 Роботу необходимо проехать в следующий по направлению движения сектор (“F”);

      Ожидаемый результат: После запуска программы робот выполнил все команды, остановился и вывел на экран `finish`.
   1.2. В качестве второй задачи:
      1.2.1. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Задача робота проехать из точки старта в точку финиша. Робот проехал из точки старта в точку финиша и вывел `finish` на всех проверочных полигонах. Координаты сектора старта, включая направление и финиша передаются через входной файл. Структура лабиринта не известна.
      Входные данные: через файл `input.txt` управляющей программе передаются:
         A. В первой строке через пробел — координаты старта X_s, Y_s, направление старта D_s ($0 \leqslant X_s, Y_s \leqslant 7$, $D_sin[0, 3]$):
            0 Робот направлен вверх (в сторону отрицательного направления оси Y);
            1 Робот направлен направо (в сторону положительного направления оси X);
            2 Робот направлен вниз (в сторону положительного направления оси Y);
            3 Робот направлен налево (в сторону отрицательного направления оси X);
         B. Во второй строке через пробел — координаты финиша X_f, Y_f, $0 \leqslant X_f, Y_f \leqslant 7$;

      Ожидаемый результат: После запуска программы робот доехал до финиша, остановился и вывел на экран `finish`.

1.3. Правила именования файлов с управляющей программой для проверки решений в симуляторе:
   1.3.1. Для первой задачи: `sim_1.js`;
   1.3.2. Для второй задачи: `sim_2.js`.

2. Конфигурация робота в симуляторе следующая:
   2.1. Конфигурация моторов:
      2.1.1. M3 — правый мотор;
      2.1.2. M4 — левый мотор.
   2.2. Конфигурация датчиков:
      2.2.1. D1 — ультразвуковой датчик, направленный вперёд;
      2.2.2. D2 — ультразвуковой датчик, направленный назад;
      2.2.3. A1 — инфракрасный датчик, направленный вправо;
      2.2.4. A2 — инфракрасный датчик, направленный влево.
3. Команде необходимо будет подготовить решения для двух разных подзадач для реальных роботов. На демонстрацию каждого решения предоставляется 2 попытки.
4. Все попытки осуществляются 18 марта.
5. За 5 мин до сдачи в карантин для 1ой подзадачи судья определяет количество и набор команд для выполнения роботом.
6. После сдачи в карантин для 1ой подзадачи судья определяет сектор старта и направление робота в секторе старта.
7. За 5 мин до сдачи в карантин для 2ой подзадачи судья определяет сектор старта, направление старта и сектор финиша для робота. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
8. Секторы старта могут быть различными в разных попытках. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
9. Правила именования файлов с программой управления:
   9.1. для первой попытки первой подзадачи: `real_1_1.js`;
   9.2. для второй попытки первой подзадачи: `real_1_2.js`;
   9.3. для первой попытки второй подзадачи: `real_2_1.js`;
   9.4. для второй попытки второй подзадачи: `real_2_2.js`.
10. Merge Request необходимо назвать следующим образом: “код команды_day1”. Например: “irs202000_day1”.
11. Максимальное время выполнения одной попытки для 1ой подзадачи — 2 минуты.
12. Максимальное время выполнения одной попытки для 2ой подзадачи — 5 минут.
13. Баллы за решение задач этапа:
   13.1. **Первая задача в симуляторе**: робот смог выполнить всю задачу на всех проверочных полигонах — 12 баллов.
   13.2. **Вторая задача в симуляторе**: робот смог выполнить всю задачу на всех проверочных полигонах — 20 баллов.
   13.3. **Первая подзадача на реальном роботе:** Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Задача робота выполнить все команды. Робот выполнил все команды и вывел `finish`. — 16 баллов.
   13.4. **Вторая подзадача на реальном роботе:** Робот устанавливается в мо-

дели логистического центра в заранее неизвестном секторе. Задача робота проехать из точки старта в точку финиша. Робот проехал из точки старта в точку финиша и вывел `finish`. Координаты сектора старта, включая направление и финиша передаются через входной файл. Структура лабиринта не известна. — 20 баллов.

14. Выводить `finish` следует не менее 10 секунд.
15. Баллы за первую подзадачу не начисляются, если не было частично засчитано решение первой задачи в симуляторе
16. Баллы за вторую подзадачу не начисляются, если не было частично засчитано решение второй задачи в симуляторе.
17. Баллы за все попытки в каждой подзадаче суммируются.
18. Выполнение всех критериев в каждой из двух попыток всех двух подзадач дает дополнительные 8 баллов.
19. Максимальное количество баллов за этап — 112.

Второй этап

1. Командам необходимо подготовить две задачи для симулятора:
   1.1. В качестве первой задачи участникам необходимо выполнить следующее:
      1.1.1. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Необходимо в процессе движения опредлить своё местоположение, остановиться и вывести на экран координаты робота в формате "(X, Y)" без пробелов и без кавычек. Направление старта передаётся через входной файл.
      Входные данные: через файл `input.txt` управляющей программе передаются:
         A. В первой строке — направление старта D_s:
            0 Робот направлен вверх (в сторону отрицательного направления оси Y);
            1 Робот направлен направо (в сторону положительного направления оси X);
            2 Робот направлен вниз (в сторону положительного направления оси Y);
            3 Робот направлен налево (в сторону отрицательного направления оси X);
      Ожидаемый результат: После запуска программы робот определил своё местоположение, остановился и вывел на экран `finish`.
   1.2. В качестве второй задачи:
      1.2.1. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Известно что один сектор данного логистического центра заблокирован другим роботом. Координаты этого сектора передаются через входной файл. Необходимо в процессе движения определить своё местоположение и приехать в сектор, смежный с сектором нахождения другого робота и вывести `finish`. Структура лабиринта заранее не известна.
      Входные данные: через файл `input.txt` управляющей программе передаются:
         A. В первой строке через пробел — направление старта D_s

0 Робот направлен вверх (в сторону отрицательного направления оси Y);

1 Робот направлен направо (в сторону положительного направления оси X);

2 Робот направлен вниз (в сторону положительного направления оси Y);

3 Робот направлен налево (в сторону отрицательного направления оси X);

B. Во второй строке через пробел — координаты расположения другого робота X, Y ($0 \leqslant X_s, Y_s \leqslant 7$);

Ожидаемый результат: После запуска программы робот доехал до одного из смежных секторов с сектором нахождения другого робота, остановился и вывел на экран `finish`.

1.3. Правила именования файлов с управляющей программой для проверки решений в симуляторе:

1.3.1. Для первой задачи: `sim1.js`;

1.3.2. Для второй задачи: `sim2.js`.

2. Конфигурация робота в симуляторе следующая:

2.1. Конфигурация моторов:

2.1.1. M3 — правый мотор;

2.1.2. M4 — левый мотор.

2.2. Конфигурация датчиков:

2.2.1. D1 — ультразвуковой датчик, направленный вперёд;

2.2.2. D2 — ультразвуковой датчик, направленный назад;

2.2.3. A1 — инфракрасный датчик, направленный вправо;

2.2.4. A2 — инфракрасный датчик, направленный влево.

3. Команде необходимо будет подготовить решения для одной задачи для реальных роботов. На демонстрацию решения предоставляется 2 попытки.

4. Все попытки осуществляются 19 марта.

5. Merge Request необходимо назвать следующим образом: “код команды_day2”. Например: “irs202000_day2”.

6. За 10 мин до сдачи в карантин судья определяет направление старта для робота. Данное значение команда должна внести в программу перед тем, как сдать робота в карантин.

7. Направление старта могут быть различными в разных попытках. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.

8. После сдачи в карантин судья определяет секторы старта обоих роботов.

9. Правила именования файлов с программой управления:

9.1. для первой попытки задачи для реального робота: `real_1_1.js`;

9.2. для второй попытки задачи для реального робота: `real_1_2.js`;

10. Максимальное время выполнения одной попытки - 7 минут.

11. Баллы за решение задач этапа:

11.1. **Первая задача в симуляторе**: робот смог определить своё местоположение на всех проверочных полигонах — 16 баллов.

11.2. **Вторая задача в симуляторе**: робот смог достичь смежного сектора

на всех проверочных полигонах — 20 баллов.

11.3. **Задача на реальном роботе:** Роботы располагаются в случайных секторах робототехнического полигона.

11.3.1. Первый робот определил своё местоположение на поле, остановился, издал звуковой сигнал, вывел на экран свои координаты в формате (X, Y). — 14 баллов.

11.3.2. Координаты второго робота также определены. Второй робот издал звуковой сигнал, вывел на экран свои координаты в формате (X, Y). — 10 баллов.

11.3.3. Во время решения посталенной задачи второй робот не покидал изначальных сектор. — 8 баллов.

12. Выводить значение метки следует не менее 10 секунд.
13. Баллы за задачу на реальном роботе не начисляются, если не было частично засчитано решение первой задачи в симуляторе.
14. Баллы за все попытки на реальном роботе суммируются.
15. Выполнение всех критериев в каждой из двух попыток всех двух подзадач дает дополнительные 8 баллов.
16. Максимальное количество баллов за этап — 108.

Третий этап

1. В качестве задачи для симулятора участникам необходимо выполнить следующее:

1.1. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. При этом структура логистического центра не известна заранее. Задача робота проехать из точки старта в точку финиша, согласно маршруту, заданному изображением ARTag маркера, заранее считанным с камеры реального устройства. На финише необходимо вывести на экран слово `finish`.

Входные данные: через файл `input.txt` управляющей программе передаются:

- В первой строке 19200, разделенных пробелом, целых чисел $P_{1,i}$ $(0 \leqslant P_{1,i} \leqslant 2^{24})$ — изображение ARTag маркера;

Каждое число в маркере — точка, закодированная в формате RGB, т.е. строка с изображением маркера эквивалентна снимку разрешением 160×120 точек.

Ожидаемый результат: После запуска программы робот доехал до финиша, остановился и вывел на экран `finish`.

1.2. Имя файла с управляющей программой для проверки решения в симуляторе: `sim1.js`.

2. Конфигурация робота в симуляторе следующая:

2.1. Конфигурация моторов:

2.1.1. M3 — правый мотор;

2.1.2. M4 — левый мотор.

2.2. Конфигурация датчиков:

2.2.1. D1 — ультразвуковой датчик, направленный вперёд;

2.2.2. D2 — ультразвуковой датчик, направленный назад;

2.2.3. A1 — инфракрасный датчик, направленный вправо;

2.2.4. A2 — инфракрасный датчик, направленный влево.

3. Команде необходимо будет подготовить решения для двух разных подзадач для реального робота. На демонстрацию каждого решения предоставляется 2 попытки.
4. Все попытки осуществляются 20 марта.
5. Merge Request необходимо назвать следующим образом: “код команды_day3”. Например: “irs202000_day3”.
6. Допускается возможность демонстрации первой подзадачи во время периода отладки.
7. Сектор старта может быть разным для каждой попытки.
8. Правила именования файлов с программой управления:
   8.1. для первой подзадачи: `real_1.js`;
   8.2. для второй подзадачи: `real_2.js`;
9. Максимальное время выполнения одной попытки - 3 минуты.
10. Баллы за решение задач этапа:
    10.1. **Задача в симуляторе**: робот проехал из точки старта в точку финиша на всех проверочных полигонах — 16 баллов.
    10.2. **Первая подзадача на реальном роботе:** Робот распознал верно ARTag метку, которая установлена прямо перед камерой, и вывел на экран верные команды, закодированные в ARTag метке. Допускается ручная настройка (наведение) камеры на маркер (включая визуальную корректировку средствами просмотра). — 8 баллов.
    10.3. **Вторая подзадача на реальном роботе:** Робот располагается в секторе старта, с неизвестной стороны которого располагается ARTag метка. Робот нашел метку, издал звуковой сигнал и вывел на экран верные команды, закодированные в ARTag метке — 12 баллов.
11. Цифры, означающие команды, необходимо выводить на экран через пробел. Допускается вывод в несколько строк.
12. За вторую подзадачу начисляется только половина возможных баллов, если не было засчитано решение в симуляторе, а также если не сдана первая подзадача.
13. Выводить значение метки следует не менее 10 секунд.
14. Баллы за все попытки в каждой подзадаче суммируются.
15. Выполнение всех критериев в каждой из двух попыток всех двух подзадач дает дополнительные 4 балла.
16. Максимальное количество баллов за этап — 60.

Четвёртый этап

1. В качестве задачи для симулятора участникам необходимо выполнить следующее:
   1.1. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Известно что один сектор данного логистического центра заблокирован другим роботом. Необходимо в процессе движения опредлить своё местоположение, доехать в сектор старта и вывести на экран координаты одного из достижимых смежных секторов заблокиро-

ванного робота в формате "(X, Y)" без пробелов и без кавычек.
Входные данные: через файл `input.txt` управляющей программе передаются:

1.1.1. В первой строке — направление старта D_s:
- 0 Робот направлен вверх (в сторону отрицательного направления оси Y);
- 1 Робот направлен направо (в сторону положительного направления оси X);
- 2 Робот направлен вниз (в сторону положительного направления оси Y);
- 3 Робот направлен налево (в сторону отрицательного направления оси X);

1.1.2. Во второй строке через пробел — координаты расположения второго робота X, Y ($0 \leqslant X, Y \leqslant 7$);

Ожидаемый результат: После запуска программы робот определил своё местоположение, вернулся в сектор старта и вывел на экран координаты.

1.2. Имя файла с управляющей программой для проверки решения в симуляторе: `sim1.js`.

2. Конфигурация робота в симуляторе следующая:
   2.1. Конфигурация моторов:
      2.1.1. M3 — правый мотор;
      2.1.2. M4 — левый мотор.
   2.2. Конфигурация датчиков:
      2.2.1. D1 — ультразвуковой датчик, направленный вперёд;
      2.2.2. D2 — ультразвуковой датчик, направленный назад;
      2.2.3. A1 — инфракрасный датчик, направленный вправо;
      2.2.4. A2 — инфракрасный датчик, направленный влево.
3. Команде необходимо будет подготовить решения для одной подзадачи для реальных роботов. На демонстрацию решения предоставляется 1 попытка.
4. Все попытки осуществляются 21 марта.
5. Merge Request необходимо назвать следующим образом: "код команды_day4". Например: "irs202000_day4".
6. Стартовым направлением для первого робота является направление 0.
7. После сдачи в карантин для подзадачи судья определяет секторы старта и направление старта 2го робота.
8. Правила именования файлов с программой управления:
   8.1. для первой попытки подзадачи: `real_1_1.js`;
9. Максимальное время выполнения одной попытки - 15 минут.
10. Баллы за решение задач этапа:
    10.1. **Задача в симуляторе**: робот смог выполнить всю задачу на всех проверочных полигонах — 20 баллов.
    10.2. **Задача на реальном роботе**: Роботы располагаются в секторах старта, задача определить свое местоположение, выполнить вторым роботом команды (относительно начального направления робота), переданные через ARTag метку, расположенную на случайном стеллаже вокруг сектора старта, доехать первым роботом до второго и после до любого сектора

старта.

10.2.1. Второй робот успешно считал ARTag метку, выполнил команды, остановился, издал звук и вывел на экран *finish* и более не двигался. — 24 балла.

10.2.2. Первый робот локализовался, доехал до смежного сектора, издал звук. — 24 балла.

10.2.3. Первый робот доехал до смежного сектора, после перемещения второго робота — 8 баллов.

10.2.4. Первый робот локализовался, доехал до смежного сектора, издал звук. Через 10 сек продолжил движение до любого сектора старта, остановился, издал звук и вывел на экран *finish*.—24 балла.

10.2.5. Первый робот доехал до ближайшей (с точки зрения количества секторов) зоны старта. Перед этим, первый робот доехал до смежного сектора после перемещения второго робота — 16 баллов.

11. Баллы за задачу на реальном роботе не начисляются, если не было частично засчитано решение задачи в симуляторе.
12. Частичным решением в симуляторе считается успешное прохождение 30% всех тестов.
13. Во всемя решения задачи возможен перезапуск. При перезапуске команда может поправить своё решение, потратив на это `не более трёх минут`.
14. При перезапуске набранные баллы не пропадают, время не останавливается.
15. В случае если команда использовала перезапуск, то она получит половину из достигнутых баллов за всё попытку.
16. Баллы за все попытки в каждой подзадаче суммируются.
17. Выполнение всех критериев(не обязательно на максимум баллов) в каждой из двух попыток задачи на реальном роботе дает дополнительные 4 балла.
18. Максимальное количество баллов за этап — 120.

Решение

solution/helpers.ts

```
import { Direction } from './FieldGraph';

export function normalizeAngle(angle: number) {
    var newAngle = angle % 360;
    while (newAngle <= -180) newAngle += 360;
    while (newAngle > 180) newAngle -= 360;
    return newAngle;
}

export function normalizeDirection(direction: number): Direction {
    direction = direction % 4;
    if (direction < 0) direction += 4;
    return direction;
}

export function copy<T>(x: T): T {
    return JSON.parse(JSON.stringify(x));
```

```
}

export function run_generator(generator: Iterable<unknown>): void {
    for (const _ of generator) {
    }
}
```

solution/sensor_wrap.ts

```
import trik_brick from "trik_brick";

export abstract class SensorWrap {
    abstract update(total_elapsed_time: number, delta_time: number): void;
}

export abstract class ScalarSensor extends SensorWrap {
    abstract get_value(): number;
}

export class SmoothUltrasonicSensorWrap extends ScalarSensor {
    sensor: trik_brick.Sensor
    previous_stable_reading: number;
    previous_unstable_reading: number;
    current_reading: number;

    constructor(sensor: trik_brick.Sensor) {
        super();
        this.sensor = sensor;
        this.current_reading = sensor.read();
        this.previous_stable_reading = this.current_reading;
        this.previous_unstable_reading = this.current_reading;
    }

    update(total_elapsed_time: number, delta_time: number): void {
        const currentRead = this.sensor.read();
        if (Math.abs(currentRead - this.previous_unstable_reading) > 10)
        {
            console.log("unstable: " + this.previous_stable_reading + ", " +
            ↪  currentRead);
            this.previous_unstable_reading = currentRead;
            this.current_reading = this.previous_stable_reading;
        }

        this.previous_unstable_reading = currentRead;
        this.previous_stable_reading = this.previous_unstable_reading;

        this.current_reading = currentRead;
    }


    get_value(): number {
        return this.current_reading;
    }
}
```

solution/artag/hamming.ts

```
import { ARTAG_SIZE } from "./libmorat-ng";

const message_len = (ARTAG_SIZE - /*borders*/ 2) ** 2 - /*corners*/ 4;
const control_bits = Math.floor(Math.log2(message_len));
const hamming_masks: number[] = [];

// note: using little-endian for representing binaries

for (let control_bit = 0; control_bit < control_bits; ++control_bit) {
    const index = 2 ** control_bit;
    let bitmask = 0;
    for (let i = index - 1; i < message_len; i += index * 2) {
        for (let di = 0; di < index; ++di) {
            bitmask |= 1 << (i + di);
        }
    }

    hamming_masks.push(bitmask);
}

export function extract_message(hamming: number) {
    let ret = 0;
    let c = 0;
    for (let i = 2; i < message_len; ++i) {
        const l2 = Math.log2(i + 1);
        if (Math.floor(l2) !== l2) {
            ret |= ((hamming >> i) & 1) << c;
            c++;
        }
    }
    return ret;
}

function extend_message(message: number) {
    let ret = 0;
    let c = 0;
    for (let i = 0; i < message_len; ++i) {
        const l2 = Math.log2(i + 1);
        if (Math.floor(l2) !== l2) {
            const mi = message & 1;
            message >>= 1;
            ret |= mi << c;
        }
        c++;
    }
    return ret;
}

function calculate_control_bits(extended_message: number) {
    let ret = extended_message;
    for (
        let control_bit_number = 0;
        control_bit_number < control_bits;
        ++control_bit_number
    ) {
        const index = 2 ** control_bit_number - 1;
        let v = hamming_masks[control_bit_number] & extended_message;
        let bit = 0;
```

```
        while (v > 0) {
            bit ^= v & 1;
            v >>= 1;
        }
        ret |= bit << index;
    }

    return ret;
}

export function encode(message: number) {
    return calculate_control_bits(extend_message(message));
}

export function decode(hamming: number) {
    const recomputed = encode(extract_message(hamming));
    let diff = recomputed ^ hamming;
    let c = 0;
    const different_bits: number[] = [];
    while (diff > 0) {
        c++;
        const bit = diff & 1;
        diff >>= 1;
        if (bit) different_bits.push(c);
    }

    if (different_bits.length > 0) {
        console.log("Found a encode mistake, fixing...")
        const to_correct =
            different_bits.reduce((acc, curr) => acc + curr, 0) - 1;
        return extract_message(hamming ^ (1 << to_correct));
    }
    return extract_message(hamming);
}
```

solution/artag/artag_ decode.ts

```
import { ARTAG_SIZE } from './libmorat-ng';
import { Command } from '../Command';
import { decode } from './hamming';

function decode_to_number(bin_artag: NdArray) {
    let ret = 0;
    let c = 0;
    for (let i = 1; i < ARTAG_SIZE - 1; ++i) {
        for (let j = 1; j < ARTAG_SIZE - 1; ++j) {
            if (
                (i === 1 && j === 1) ||
                (i === 1 && j === ARTAG_SIZE - 2) ||
                (i === ARTAG_SIZE - 2 && j === 1) ||
                (i === ARTAG_SIZE - 2 && j === ARTAG_SIZE - 2)
            ) {
                continue;
            }
            ret |= bin_artag.get(i, j) << c;
            c++;
        }
    }
```

```
    return ret;
}

export function artag_decode(bin_artag: NdArray): Command[] {
    let num = decode_to_number(bin_artag);
    num = decode(num);
    const commands = Array(13)
        .fill(0)
        .map(
            (_, i) => (((num >> (i * 2)) & 1) << 1) | ((num >> (i * 2 + 1)) & 1)
        );
    return commands as Command[];
}
```

solution/artag/libmorat-ng.ts

```
import ndarray from "ndarray";

export const ARTAG_SIZE = 8;

const SHAPE: [number, number] = [120, 160];
const LIGHT_NORM: [number, number] = [0.8741, -16.94];
const DARK_NORM: [number, number] = [1, 0];
const XDARK_NORM: [number, number] = [1.582, 0];

const MAP: [number, number][] = new Array(SHAPE[1]);
MAP.fill(LIGHT_NORM);
MAP[3] = XDARK_NORM;
for (let i = 5; i < SHAPE[1]; i+= 2)
    MAP[i] = DARK_NORM;
MAP[7] = XDARK_NORM;

type pos = [number, number];

export class InvalidArtagImageError extends Error {
}

function filter(image: NdArray) {
    for (let j = 0; j < image.shape[1]; j++) {
        const [slope, intercept] = MAP[j];
        for (let i = 0; i < image.shape[0]; i++) {
            let v = slope * image.get(i, j) + intercept;
            v = Math.max(0, v);
            v = Math.min(255, v);
            image.set(i, j, v);
        }
    }
    for (let j = 0; j < image.shape[1]; j += 2) {
        for (let i = 0; i < image.shape[0]; i++) {
            let v = image.get(i, j) + image.get(i, j + 1);
            v /= 2;
            image.set(i, j, v | 0);
            image.set(i, j + 1, v | 0);
        }
    }
}

function histogram(image: NdArray): number[] {
    const res: number[] = new Array(256);
```

```
    res.fill(0);
    for (let i = 0; i < image.shape[0]; i++) {
        for (let j = 0; j < image.shape[1]; j++)  {
            const v = image.get(i, j);
            res[v]++;
        }
    }
    return res;
}

function threshhold(image: NdArray, thresh: number) {
    for (let i = 0; i < image.shape[0]; i++) {
        for (let j = 0; j < image.shape[1]; j++)  {
            const v = image.get(i, j);
            const nv = (v < thresh) ? 0 : 255;
            image.set(i, j, nv);
        }
    }
}

function otsu(image: NdArray) {
    /* http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html */
    const total = image.shape[0] * image.shape[1];
    const hist = histogram(image);

    const hist_sum = hist.reduce((a, x, i) => i * x + a, 0);
    let wB = 0, wF = 0, sumB = 0;
    let best_thresh = 0;
    let best_thresh_variance = 0;
    for (let i = 0; i < 256; i++) {
        wB += hist[i];
        if (wB == 0) continue;

        wF = total - wB;
        if (wF == 0) break;

        sumB += i * hist[i];

        const mB = sumB / wB;
        const mF = (hist_sum - sumB) / wF;
        const variance = wB * wF * (mB - mF) * (mB - mF);
        if (variance > best_thresh_variance) {
            best_thresh = i;
            best_thresh_variance = variance;
        }
    }
    threshhold(image, best_thresh);
}

function get_ccs(image: NdArray): pos[][] {
    function* nei(p: pos): Iterable<[number, number]> {
        const [i, j] = p;
        if (i > 0)                  yield [i - 1, j];
        if (i < image.shape[0] - 1) yield [i + 1, j];
        if (j > 0)                  yield [i, j - 1];
        if (j < image.shape[1] - 1) yield [i, j + 1];
    }
    const seen = ndarray(new Array(image.shape[0] * image.shape[1]), image.shape);
    seen.data.fill(0);
    const ccs: pos[][] = [];
```

```
    function bfs(p: pos, cc: number) {
        if (seen.get(...p) !== 0)
            return;
        seen.set(...p, 1);
        const queue: pos[] = [ p ];
        while (queue.length > 0) {
            const p = queue.shift()!;
            ccs[cc].push(p);
            function put(i: number, j: number) {
                if (image.get(i, j) === 0 && seen.get(i, j) === 0) {
                    seen.set(i, j, 1);
                    queue.push([i,j]);
                }
            }
            //for (let x of nei(p)) {
            const [i, j] = p;
            if (i > 0)                  put(i - 1, j);
            if (i < image.shape[0] - 1) put(i + 1, j);
            if (j > 0)                  put(i, j - 1);
            if (j < image.shape[1] - 1) put(i, j + 1);
            //}
        }
    }
    let cc = 0;
    for (let i = 0; i < image.shape[0]; i++) {
        for (let j = 0; j < image.shape[1]; j++)  {
            const p: pos = [i, j];
            if (image.get(i, j) < 128 && seen.get(i, j) == 0) {
                console.log("bfs(" + p + ")");
                ccs.push([]);
                bfs(p, cc);
                cc++;
            }
            //console.log("mining...");
        }
    }
    return ccs;
}

function slice(image: NdArray, from_i: number = 0, to_i: number = image.shape[0],
↪   from_j: number = 0, to_j: number = image.shape[1]): NdArray {
    const si = to_i - from_i;
    const sj = to_j - from_j;
    const res = ndarray(new Array(si * sj), [si, sj]);
    for (let i = 0; i < si; i++)
        for (let j = 0; j < sj; j++)
            res.set(i, j, image.get(from_i + i, from_j + j));
    return res;
}

function crop(image: NdArray): NdArray {
    const ccs = get_ccs(image)
        //.filter(x => image.get(...x[0]) < 128)
        .sort((a, b) => b.length - a.length);
    const cc = ccs[0];
    const mini = Math.min(...cc.map(x => x[0]));
    const minj = Math.min(...cc.map(x => x[1]));
    const maxi = Math.max(...cc.map(x => x[0]));
    const maxj = Math.max(...cc.map(x => x[1]));
    return slice(image, mini, maxi + 1, minj, maxj + 1);
```

```
}

function get_verts(image: NdArray): [pos, pos, pos, pos] {
    function get_one_vert(i: (p: number, d: number) => number, j: (p: number, d:
    ↪  number) => number): pos {
        for (let d = 0; d < Math.max(...image.shape); d++) {
            for (let p = 0; p <= d; p++) {
                const i1 = i(p, d);
                const j1 = j(p, d);
                if (i1 < image.shape[0] && j1 < image.shape[1] && image.get(i1, j1) <
                ↪  128)
                    return [i1, j1];
            }
        }
        throw new InvalidArtagImageError("Oh shit!");
    }
    return [
        get_one_vert((p, d) => p, (p, d) => d - p),
        get_one_vert((p, d) => p, (p, d) => image.shape[1] - (d - p)),
        get_one_vert((p, d) => image.shape[0] - p, (p, d) => d - p),
        get_one_vert((p, d) => image.shape[0] - p, (p, d) => image.shape[1] - (d -
        ↪  p)),
    ];
}

function pos_sub(a: pos, b: pos): pos {
    return [a[0] - b[0], a[1] - b[1]];
}

function pos_add_one(a: pos, b: pos): pos {
    return [a[0] + b[0], a[1] + b[1]];
}

function pos_add(...args: pos[]): pos {
    return args.reduce((x, a) => pos_add_one(x, a), [0,0]);
}

function pos_mul(a: number, b: pos): pos {
    return [a * b[0], a * b[1]];
}

function get_points(verts: [pos, pos, pos, pos]): pos[] {
    const [p0, p1, p2, p3] = verts;
    const v1 = pos_sub(p1, p0);
    const v2 = pos_sub(p3, p1);
    const v3 = pos_sub(p2, p0);
    const v4 = pos_sub(p3, p2);
    const ib = pos_mul(0.5, pos_add(v1, v4));
    const jb = pos_mul(0.5, pos_add(v3, v2));
    function get_point(i: number, j: number): pos {
        i = (i + 0.5) / ARTAG_SIZE;
        j = (j + 0.5) / ARTAG_SIZE;
        return pos_add(p0, pos_mul(i, ib), pos_mul(j, jb));
    }
    const res: pos[] = [];
    for (let j = 0; j < ARTAG_SIZE; j++) {
        for (let i = 0; i < ARTAG_SIZE; i++)
            res.push(get_point(i, j));
    }
    return res;
```

```
}

function inplace_rotate(pixels: NdArray) {
    /* https://www.geeksforgeeks.org/inplace-rotate-square-matrix-by-90-degrees/ */
    const size = pixels.shape[0];
    const hs = (size / 2) | 0;
    for (let p = 0; p < hs; p++) {
        for (let k = 0; k < size - p * 2 - 1; k++) {
            const p0: pos = [p + k, p];
            const p1: pos = [size - p - 1, p + k];
            const p2: pos = [size - p - 1 - k, size - p - 1];
            const p3: pos = [p, size - p - 1 - k];
            const t = pixels.get(...p0);
            pixels.set(...p0, pixels.get(...p3));
            pixels.set(...p3, pixels.get(...p2));
            pixels.set(...p2, pixels.get(...p1));
            pixels.set(...p1, t);
        }
    }
}

function rotate_artag(pixels: NdArray) {
    let lc_marker_count = 0;
    for (let i = 0; i < 4; i++) {
        if (pixels.get(ARTAG_SIZE - 2, ARTAG_SIZE - 2))
            lc_marker_count++;
        inplace_rotate(pixels);
    }
    if (lc_marker_count != 1)
        throw new InvalidArtagImageError("expected to have 1 left-bottom-corner
        ↪  marker, but found " + lc_marker_count.toString());
    while (!pixels.get(ARTAG_SIZE - 2, ARTAG_SIZE - 2))
        inplace_rotate(pixels);
}

function sanitize_artag(pixels: NdArray) {
    for (let i = 0; i < 4; i++) {
        for (let j = 0; j < pixels.shape[0] - 1; j++) {
            if (pixels.get(j, 0) != 0)
                throw new InvalidArtagImageError("No border around artag");
        }
        inplace_rotate(pixels);
    }
}

/**
 * converts camera output to NdArray
 */
export function numbers_to_image(image: number[]): NdArray {
    const SIZE = [120, 160];
    const result = ndarray(new Array(SIZE[0] * SIZE[1]), SIZE);
    for (let i = 0; i < SIZE[0]; i++) {
        for (let j = 0; j < SIZE[1]; j++) {
            /* (historically) use only the green channel is used */
            const val = image[j + i * SIZE[1]]
            const green = (val >> 8) & 0xff;
            result.set(i, j, green);
        }
    }
    return result;
```

```
}
/**
 * converts input from tests to NdArray
 **/
export function string_to_image(str: string): NdArray {
    const SIZE = [120, 160];
    const splitted = str.trim().split(' ');
    const result = ndarray(new Array(SIZE[0] * SIZE[1]), SIZE);
    for (let i = 0; i < SIZE[0]; i++) {
        for (let j = 0; j < SIZE[1]; j++) {
            /* we (historically) use only the green channel */
            const val = splitted[j + i * SIZE[1]].slice(2, 4);
            result.set(i, j, parseInt(val, 16));
        }
    }
    return result;
}

export function read_artag(image: NdArray): NdArray {
    filter(image);
    image = slice(image, undefined, undefined, 4, undefined);
    otsu(image);
    image = crop(image);
    const verts = get_verts(image);
    const points = get_points(verts);
    const pixels = ndarray(points.map(x => image.get(...x.map(p => p | 0)) > 128 ? 1 :
    ↪  0), [ARTAG_SIZE, ARTAG_SIZE]);
    sanitize_artag(pixels);
    rotate_artag(pixels);
    return pixels;
}
```

solution/low_level_control/util/sensor_wrap.ts

```
import trik_brick from "trik_brick";
import { Timer } from "./timer";
import { setup_gyro, packed_uint_to_sec } from "./common";

export abstract class SensorWrap {
    abstract update(total_elapsed_time: number, delta_time: number): void;
}

export abstract class ScalarSensor extends SensorWrap {
    abstract get_value(): number;
}

export abstract class VectorSensor extends SensorWrap {
    abstract get_value(): number[];
}

export class DumbScalarSensor extends ScalarSensor {
    sensor: trik_brick.Sensor;
    current_reading: number = 0;
    constructor(sensor: trik_brick.Sensor, count: number) {
        super();
        this.sensor = sensor;
    }

    get_value(): number {
```

```
        return this.current_reading;
    }

    update(total_elapsed_time: number, delta_time: number): void {
        this.current_reading = this.sensor.read();
    }

}

export class MedianScalarSensor extends SensorWrap {
    sensor: trik_brick.Sensor
    buffer: number[];
    current_reading: number = 0;

    constructor(sensor: trik_brick.Sensor, count: number) {
        super();
        this.sensor = sensor;
        this.buffer = [];
        for (let i = 0; i < count; i++)
            this.buffer.push(sensor.read());
    }

    update(total_elapsed_time: number, delta_time: number): void {
        this.buffer.push(this.sensor.read());
        this.buffer.shift();
        const sorted = this.buffer.slice();
        sorted.sort();
        this.current_reading = sorted[((sorted.length + 1) / 2) | 0];
    }


    get_value(): number {
        return this.current_reading;
    }
}

export class SmoothUltrasonicSensorWrap extends ScalarSensor {
    sensor: trik_brick.Sensor
    previous_stable_reading: number;
    previous_unstable_reading: number;
    current_reading: number;

    constructor(sensor: trik_brick.Sensor) {
        super();
        this.sensor = sensor;
        this.current_reading = sensor.read();
        this.previous_stable_reading = this.current_reading;
        this.previous_unstable_reading = this.current_reading;
    }

    update(total_elapsed_time: number, delta_time: number): void {
        const currentRead = this.sensor.read();
        if (Math.abs(currentRead - this.previous_unstable_reading) > 10)
        {
            console.log("unstable: " + this.previous_stable_reading + ", " +
            ↪  currentRead);
            this.previous_unstable_reading = currentRead;
            this.current_reading = this.previous_stable_reading;
        }
```

```
        this.previous_unstable_reading = currentRead;
        this.previous_stable_reading = this.previous_unstable_reading;

        this.current_reading = currentRead;
    }


    get_value(): number {
        return this.current_reading;
    }
}

export interface GyroSensor extends SensorWrap {
    get_value(): number[];
    get_time(): number;
    get_pitch(): number;
    get_roll(): number;
    get_yaw(): number;
    translate(new_zero: number): ContinuousYawGyroSensor;
    reset(): void;
}

export interface ContinuousYawGyroSensor extends GyroSensor {}

export class RawGyroSensorWrap extends VectorSensor {
    sensor: trik_brick.GyroSensor;
    value: number[];
    timer: Timer;

    constructor(sensor: trik_brick.GyroSensor) {
        super();
        this.sensor = sensor;
        this.value = [0,0,0,0];
        setup_gyro(sensor);
        //sensor.newData.connect((read, time) => this.sensor_callback(read, time));
        this.timer = new Timer(() => this.sensor_callback(), 0.050);
        //calibrate_gyro(sensor);
        //console.log(sensor.getCalibrationValues());
    }

    sensor_callback(/*values: number[], time: number*/) {
        const MDEG_TO_RAD = Math.PI / 1000 / 180;
        const raw_value = this.sensor.read();
        this.value[0] = packed_uint_to_sec(raw_value[3]);
        this.value[1] = raw_value[4] * MDEG_TO_RAD;
        this.value[2] = raw_value[5] * MDEG_TO_RAD;
        this.value[3] = -raw_value[6] * MDEG_TO_RAD;
    }

    get_value(): number[] {
        return this.value;
    }

    get_time() { return this.value[0]; }
    get_pitch() { return this.value[1]; }
    get_roll() { return this.value[2]; }
    get_yaw() { return this.value[3]; }

    /** It's updated by the timer, so there is no point in doing anything here
     */
```

```
    update(total_elapsed_time: number, delta_time: number): void {}
};

class TranslatedGyroSensorWrap implements ContinuousYawGyroSensor {
    gyro: ContinuousYawGyroSensorWrap;
    zero: number;
    constructor(gyro: ContinuousYawGyroSensorWrap, zero: number) {
        this.gyro = gyro;
        this.zero = zero;
    }

    get_value(): number[] {
        let val = this.gyro.get_value().slice();
        val[3] -= this.zero;
        return val;
    }
    get_time(): number { return this.gyro.get_time(); }
    get_pitch(): number { return this.gyro.get_pitch(); }
    get_roll(): number { return this.gyro.get_roll(); }
    get_yaw(): number { return this.gyro.get_yaw() - this.zero; }
    translate(zero: number): ContinuousYawGyroSensor { return new
    ↪  TranslatedGyroSensorWrap(this.gyro, this.zero + zero); }
    update(total_elapsed_time: number, delta_time: number): void {
        this.gyro.update(total_elapsed_time, delta_time);
    }

    reset() {
        this.gyro.wrapped_yaw = this.zero;
        //this.zero = this.gyro.get_value()[3];
    }
}

/**
 * trik's gyro gives angles in range [-pi;pi], but it is more convienment to have
 ↪  "continuous" yaw axis (other axis do not matter anyways :shrug:)
 */
export class ContinuousYawGyroSensorWrap extends RawGyroSensorWrap implements
↪  ContinuousYawGyroSensor {
    previous_yaw: number = 0;
    wrapped_yaw: number = 0;
    static circle_delta(from: number, to: number) {
        /* gives a shortest by absolute value delta on unit circle */
        if (from < 0) from += Math.PI;
        if (to < 0) to += Math.PI;
        const d1 = to - from;
        const d2 = d1 + Math.PI;
        const d3 = d1 - Math.PI;
        const a1 = Math.abs(d1), a2 = Math.abs(d2), a3 = Math.abs(d3);
        if (a1 < a2) {
            if (a1 < a3)
                return d1;
            else
                return d3;
        } else if (a2 < a3)
            return d2;
        else
            return d3;
    }

    sensor_callback(/*values: number[], time: number*/) {
```

```
        super.sensor_callback();
        let val = this.value[3];
        let pval = this.previous_yaw;
        const delta = ContinuousYawGyroSensorWrap.circle_delta(pval, val);
        this.previous_yaw = this.value[3];
        this.wrapped_yaw += delta;
        // console.log(this.wrapped_yaw);
        this.value[3] = this.wrapped_yaw;
    }

    reset() {
        this.wrapped_yaw = 0;
    }

    translate(zero: number): ContinuousYawGyroSensor { return new
    ↪  TranslatedGyroSensorWrap(this, zero); }
}
```

solution/low_level_control/util/trik_wrap.ts

```
import trik_brick from "trik_brick";
import trik_script from "trik_script";

export function take_raw_image(): number[] {
    let image = trik_brick.getStillImage();
    if (image.length == 0)
        throw new Error("Camera is not available");
    return image;
}

export function take_filtered_image(): number[] {
    let image = trik_script.getPhoto();
    if (image.length == 0)
        throw new Error("Camera is not available");
    return image;
}
```

solution/low_level_control/util/common.ts

```
import trik_script from "trik_script";
import trik_brick, { GyroSensor } from "trik_brick";

export function delay(seconds: number) {
    trik_script.wait(seconds * 1000);
}

export function randint(lo: number, hi: number): number {
    return (Math.random() * (hi - lo) ) << 0;
}

export function get_time(): number {
    return new Date().getTime() / 1000;
}

export function beep(hz: number, ms: number) {
    trik_brick.playTone(hz, ms);
}
```

```
export function calibrate_gyro(gyro: GyroSensor) {
    const TIME = 30;
    gyro.calibrate(TIME * 1000);
    delay(TIME);
}

export function setup_gyro(gyro: GyroSensor) {
    const GYRO_FILE = "gyroscope.txt";
    let cal_text = trik_script.readAll(GYRO_FILE);
    if (cal_text.length === 0) {
        console.log("recalibrating gyroscope...");
        calibrate_gyro(gyro);
        const cal_val = gyro.getCalibrationValues();
        console.log("calibrated to " + JSON.stringify(cal_val));
        cal_text = [JSON.stringify(cal_val)];
        trik_script.removeFile(GYRO_FILE);
        trik_script.writeToFile(GYRO_FILE, cal_text[0]);
    }
    const cal_val = JSON.parse(cal_text[0]);

    gyro.setCalibrationValues(cal_val);
}

export function packed_uint_to_sec(val: number) {
    return val * Math.pow(2, 8) / 1000000;
}
```

solution/low_level_control/util/timer.ts

```
export class Timer {
    private id: number;

    constructor(callback: () => void, interval: number) {
        this.id = <any>setInterval(callback, interval * 1000);
    }

    clear() {
        clearInterval(<any>this.id);
    }
}
```

solution/low_level_control/util/pid.ts

```
export class PID {
    readonly p: number;
    readonly i: number;
    readonly d: number;

    integral: number;
    previous_error: number;

    constructor(p: number, i: number, d: number) {
        this.p = p;
        this.i = i;
        this.d = d;
        this.integral = 0;
```

```
        this.previous_error = 0;
    }

    get_next(setpoint:number, value: number, delta_time: number): number {
        const error = setpoint - value;
        this.integral += error * delta_time;
        const derivative = (error - this.previous_error) / delta_time;
        this.previous_error = error;
        const output = this.p * error + this.i * this.integral + this.d * derivative;
        return output;
    }

    reset() {
        this.previous_error = 0;
        this.integral = 0;
    }
}

export class LimitedPID extends PID {
    min: number;
    max: number;

    constructor(k_p: number, t_i: number, t_d: number, min: number, max: number) {
        super(k_p, t_i, t_d);
        this.min = min;
        this.max = max;
    }

    get_next(setpoint:number, value: number, delta_time: number): number {
        let val = super.get_next(setpoint, value, delta_time);
        if (val > this.max) val = this.max;
        if (val < this.min) val = this.min;
        return val;
    }
}
```

solution/low_level_control/low_level_interaction_proto.ts

```
import { WallsState } from "../FieldGraph";
import { Command } from "../Command";

export class LLCommand {
    public readonly xtype: string = 'Я - автор!';
    public result: LLResult | undefined;
}

export class ForwardCommand extends LLCommand {
    xtype = 'f';
    public amount: number = 0;
}
export class InspectWallsCommand extends LLCommand {
    xtype = 'w';
}
export class TurnCommand extends LLCommand {
    xtype = 't';
    public degrees: number = 0;
}
export class ScanArtagCommand extends LLCommand {
    xtype = 'a';
```

```
}
export class StopCommand extends LLCommand {
    xtype = 's';
}

export class LLResult {}

export class InspectWallsResult extends LLResult {
    public constructor(readonly walls: WallsState) { super(); }
}

export class ScanArtagResult extends LLResult {
    public constructor(readonly commands: Command[] | undefined) { super(); }
}
```

solution/low_level_control/low_level_interaction.ts

```
import { WallsState } from "../FieldGraph";
import ipc_entrypoint from "./movement";
import { LLCommand, LLResult, ForwardCommand, InspectWallsResult, InspectWallsCommand,
↪  TurnCommand, ScanArtagCommand, StopCommand, ScanArtagResult } from
↪  "./low_level_interaction_proto";

// Provides a bridge between low-level control and high-level controll

export class LLInteractionClient {
    constructor(private readonly low_level_loop_coro: Generator<void, void,
    ↪  LLCommand>) {
        console.log("LLInteractionClient constructor");
    }

    execute_command(command: LLCommand): LLResult | undefined {
        const res = this.low_level_loop_coro.next(command);
        if (res.done)
            throw new Error("low-level control stopped, but command was issued");
        //console.log("execute result is", command.result);
        return command.result;
    }

    execute_forward(amount: number) {
        const gg = new ForwardCommand();
        gg.amount = amount;
        this.execute_command(gg);
    }

    execute_inspect_walls(): WallsState {
        const result: InspectWallsResult =
        ↪  <InspectWallsResult>this.execute_command(new InspectWallsCommand());
        if (result === undefined) throw new Error("Result is undefined in
        ↪  execute_inspect_walls()")
        return result.walls;
    }

    execute_turn(degree: number) {
        const gg = new TurnCommand();
        gg.degrees = degree;
        this.execute_command(gg);
    }
```

```
    execute_scan_artag(): ScanArtagResult {
        return <ScanArtagResult>this.execute_command(new ScanArtagCommand());
    }

    execute_stop() {
        this.execute_command(new StopCommand());
    }
}

export function create_low_level_interation_client() {
    return new LLInteractionClient(ipc_entrypoint());
}
```

solution/low_level_control/robot_constants.ts

```
import trik_mailbox from 'trik_mailbox';

/* specific to construction */
export const WHEEL_DISTANCE = 0.175;
export const WHEEL_RADIUS = 0.03969885586091524; /* computed by moving trik and
↪  measuring encoders' values */ //0.0425
export const COUNTS_PER_REVOLUTION = 360;
export const COUNTS_PER_METER =
    COUNTS_PER_REVOLUTION / (2 * WHEEL_RADIUS * Math.PI);
export const METERS_PER_COUNT = 1 / COUNTS_PER_METER;
export const WHEEL_TO_CENTER_DISTANCE = 0.04;

export const CELL_FRONT_DISTANCE = 7.5;
export const MOVE_SPEED = 300;
export const ROTATE_SPEED = 200;
export const MOVE_SPEEDUP = 300;
export const ROTATE_SPEEDUP = 100;

/* specific to robot instance */
export const RIGHT_MOTOR_COEFFICIENT = 1;
export const LEFT_MOTOR_COEFFICIENT = 1;

/* some PID coefficients */
// 0.2   0.2 0     for non-accumulating speedy regulator
// 0.05  0   0.001 for accumulating speedy regulator
// 0.8   0.1 0     for positional regulator
// validated at 11.1824 V
export const MOTOR_PID_COEEFICIENTS = [0.8, 0.1, 0.005];

/* Reduces overshoot when rotating using gyro */
export let TRIK_90: number;
export let TRIK_180: number;

if (trik_mailbox.myHullNumber() == 0) {
    TRIK_90 =
        //0.475
        //0.5
        0.48 * Math.PI;
    TRIK_180 = 0.95 * Math.PI;
} else {
    TRIK_90 =
        //0.475
        0.485
        // 0.48
```

```
        * Math.PI;
    TRIK_180 =
        // 0.95
        1
        * Math.PI;
}
```

solution/low_level_control/movement/movement.ts

```
import { SmoothUltrasonicSensorWrap, ContinuousYawGyroSensorWrap, SensorWrap,
↪  ScalarSensor, ContinuousYawGyroSensor } from "../util/sensor_wrap";
import { make_motor, MotorController } from "./motors";
import { WHEEL_DISTANCE } from "../robot_constants";
import { PID } from "../util/pid";



// rotate k: 2.555

function cap_speed(speed: number) {
    return Math.sign(speed) * Math.min(500, Math.abs(speed));
}

export class Motors {
    readonly motor_left: MotorController;
    readonly motor_right: MotorController;
    constructor(motor_left: MotorController, motor_right: MotorController) {
        this.motor_left = motor_left;
        this.motor_right = motor_right;
    }
    update(left_speed: number, right_speed: number, delta_time: number) {
        this.motor_left.update(left_speed, delta_time);
        this.motor_right.update(right_speed, delta_time);
    }

    brake() {
        this.motor_left.brake();
        this.motor_right.brake();
    }
}

export class RulingMovementController  {
    readonly motors: Motors;
    static readonly D = WHEEL_DISTANCE;

    readonly speedup: number;
    constructor(motors: Motors, speedup: number) {
        this.motors = motors;
        this.speedup = speedup;
    }
    /**
     * Positive radius means turning right, negative - left, +-Infinity - forward
     */

    update(target_speed: number, rule_radius: number, total_elapsed_time: number,
    ↪  delta_time: number): void {
        target_speed = cap_speed(target_speed);
        let actual_target_speed = Math.sign(target_speed) *
        ↪  Math.min(Math.abs(target_speed), total_elapsed_time * this.speedup);
```

```
        if (rule_radius == Infinity || rule_radius == -Infinity)
        {
            this.motors.update(actual_target_speed, actual_target_speed, delta_time);
        }
        else
        {
            const R_m = Math.abs(rule_radius);
            const R_D = R_m + RulingMovementController.D / 2;
            const R = R_m - RulingMovementController.D / 2;
            const speed1 = cap_speed(actual_target_speed * R / R_m);
            const speed2 = cap_speed(actual_target_speed * R_D / R_m);

            //console.log(total_elapsed_time, actual_target_speed, speed1, speed2);

            // speed2 >= speed1

            if (rule_radius > 0)
                this.motors.update(speed2, speed1, delta_time);
            else
                this.motors.update(speed1, speed2, delta_time);
        }
    }

    brake() {
        this.motors.brake();
    }
}

export abstract class DirectionController {
    /** Returns desired curvature or null if don't have information
     */
    abstract update(target_speed: number, total_elapsed_time: number, delta_time:
    ↪  number): number | null;
    reset() {}
}

export class GyroDirectionController extends DirectionController {
    readonly gyro: ContinuousYawGyroSensor;
    static readonly coefficient: number = 20;

    constructor(gyro: ContinuousYawGyroSensor) {
        super();
        this.gyro = gyro;
    }
    update(target_speed: number, total_elapsed_time: number, delta_time: number):
    ↪  number | null {
        const error = this.gyro.get_yaw();
        const curvature = Math.sign(target_speed) * error *
        ↪  GyroDirectionController.coefficient;
        return curvature;
    }

    reset_gyro() {
        this.gyro.reset();
    }
}

export class WallProximityDirectionController extends DirectionController {
    readonly proximity: ScalarSensor;
    static readonly COEFFICIENT: number = 1;
```

```
    readonly coefficient: number;
    readonly target_proximity: number;
    readonly regulator: PID;
    error: number = 0;
    active: boolean = false;

    constructor(proximity: ScalarSensor, target_proximity: number, is_right: boolean)
    ↪  {
        super();
        this.proximity = proximity;
        this.regulator = new PID(0.4, 0, 0.1);
        if (!is_right)
            this.coefficient = -WallProximityDirectionController.COEFFICIENT;
        else
            this.coefficient = WallProximityDirectionController.COEFFICIENT;
        this.target_proximity = target_proximity;
    }

    update(target_speed: number, total_elapsed_time: number, delta_time: number):
    ↪  number | null {
        const prox_value = this.proximity.get_value();
        this.error = this.target_proximity - prox_value;
        //console.log(prox_value, error);
        this.active = prox_value < 25;
        if (!this.active)
            return null;
        const regulated_value = this.regulator.get_next(0, this.error, delta_time);
        return this.coefficient * regulated_value;
    }

    reset() {
        this.regulator.reset();
    }
}

export class CompositeDirectionController extends DirectionController {
    elements: [DirectionController, number][];

    constructor(elements: [DirectionController, number][]) {
        super();
        this.elements = elements;
    }

    update(target_speed: number, total_elapsed_time: number, delta_time: number):
    ↪  number | null {
        const results = this.elements
            .map((v) => [v[0].update(target_speed, total_elapsed_time, delta_time),
            ↪  v[1]])
            .filter((p): p is [number, number] => p[0] !== null);
        if (results.length == 0)
            return null;
        return results.reduce((a, v) => a + v[0] * v[1], 0) / results.reduce((a, v) =>
        ↪  a + v[1], 0);
    }

}

export class ForwardMovementController {
    readonly direction: CompositeDirectionController;
    readonly ruler: RulingMovementController;
```

```
static readonly coefficient: number = 20;
private previous_errors: number[] = [];
private readonly wall_controllers: WallProximityDirectionController[];
private readonly gyro_controller: GyroDirectionController;

constructor(ruler: RulingMovementController, direction:
↪  CompositeDirectionController) {
    this.direction = direction;
    this.ruler = ruler;
    this.wall_controllers =
    ↪  <WallProximityDirectionController[]>direction.elements.map(x =>
    ↪  x[0]).filter(x => x instanceof WallProximityDirectionController);
    this.gyro_controller = <GyroDirectionController>direction.elements.map(x =>
    ↪  x[0]).filter(x => x instanceof GyroDirectionController)[0];
}

private reset_gyro() {
    this.gyro_controller.reset_gyro();
    console.log("gyro reset");
}

private update_gyro_reset(total_elapsed_time: number) {
    const current_errors = this.wall_controllers.filter(x => x.active).map(x =>
    ↪  x.error);
    const ok = Math.max(...this.previous_errors.map(x =>
    ↪  Math.abs(x)).slice(undefined, -1)) < 3 && this.previous_errors.length >
    ↪  20;
    if (this.previous_errors.length >= 60 && ok) {
        console.log("we are stable for a pretty long time. resetting gyro...;
        ↪  previous_errors =", JSON.stringify(this.previous_errors))
        this.previous_errors = [];
        this.reset_gyro();
        return;
    }
    if (current_errors.length == 0) {
        if (this.previous_errors.length != 0) {
            console.log("we have lost walls! Were we moving ok?", (ok ? "YES;" :
            ↪  "no...;"), "previous_errors =",
            ↪  JSON.stringify(this.previous_errors));
            this.previous_errors = [];
            if (ok)
                this.reset_gyro();
        }
        return;
    }
    this.previous_errors.push(Math.max(...current_errors));
    //return false;
}

update(target_speed: number, total_elapsed_time: number, delta_time: number): void
↪  {
    const curvature = this.direction.update(target_speed, total_elapsed_time,
    ↪  delta_time) ?? 0;

    this.update_gyro_reset(total_elapsed_time);
    //console.log(target_speed, curvature, 1 / curvature);

    this.ruler.update(target_speed, 1 / curvature, total_elapsed_time,
    ↪  delta_time);
}
```

```
    brake(): void {
        this.ruler.brake();
    }
}

export class RotationMovementController {
    constructor(readonly motors: Motors, readonly speedup: number) {
        this.motors = motors;
        this.speedup = speedup;
    }

    update(target_speed: number, total_elapsed_time: number, delta_time: number): void
    ↪  {
        target_speed = cap_speed(target_speed);
        let actual_target_speed = Math.sign(target_speed) *
        ↪  Math.min(Math.abs(target_speed), total_elapsed_time * this.speedup);
        this.motors.update(actual_target_speed, -actual_target_speed, delta_time);
    }
    brake() {
        this.motors.brake();
    }
}
```

solution/low_level_control/movement/motors.ts

```
import trik_brick, { Encoder, encoder } from "trik_brick";

import { PID } from "../util/pid";
import { METERS_PER_COUNT } from "../robot_constants";

export abstract class MotorController {
    motor: trik_brick.Motor;
    encoder: trik_brick.Encoder;
    pid: PID;
    name: string;
    power_coefficient: number;
    constructor(name: string, power_coefficient: number, motor: trik_brick.Motor,
    ↪  encoder: trik_brick.Encoder, pid: PID) {
        this.name = name;
        this.motor = motor;
        this.encoder = encoder;
        this.pid = pid;
        this.power_coefficient = power_coefficient;

        encoder.reset();
    }

    protected validate(power: number): number {
        if (power > 100)
            power = 100;
        if (power < -100)
            power = -100;
        return power * this.power_coefficient;
    }

    protected set_motor_power(power: number) {
        this.motor.setPower(power * this.power_coefficient);
    }
```

```
    abstract update(target_speed: number, delta_time: number): void;

    brake() {
        this.motor.brake();
        this.pid.reset();
    }
}

export class SpeedyMotorController extends MotorController {
    last_encoder_position: number = 0;
    last_motor_output: number = 0;

    update(target_speed: number, delta_time: number) {
        const encoder_position = this.encoder.read();
        const encoder_velocity = (encoder_position - this.last_encoder_position) /
        ↪  delta_time;
        this.last_encoder_position = encoder_position;

        this.last_motor_output = this.last_motor_output +
        ↪  this.pid.get_next(target_speed, encoder_velocity, delta_time);
        this.last_motor_output = this.validate(this.last_motor_output);

        //console.log(0, this.name, target_speed, encoder_velocity,
        ↪  this.last_motor_output);

        this.set_motor_power(this.last_motor_output);
    }
}

export class PositionalMotorController extends MotorController {
    target_position: number = 0;

    update(target_speed: number, delta_time: number): void {
        const encoder_position = this.encoder.read();

        this.target_position = this.target_position + target_speed * delta_time;
        const new_power = this.pid.get_next(this.target_position, encoder_position,
        ↪  delta_time);

        //console.log(0, this.name, this.target_position, encoder_position,
        ↪  new_power);

        this.set_motor_power(this.validate(new_power));
    }

    brake() {
        super.brake();
        this.target_position = 0;
        this.encoder.reset();
    }
}

export class Encoders {
    left: Encoder;
    right: Encoder;
    constructor(left: Encoder, right: Encoder) {
        this.left = left;
        this.right = right;
    }
    get_position(): [number, number] {
```

```
        return [this.left.read(), this.right.read() ];
    }
    get_distance_from(position: [number, number]): number {
        const [pl, pr] = position;
        const [l, r] = this.get_position();
        const [dl, dr] = [METERS_PER_COUNT * (l - pl), METERS_PER_COUNT * (r - pr)];
        //console.log(pl, pr, l, r, dl, dr);
        return Math.abs(dl + dr) / 2;
    }
}

export function make_motor(name: string, port_number: string, power_coefficient:
↪  number, p: number, i: number, d: number): MotorController {
    const pid = new PID(p, i, d);
    const encoder = trik_brick.encoder("E" + port_number);
    const motor = trik_brick.motor("M" + port_number);
    return new PositionalMotorController(name, power_coefficient, motor, encoder,
    ↪  pid);
}

export function make_encoders(left_port: string, right_port: string) {
    const left = trik_brick.encoder("E" + left_port);
    const right = trik_brick.encoder("E" + right_port);
    return new Encoders(left, right);
}
```

solution/low_level_control/movement/index.ts

```
import trik_brick, { display } from "trik_brick";

import { get_time, delay } from "../util/common";
import { ScalarSensor, ContinuousYawGyroSensorWrap, MedianScalarSensor,
↪  ContinuousYawGyroSensor } from "../util/sensor_wrap";
import { make_motor, Encoders, make_encoders } from "./motors";
import { ForwardMovementController, GyroDirectionController, RulingMovementController,
↪  WallProximityDirectionController, CompositeDirectionController, Motors,
↪  RotationMovementController } from "./movement";
import { LEFT_MOTOR_COEFFICIENT, RIGHT_MOTOR_COEFFICIENT, TRIK_90, MOVE_SPEED,
↪  WHEEL_TO_CENTER_DISTANCE, ROTATE_SPEED, TRIK_180, CELL_FRONT_DISTANCE,
↪  MOTOR_PID_COEEFICIENTS, MOVE_SPEEDUP as MOVE_SPEEDUP, ROTATE_SPEEDUP } from
↪  "../robot_constants";
import { InspectWallsResult, ScanArtagCommand, LLCommand, ForwardCommand, LLResult,
↪  InspectWallsCommand, TurnCommand, StopCommand, ScanArtagResult } from
↪  "../low_level_interaction_proto";
import { take_filtered_image } from "../util/trik_wrap";
import { numbers_to_image, read_artag, InvalidArtagImageError } from
↪  "../../artag/libmorat-ng";
import { artag_decode } from "../../artag/artag_decode";

const MOTOR_LEFT_PORT = "2"
const MOTOR_RIGHT_PORT = "1"

const FRONT_SENSOR_PORT = "D1"
const BACK_SENSOR_PORT = "D2"
const LEFT_SENSOR_PORT = "A3"
const RIGHT_SENSOR_PORT = "A2"

abstract class MovementOperation {
    constructor(name: string) {
```

```
        this.name = name;
    }

    abstract update(total_elapsed_time: number, delta_time: number): boolean;
    readonly name: string;
}

class MoveToProximityOperation extends MovementOperation {
    readonly proximity_sensor: ScalarSensor;
    readonly target_distance: number;
    readonly movement_controller: ForwardMovementController;
    readonly target_speed: number;
    stop_counter = 0;


    constructor(target_speed: number, proximity_sensor: ScalarSensor,
    ↪  movement_controller: ForwardMovementController, target_distance: number, name:
    ↪  string) {
        super(name);

        this.proximity_sensor = proximity_sensor;
        this.movement_controller = movement_controller;
        this.target_distance = target_distance;
        this.target_speed = target_speed;
    }

    update(total_elapsed_time: number, delta_time: number): boolean {
        const current_distance = this.proximity_sensor.get_value()
        if (current_distance == -1)
            throw new Error("current_distance is -1");

        const error = Math.max(current_distance - this.target_distance, 0);

        //console.log(current_distance);

        //, error, regulated_target_speed, this.stop_counter);

        if (error < 2 || this.stop_counter > 0)
        {
            this.stop_counter += 1;
            this.movement_controller.update(0, total_elapsed_time, delta_time);
            if (this.stop_counter > 0)
            {
                this.movement_controller.brake();
                return false
            };
        }
        else
        {
            const regulated_target_speed = Math.sign(this.target_speed) *
            ↪  Math.min(Math.abs(this.target_speed), error * 10);
            this.movement_controller.update(regulated_target_speed,
            ↪  total_elapsed_time, delta_time);
        }
        return true;
    }
}

class RotateOperation extends MovementOperation {
```

```
    constructor(readonly target_speed: number, readonly controller:
    ↪  RotationMovementController, readonly gyro: ContinuousYawGyroSensor, name:
    ↪  string) {
        super(name);
    }

    update(total_elapsed_time: number, delta_time: number): boolean {
        const SLOWDOWN_BEGIN = Math.PI / 4;
        const error = this.gyro.get_yaw();
        const capped_error = Math.sign(error) * Math.min(Math.abs(error),
        ↪  SLOWDOWN_BEGIN);
        //console.log(error, capped_error);
        if (Math.abs(error) < Math.PI / 180) {
            this.controller.brake();
            return false;
        }
        this.controller.update(capped_error * this.target_speed / SLOWDOWN_BEGIN,
        ↪  total_elapsed_time, delta_time);
        return true;
    }
}


class MoveMetersOperation extends MovementOperation {
    readonly movement_controller: ForwardMovementController;
    readonly target_speed: number;
    readonly initial_position: [number, number];
    readonly target_distance: number;
    readonly encoders: Encoders;

    constructor(target_speed: number, movement_controller: ForwardMovementController,
    ↪  encoders: Encoders, distance: number, name: string) {
        super(name);
        this.movement_controller = movement_controller;
        this.target_speed = target_speed;
        this.initial_position = encoders.get_position();
        this.target_distance = distance;
        this.encoders = encoders;
    }

    update(total_elapsed_time: number, delta_time: number): boolean {
        const position = this.encoders.get_distance_from(this.initial_position);
        const error = this.target_distance - position;
        if (error < 0.005) {
            this.movement_controller.update(0, total_elapsed_time, delta_time);
            this.movement_controller.brake();
            return false;
        }

        const regulated_target_speed = Math.sign(this.target_speed) *
        ↪  Math.min(Math.abs(this.target_speed), error * 10000);

        //console.log(error, regulated_target_speed);
        this.movement_controller.update(regulated_target_speed, total_elapsed_time,
        ↪  delta_time);

        return true;
    }
}
```

```
class DelayOperation extends MovementOperation {
    constructor(readonly delay: number, name: string) {
        super(name);
    }

    update(total_elapsed_time: number, delta_time: number): boolean {
        return total_elapsed_time < this.delay;
    }
}


let [p, i, d] = MOTOR_PID_COEEFICIENTS;
const motor_left = make_motor("L", MOTOR_LEFT_PORT, LEFT_MOTOR_COEFFICIENT, p, i, d);
↪  // new SpeedyMotorWrapper("L", motor1, encoder1, pid1);
const motor_right = make_motor("R", MOTOR_RIGHT_PORT, RIGHT_MOTOR_COEFFICIENT, p, i,
↪  d); // new SpeedyMotorWrapper("R", motor2, encoder2, pid2);
const motors = new Motors(motor_left, motor_right);

const encoders = make_encoders(MOTOR_LEFT_PORT, MOTOR_RIGHT_PORT);

const front_sensor = new MedianScalarSensor(trik_brick.sensor(FRONT_SENSOR_PORT), 3);
const back_sensor = new MedianScalarSensor(trik_brick.sensor(BACK_SENSOR_PORT), 3);
const left_sensor = new MedianScalarSensor(trik_brick.sensor(LEFT_SENSOR_PORT), 3);
const right_sensor = new MedianScalarSensor(trik_brick.sensor(RIGHT_SENSOR_PORT), 3);


let gyro: ContinuousYawGyroSensor = new
↪  ContinuousYawGyroSensorWrap(trik_brick.gyroscope());
function make_forward_movement_controller() {
    const ruler = new RulingMovementController(motors, MOVE_SPEEDUP);
    const direction_rwall = new WallProximityDirectionController(right_sensor, 13.5,
    ↪  true);
    const direction_lwall = new WallProximityDirectionController(left_sensor, 13.5,
    ↪  false);
    const direction_gyro = new GyroDirectionController(gyro);
    const composite_ruler = new CompositeDirectionController([[direction_gyro, 0.2],
    ↪  [direction_rwall, 0.4], [direction_lwall, 0.4]]);
    const current_controller = new ForwardMovementController(ruler, composite_ruler);
    return current_controller;
}

function* make_move_until(speed: number, proximity: number, name: string) {
    yield new MoveToProximityOperation(speed, speed < 0 ? back_sensor : front_sensor,
    ↪  make_forward_movement_controller(), proximity, name);
}

function* make_move_to(speed: number, distance: number, name: string) {
    yield new MoveMetersOperation(speed, make_forward_movement_controller(), encoders,
    ↪  distance, name);
}

function* make_raw_rotation(speed: number, rotation: number, name: string) {
    yield new RotateOperation(speed, new RotationMovementController(motors,
    ↪  ROTATE_SPEEDUP), gyro.translate(-rotation), name);
}

function* make_noop(name: string) {
    yield new DelayOperation(1 / 60 + 0.01, name);
}
```

```
function* definite_proximity(sensor: ScalarSensor[]): Generator<MovementOperation,
↪  [number, number][]> {
    /* TODO: maybe add some filtering */
    const vv = [];
    for (let i = 0; i < 5; i++) {
        vv.push(sensor.map(x => x.get_value()));
        yield* make_noop("definite_proximity_yield");
    }

    console.log("vv is", JSON.stringify(vv));
    let res: [number, number][] = new Array(vv[0].length);
    for (let i = 0; i < res.length; i++)
        res[i] = [0, 0];
    for (let i = 0; i < vv.length; i++)
        for (let j = 0; j < vv[i].length; j++)
            res[j][0] += vv[i][j];

    res = res.map(x => [x[0] / vv.length, x[1]]);

    for (let i = 0; i < vv.length; i++)
        for (let j = 0; j < vv[i].length; j++)
            res[j][1] += Math.pow(res[j][0] - vv[i][j], 2);
    res = res.map(x => [x[0], x[1] / vv.length]);

    console.log("res is", JSON.stringify(res));
    return res;
}

function* make_cell_rotation(rotation: number, name: string) {
    rotation = rotation | 0;
    let degree = 0;
    switch (rotation) {
        case 0:
            degree = 0;
            break;
        case 1:
            degree = TRIK_90;
            break;
        case 2:
            degree = TRIK_180;
            break;
        case 3:
            degree = -TRIK_90;
            break;
    }
    yield* make_move_to(ROTATE_SPEED, WHEEL_TO_CENTER_DISTANCE,
    ↪  "move_forward_before_rotation");
    yield* make_raw_rotation(ROTATE_SPEED, degree, "rotate");
    gyro = gyro.translate(-degree);
    yield* make_move_to(-ROTATE_SPEED, WHEEL_TO_CENTER_DISTANCE,
    ↪  "move_backwards_after_rotation");
}

function* make_onwards_one(): Iterable<MovementOperation> {
    const f = (yield* definite_proximity([front_sensor]))[0][0];
    console.log("Forward distance:", f);
    if (f < 20 && f != -1) throw Error("Won't move into wall");
    else if (f - 40 < 15) yield* make_move_until(MOVE_SPEED, CELL_FRONT_DISTANCE,
    ↪  "move_forward_until_wall");
    else yield* make_move_to(MOVE_SPEED, 0.4, "move_forward_tacho");
```

```
}

function inspect_wall(distance: number, error: number, is_infrared: boolean): number |
↪  undefined {
    if (is_infrared) {
        if (distance < 35)
            return 0;
        else
            return 1;
    } else {
        if (error > 5)
            return undefined;
        if (distance == -1)
            throw new Error("distance is -1");
        if (distance < 30)
            return 0;
        //if (distance < 70)
            return 1;
        //if (distance < 100)
        //    return 2;
        //return 3;
    }
}

function* make_onwards_many(count: number): Iterable<MovementOperation> {
    if (count == 1)
        yield* make_onwards_one();
    else {
        const f = (yield* definite_proximity([front_sensor]))[0];
        console.log("Forward distance:", f);
        const free = inspect_wall(f[0], f[1], false) ?? 0;
        if (f[0] < 20 && f[0] != -1) throw Error("Won't move into wall");
        if (free == count && free < 2)
            yield* make_move_until(MOVE_SPEED, CELL_FRONT_DISTANCE,
            ↪  "move_forward_until_wall");
        else
            yield* make_move_to(MOVE_SPEED, 0.4 * count, "move_forward_tacho");
    }
}

function* inspect_walls(): Generator<MovementOperation, InspectWallsResult> {
    //const aaa: [number, boolean][] = [];
    const sens = [front_sensor, back_sensor, left_sensor, right_sensor];
    const bbb = yield* definite_proximity(sens);
    const ccc: [number, number, boolean][] = new Array(bbb.length);
    for (let i = 0; i < sens.length; i++) {
        ccc[i] = [bbb[i][0], bbb[i][1], sens[i] == left_sensor || sens[i] ==
        ↪  right_sensor];
    }

    console.log("inspect_walls:", bbb);
    let [f, b, l, r] = ccc.map(x => inspect_wall(x[0], x[1], x[2]));
    if (f == undefined || b == undefined)
        return yield* inspect_walls();
    return new InspectWallsResult({
        f_free: f,
        b_free: b,
        l_free: l ?? 1,
        r_free: r ?? 1,
        // f_wall: f < 2 ? f : undefined,
```

```
        // b_wall: b < 2 ? b : undefined,
        // l_wall: (l ?? 1) < 1 ? l : undefined,
        // r_wall: (r ?? 1) < 1 ? r : undefined,
    });
}

function scan_artag(): ScanArtagResult {
    console.log("scanning artag...");
    const raw_image = take_filtered_image();
    display().show(raw_image, 160, 120, "rgb32");
    const image = numbers_to_image(raw_image);
    try {
        const data = read_artag(image);
        const commands = artag_decode(data);
        return new ScanArtagResult(commands);
    } catch (e) {
        if (e instanceof InvalidArtagImageError)
            return new ScanArtagResult(undefined);
        throw e;
    }
}



function* ipc_logic(): Generator<MovementOperation | undefined, void, LLCommand |
↪  undefined> {
    console.log("ipc_logic");
    while (true) {
        const command = yield;

        console.log("got command (2):", command);

        if (command instanceof ForwardCommand) {
            for (const p of make_onwards_many(command.amount))
                yield p;
            command.result = new LLResult();
        } else if (command instanceof InspectWallsCommand) {
            yield* make_noop("yield_for_sensors");
            command.result = yield* inspect_walls();
        } else if (command instanceof TurnCommand) {
            const magic = (((-command.degrees / 90) | 0) + 4) % 4;
            yield* make_cell_rotation(magic, "rotate_" + magic.toString());
            command.result = new LLResult();
        } else if (command instanceof ScanArtagCommand) {
            yield* make_noop("yield_for_sensors");
            command.result = scan_artag();
        } else if (command instanceof StopCommand) {
            command.result = new StopCommand();
            break;
        } else {
            throw new Error("Got unknown command");
        }
    }
}

/* some coroutine spaghetti is going on. But it is better than having two threads */
function* execute_logic(ipc_logic_generator: Generator<MovementOperation | undefined,
↪  void, LLCommand | undefined>): Generator<void, void, LLCommand>
{
```

```
    const logic = ipc_logic_generator[Symbol.iterator]();
    let current_operation: MovementOperation | null = null;
    let current_command: LLCommand = yield;
    logic.next(current_command);
    //console.log("got initial command", current_command)

    let prev_time = get_time();
    let start_time = prev_time;
    delay(0.01);

    while (true) {
        const time = get_time();
        const delta_time = time - prev_time;
        const total_elapsed_time = time - start_time;

        prev_time = time;

        front_sensor.update(total_elapsed_time, delta_time);
        back_sensor.update(total_elapsed_time, delta_time);
        left_sensor.update(total_elapsed_time, delta_time);
        right_sensor.update(total_elapsed_time, delta_time);
        gyro.update(total_elapsed_time, delta_time);

        if (current_operation == null) {
            const value = logic.next(current_command);
            //console.log("got operation", value.value)
            if (value.done)
                break;
            if (value.value == undefined) {
                //console.log("yielding to high level...")
                current_command = yield;
                //console.log("got:", current_command)
                continue;
            }
            start_time = prev_time = get_time();
            current_operation = value.value!;
            delay(0.01);
            console.log("new operation started:", start_time, current_operation.name);
            continue;
        }
        if (!current_operation.update(total_elapsed_time, delta_time)) {
            console.log("Operation '" + current_operation.name + "' has Completed");
            current_operation = null;
        }

        delay(1 / 60);
    }
    console.log("Control loop exited");
}

/* Just in case */
delay(1);

//execute_logic(test_logic());
//execute_logic(left_handle_rule_logic());

function ipc_entrypoint(): Generator<void, void, LLCommand> {
    console.log("worker entry");
    const co = execute_logic(ipc_logic());
    co.next();
```

```
    return co;
}

export default ipc_entrypoint;
```

solution/entry_points/day4/real_1.ts

```
import '../../../shim-global';

import brick from 'trik_brick';
import script from 'trik_script';
import mailbox from 'trik_mailbox';
import { Direction, negateDirection, Node } from '../../FieldGraph';
import { RobotReal } from '../../robots/RobotReal';
import { delay } from '../../low_level_control/util/common';
import {
    run_generator,
    normalizeAngle,
    normalizeDirection,
} from '../../helpers';
import { Command } from '../../Command';

const FIRST_ROBOT_HULL_NUM = 0;
const FIRST_ROBOT_DIR = Direction.Up;

const SECOND_ROBOT_HULL_NUM = 1;
const SECOND_ROBOT_DIR = Direction.Left;

switch (mailbox.myHullNumber()) {
    case FIRST_ROBOT_HULL_NUM: {
        const robot = new RobotReal(FIRST_ROBOT_DIR);
        const localMapStart = robot.localNode;

        mailbox.receive(true); // wait for the second robot finish scanning artag &
        ↪ moving

        robot.scan();
        let relative_dir: Direction;
        let scanned_node: Node;

        for (const cell of robot.walk()) {
            run_generator(robot.moveToLocalMapNode(cell));

            console.log('sending stuff to', SECOND_ROBOT_HULL_NUM);
            mailbox.send(
                SECOND_ROBOT_HULL_NUM,
                JSON.stringify({ type: 'rescan' })
            );
            const msg = JSON.parse(mailbox.receive(true));
            if (msg !== null) {
                console.log('WHOOOOP!', JSON.stringify(msg));
                const { dwall }: { dwall: Direction } = msg;
                relative_dir = negateDirection(dwall);
                scanned_node = robot.localNode;

                break; // from now on we should use optimal localization algorithm
            }
        }
```

```
robot.localize();

console.log('localized');

let localCell: Node; // second robot coords in the first robot local
↪  coordinate system
switch (relative_dir!) {
    case Direction.Right:
        localCell = [scanned_node![0], scanned_node![1] + 1];
        break;
    case Direction.Up:
        localCell = [scanned_node![0] - 1, scanned_node![1]];
        break;
    case Direction.Left:
        localCell = [scanned_node![0], scanned_node![1] - 1];
        break;
    case Direction.Down:
        localCell = [scanned_node![0] + 1, scanned_node![1]];
        break;
}

run_generator(robot.moveNearOccupiedCell(localCell!));

brick.display().clear();
brick.display().addLabel(`(${robot.xReal},${robot.yReal})`, 1, 1);
brick.display().redraw();
brick.playTone(1000, 100);

delay(10);

mailbox.send(
    SECOND_ROBOT_HULL_NUM,
    JSON.stringify({
        type: 'coords',
        args: { node: localCell!, dx: robot.dx, dy: robot.dy },
    })
);

const secondStart: Node = JSON.parse(mailbox.receive(true));
const secondStartLocal: Node = [
    secondStart[0] + robot.dy!,
    secondStart[1] + robot.dx!,
];

console.log(
    'Starts:',
    JSON.stringify([secondStartLocal, localMapStart])
);

const dfirst =
    Math.abs(robot.yLocal - localMapStart[0]) +
    Math.abs(robot.xLocal - localMapStart[1]);
const dsecond =
    Math.abs(robot.yLocal - secondStartLocal[0]) +
    Math.abs(robot.xLocal - secondStartLocal[1]);

if (dfirst <= dsecond) {
    try {
        run_generator(robot.moveToLocalMapNode(localMapStart));
    } catch (e) {
```

```
            if (e instanceof RangeError) {
                run_generator(robot.moveToLocalMapNode(secondStartLocal));
            } else throw e;
        }
    } else {
        try {
            run_generator(robot.moveToLocalMapNode(secondStart));
        } catch (e) {
            if (e instanceof RangeError) {
                run_generator(robot.moveToLocalMapNode(localMapStart));
            } else throw e;
        }
    }

    mailbox.send(
        SECOND_ROBOT_HULL_NUM,
        JSON.stringify({
            type: 'stop',
            args: null,
        })
    );

    brick.display().clear();
    brick.display().addLabel('finish', 1, 20);
    brick.display().redraw();
    delay(10);
    break;
}
case SECOND_ROBOT_HULL_NUM: {
    /* oh well */
    mailbox.connect('192.168.77.1');

    const robot = new RobotReal(SECOND_ROBOT_DIR, false);

    const secondStart = robot.localNode;

    const dir_initial = robot.direction;
    const commands = robot.findArtag();
    if (commands === undefined) throw new Error('Cannot read artag!!1');
    const dir_delta = normalizeDirection(dir_initial - robot.direction);
    const rot_cmds = Array(dir_delta).fill(Command.TurnLeft);
    run_generator(robot.runCommands(rot_cmds));

    // run_generator(robot.runCommandsRaw(commands));
    run_generator(robot.runCommands(commands));

    mailbox.send(FIRST_ROBOT_HULL_NUM, 'null');

    brick.playTone(1000, 100);

    brick.display().clear();
    brick.display().addLabel('finish', 1, 20);
    brick.display().redraw();

    const walls_initial = robot.get_walls_neighbors();
    let wait = true;

    while (wait) {
        const messageJson = mailbox.receive(true);
        const message: { type: string; args: any } = JSON.parse(
```

```
    messageJson
);
switch (message.type) {
    case 'rescan': {
        const walls = robot.get_walls_neighbors();
        // only one wall can change
        const newWalls = walls.filter(
            (x) => !walls_initial.includes(x)
        );
        const removedWalls = walls_initial.filter(
            (x) => !walls.includes(x)
        );
        let dwall: Direction | undefined = undefined;
        if (newWalls.length) {
            dwall = newWalls[0];
        } else if (removedWalls.length) {
            dwall = newWalls[0];
        }

        if (dwall !== undefined) {
            mailbox.send(
                FIRST_ROBOT_HULL_NUM,
                JSON.stringify({
                    dwall,
                })
            );
        } else mailbox.send(FIRST_ROBOT_HULL_NUM, 'null');
        break;
    }

    case 'coords': {
        const {
            args: { node, dx, dy },
        }: {
            args: { node: Node; dx: number; dy: number };
        } = message;

        // local_first = real + d
        // real = local_first - d
        // d_second = local_second - real

        const yReal = node[0] - dy;
        const xReal = node[1] - dx;

        robot.dx = robot.localNode[1] - yReal;
        robot.dy = robot.localNode[0] - xReal;

        console.log(
            'sending to first: ',
            JSON.stringify([
                secondStart[0] - robot.dy,
                secondStart[1] - robot.dx,
            ])
        );
        mailbox.send(
            FIRST_ROBOT_HULL_NUM,
            JSON.stringify([
                secondStart[0] - robot.dy,
                secondStart[1] - robot.dx,
            ])
```

```
                    );

                    break;
                }

                case 'stop': {
                    wait = false;
                    break;
                }
            }
        }
        break;
    }
    default: {
        throw new Error(`Unknown hull number: ${mailbox.myHullNumber()}`);
    }
}
```

solution/robots/Robot.ts

```
import { Command } from '../Command';
import { normalizeDirection, run_generator } from '../helpers';
import {
    FieldGraph,
    FIELD_SIZE,
    Direction,
    WallsState,
    Node,
    CellState,
} from '../FieldGraph';

export abstract class Robot {
    private readonly verbose = true;
    private localization_data = {
        xmax: 0,
        ymax: 0,
        xmin: FIELD_SIZE,
        ymin: FIELD_SIZE,
    };

    protected readonly direction_initial: Direction;

    constructor(
        public direction: Direction,
        public autoScanField: boolean = true
    ) {
        this.direction_initial = direction;
    }

    protected localMap = new FieldGraph();

    public yLocal = FIELD_SIZE;
    public xLocal = FIELD_SIZE;
    public get localNode(): Node {
        return [this.yLocal, this.xLocal];
    }

    public get realNode(): Node {
        return [this.yReal, this.xReal];
```

```
}

public dx?: number = undefined;
public dy?: number = undefined;

public get located() {
    return this.dx !== undefined && this.dy !== undefined;
}

get xReal() {
    if (!this.located) throw new Error('Not located yet');
    return this.xLocal - this.dx!;
}

get yReal() {
    if (!this.located) throw new Error('Not located yet');
    return this.yLocal - this.dy!;
}

/**
 * Move forward by `n` cells
 */
protected abstract _forward(n?: number): void;

/**
 * Move forward by n cells
 * Yields the position after each rescan
 */
public *forward(n: number = 1): Iterable<Node> {
    if (this.direction === Direction.Up) this.yLocal -= n;
    else if (this.direction === Direction.Down) this.yLocal += n;
    else if (this.direction === Direction.Left) this.xLocal -= n;
    else if (this.direction === Direction.Right) this.xLocal += n;
    this._forward(n);
    if (this.autoScanField) {
        this.scan();
        this.localize_iteration();
    }

    yield this.localNode;
}

/**
 * Turn left by `deg` degrees
 */
public abstract turn(deg: number): void;

/**
 * Get the number of free cells at each direction:
 *   - `r:` right
 *   - `l:` left
 *   - `f:` forward
 *
 * If counting of free cells is impossible, the method should output
 * whether the wall at the direction exist (0 if exist, 1 otherwise)
 *
 * If you can't say anything about the walls, set the corresponding property to
 ↪  `undefined`
 */
public abstract getWalls(): WallsState;
```

```
public get_walls_neighbors(): Direction[] {
    const ret: Direction[] = [];
    const w = this.getWalls();
    const ifWall = (x?: number) => x !== undefined && x === 0;
    switch (this.direction) {
        case Direction.Right:
            if (ifWall(w.f_free)) {
                ret.push(Direction.Right);
            }
            if (ifWall(w.l_free)) {
                ret.push(Direction.Up);
            }
            if (ifWall(w.b_free)) {
                ret.push(Direction.Left);
            }
            if (ifWall(w.r_free)) {
                ret.push(Direction.Down);
            }
            break;
        case Direction.Up:
            if (ifWall(w.r_free)) {
                ret.push(Direction.Right);
            }
            if (ifWall(w.f_free)) {
                ret.push(Direction.Up);
            }
            if (ifWall(w.l_free)) {
                ret.push(Direction.Left);
            }
            if (ifWall(w.b_free)) {
                ret.push(Direction.Down);
            }
            break;
        case Direction.Left:
            if (ifWall(w.b_free)) {
                ret.push(Direction.Right);
            }
            if (ifWall(w.r_free)) {
                ret.push(Direction.Up);
            }
            if (ifWall(w.f_free)) {
                ret.push(Direction.Left);
            }
            if (ifWall(w.l_free)) {
                ret.push(Direction.Down);
            }
            break;
        case Direction.Down:
            if (ifWall(w.l_free)) {
                ret.push(Direction.Right);
            }
            if (ifWall(w.b_free)) {
                ret.push(Direction.Up);
            }
            if (ifWall(w.r_free)) {
                ret.push(Direction.Left);
            }
            if (ifWall(w.f_free)) {
                ret.push(Direction.Down);
```

```
            }
            break;
    }

    return ret;
}

/**
 * Turns left
 * Yields the position after each rescan
 */
public *turnLeft(n: number = 1): Iterable<Node> {
    this.turn(90 * n);
    this.direction = normalizeDirection(this.direction + n);
    if (this.autoScanField) {
        this.scan();
        this.localize_iteration();
    }

    yield this.localNode;
}

/**
 * Turns right
 * Yields the position after each rescan
 */
public *turnRight(n: number = 1): Iterable<Node> {
    yield* this.turnLeft(-n);
}

public *runCommandsRaw(commands: Command[]): Iterable<Node> {
    for (const command of commands) {
        yield* this.runCommands([command]);
    }
}

public *runCommands(commands: Command[]): Iterable<Node> {
    let i = 0;
    while (i < commands.length) {
        switch (commands[i]) {
            case Command.MoveForward:
                let fwdCounter = 1;
                i += 1;
                while (
                    commands[i] == Command.None ||
                    commands[i] == Command.MoveForward
                ) {
                    if (commands[i] == Command.MoveForward) fwdCounter += 1;
                    i += 1;
                }

                yield* this.forward(fwdCounter);
                break;

            case Command.None:
                i += 1;
                break;

            case Command.TurnLeft:
            case Command.TurnRight:
```

```
                let rotCounter = 0;
                while (
                    commands[i] == Command.TurnLeft ||
                    commands[i] == Command.TurnRight ||
                    commands[i] == Command.None
                ) {
                    if (commands[i] == Command.TurnLeft) {
                        rotCounter += 1;
                    } else if (commands[i] == Command.TurnRight) {
                        rotCounter -= 1;
                    }
                    i += 1;
                }

                yield* this.turnLeft(rotCounter);
                break;
            }
        }
    }

    public scan(): void {
        for (const node of this.localMap.updateField(
            this.localNode,
            this.getWalls(),
            this.direction
        )) {
            this.localize_iteration(node);
        }

        if (this.verbose) {
            console.log('Map:');
            console.log(this.localMap);
        }
    }

    private localize_iteration([y, x]: Node = this.localNode) {
        let { xmin, xmax, ymin, ymax } = this.localization_data;

        ymin = Math.min(ymin, y);
        ymax = Math.max(ymax, y);
        // if (this.verbose) console.log('ymin, ymax', ymin, ymax);
        xmin = Math.min(xmin, x);
        xmax = Math.max(xmax, x);
        // if (this.verbose) console.log('xmin, xmax', xmin, xmax);

        if (ymax - ymin + 1 === FIELD_SIZE) {
            this.dy = ymin;

            // set the borders of the field when found Y coordinate
            for (let jj = 0; jj < this.localMap.length; ++jj) {
                this.localMap.field[ymin - 1][jj] = CellState.Wall;
                this.localMap.field[ymax + 1][jj] = CellState.Wall;
            }
        }

        if (xmax - xmin + 1 === FIELD_SIZE) {
            this.dx = xmin;

            // set the borders of the field when found X coordinate
            for (let ii = 0; ii < this.localMap.length; ++ii) {
```

```
            this.localMap.field[ii][xmin - 1] = CellState.Wall;
            this.localMap.field[ii][xmax + 1] = CellState.Wall;
        }
    }

    this.localization_data = { xmin, xmax, ymin, ymax };
}

/**
 * Walk all cells of the local map
 */
public *walk(): Iterable<Node> {
    yield* this.localMap.dfs(this.localNode);
}

public localize(fullMapScan = false): void {
    this.scan();
    for (const [i, j] of this.walk()) {
        const hasUnscannedNeighbor = () =>
            [...this.localMap.neighbors([i, j], true)].reduce(
                (has, [i, j]) =>
                    has || this.localMap.field[i][j] === CellState.Unknown,
                false
            );

        if (hasUnscannedNeighbor()) {
            for (const _ of this.moveToLocalMapNode([i, j])) {
                if (this.located && !fullMapScan) return;
                if (!hasUnscannedNeighbor()) break;
            }
        }

        if (this.located && !fullMapScan) return;
    }
}

private moveByAbsoluteDirectionCommands(
    to: Direction,
    currentDirection: Direction = this.direction
): Command[] {
    const diff = normalizeDirection(to - currentDirection);
    const commands: Command[] = [];
    // for (let i = 0; i < diff; ++i) commands.push(Command.TurnLeft);
    switch (diff) {
        case 0:
            break;
        case 1:
            commands.push(Command.TurnLeft);
            break;
        case 2: // Rotate by 180°
            commands.push(Command.TurnLeft, Command.TurnLeft);
            break;
        case 3:
            commands.push(Command.TurnRight);
            break;
    }
    commands.push(Command.MoveForward);

    return commands;
}
```

```
public *moveByAbsoluteDirection(to: Direction): Iterable<Node> {
    yield* this.runCommands(this.moveByAbsoluteDirectionCommands(to));
}

private moveToNeighborNodeCommands(
    to: Node,
    currentPosition: Node,
    currentDirection: Direction = this.direction
): Command[] {
    const rel = this.localMap.relativePosition(to, currentPosition);
    if (rel === Direction.Unknown)
        throw new Error("Can't move to node " + JSON.stringify(to));
    return this.moveByAbsoluteDirectionCommands(rel, currentDirection);
}

public *moveToNeighborNode(to: Node): Iterable<Node> {
    yield* this.runCommands(
        this.moveToNeighborNodeCommands(to, this.localNode)
    );
}

/**
 * Converts path to robot commands
 * @param path The required path. Should start from current node.
 * @param currentDirection
 */
private pathToCommands(
    path: Node[],
    currentDirection = this.direction
): Command[] {
    let commands: Command[] = [];
    let currentPosition = path[0];
    for (const nextNode of path.slice(1)) {
        commands = commands.concat(
            this.moveToNeighborNodeCommands(
                nextNode,
                currentPosition,
                currentDirection
            )
        );
        currentDirection = this.localMap.relativePosition(
            nextNode,
            currentPosition
        );
        currentPosition = nextNode;
    }
    return commands;
}

private shortestCommandSetMetric(path: Command[]): number {
    const turns = path.reduce(
        (acc, command) =>
            command === Command.TurnLeft || command === Command.TurnRight
                ? acc + 1
                : 0,
        0
    );
    return path.length + turns;
}
```

```
public getReachableCellNearOccupiedCell(cell: Node): Node | undefined {
    const neighbors = [...this.localMap.neighbors(cell, true)];

    const ret = neighbors.reduce<{ neighbor?: Node; plen: number }>(
        (acc, neighbor) => {
            const { known, path } = this.localMap.bfs_path(
                this.localNode,
                neighbor
            );
            if (!known || path === undefined) return acc;

            const plen = this.shortestCommandSetMetric(
                this.pathToCommands(path)
            );
            if (plen < acc.plen) {
                return { plen, neighbor };
            }

            return acc;
        },
        { neighbor: undefined, plen: Infinity }
    );

    return ret.neighbor;
}

public *moveNearOccupiedCell(cell: Node): Iterable<Node> {
    if (this.verbose) console.log('Moving near:', cell);
    const known_neighbor = this.getReachableCellNearOccupiedCell(cell);
    if (known_neighbor !== undefined) {
        yield* this.moveToLocalMapNode(known_neighbor);
        return;
    }

    const unknown_neighbors = [...this.localMap.neighbors(cell, true)];
    for (const node of unknown_neighbors) {
        console.log('unknown neighbor: ', node);

        try {
            yield* this.moveToLocalMapNode(node);
        } catch (e) {
            if (e instanceof RangeError) {
            } else throw e;
        }

        console.log('localNode: ', this.localNode);

        if (
            this.localMap.relativePosition(this.localNode, cell) !==
            Direction.Unknown
        )
            return;
    }

    throw new Error('Cannot move near cell: ' + JSON.stringify(cell));
}

/**
 * Move to the node in local robot map
```

```
     */
    public *moveToLocalMapNode(node: Node): Iterable<Node> {
        if (this.verbose) console.log('Moving:', node);
        while (this.yLocal !== node[0] || this.xLocal !== node[1]) {
            const { known, path } = this.localMap.bfs_path(
                this.localNode,
                node
            );

            if (path === undefined) {
                throw new RangeError('Path not found');
            }

            if (!known) {
                const known_path = this.localMap.known_path_part(path);
                const commands = this.pathToCommands(
                    known_path,
                    this.direction
                );

                if (this.verbose)
                    console.log('partial path:', JSON.stringify(known_path));

                yield* this.runCommands(commands);
            } else {
                const commands = this.pathToCommands(path, this.direction);

                if (this.verbose)
                    console.log('fin path:', JSON.stringify(path));

                yield* this.runCommands(commands);
            }
        }
    }

    public findArtag(): Command[] | undefined {
        for (let i = 0; i < 4; ++i) {
            const commands = this.getArtag();
            if (commands !== undefined) return commands;
            run_generator(this.turnLeft());
        }

        return undefined;
    }

    public abstract getArtag(): Command[] | undefined;
}
```

solution/robots/RobotReal.ts

```
import ipc_entrypoint from "../low_level_control/movement/index"
import { Robot } from "./Robot";
import { Direction, WallsState } from "../FieldGraph";
import { Command } from "../Command";
import { LLInteractionClient, create_low_level_interation_client } from
↪  "../low_level_control/low_level_interaction";

export class RobotReal extends Robot {
    private ipc_client: LLInteractionClient;
```

```
    constructor(dir: Direction, auto_scan_field: boolean = true) {
        super(dir, auto_scan_field);
        this.ipc_client = create_low_level_interation_client();
    }

    protected _forward(n: number = 1): void {
        this.ipc_client.execute_forward(n);
        console.log("RobotReal._forward(" + n.toString() + ")");
    }
    public turn(deg: number): void {
        this.ipc_client.execute_turn(deg);
        console.log("RobotReal.turn(" + deg.toString() + ")");
    }
    public getWalls(): WallsState {
        console.log("RobotReal.getWalls()..");
        const r = this.ipc_client.execute_inspect_walls()
        console.log("RobotReal.getWalls():", JSON.stringify(r));
        return r;
    }
    public getArtag(): Command[] | undefined {
        const r = this.ipc_client.execute_scan_artag();
        console.log("RobotReal.getArtag():", JSON.stringify(r.commands));
        return r.commands;
    }
}
```

solution/Command.ts

```
export const enum Command {
    None = 0,
    TurnLeft = 1,
    TurnRight = 2,
    MoveForward = 3,
}
```

solution/FieldGraph.ts

```
import { copy } from './helpers';

/** - The number of free cells at each direction:
 *     - `r_free:` right
 *     - `l_free:` left
 *     - `f_free:` forward
 *     - `b_free:` back
 *
 *  - The number of free cells before wall at each direction (if wall detected):
 *     - `r_wall`, `l_wall`, `f_wall`, `b_wall`
 */
export interface WallsState {
    // TODO: optimize localization using new walls
    // TODO: add better graph walking algorithm
    // TODO: add better localization algorithm
    r_free: number;
    l_free: number;
    f_free: number;
    b_free: number;
```

```
    r_wall?: number;
    l_wall?: number;
    f_wall?: number;
    b_wall?: number;
}

export enum CellState {
    Unknown = 0,
    Wall,
    Free,
}

/**
 * The direction of the robot:
 *   - `0`: right
 *   - `1`: up
 *   - `2`: left
 *   - `3`: bottom
 */
export enum Direction {
    Unknown = 100,
    Right = 0,
    Up = 1,
    Left = 2,
    Down = 3,
}

export function negateDirection(dir: Direction) {
    switch (dir) {
        case Direction.Unknown:
            return Direction.Unknown;
        case Direction.Right:
            return Direction.Left;
        case Direction.Left:
            return Direction.Right;
        case Direction.Up:
            return Direction.Down;
        case Direction.Down:
            return Direction.Up;
    }
}

export type Node = [number, number];

export const FIELD_SIZE = 8;

export class FieldGraph {
    public field: CellState[][] = [];

    get length(): number {
        return this.field.length;
    }

    constructor() {
        for (let i = 0; i < FIELD_SIZE * 3; ++i) {
            this.field.push([]);
            for (let j = 0; j < FIELD_SIZE * 3; ++j) {
                this.field[i].push(CellState.Unknown);
            }
        }
```

```
    }

    private *clearWallsVertical(
        [i, j]: Node,
        di_start: number,
        di_end: number
    ): Iterable<Node> {
        for (let di = di_start; di <= di_end; ++di) {
            this.updateCell(i + di, j, CellState.Free);
            yield [i + di, j];
        }
    }

    private *clearWallsHorizontal(
        [i, j]: Node,
        dj_start: number,
        dj_end: number
    ): Iterable<Node> {
        for (let dj = dj_start; dj <= dj_end; ++dj) {
            this.updateCell(i, j + dj, CellState.Free);
            yield [i, j + dj];
        }
    }

    private updateCell(i: number, j: number, val: CellState) {
        if (this.field[i][j] !== CellState.Unknown && this.field[i][j] !== val)
            throw new RangeError(`Updated scanned cell: ${[i, j]}, ${val}`);
        this.field[i][j] = val;
    }

    /**
     * Yields free nodes (for localization)
     */
    public *updateField(
        node: Node,
        walls: WallsState,
        direction: number
    ): Iterable<Node> {
        const [i, j] = node;

        switch (direction) {
            case 0:
                if (walls.l_free === 0)
                    this.updateCell(i - 1, j, CellState.Wall);
                if (walls.f_free === 0)
                    this.updateCell(i, j + 1, CellState.Wall);
                if (walls.b_free === 0)
                    this.updateCell(i, j - 1, CellState.Wall);
                if (walls.r_free === 0)
                    this.updateCell(i + 1, j, CellState.Wall);

                if (walls.l_wall !== undefined)
                    this.updateCell(i - 1 - walls.l_wall, j, CellState.Wall);
                if (walls.f_wall !== undefined)
                    this.updateCell(i, j + 1 + walls.f_wall, CellState.Wall);
                if (walls.b_wall !== undefined)
                    this.updateCell(i, j - 1 - walls.b_wall, CellState.Wall);
                if (walls.r_wall !== undefined)
                    this.updateCell(i + 1 + walls.r_wall, j, CellState.Wall);
```

```
            yield* this.clearWallsVertical(
                node,
                -walls.l_free,
                walls.r_free
            );
            yield* this.clearWallsHorizontal(
                node,
                -walls.b_free,
                walls.f_free
            );
            break;
        case 1:
            if (walls.l_free === 0)
                this.updateCell(i, j - 1, CellState.Wall);
            if (walls.f_free === 0)
                this.updateCell(i - 1, j, CellState.Wall);
            if (walls.b_free === 0)
                this.updateCell(i + 1, j, CellState.Wall);
            if (walls.r_free === 0)
                this.updateCell(i, j + 1, CellState.Wall);

            if (walls.l_wall !== undefined)
                this.updateCell(i, j - 1 - walls.l_wall, CellState.Wall);
            if (walls.f_wall !== undefined)
                this.updateCell(i - 1 - walls.f_wall, j, CellState.Wall);
            if (walls.b_wall !== undefined)
                this.updateCell(i + 1 + walls.b_wall, j, CellState.Wall);
            if (walls.r_wall !== undefined)
                this.updateCell(i, j + 1 + walls.r_wall, CellState.Wall);

            yield* this.clearWallsVertical(
                node,
                -walls.f_free,
                walls.b_free
            );
            yield* this.clearWallsHorizontal(
                node,
                -walls.l_free,
                walls.r_free
            );
            break;
        case 2:
            if (walls.l_free === 0)
                this.updateCell(i + 1, j, CellState.Wall);
            if (walls.f_free === 0)
                this.updateCell(i, j - 1, CellState.Wall);
            if (walls.b_free === 0)
                this.updateCell(i, j + 1, CellState.Wall);
            if (walls.r_free === 0)
                this.updateCell(i - 1, j, CellState.Wall);

            if (walls.l_wall !== undefined)
                this.updateCell(i + 1 + walls.l_wall, j, CellState.Wall);
            if (walls.f_wall !== undefined)
                this.updateCell(i, j - 1 - walls.f_wall, CellState.Wall);
            if (walls.b_wall !== undefined)
                this.updateCell(i, j + 1 + walls.b_wall, CellState.Wall);
            if (walls.r_wall !== undefined)
                this.updateCell(i - 1 - walls.r_wall, j, CellState.Wall);
```

```
            yield* this.clearWallsVertical(
                node,
                -walls.r_free,
                walls.l_free
            );
            yield* this.clearWallsHorizontal(
                node,
                -walls.f_free,
                walls.b_free
            );
            break;
        case 3:
            if (walls.l_free === 0)
                this.updateCell(i, j + 1, CellState.Wall);
            if (walls.f_free === 0)
                this.updateCell(i + 1, j, CellState.Wall);
            if (walls.b_free === 0)
                this.updateCell(i - 1, j, CellState.Wall);
            if (walls.r_free === 0)
                this.updateCell(i, j - 1, CellState.Wall);

            if (walls.l_wall !== undefined)
                this.updateCell(i, j + 1 + walls.l_wall, CellState.Wall);
            if (walls.f_wall !== undefined)
                this.updateCell(i + 1 + walls.f_wall, j, CellState.Wall);
            if (walls.b_wall !== undefined)
                this.updateCell(i - 1 - walls.b_wall, j, CellState.Wall);
            if (walls.r_wall !== undefined)
                this.updateCell(i, j - 1 - walls.r_wall, CellState.Wall);

            yield* this.clearWallsVertical(
                node,
                -walls.b_free,
                walls.f_free
            );
            yield* this.clearWallsHorizontal(
                node,
                -walls.r_free,
                walls.l_free
            );
            break;
    }
}

public relativePosition(of: Node, to: Node): Direction {
    const [i0, j0] = to;
    const [i1, j1] = of;

    if (i1 === i0 + 1 && j1 === j0) return Direction.Down;
    if (i1 === i0 - 1 && j1 === j0) return Direction.Up;
    if (j1 === j0 + 1 && i1 === i0) return Direction.Right;
    if (j1 === j0 - 1 && i1 === i0) return Direction.Left;

    return Direction.Unknown;
}

/**
 * Return reachable neighbor nodes
 */
public *neighbors(
```

```
        [i, j]: Node,
        allowUnknown: boolean = false
    ): Iterable<Node> {
        const allowedVals = [CellState.Free];
        if (allowUnknown) allowedVals.push(CellState.Unknown);
        yield* ([
            [i - 1, j],
            [i, j - 1],
            [i + 1, j],
            [i, j + 1],
        ] as Node[]).filter(([i_, j_]) =>
            i_ < 0 || j_ < 0 || i_ >= this.length || j_ >= this.length
                ? false
                : allowedVals.includes(this.field[i_][j_])
        );
    }

    public *dfs(
        startNode: Node,
        allowUnknown: boolean = false
    ): Iterable<Node> {
        const stack = [startNode];

        // List are just links, so we can't use list as a hashable type in JS
        const visited: { [node: string]: boolean } = {};

        while (stack.length > 0) {
            const [i, j] = stack.pop()!;
            if (`${i}:${j}` in visited) {
                continue;
            }

            yield [i, j];
            visited[`${i}:${j}`] = true;

            stack.push(...this.neighbors([i, j], allowUnknown));
        }
    }

    private _bfs_path(
        startNode: Node,
        endNode: Node,
        allowUnknown: boolean = false
    ): Node[] | undefined {
        const q = [[startNode]];
        const visited: { [node: string]: boolean } = {};

        while (q.length > 0) {
            const path = q.shift()!;
            const [i, j] = path[path.length - 1];

            if (`${i}:${j}` in visited) continue;
            visited[`${i}:${j}`] = true;

            if (i === endNode[0] && j === endNode[1]) {
                return path;
            }

            q.push(
                ...[...this.neighbors([i, j], allowUnknown)].map((neighbor) =>
```

```
                    copy(path).concat([neighbor])
                )
            );
        }

        return undefined;
    }

    public known_path_part(path: Node[]): Node[] {
        const ret: Node[] = [];
        for (const [i, j] of path) {
            if (this.field[i][j] !== CellState.Unknown) {
                ret.push([i, j]);
            } else {
                break;
            }
        }

        return ret;
    }

    public bfs_path(
        startNode: Node,
        endNode: Node
    ): { path: Node[] | undefined; known: boolean } {
        const p1 = this._bfs_path(startNode, endNode, false);
        if (p1 !== undefined) return { path: p1, known: true };

        return {
            path: this._bfs_path(startNode, endNode, true),
            known: false,
        };
    }

    public toString(): string {
        return this.field.map((line) => line.join(', ')).join('\n');
    }
}
```

Критерии

Критерии определения победителей и призеров заключительного этапа

Первый отборочный этап

Количество баллов, набранных при решении задач одной попытки, суммируется. Если участник решал задачи в нескольких попытках, то выбирается попытка с большей суммой баллов. Призерам первого отборочного этапа необходимо было набрать 80 баллов (для 9 класса) и 100 баллов (для 10-11 класса). Победители первого отборочного этапа должны были набрать 200 баллов.

Второй отборочный этап

Количество баллов, набранных при решении всех задач, суммируется. Призерам второго отборочного этапа было необходимо набрать 27 баллов. Победители второго отборочного этапа должны были набрать 50 баллов и выше.

Заключительный этап

В заключительном этапе олимпиады баллы участника складываются из двух частей: он получает баллы за индивидуальное решение задач по предметам (математика и информатика) и за командное решение практической задачи.

$$S = S_i + S_m + S_t$$

где S_i — сумма баллов по информатике, набранная в рамках индивидуального тура заключительного этапа (максимум 300 баллов), поделённая на 3; S_i — сумма баллов о математике, набранная в рамках индивидуального тура заключительного этапа (максимум 100 баллов); S_t — количество баллов, набранное в рамках командного тура заключительного этапа (максимум 400 баллов).

Критерий определения победителей и призеров:

Категория	**Количество баллов**
Победители	От 234 баллов и выше
Призеры	От 167 до 234 баллов