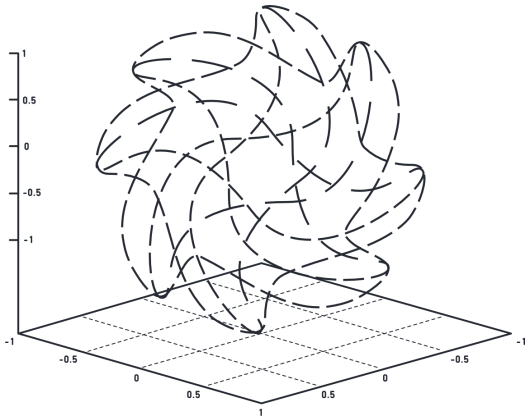


МАТЕРИАЛЫ ЗАДАНИЙ

командной инженерной олимпиады школьников

«Олимпиада Национальной технологической инициативы»

по профилю

ИНЖЕНЕРНЫЕ БИОЛОГИЧЕСКИЕ СИСТЕМЫ

2017/18 учебный год

http://nti-contest.ru

Оглавление

ВВЕДЕНИЕ

Олимпиада Национальной технологической инициативы[1] (далее – Олимпиада 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 инженерно-ориентированных школьников в образовательные программы высшего образования, ориентированные на рынки НТИ. Оператором Олимпиады НТИ является некоммерческая организация - Ассоциация участников технологических кружков. Профили Олимпиады НТИ выбраны на основе приоритетов Национальной технологической инициативы: «Автономные транспортные системы», «Большие данные и машинное обучение», «Системы связи и дистанционного зондирования Земли», «Интеллектуальные энергетические системы», «Создание систем протезирования (Нейротехнологии)», «Инженерные биологические системы», «Интеллектуальные робототехнические системы», «Беспилотные авиационные системы», «Ядерные технологии», «Наносистемы и наноинженерия», «Технологии беспроводной связи», «Электронная инженерия: Умный дом», «Передовые производственные технологии», «Виртуальная и дополненная реальность», «Новые материалы и сенсоры», «Водные робототехнические системы» и «Программная инженерия финансовых технологий».

Целевыми победителями Олимпиады НТИ являются школьники, способные реализовывать сложные технические проекты в прорывных областях. Олимпиада должна выделять команды участников с особыми характеристиками мышления, коммуникации и действия, необходимыми для решения задач НТИ. Победители и призеры Олимпиады НТИ должны показывать высокие результаты в области применения предметных знаний в практической работе. Одновременно с этим, система подготовки Олимпиады НТИ должна предоставлять участникам инструменты для подготовки и получения недостающих знаний и практических навыков.

Первый год проведения Олимпиады

Олимпиада НТИ была впервые проведена в 2015/16 учебном году. В отборочных этапах олимпиады приняли участие несколько тысяч школьников, около ста из них были приглашены к участию в заключительном этапе по профилям «Большие данные и машинное обучение», «Системы связи и дистанционного зондирования Земли», «Интеллектуальные энергетические системы», «Автономные транспортные системы». Заключительный этап Олимпиады и торжественные мероприятия проводились в ВДЦ «Орленок».

В 2015-16 учебном году победители и призеры олимпиады могли воспользоваться возможностью добавить дополнительные 10 баллов к сумме баллов за вступительные экзамены, в случае если они поступали в вузы-организаторы Олимпиады НТИ.

[1]Национальная технологическая инициатива (НТИ) — это программа мер, нацеленная на формирование принципиально новых рынков и создание условий для глобального технологического лидерства России к 2035 году. Задача по созданию НТИ поставлена Президентом Российской Федерации 4 декабря 2014 года в Послании к Федеральному собранию.

Второй год проведения Олимпиады

В 2016/17 учебном году Олимпиада проводилась во второй раз по 12 профилям, количество зарегистрированных для участия школьников увеличилось более чем в три раза и достигло 12 тыс., в отборочных этапах приняли активное участие 4 тыс. школьников, на заключительный этап прибыло 306 участников.

Заключительный этап Олимпиады и торжественные мероприятия проводились на площадке ОЦ «Сириус», в лабораториях и помещениях Парка Наук и Искусств. Вечером проходили лекции и неформальные встречи с представителями технологических компаний.

В 2016-17 учебном году четыре профиля Олимпиады НТИ («Автономные транспортные системы», «Большие данные и машинное обучение», «Системы связи и дистанционного зондирования Земли», «Интеллектуальные энергетические системы») входили в «Перечень олимпиад школьников», таким образом победители и призеры смогли воспользоваться льготами при поступлении в вузы России (зависит от правил приема конкретного вуза). Победители и призеры новых профилей также могли воспользоваться бонусами при поступлении в вузы, которые имеют статус «организатор Олимпиады НТИ».

Третий год проведения Олимпиады

В отборе на Олимпиаду 2017/18 учебного года приняло участие более 20 тыс. школьников, подавших более 50 тыс. заявок на различные профили, число которых увеличилось до 17. В финал вышли 578 участников Олимпиады из 51 региона РФ:

Финал стал распределенным и проходил с февраля по апрель 2018 года: Олимпиаду приняли ОЦ «Сириус», МАИ, МИФИ, ТПУ, Университет Иннополис, СПбПУ, ДВФУ, УрФУ. В 2017-18 учебном году девять из 17 профилей Олимпиады НТИ входят в «Перечень олимпиад школьников» (приказ Минобрнауки России от 30.08.2017 № 866) и дают льготы при поступлении в ВУЗы.

Важная составляющая подготовки участников к финалу Олимпиады – открытые для всех желающих хакатоны, вебинары и мастер-классы. Программы этих мероприятий разработаны педагогами профилей Олимпиады НТИ специально для регионов так, чтобы их

можно было провести на минимальном количестве оборудования. Сеть региональных партнеров Олимпиады со статусом Методическая площадка или Площадка подготовки растет с каждым годом, и в 2018 году, на третий год проведения Олимпиады, их количество достигло 74, всего проведенных мероприятий по подготовке (соревнований, хакатонов, сборов) – более 50. Информация о партнерских площадках размещена в специальном разделе официального сайта олимпиады: http://nti-contest.ru/places_to_prepare/.

Организаторы и партнеры Олимпиады НТИ

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

Вузы-соорганизаторы олимпиады:

- ФГБОУ ВО «Московский политехнический университет»;
- ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»;
- ФГАОУ ВО «Национальный исследовательский Томский политехнический университет».

Технологические партнеры

Олимпиада НТИ проводится при поддержке технологических партнеров, количество которых увеличилось, по сравнению с прошлым годом, среди них: РВК (Российская венчурная компания) и АСИ (Агентство стратегических инициатив по продвижению новых проектов) – в роли со-организаторов и генеральных партнеров: Сбербанк, ПАО «Сухой», ОАК, ФИОП Роснано, сеть детских технопарков «Кванториум», Спутникс, Полюс-НТ, BiTronicsLab, КРОК и др. Полный список организаторов и партнеров олимпиады размещен в соответствующем разделе на официальном сайте: http://nti-contest.ru/about/.

Вузы-организаторы профилей Олимпиады НТИ:

- ФГАОУ ВО «Дальневосточный федеральный университет»;
- АНО ВО «Университет Иннополис»;
- ФГАОУ ВПО «Национальный исследовательский ядерный университет «МИФИ»;
- ФГАОУ ВО «Московский физико-технический институт (государственный университет)»;
- ФГАОУ ВО «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики» (Университет ИТМО);
- ФГБОУ ВО «Московский авиационный институт (национальный исследовательский университет)»;
- ФГБОУ ВО «Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»;
- ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина.

К работе методической комиссии был привлечен профессорско-преподавательский состав вузов-организаторов и представители реального сектора экономики. Объективную оценку работы осуществляет жюри, представленное основателями технологических компаний, а также представителями вузов-организаторов.

Вузы-организаторы, входящие в Оргкомитет Олимпиады НТИ, ведут непрерывную работу с талантливыми школьниками.

Московский политехнический университет

Московский политехнический университет при активном участии Инженерной школы (факультета) регулярно ведет мероприятия для школьников. Факультет "Инженерная школа" создан в 2016 году в целях развития работы Московского Политеха с детьми старшего школьного возраста и организации участия университета во всероссийских и региональных программах по поддержке талантливых школьников.

Факультет курирует проект «Инженерный класс в московской школе», инициированный в 2015 году Департаментом образования города Москвы в целях повышения количества выпускников московских школ, поступивших в инженерные вузы столицы. В 2017 году под научно-методическим руководством сотрудников факультета открыты инженерные классы в 26 школах города Москвы (более 1500 учащихся 10-11 классов). Преподаватели инженерной школы ведут занятия в технологических кружках на базе Московского Политеха, курсы повышения квалификации для преподавателей московских школ, организуют инженерные соревнования и профориентационные мероприятия: экскурсии на предприятия, встречи с носителями профессионального опыта, инженерные турниры и соревнования.

Для подготовки учащихся к инженерной проектной деятельности и вовлечения их в техническое творчество инженерная школа с октября 2016 г. открыла кружки для старшеклассников (8-11 класс):

- Космическая инженерия;
- Кружок радиоэлектроники и программирования;
- Программирование на С++;
- Аквапонные системы;
- Кружок 3D моделирования и прототипирования.

Важнейшим направлением работы факультета является организация выездных инженерных школ и профильных смен, попасть на которые имеют возможность дети из любых регионов России, проявившие уникальные способности в научно-технической сфере.

Образовательный центр «Сириус» и Московский политехнический университет являются официальными партнерами. Летом 2016 года Московский Политех выступил соорганизатором трех направлений проектной деятельности в ОЦ «Сириус» и осуществил экспертную поддержку деятельности центра по направлениям «Транспорт» и «Космос». В июле 2017 Московский Политех стал соорганизатором направления «Наука», в котором приняли участие 400 учеников 8-10 классов, прошедшие конкурсный отбор. Преподаватели и студенты университета приняли участие в организации направлений «Беспилотный транспорт и логистические системы», «Спутники и пилотируемая космонавтика», «Персонализированная медицина» и «Современная энергетика». В планах факультета - создание инженерных кружков Московского Политеха на базе Сириуса (радиотехника, аэрокосмическая инженерия) и лаборатория беспилотного транспорта.

С 2016 г. факультет организует участие Московского Политеха в ежегодном форуме талантливых детей «Проектория» в гор. Ярославле. Форум проводится под руководством аппарата полномочного представителя Президента РФ в ЦФО и Министерства образования и науки РФ. В форуме принимают участие до 500 школьников со всей страны. В ноябре 2016 г. и сентябре 2017 г. Московский Политех выступил партнером образовательной программы форума в рамках направления «Технологии движения».

С 2016 г. Инженерная школа (факультет) является соорганизатором и партнером инженерно-конструкторских школ «Лифт в будущее» - программа БФ «Система» по поддержке талантливой молодежи. В течение октября 2017 г. вместе с преподавателями Московского Политеха в ВДЦ «Орленок» дети разрабатывали технологические проекты, две из восьми лабораторий курировали сотрудники и студенты университета.

В январе 2018 г. совместно с Центром педагогического мастерства (Департамент образования гор. Москвы) факультет провел зимнюю инженерную школу Московского Политеха для учащихся инженерных классов города Москвы, приняли участие 50 человек.

В марте 2018 г. открылся детский технопарк Центра развития инжиниринга-инженерно-технологический комплекс, на базе которого проводятся углубленные технико-ориентированные курсы дополнительного образования для школьников; является частью Московского политехнического университета, создан при поддержке Департамента науки, промышленной политики и предпринимательства города Москвы. На данный момент запущены 4 образовательных программы: Транспортный дизайн, Введение в автомобилестроение, Беспилотный транспорт и Современная космонавтика.

Вместе с тем Московский Политех принимает участие в организации других олимпиад, входящих в перечень олимпиад школьников Минобрнауки России 2017/2018 учебного года:

1. Объединенная межвузовская математическая олимпиада (ОММО). Проводится для одиннадцатиклассников по инициативе группы московских вузов с 2009 года.

2. Международная олимпиада школьников «Искусство графики». Проводится с целью выявления и привлечения наиболее подготовленных, талантливых и профессионально ориентированных учащихся средних художественных училищ РФ и ближнего зарубежья, школьников, слушателей подготовительных курсов, развитие у обучающихся творческих способностей, содействие профессиональной ориентации школьников.

Московский Политех также участвует (имеет статус организатора или со-организатора) в следующих мероприятиях: инженерно-конструкторское направление предпрофессиональной олимпиады Московской олимпиады школьников 2016-18 гг., предпрофессиональный экзамен для инженерных классов в Москве 2016-2018 гг., инженерное направление Московского конкурса научно-исследовательских и проектных работ учащихся, проектные смены ОЦ «Сириус», турнир двух столиц по робототехнике и т.д.

Санкт-Петербургский политехнический университет Петра Великого

Санкт-Петербургский политехнический университет Петра Великого с целью популяризации инженерных наук и привлечения талантливых абитуриентов проводит ряд масштабных мероприятий:

- летняя и зимние политехнические школы;
- олимпиады для школьников, в которых в прошлом году приняло участие более 1800 школьников;
- организовано взаимодействие с региональными центрами развития творчества одаренных детей;
- в различных школах и кружках, организованных цифровой мастерской Фаблаб Политех ежегодно принимает участие более 5000 школьников из Санкт-Петербурга и Ленинградской области. С 2015 года Фаблаб помогает развивать центры технического творчества и в регионах, наиболее крупный проект представлен в городе Норильск;
- для привлечения и подготовки школьников к поступлению созданы 5 массовых онлайн-курсов, размещенных в настоящее время на портале Лекториум: «Инженерное дело», «Математика», «Физика», «Химия», «Обществознание».

Одним из ключевых событий 2015 года стало проведение Фестиваля научно-технического творчества молодежи на территории Политехнического университета Петра Великого. Мероприятие, ориентированное на школьников 10 и 11 классов, посетило более 15 000 абитуриентов.

Политехнический университет Петра Великого является организатором олимпиад, входящих в перечень Минобрнауки РФ: Межрегиональная олимпиада школьников по математике и криптографии, Политехническая олимпиада школьников, Объединенная межвузовская математическая олимпиада).

Томский политехнический университет

Томский политехнический университет является одним из опорных вузов в программе инновационного развития ПАО «Газпром». В 2009 году ТПУ и ПАО «Газпром» подписали соглашение о сотрудничестве. В Институте природных ресурсов создан и действует полный учебно-научный цикл, включающий обучение (рабочим профессиям, прохождение производственных преддипломных практик, переподготовку и повышение квалификации специалистов компании, создание кадрового задела) и исследования (Инновационный научно-образовательный центр трубопроводного транспорта нефти и газа, в составе которого действует Международная научно-образовательная лаборатория «Нефтегазовая гидродинамика и теплообмен», лаборатория неразрушающего контроля).

Участие со своим треком «Электронная инженерия: Умный дом» в рамках Олимпиады НТИ стало логичным продолжением работы Томского политехнического университета по развитию инженерных навыков у школьников и студентов. На траектории элитного технического образования ТПУ студенты воплощают в жизнь свои самые смелые инженерные идеи. Однако многолетний опыт показывает, что деятельность по развитию интереса к инженерии и закладыванию базовых знаний в предметной области нужно прививать гораздо раньше, поэтому Томский политехнический университет ведет постоянную работу по привлечению в инженерные профессии талантливых школьников с более раннего возраста. Одним из таких мероприятий стал проект «Юный инженер», который проводится совместно с Департаментом образования Томской области и НП “Лифт в будущее”. Суть проекта в том, чтобы студенты ТПУ, уже имеющие опыт в реализации инженерных проектов, готовили и проводили учебные занятия со школьниками г. Томска и Томской области. Цель проекта сделать доступным современное дополнительное образование для детей всего региона. А также развивать практику наставничества, которая реализуется в ТПУ с 2012 г. и в 2018 г. вошла в шорт-лист лучших практик на форуме “Наставник”.

Для знакомства с инженерными профессиями будущего с 2016 г. в ТПУ запущена игра «Агенты будущего» (http://corpus-future.net/) - это образовательная online-система, построенная на технологиях игрового моделирования, геймификации и многопользовательского онлайн-обучения. Игра построена на выполнении квестов (заданий). Каждое задание позволяет изучить аспект или технологии отдельной отрасли инженерии/техники. В ходе прохождения сюжета школьник проходит путь к будущей инженерной специальности в качестве исследователя, технолога, стратега или оператора. Квест-задания проходят в энергоблоке, в виртуальных лабораториях, вычислительном центре и других симуляторах.

Важным этапом в ходе подготовки к Олимпиаде НТИ стала Школа инженерных проектов ТПУ (ШИП), во время которой школьники от 14 до 17 лет создавали макет узла космического аппарата, макет спутника, могли разработать и реализовать отдельные комплексы, которые по алгоритмам, разработанными школьниками, рассчитывают оптимальные характеристики передачи, позволяющие снизить потери электроэнергии.

Целевая аудитория ШИП – учащиеся образовательных учреждений от 14 до 17 лет. Преподаватели ТПУ выезжают к школьникам других городов с проектами, которые в течении недели реализуют с ребятами. Тематики проектов: «Проектирование малых космических аппаратов с использованием 3D моделирования». Результат: макет узла космического аппарата и макет спутника, выполненный посредством 3D печати. «Технологии Smart Grid в электросетях нового поколения». Результат: разработка и реализация отдельных комплексов программно-аппаратных средств, которые по алгоритмам, разработанными школьниками, рассчитывают оптимальные характеристики передачи, позволяющие снизить потери электроэнергии. «Теплотехнологическая переработка биомассы в энергетические продукты и энергию». Результат: собственный вариант теплотехнологической энергоустановки и созданный на её основе экспериментальный стенд. План на 2018/19 уч. год разработка ШИП по тематике «Газовый комплекс».

Со школьниками проводится много других интересных и полезных мероприятий:

«Университетские субботы» - проект, позволяющий школьнику вникнуть в естественные и точные науки (физика, химия, математика, информатика), узнать о потенциале ТПУ, его лабораториях и передовых разработках. Здесь школьникам предоставляются физические задачи с техническим содержанием, которые разрабатываются в колоборации с преподавателями-физиками и преподавателями исследовательских школ (институтов). Решение практических задач для школьников младших классов представлено в виде красочного шоу. Такой подход позволяет школьникам увидеть законы физики в действии.

В ТПУ сформирована единая система довузовской подготовки инженерно-технического образования – подготовительные курсы ТПУ, профильные классы Газпрома, центр занимательных наук «Склад ума».

Школьники имеют возможность работать в KIDsLAB, где проводится обучение в кружках начального технического творчества, авиамодельном, стендового моделизма, робототехники. KIDsLAB - это производственно-образовательная мастерская, где можно изготовить практически все. Основной задачей KIDsLAb является помощь школьникам и студентам первого курса в приобретении опыта изготовления инженерных предметов, внедрения своей разработки - от идеи до создания прототипа, поиске маркетинговых решений и создании инновационных продуктов.

Томский политехнический университет принимает участие в организации олимпиад, входящих в перечень олимпиад школьников Минобрнауки России 2017/2018 учебного года:

1. Интернет-олимпиада школьников по физике, целью олимпиады является повышение качества знаний по физике, повышение мотивации к поступлению в технические вузы. С 30.03.2017 г. олимпиада получила статус международной.
2. Многопрофильная инженерная олимпиада «Звезда». Участие в данной олимпиаде дает возможность школьнику существенно расширить свой кругозор, получить полное представление о тенденциях развития экономики и техники.
3. Межрегиональная олимпиада школьников по математике САММАТ.
4. Олимпиада школьников Сибирского Федерального округа «Будущее Сибири» - мотивирует школьников для поступления в технические вузы и расширяет знания в предметной области.
5. Межрегиональная олимпиада школьников «Высшая проба» дает возможность учащимся попробовать себя в разных областях знаний, как в технических, так и в гуманитарных, что способствует гармоничному развитию личности.

Также ТПУ является соорганизатором таких олимпиад, конкурсов и конференций как:

1. Политехническая олимпиада по математике, физике, химии, информатике, русскому языку, обществознанию, проводимая на территории стран СНГ. Данная олимпиада нацелена на выявление талантливых абитуриентов в странах СНГ, развитие и проверку глубоких знаний по предложенным дисциплинам.
2. Олимпиада ПАО «Газпром», направленная на развитие креативного подхода к решению инженерных задач, проверки знаний и мотивации к изучению газодобывающей отрасли.
3. Олимпиада для детей с ОВЗ. Совместно с ОГБОУ «Школа-интернат для обучающихся, нуждающихся в психолого-педагогической и медико-социальной помощи» в декабре 2017 года проведена VI Межрегиональная дистанционная олимпиада школьников с ограниченными возможностями здоровья, в которой приняли участие 600 школьников из 19 регионов РФ, Казахстана, Беларуси.
4. Всероссийская конференция-конкурс исследовательских работ школьников "Юные исследователи - науке и технике" направлена на выявление и развитие у молодежи творческих способностей и интереса к научно-исследовательской деятельности, создание необходимых условий для поддержки одаренной молодежи, распространение и популяризация научных знаний среди молодежи.
5. Региональные конкурсы «Мой выбор — химия», «От школьной физики – к высоким технологиям», «Царица наук - математика», «Выбор будущего - информатика»

направлены на стимулирование учителей химии, физики, математики и информатики в школах за высокие показатели учеников в сдаче единого государственного экзамена (ЕГЭ) в 11 классах и во Всероссийских олимпиадах. В 2017 году приняли участие 263 учителя.

6. Ежегодно летом с июня по август на базе ТПУ проводятся летние смены. Каждая смена имеет свою тематику. Ребятам предлагают и усилить свои знания по профильным для ТПУ предметам, и принять участие в проектной и исследовательской деятельности, а также начать изучение китайского языка.

Дальневосточный Федеральный Университет

Дальневосточный Федеральный Университет регулярно проводит мероприятия по привлечению абитуриентов, выявлению и поддержке талантливых школьников в области инженерных и естественных наук.

1. Программы дополнительного образования:

Программы «Малых академий» для школьников 9-11 классов. В период с 2013 по 2017 годы было организовано 8 профильных инженерных и естественнонаучных «малых академий» (охват школьников составил более 6000 человек):

- малая инженерная академия;
- школа юного системотехника;
- школа юных биологов и экологов;
- школа юных химиков;
- школа юных математиков;
- школа юных физиков;
- школа юных программистов;
- малая академия биомедицины по профилям: медицинское направление и пищевые биотехнологические и биотехнические направления.

На базе Университета организованы курсы подготовки к ЕГЭ по профильным предметам:

- физика;
- математика;
- биология;
- химия.

Важными мероприятиями для Университета являются краткосрочные интенсивные программы, такие как «Тихоокеанская математическая школа». В марте 2017 года такой интенсив прошли 59 школьников 10-11 классов со всего Приморского края.

2. Олимпиады и конкурсы:

В 2015 - 2016 уч. году в ДВФУ организовано и проведено 33 Олимпиады школьников: в том числе 11 олимпиад, в которых ДВФУ является организатором и 22 олимпиады, в которых университет выступает в роли региональной площадки.

В 2015-16 учебном году Дальневосточный федеральный университет послужил в качестве региональной площадки для проведения Всероссийских олимпиад школьников инженерной и естественно-научной направленности:

- «Полуфинал Всероссийской командной школьной олимпиады по программированию»;
- Всероссийская олимпиада школьников (муниципальный и региональный этапы);
- Межрегиональная Всероссийская олимпиада школьников «Ломоносов» по биологии и математике (2016 год - 48 участников, из них – 1 победитель, 6 призеров из территорий Приморского края, Хабаровского край, Республики Саха (Якутия));
- Всероссийская «Объединенная межвузовская математическая олимпиада школьников» по математике (2016 год – 24 участника из территорий Приморского края, Хабаровского края, Амурской области);
- Многопрофильная инженерная олимпиада школьников «Звезда» по технике и

технологии кораблестроения и морской техники, машиностроению, технике и технологии наземного транспорта, авиационной и ракетно-космической технике, технологии материалов, ядерной энергетике, электронике, радиотехнике и системе связи (2016 год - 250 участников (7 - 11 классы), призеры – 30, победители – 10);

- Всероссийская «Северо-Восточная олимпиада школьников» по математике, информатике и химии (2016 год - 36 участников, из них – 1 победитель, 9 призеров из территорий Приморского края, Хабаровского края, Амурской области).

Дальневосточный федеральный университет также выступил как организатор следующих олимпиад:

- «Океан знаний» по математике, физике, химии, биологии, экологии.
- Турнир юных программистов.

На протяжении 20 лет Университет выступает соорганизатором Регионального этапа Всероссийской олимпиады школьников «Приморский интеллект» организованной на базе ВДЦ «Океан» (500 участников ежегодно).

С 2012 года реализуется уникальная совместная образовательная программа ДВФУ и ВДЦ «Океан» в рамках смены «Российский интеллект» для победителей и призеров региональных этапов Всероссийской олимпиады школьников и победителей/призеров отборочного этапа олимпиады школьников «Океан знаний» 9, 10 и 11 классов (2015 год - 250 учащихся из 22 территорий, 2016 год - 400 учащихся из 21 территории России).

Структура отбора участников Олимпиады НТИ

Соревнование проходит в три этапа. Первый и второй отборочные этапы проходили с октября по декабрь 2017 года в заочной форме на интернет-платформе «Stepik» (http://stepik.org) и в инженерных онлайн-симуляторах.

Отборочные этапы сопровождались различными подготовительными мероприятиями, среди которых были дистанционные мероприятия (вебинары), мероприятия для самостоятельной подготовки (онлайн-курсы), мероприятия направленные на командообразующую деятельность (специальные встречи, интенсивы, очные курсы на площадках по подготовке), мероприятия, направленные на получение практических навыков (интенсивы).

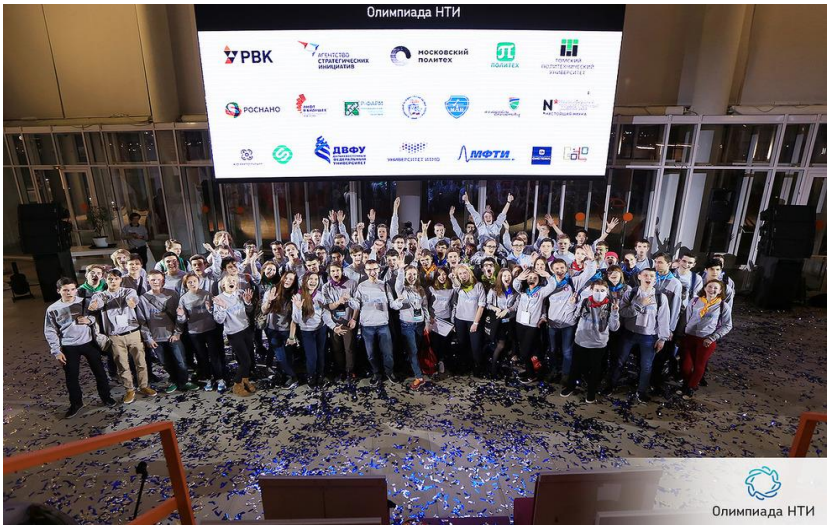
Заключительный этап Олимпиады состоит из двух частей: индивидуальное решение предметных задач по выбранным профилям и командная разработка инженерного решения с испытанием его на стенде. Задание второй части заключительного этапа имеет свою специфику для каждого профиля.

Работа с участниками

Организаторы Олимпиады заинтересованы в дальнейшем сопровождении ее участников. Практика показывает, что школьники – участники Олимпиады НТИ также заинтересованы в дальнейшем сотрудничестве. В организации заключительного этапа Олимпиады НТИ 2016\17 в качестве волонтеров приняли участие победители и призеры Олимпиады НТИ 2015\16, студенты первых курсов из различных регионов России. Участники заключительного этапа 2016\17 из числа учеников одиннадцатого класса также выразили желание принять участие в организации олимпиады и подготовке участников в качестве волонтеров.

В 2017/18 годах число партнерских мероприятий Олимпиады увеличилось: на странице http://nti-contest.ru/participants/posle_finala/ представлен список мероприятий организаторы которых специально приглашают участников Олимпиады и дают им бонусы при конкурсном отборе.

Так, на конкурсном отборе на авиационную смену в ВДЦ «Океан», призёры и победители профилей по «Интеллектуальным робототехническим системам» и «Беспилотным авиационным системам» получают весомые бонусы (70 и 130 баллов соответственно). На авиационную смену в «Артеке» получили приглашение лучшие участники профиля «Беспилотные авиационные системы». Отбор на июльскую проектную смену в ОЦ «Сириус» предполагает дополнительные баллы для призеров и победителей Олимпиады НТИ.



На фото: участники заключительного этапа Олимпиады во время торжественной церемонии закрытия.

Партнерство с инженерными соревнованиями

Оргкомитет Олимпиады НТИ, в свою очередь, ежегодно утверждает перечень инженерных мероприятий и конкурсов, победители которых, могут принять участие в заключительном этапе Олимпиады, минуя отборочные. В 2016/17 и 2017/18 таковыми мероприятиями являлись: IT-хакатон GoTo, инженерно-конструкторские школы «Лифт в будущее», всероссийский форум «Будущие интеллектуальные лидеры России» и World Skills High Tech.

Подготовка участников

Чтобы участники могли восполнить недостаток практических компетенций и изучить оборудование, на котором им предстоит работать на заключительном этапе Олимпиады НТИ, разработчики направлений представляют методические материалы для самостоятельной практики и самоподготовки, проводят вебинары для участников и педагогов с ответами на вопросы и подбирают подготовительные курсы. Все указанные материалы находятся в свободном доступе и размещены на официальном сайте олимпиады, на страницах профилей в разделе «Материалы для участников».

Прошедшая олимпиада является промежуточным итогом работы по реализации дорожной карты НТИ «Кружковое движение»: подготовка к ней велась в фаблабах, ЦМИТах, детских технопарках, на базе активных школ и лицеев, центров дополнительного образования по всей России. Рабочая группа «Кружковое движение» НТИ направлена на развитие 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.

Популяризация Олимпиады НТИ

Олимпиада НТИ проходит с широким освещением мероприятий в различных средствах массовой информации (телевидение, печатные издания, электронные издания), среди которых как узкопрофильные издания, аудитория которых включает школьников старших классов, учителей, преподавателей в определенной научной среде, так и крупные федеральные издания общего характера. Во время проведения отборочных этапов Олимпиада НТИ освещалась в публикациях специализированных научных и общеобразовательных порталов (Занимательная робототехника, Дневник.ру, НИА Наука, Научная Россия, 5 углов, Нейроновости), официальных региональных порталов (Калининград, Тюмень, Курск, Курган, Тамбов, Мурманск, Новогород, Вологда и т.д.), в печатных изданиях («Качество образования»). Заключительный этап Олимпиады НТИ проходил при участии журналистов таких печатных изданий, как Российская газета, Комсомольская правда, Огонек, Сноб, Известия, Учительская газета; федеральных телевизионных каналов (Первый канал, Россия 24), научно-популярных блогов Мел, Постоянная Планка, Химия-Просто, Первый научный. Широкое освещение мероприятий заключительного этапа имеет своей целью распространение информации среди потенциальных участников Олимпиады НТИ будущего года - учеников 6-10 классов и направлено на 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 2017-2018 учебного года, был посвящен изучению алгоритмов компьютерного зрения и определения изменений в, заранее заложенной в робототехническое устройство, карте местности посредством исследования данной местности с использованием экстероцептивных сенсоров. 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 продемострировать простейшие навыки составления и отладки программ, обрабатывающих массивы данных, и понимание таких тем, как битовые операции, вычислительная геометрия, теория графов. Поскольку именно уверенные знания в этих темах позволят участникам получить навыки, необходимые для решения фианльной задачи. Задачи по математике проверяли у участников знания по комбинаторике, геометрии, тригонометрии.

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 делились на два блока: задачи на изучение сложных тем математики, такие как интерполяция, параметризованная кривая и геометрическое место точек - те знания, которые применяются при навигации и планировании пути перемещения в робототехнических системах, а также задачи на составление алгоритмов. Задачи по программированию требовали от участников погружения в такие робототехнические темы, как навигация робототехнических устройств и планирование маршрутов перемещения, построение карты с использованием экстероцептивных сенсоров, цифровая обработка сигналов и компьютерное зрение. Для получения дополнительной информации, необходимой для решения задач второго этапа, командам были предложены образовательные материалы, разработанные в Университете Иннополис.

Команды, прошедшие в финал профиля, приглашались на очный хакатон (учебно-тренировочные сборы), на котором 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, а также реализовывали простейшие алгоритмы обработки изображения, захваченного с камеры робототототехнического устройства.

Таким образом при решении финальной задачи в очном заключительном этапе 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 перекликались с темами задач второго этапа. А задачи по информатике предлагали поэтапное решение задачи планирования перемещения роя робототехнических устройств с использованием вычислительной геометрии.

1. ПЕРВЫЙ ОТБОРОЧНЫЙ ЭТАП

Первый отборочный тур проводится индивидуально в сети Интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Для каждого 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 три временных слота по 2 дня каждый, когда они могли решать задачи по предмету. Решение каждой задачи дает определенное количество баллов. За каждую следующую попытку сдать решение баллы начисляются по формуле $\frac{P}{N}$, где P - максимальное количество очков, которые можно получить за решение задачи, N - количество попыток. Всего на каждую задачу математике давалось по 2 попытки, на каждую задачу по информатике - 4 попытки.

Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и информатика) — суммарно от 0 до 30 баллов.

Первая попытка

2.1. Задачи по математике (9 класс)

Задача 2.1.1. (2 балла)

Найдите сумму всех полных квадратов, которые при делении на 11 в частном дают простое число и в остатке 4.

Решение

$n^2 = 11p + 4 \Longrightarrow 11p = (n-2)(n+2)$. $n - 2 = 1$ не подходит, значит $n - 2 = 11$ и $n + 2 = p$ или $n - 2 = p$ и $n + 2 = 11$. В первом случае $p = 15$ не простое, во втором случае $n = 9$. (в наших рассуждениях n натуральное, p простое число)

Ответ: 81

Задача 2.1.2. (3 балла)

Вася и Петя живут в одном доме у прямой автодороги. Чтобы сесть в школьный автобус, они одновременно вышли из дома, Петя пошел в сторону остановки А навстречу автобусу со скоростью 6 км/час, а Вася в сторону остановки В по направлению движения автобуса со скоростью 4 км/час. Они оба подошли к остановкам как раз к моменту прибытия автобуса. Найдите отношение расстояния пройденного Петей к расстоянию пройденного Васей, если скорость автобуса 60 км/час. Результат округлите до сотых.

Решение

Пусть Петя шел к своей остановке t часов и прошел расстояние $6t$ км. В момент, когда автобус подобрал его в остановке В, Вася находился на расстоянии $10t$. Автобус догоняет Васю со скоростью 56км/час и догонит через $\dfrac{10t}{56}$ часов. Вася к этому моменту пройдет $4t + \dfrac{4 \cdot 10t}{56} = \dfrac{66}{14}t$ километров. Отношение расстояний тогда равно $6t : \dfrac{66}{14}t = \dfrac{14}{11} = 1,(27)$.

Ответ: 1,27

Задача 2.1.3. (3 балла)

В автомат для размена денег загружены монеты в 2, 5 и 10 рублей. Сколько всего способов разменять 100 рублевую купюру в этом автомате?

Решение

Пусть x, y, z количество монет в 2, 5 и 10 рублей соответственно. Тогда $2x+5y+10z = 100$. Видим, что $x = 5k$, $y = 2n$, иначе уравнение не имеет решения в целых числах. Осталось найти количество неотрицательных целых решений уравнения $k + n + z = 10$. А это число сочетаний с повторениями $\overline{C}_3^{10} = C_{12}^{10} = 66$.

Ответ: 66

Задача 2.1.4. (3 балла)

На доске были написаны n первых натуральных чисел $(1, 2, \ldots n)$. Вася стер одно число. Петя заметил, что среднее арифметическое оставшихся стало $\dfrac{45}{4}$. Какое число стер Вася?

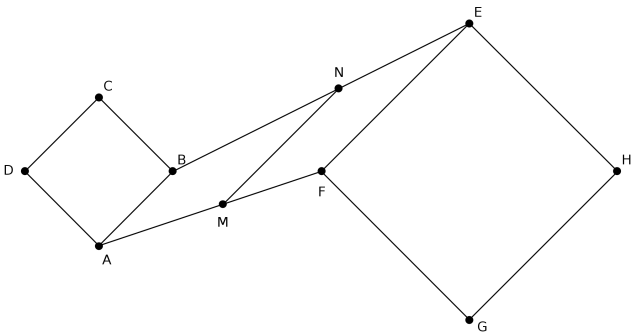
Решение

Из условия ясно, что $n > 1$. Пусть стерли число x. Тогда $x = \frac{n(n+1)}{2} - \frac{45}{4}(n-1) = \frac{2n^2 - 43n + 45}{4} = \frac{(2n-41)(n-1)}{4} + 1$. По условию $1 \leqslant x \leqslant n$. Отсюда $0 \leqslant \frac{(2n-41)(n-1)}{4} \leqslant n-1$, $0 \leqslant 2n - 41 \leqslant 4$. Решением неравенства являются только $n = 21$ или $n = 22$. При $n = 22$ число x не является целым, при $n = 21$ $x = 6$.

Ответ: 6

Задача 2.1.5. (4 балла)

На рисунке внизу $ABCD$ и $EFGH$ квадраты, их площади равны 32 см^2 и 66 см^2 соответственно. Отрезок MN параллелен основаниям трапеции $ABEF$ и делит её площадь пополам. Найдите длину отрезка MN.



Решение

Пусть прямые AB и EF пересекаются в точке K. Тогда $2S_{KMN} = S_{KBE} + SKAF$. Отсюда $2 = \frac{S_{KBE}}{S_{KMN}} + \frac{S_{KAF}}{S_{KMN}} = \frac{BE^2}{MN^2} + \frac{AF^2}{MN^2} = \frac{98}{MN^2}$.

Ответ: 7

2.2. Задачи по математике (10-11 класс)

Задача 2.2.1. (2 балла)

На плоскости отмечены 17 точек, не лежащих на одной прямой (т.е. найдутся хотя бы 3 точки, не лежащие на одной прямой). Через каждые две точки провели прямую. Какое наименьшее количество различных прямых могло получиться?

Решение

Пусть на одной прямой x точек, а на другой — $k-x$ точек (не считая общую). Прямые, проходящие через точки на различных прямых, не совпадают. Поэтому их $x(k-x)$. Эта величина уменьшается когда x стремится к 0 или k. Если точки перенесем на одну из отмеченных прямых, то количество прямых будет наименьшим. Все точки, кроме одной, лежат на одной прямой. Тогда количество раздичных прямых равно 17.

Ответ: 17

Задача 2.2.2. (3 балла)

Для проведения олимпиады преподаватели разбивают 60 школьников следующим образом: список в алфавитном порядке разбивается на 4 части, первая идет в первую аудиторию, вторая — во вторую и т. д. При этом в каждую аудиторию отправляется хотя бы один школьник. Сколькими способами можно произвести распределение?

Решение

В списке надо провести 3 разделительные черты. Эти черты можно проводить в промежутках между фамилиями. Надо выбрать 3 промежутка из 59 и поставить там разделительную черту. Всего способов $C_5 9^3 = \dfrac{59!}{3!56!}$.

Ответ: 32509

Задача 2.2.3. (3 балла)

Имеется таблица 100×100 натуральных чисел, в которой строки пронумерованы сверху вниз числами от 1 до 100. В k-ой строке таблицы слева направо написаны числа, которые являются арифметической прогрессией с первым членом 1 и разностью k. Какое наибольшее число встречается среди элементов диагонали, идущей из левой нижней клетки в правую верхнюю?

Решение

На отмеченной диагонали окажется $(101-k)$-ый член арифметической прогрессии, находящейся на k-ой строке. Тогда $x_{101-k} = 1+(100-k)k = 2501-(x-50)^2 \leqslant 2501$.

Ответ: 2501

Задача 2.2.4. (3 балла)

Бочка в виде прямого кругового цилиндра полностью заполнена водой. Радиус основания 40 см, а высота 120 см; бочка стоит на горизонтальной поверхности. На

какой угол необходимо наклонить эту бочка, чтобы вылить ровно половину воды. В ответ напишите тангенс этого угла, если необходимо, округлите до сотых.

Решение

Любая плоскость, проходящая через центр цилиндра, делит её на 2 равные части. Поэтому бочку надо наклонить так, чтобы плоскость, проходящая через точку на границе верхнего основания и центр цилиндра, стала горизонтальной. Тогда угол наклона станет равным углу между этой плоскостью и основанием. Тангенс этого угла равен отношению высоты бочки на диаметр основания, т.е. $\frac{120}{80}$.

Ответ: 1,5

Задача 2.2.5. (4 балла)

Найдите наибольшее натуральное значение n такое, что 120! делится без остатка на 12^n.

Решение

В 120! простое число p входит в степени $\left[\frac{120}{p}\right] + \left[\frac{120}{p^2}\right] + \left[\frac{120}{p^3}\right] + \ldots$. Таким образом находим степени 2 и 3 входящие в разложение 120! на простые множители:

$$120! = 2^{60+30+15+7+3+1} \cdot 3^{40+13+4+1} \cdot A = 12^{58} \cdot A,$$

где A не делится на 12.

Ответ: 58

2.3. Задачи по информатике

Задача 2.3.1. (1 балл)

Вам дана шахматная доска размером 8×8, в поле a которого находится король. Определите, за какое минимальное количество ходов король может добраться до клетки b.

Шахматный король может переместиться на любое соседнее поле, с которым есть хотя бы одна общая точка.

Формат входных данных

Первая строка входных данных содержит два числа a_v, a_h — координаты, где изначально находится король ($1 \leqslant a_v, a_h \leqslant 8$). Вторая строка также содержит два числа b_v, b_h — координаты клетки b, в таком же формате.

Формат выходных данных

Выведите одно целое число — ответ на задачу.

Примеры

Пример №1

Стандартный ввод
2 3 5 8
Стандартный вывод
5

Решение

Для минимизации количества ходов выгодно перемещать короля по диагонали, пока не дойдем до строки b_v или столбца b_h. Это $min(|b_v - a_v|, |b_h - a_h|)$ ходов. Оставшуюся часть доходим ходами по строке/столбцу, их всего $max(|b_v - a_v|, |b_h - a_h|) - min(|b_v - a_v|, |b_h - a_h|)$. Суммируем два ответа и получаем, что ответ равен $max(|b_v - a_v|, |b_h - a_h|)$.

Пример программы

Ниже представлено решение на языке C++

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    int av, ah, bv, bh;
    cin >> av >> ah >> bv >> bh;
    cout << max(abs(av - bv), abs(ah - bh)) << '\n';
}
```

Задача 2.3.2. (2 балла)

Фёдор перешел в 11 класс, и подготовка к ЕГЭ — его приоритетная цель в этом году.

Сегодня он узнал, что такое маска подсети, IP-адрес и адрес сети. Адрес сети получается как применение *поразрядной конъюнкции* (логическая операция И) к IP-адресу и маске подсети.

Поразрядная конъюнкция — применение к соответствующим битам операции логического И. Другими словами, если соответствующие биты равны 1, результирующий двоичный разряд будет равен 1, иначе 0.

Применение поразрядной конъюнкции к маске подсети и IP-адресу выглядит так:

```
11111111.11111111.11111111.00000000 (255.255.255.0)
&
11000000.10101000.00000000.00000001 (192.168.0.1)
-----------------------------------
11000000.10101000.00000000.00000000 (192.168.0.0) <--- Адрес сети
```

Вам дана маска подсети и n IP-адресов. Ваша задача определить сколько различных адресов сети получится.

Формат входных данных

В первой строке вам дана маска сети. Во второй строке дано числа n — количество IP-адресов ($1 \leqslant n \leqslant 1000$). Следующие n строк содержат IP-адреса.

Формат выходных данных

Выведите количество различных адресов сети.

Примеры

Пример №1

Стандартный ввод
255.255.255.0 3 192.168.0.1 192.168.0.28 192.168.1.1
Стандартный вывод
2

Решение

Давайте представим маску и IP-адреса как простые 32 битные числа, чтобы было удобно с ними работать.

Теперь получим все возможные адреса сети путем $\widehat{\Pi}$оразрядная конъюнкция всех IP-адресов с маской. Дальше нужно убрать одинаковый адреса сети. Давайте отсортируем все получившееся адреса сети. Дальше не трудно посчитать сколько будет различных, так как одинаковые теперь стоять друг за другом.

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <vector>
#include <set>

using namespace std;

int address(int p1, int p2, int p3, int p4) {
        int data = p1;
        data <<= 8;
        data |= p2;
        data <<= 8;
```

```
        data |= p3;
        data <<= 8;
        data |= p4;
        return data;
}

int read_ad() {
        int m1, m2, m3, m4;
        char c;
        cin >> m1 >> c >> m2 >> c >> m3 >> c >> m4;
        return address(m1, m2, m3, m4);
}

set <int> s;

int main() {
        int mask = read_ad();
        int n;
        cin >> n;
        for (int i = 0; i < n; i++) {
                int x = read_ad();
                s.insert(mask & x);
        }
        cout << s.size();
        return 0;
}
```

Задача 2.3.3. (3 балла)

Юный поэт Марк хочет выпустить свой первый сборник стихотворений. Как вы знаете, рифма — одна из важных деталей любого стихотворения. И Марк хочет, чтобы во всех его стихах сохранялась рифма; но из-за недостатка опыта он еще не научился определять, имеет ли его стихотворение рифму. И вам необходимо ему помочь.

Стихотворение Марка представляет из себя n непересекающихся четверостиший. Четверостишие — 4 строки, состоящие из латинских букв и пробелов. Словом считается последовательность букв, не имеющая пробелов. Два слова имеют рифму, если не менее двух их последних букв совпадают. Две строки имеют рифму, если их последние слова рифмуются. Существует 3 способа задания рифмы в четверостишье:

1. Если независимо рифмуются строки номеров (1, 2) и (3, 4).
2. Если независимо рифмуются строки номеров (1, 3) и (2, 4).
3. Если независимо рифмуются строки номеров (1, 4) и (2, 3).

И наконец, стихотворение рифмуется, если все его четверостишья рифмуются одним и тем же способом. Получив стихотворение Марка, вам необходимо ответить, имеет ли оно рифму.

Формат входных данных

Первая строка входных данных содержит целое число $k = 4 * n$ — количество строк в стихотворении Марка ($4 \leqslant k \leqslant 1000$).

Далее следует стихотворение, в виде k строк, состоящих из латинских букв и

пробелов (ни одна строка не начинается и не оканчивается пробелом). Длина каждой строки не превышает 1000 символов.

Формат выходных данных

Необходимо вывести *Yes*, если стихотворение рифмуется, а иначе — *No*.

Примеры

Пример №1

Стандартный ввод
4 If bees stay at home Rain will soon come If they fly away Fine will be the day

Стандартный вывод
Yes

Пример №2

Стандартный ввод
4 One two three Let me see I think about u I like u

Стандартный вывод
No

Пример №3

Стандартный ввод
8 I want a tree For shade and rest I want a tree Where birds can nest So I planted A green tree You must see Rest guaranteed

Стандартный вывод
No

Решение

Для того, чтобы понять рифмуются ли две строки, необходимо всего лишь сравнить на равенство по два последних символа строки(не должно быть пробелов). Теперь можно для каждого типа рифмы проверить, поддерживается ли он в каждом четверостишии стихотворения.

Пример программы

Ниже представлено решение на языке Python3

```
k = int(input())
k = k // 4

def is_ryf(s1, s2):
        if len(s1[-1]) < 2 or len(s2[-1]) < 2:
                return False

        if (s1[-1][-2:] == s2[-1][-2:]):
                return True
        return False

t1 = True
t2 = True
t3 = True

for i in range(0, k):
        s1 = input().split()
        s2 = input().split()
        s3 = input().split()
        s4 = input().split()

        if not(is_ryf(s1, s2) and is_ryf(s3, s4) and t1):
                t1 = False
        if not(is_ryf(s1, s3) and is_ryf(s2, s4) and t2):
                t2 = False
        if not(is_ryf(s1, s4) and is_ryf(s2, s3) and t3):
                t3 = False

if t1 or t2 or t3:
        print("Yes")
else:
        print("No")
```

Задача 2.3.4. (4 балла)

Вам дан невырожденный треугольник и точка. Ваша задача состоит в том, чтобы определить периметр видимой части треугольника из данной точки.

Гарантируется, что точка находится строго вне треугольника.

Формат входных данных

В первой строке вам дано 6 чисел: $(x_1; y_1)$, $(x_2; y_2)$ и $(x_3; y_3)$ — точки образующие треугольник соответственно.

На следующей строке дано 2 числа — координаты точки $(x; y)$.

Все координаты точек целые и не превосходят по модулю 10^5.

Формат выходных данных

Вывести одно число: периметр видимой части треугольника из точки $(x; y)$.

Ответ будет засчитан, если он отличается от правильного не более чем на 10^{-6} относительной или абсолютной погрешности.

Примеры

Пример №1

Стандартный ввод
0 1 1 0 1 1 2 2
Стандартный вывод
2.0000000000

Пример №2

Стандартный ввод
0 1 1 0 1 1 2 1
Стандартный вывод
1.0000000000

Пример №3

Стандартный ввод
0 0 0 1 1 0 1 1
Стандартный вывод
1.4142135624

Решение

Если мы зафиксируем какой-нибудь отрезок треугольника, то когда его будет видно ? Логично, что его будет видно, если точка, из которой мы смотрим, находится по другую сторону прямой, образованной этим отрезком, для третьей точки треугольника.

Осталось научиться проверять, лежат ли две точки по разные стороны от прямой. Это можно делать с помощью длины векторного произведения. Для удобства введем обозначения: t_1, t_2, t_3 — точки треугольника, p — точка, из которой смотрим. Возьмем длину векторного произведения c_1 для векторов из точки t_1 в точку t_2 и из t_1 в точку p. Аналогично c_2 для векторов из точки t_1 в точку t_2 и из t_1 в точку t_3. Если знаки c_1 и c_2 отличаются, то этот отрезок виден из точки, и нужно прибавить к ответу длину этого отрезка.

Пример программы

Ниже представлено решение на языке C++

```cpp
#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>

using namespace std;

typedef long long ll;

struct Point {
        int x, y;
        Point() {}
        Point(int x, int y) : x(x), y(y) {}
};

Point operator-(Point &p1, Point &p2) {
        return Point(p2.x - p1.x, p2.y - p1.y);
}

istream & operator >>(istream &is, Point &p) {
        is >> p.x >> p.y;
        return is;
}

ostream & operator <<(ostream &os, Point &p) {
        os << p.x << ' ' << p.y;
        return os;
}

ll crossproduct(Point const &p1, Point const &p2) {
        return (ll)p1.x * p2.y - p2.x * p1.y;
}

int side(Point &p1, Point &p2, Point &p) {
        ll cross = crossproduct(p2 - p1, p - p1);
        if (cross == 0) return 0;
        if (cross > 0) return 1;
        else return -1;
}

double dis(Point &p1, Point &p2) {
        return sqrt(((double)p2.x - p1.x) * (p2.x - p1.x) +
                    ((double)p2.y - p1.y) * (p2.y - p1.y));
}

int main() {
        vector <Point> tri(3);
        for (auto &i : tri) cin >> i;
        Point p;
        cin >> p;
        double ans = 0;
        for (int i = 0; i < 3; i++) {
        if (side(tri[i], tri[(i + 1) % 3], p) *
            side(tri[i], tri[(i + 1) % 3], tri[(i + 2) % 3]) == -1)
                ans += dis(tri[i], tri[(i + 1) % 3]);
        }
```

```
57          cout << fixed << setprecision(15) << ans;
58      return 0;
59  }
```

Задача 2.3.5. (5 баллов)

Программист Искандер уже 14-е сутки подряд пишет программу, и вот, наконец-то, весь проект был готов. Однако, скомпилировав и запустив программу, результат выполнения оказался не таким, как ожидал Искандер...

Искандер — опытный программист, и для удобства он разделил программу на n исходных файлов, каждый из которых мог использовать некоторые из остальных исходных файлов (для простоты он пронумеровал файлы целыми числами от 1 до n). После завершения выполнения программы Искандер тут же осознал, в каких k исходных файлах допущены ошибки. Исправив всё, что нужно, теперь осталась только перекомпиляция, но в повторной компиляции нуждались не только исправленные исходные файлы, но и те файлы, которые непосредственно или через другие файлы используют какие-либо файлы из изменённых.

Искандер — ленивый программист, поэтому он хочет перекомпилировать как можно меньше исходных файлов. Помогите ему посчитать количество таких файлов.

Вспоминая, что Искандер – опытный программист, можно гарантированно сказать, что он не допустил ситуации, когда один из исходных файлов использует самого себя (непосредственно или через другие файлы).

Формат входных данных

В первой строке задано целое число n ($1 \leqslant n \leqslant 5000$).

Далее идут n строк. Каждая следующая i-я строка начинается с числа p – количество исходных файлов, которые используются в i-м файле ($0 \leqslant p \leqslant n - 1$). Затем следуют p различных целых чисел – номера исходных файлов, которые используются в i-м файле. Все числа положительные и не превосходят n. Суммарное количество p по всем строкам не превышает 10^5.

Затем задано целое число k ($1 \leqslant k \leqslant n$). Следующая строка содержит k различных целых чисел – номера файлов, которые исправил Искандер. Все числа положительные и не превосходят n.

Формат выходных данных

Выведите одно число — минимальное количество файлов, требующих перекомпиляции.

Примеры

Пример №1

Стандартный ввод
5 2 2 3 0 1 4 0 0 2 2 4

Стандартный вывод
4

Решение

Представим зависимости файлов между собой как ориентированный граф. Вершина с номером i будет соответствовать файлу с таким же номером. Если в текущем файле i используется файл j, то добавляем ребро из вершины с номером j в вершину с номером i. Теперь рассмотрим, какие из файлов надо перекомпелировать: запустим обход в глубину от каждой из k вершин, соответствующих изменённым файлам, и будем помечать все вершины на пути. Сделаем небольшую оптимизацию: не будем запускать обход в глубину из уже помеченных вершин. Таким образом, каждая вершина будет посещена не более 1 раза. Ответом на задачу является количество помеченных в графе вершин.

Пример программы

Ниже представлено решение на языке Python3

```
used = [];
v = [];

def dfs(x):
    used[x] = True
    for i in range(len(v[x])):
        to = int(v[x][i])
        if used[to] == False:
            dfs(to)

n = int(input())

v = [[] for i in range(n)]
used = [False] * n

for i in range(n) :
    p = input().split();
    for j in range(len(p)) :
        p[j] = int(p[j]) - 1
    for j in range(1, len(p)):
        v[p[j]].append(i);

k = int(input())

if k > 0:
    edit = input().split()
```

```
27
28      for i in range(k):
29          edit[i] = int(edit[i]) - 1
30
31      for i in range(k):
32          dfs(edit[i])
33
34  ans = 0;
35
36  for i in range(n):
37      if used[i] == True:
38          ans = ans + 1;
39
40  print(ans);
```

Вторая попытка

3.1. Задачи по математике (9 класс)

Задача 3.1.1. (2 балла)

С числом можно проделавать 2 типа операции: операция A умножает число на 2, операция B вычитает 3. Как из 11 при помощи этих операций получить 25 за наименьшее количество ходов? В ответ напишите последовательность операций (например последовательность операций AAABBBBBBBBBBBBBBBBBBBB за 24 хода достигает результата).

Решение

Обозначим M_n множество чисел, из которых можно получить 25 за n ходов. Тогда $M_1 = \{28\}$, $M_2 = \{14, 31\}$, $M_3 = \{7, 17, 34\}$, ... На 7-ом шаге встречаем 11.

Ответ: BBABAAB

Задача 3.1.2. (4 балла)

При каком наименьшем значении параметра k корни уравнения

$$x^2 + (k-1)x - k = 0$$

удовлетворяют условию $x_1^2 + x_2^2 = 5$.

Решение

$x_1^2 + x_2^2 = (x_1 + x_2)^2 - 2x_1x_2 = (k-1)^2 + 2k = 5$. Откуда $k = \pm 2$.

Ответ: -2

Задача 3.1.3. (3 балла)

В выпуклом пятиугольнике $ABCDE$ известно, что $AB = AE = 3$, $CD = 2$, $BC = 0,8$, $DE = 1,2$, $\angle ABC = \angle DEA = 90^\circ$. Найдите площадь этого пятиугольника.

Решение

Разделим пятиугольник на 3 треугольника ABC, ADE и ACD. Площади первых двух треугольников находяться по уже известным данным, поскольку треугольники прямоугольные, и известно две их стороны.

Стороные AC и AD треугольников прямоугльных треугольников ABC и ADE находятся также по уже известным данным. Зная все три стороны треугольника ACD, можно найти его площадь.

Останется только сложить площади всех трех треугольников.

Ответ: 6

Задача 3.1.4. (3 балла)

Натуральное число n называется «приятным», если оно удовлетворяет следующим условиям:

- состоит из 4 цифр;
- первая цифра равна третьей;
- вторая цифра равна четвертой;
- n^2 делится на произведение цифр n.

Найдите сумму всех «приятных» чисел.

Решение

$n = \overline{abab} = \overline{ab} \cdot 101$, $n^2 \vdots (a \cdot b \cdot a \cdot b) \Longrightarrow n \vdots ab$. н.о.д.$(101, ab) = 1 \Longrightarrow \overline{ab} \vdots ab$. Отсюда получаем, что $(10a + b)$ делится на a, т.е. $b \vdots a$. Небольшим перебором находим все значения $\overline{ab}$: 11, 12, 15, 24, 36. Сумма всех «приятных» чисел равна $(11 + 12 + 15 + 24 + 36) \cdot 101$.

Ответ: 9898

Задача 3.1.5. (4 балла)

Найдите количество решений в целых числах уравнения

$$\frac{1}{x}+\frac{1}{y}=\frac{1}{2017}.$$

Решение

Уравнение приводится к виду $y = 2017 + \frac{2017^2}{x-2017}$. Тогда $x - 2017$ является целым делителем 2017^2. Таких значений шесть: ± 1, ± 2017, $\pm 2017^2$. В каждом случае получаем пару целых решений.

Ответ: 6

3.2. Задачи по математике (10-11 класс)

Задача 3.2.1. (2 балла)

Найдите наименьшее число, которое можно представить в виде суммы квадратов натуральных чисел двумя различными способами.

Решение

$65 = 4^2 + 7^2 = 1^2 + 8^2$. Перебирая все суммы квадратов, меньших 64, выясняем минимальность.

Ответ: 65

Задача 3.2.2. (3 балла)

Найдите значение выражения

$$[2\sqrt{1\cdot 2\cdot 3+2\cdot 3\cdot 4+\ldots+2016\cdot 2017\cdot 2018}].$$

($[x]$ — целая часть числа x.)

Решение

$1\cdot 2\cdot 3+2\cdot 3\cdot 4+\ldots+2016\cdot 2017\cdot 2018 = \frac{1}{4}\cdot 2016\cdot 2017\cdot 2018\cdot 2019 = 4141845698256$. Остальное вычисляем на калькуляторе.

Ответ: 4070304

Задача 3.2.3. (3 балла)

На координатной плоскости отмечены точки $A(2;-1)$, $B(3;1)$ и $C(2017;4034)$. Найдите площадь треугольника ABC.

Решение

Пусть $O(0;0)$. Тогда $\overrightarrow{AB}\{1;2\} \uparrow\uparrow \overrightarrow{OC}\{2017;4034\}$. Поэтому площадь $\triangle ABC$ равна площади $\triangle OAB$.

Ответ: 2,5

Задача 3.2.4. (3 балла)

Обозначим $\tau(n)$ количество различных натуральных делителей числа n, а $\sigma(n)$ сумму различных натуральных делителей числа n. Найдите $\tau(\sigma(12)!)$.

Решение

$\sigma(12) = (1+2+4)\cdot(1+3) = 28$, $28! = 2^{25}\cdot 3^{13}\cdot 5^6\cdot 7^4\cdot 11^2\cdot 13^2\cdot 17\cdot 19\cdot 23$, $\tau(28!) = 26\cdot 14\cdot 7\cdot 5\cdot 3\cdot 3\cdot 2\cdot 2\cdot 2$.

Ответ: 917280

Задача 3.2.5. (4 балла)

Найдите все значения параметра p, при которых уравнение

$$27x^3 + (26-2p)x^2 + 16p^3 = 0$$

имеет три различных корня, образующих арифметическую прогрессию. В ответ напишите сумму полученных значений.

Решение

По теореме Виета для кубического многочлена:

$$\begin{cases} x_1+x_2+x_3 = -\dfrac{26-2p}{27} \\ x_1x_2+x_2x_3+x_3x_1 = 0 \\ x_1x_2x_3 = -\dfrac{16p^3}{27} \end{cases}$$

В эту систему добавляем условие арифметической прогрессии $x_1+x_3 = 2x_2$ и находим единственное решение.

Ответ: -0,5

3.3. Задачи по информатике

Задача 3.3.1. (1 балл)

Никита работает на автостоянке. В его обязанности входит запись номеров въезжающих машин. Это довольно скучное занятие и поэтому Никита решил оптимизировать этот процесс. Он хочет, чтобы компьютер обрабатывал изображение с камеры перед въездом в автостоянку и записывал номера. Никита уже написал софт, который обнаруживает на изображении последовательности из шести символов; осталось только проверять, является ли данная последовательность номером. Но из-за того, что Никита постоянно отвлекается на запись номеров, он просит вас о помощи.

Автомобильный номер – строка из шести символов. Первый символ – заглавная латинская буква, далее следует 3 цифры, и после — две заглавные латинские буквы. Например, строка "P142EQ" является номером. Вам будет дана строка, состоящая из шести символов, необходимо ответить, является ли строка автомобильным номером.

Формат входных данных

В единственной строке находится строка из шести символов, состоящая из цифр и заглавных латинских букв.

Формат выходных данных

Если строка является автомобильным номером, то необходимо вывести *Yes*, в ином случае — *No*.

Примеры

Пример №1

Стандартный ввод
K040LE
Стандартный вывод
Yes

Пример №2

Стандартный ввод
M3239L
Стандартный вывод
No

Решение

Для того, чтобы решить задачу необходимо проверить, что на первой и двух последних позициях строки находятся заглавные латинские буквы ($'A' \leqslant c \leqslant' Z'$); а между буквами находились три числа ($'0' \leqslant c \leqslant' 9'$).Для того, чтобы решить задачу необходимо проверить, что на первой и двух последних позициях строки находятся заглавные латинские буквы ($'A' \leqslant c \leqslant' Z'$).

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <string>

using namespace std;

bool is_dig(char c) {
        if ('0' <= c && c <= '9') {
                return true;
        }
        return false;
}

bool is_let(char c) {
        if ('A' <= c && c <= 'Z') {
                return true;
        }
        return false;
}

int main() {
        string s;
        cin >> s;
        if (is_let(s[0]) && is_dig(s[1]) && is_dig(s[2])
           && is_dig(s[3]) && is_let(s[4]) && is_let(s[5])) {
                cout << "Yes\n";
        } else {
                cout << "No\n";
        }
}
```

Задача 3.3.2. (1 балл)

Студент Денис часто прогуливает свои пары в университете. Сегодня, когда Денис пришел в деканат, он узнал, что нужно за каждую пропущенную пару написать объяснительную.

Денису выдали один бланк для написания объяснительной. Ему нужно сделать еще n копий данного бланка. Для этого в его распоряжении есть два ксерокса. Первый ксерокс тратит на копирование одного листа одну секунду, а второй — две секунды. Копию можно делать как с оригинала, так и с копии. Денис может использовать оба ксерокса одновременно. Определите, какое минимальное время необходимо Денису для получения n копий объяснительной.

Формат входных данных

В единственной строке дано число n — необходимое число копий ($1 \leqslant n \leqslant 1000$).

Формат выходных данных

Выведите одно число — минимальное время в секундах, необходимое для получения n копий.

Примеры

Пример №1

Стандартный ввод
2
Стандартный вывод
2

Пример №2

Стандартный ввод
5
Стандартный вывод
4t

Решение

Изначально нужно скопировать один экземпляр, чтобы одновременно копировать на обоих ксероксах. После этого для печати трех экземпляров необходимо 2 минуты. Получается, что для печати $n-1$ копий необходимо затратить $(n-1)//3*2$ минут в случае, если $n-1$ делится на 3. В случае остатка от деления на 3, равному k, необходимо дополнительно потратить k минут. Таким образом, итоговая формула выглядит, как $1+(n-1)//3*2+(n-1)\%3$.

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>

using namespace std;

int main() {
        int n;
        cin >> n;
        cout << 1 + (n - 1) / 3 * 2 + (n - 1) % 3<< "\n";
}
```

Задача 3.3.3. (3 балла)

Сегодня Ринат на уроке узнал про новую битовую операцию – XOR двух чисел. Напомним, что XOR или Исключающим ИЛИ, называется бинарная операция, которая применяется к каждой паре битов, стоящих на одинаковых позициях в двоичных представлениях чисел так, что, если биты равны, то в результате на этой позиции стоит 0, если же биты различны, то 1. Например, 3 XOR 5 = 6, потому что $3_{10} = 011_2$, $5_{10} = 101_2$, поэтому после применения операции второй и третий бит становятся равными 1, а первый бит – 0, таким образом получается $110_2 = 6_{10}$.

Учитель дал Ринату массив различных чисел и попросил написать массив, в котором все числа различны и нет ни одного числа из массива, который дал учитель. При этом, чтобы XOR всех чисел в массиве Рината был равен XOR всех чисел в массиве учителя.

Формат входных данных

В первой строке дано число n — количество элементов в массиве учителя ($1 \leqslant n \leqslant 10^5$).

Во второй сроке дано n чисел a_i — элементы массива ($0 \leqslant a_i \leqslant 2^{30}$).

Гарантируется, что XOR всех a_i не равен 0.

Формат выходных данных

Выведите элемента вашего массива, чтобы все числа в вашем массиве и массиве учителя отличались.

Все числа вашего массива b_i должны находиться в промежутке ($0 \leqslant b_i \leqslant 2^{30}$).

Количество элементов не должно превосходить 10^5.

Примеры

Пример №1

Стандартный ввод
1 27
Стандартный вывод
21 14

Пример №2

Стандартный ввод
2 5 6
Стандартный вывод
7 4

Пример №3

Стандартный ввод
3 1 2 4
Стандартный вывод
5 3 9 8

Пример №4

Стандартный ввод
3 3 5 1

Стандартный вывод
10 9 6 2

Решение

Стоит подумать о том, какая длина массива нам нужна. Давайте подумаем, можно ли это сделать с одним элементом ? Возьмем $x = \mathbf{XOR}\ a_i$. Тогда наш ответ и есть этот x, но он может быть в массиве учителя.

Можно ли это сделать из двух элементов ? Давайте переберем число i. Тогда вторым числом должно быть $i\ \mathbf{XOR}\ x$. Если число i и число $i\ \mathbf{XOR}\ x$ не были в массиве учителя, то мы нашли ответ.

Давайте подумает за сколько итераций завершится цикл, если просто перебирать i увеличивая на 1. А цикл завершится не больше, чем за $2 \cdot n + 1$ итераций. Потому что все числа разбиваются на пары i и $i\ \mathbf{XOR}\ x$. Всего n элементов могут покрыть не более n пар, поэтому $2 \cdot n$ чисел i могут не подойти.

Пример программы

Ниже представлено решение на языке C++

```cpp
#include <iostream>
#include <vector>
#include <set>


using namespace std;


set <int> s;

bool contains(int n) {
    return s.find(n) != s.end();
}

int main() {
        int n;
        cin >> n;
        int now = 0;
        vector <int> v(n);
        for (auto &i : v) {
            cin >> i;
            s.insert(i);
            now ^= i;
        }
        for (int i = 0; ; i++) {
            if (i != (i ^ now) && !contains(i) && !contains(i ^ now)) {
                cout << i << ' ' << (i ^ now) << '\n';
                return 0;
            }
        }
        return 0;
}
```

Задача 3.3.4. (5 баллов)

Азат уже очень давно отдыхает на островах. Сегодня он решил, что хочет перебраться на другой остров. Все острова пронумерованы целыми числами. Какие-то из островов соединены мостами. Но вот незадача, какие-то из островов ушли под воду и попасть на них никак нельзя. Азат сейчас находится на острове с номером v и хочет попасть на остров с номером u. Ваша задача проверить, сможет ли он это сделать.

Формат входных данных

В первой строке содержится 4 числа: n, m, v, и u ($1 \leqslant n \leqslant 1000, 0 \leqslant m \leqslant 5000, 1 \leqslant v, u \leqslant n$). Количество островов, количество мостов, номер острова, на котором находится Азат, и номер острова, на который он хочет попасть.

Следующие m строк описывают мосты. Каждая строка содержит 2 числа: a, b ($1 \leqslant a, b \leqslant n$). Мост между островами с номерами a и b соответственно. По мосту можно попасть с острова a на b, так и с острова b на a.

Следующая строка содержит одно число k ($0 \leqslant k \leqslant n$). Количество островов, которые затонули.

Следующая строка содержит k чисел c_i ($1 \leqslant c_i \leqslant n$) — номера затонувших островов. Гарантируется, что острова u и v не затоплены. Если k равно 0, то данная строка отсутствует.

Формат выходных данных

Если можно добраться от острова v до острова u, не посещая затопленные острова, то необходимо вывести *YES*, в ином случае — *NO*.

Примеры

Пример №1

Стандартный ввод
4 4 1 4 1 2 2 3 3 4 1 3 1 2

Стандартный вывод
YES

Пример №2

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

Стандартный вывод
NO

Решение

Давайте применим известный алгоритм поиска в глубину(DFS). Будем хранить массив $used[i]$, где будем стоять 1, если мы были на острове с номером i, или он затоплен, и 0, если мы там не были.

Изначально пометим все затопленные острова в массиве $used$ как 1. Потом запустим DFS от острова, в котором стоит Азат и будем пытаться попасть на остров, с которым мы связаны мостом, но мы не были на нем, и он не затоплен. Если таки образом мы сможем попасть на остров u, то ответ YES, иначе NO.

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <vector>

using namespace std;

int const MAX = (int)1e5;

bool used[MAX];

void dfs(int v, vector < vector <int> > &vv) {
        used[v] = true;
        for (auto i : vv[v]) {
                if (used[i]) continue;
                dfs(i, vv);
        }
}

int main() {
        int n, m, a, b;
        cin >> n >> m >> a >> b;
        --a; --b;
        vector < vector <int> > vv(n);
        for (int i = 0; i < m; i++) {
                int f, t;
                cin >> f >> t;
                --f; --t;
                vv[f].push_back(t);
```

```
28                  vv[t].push_back(f);
29          }
30          int k;
31          cin >> k;
32          for (int i = 0; i < k; i++) {
33                  int x;
34                  cin >> x;
35                  used[x - 1] = true;
36          }
37          dfs(a, vv);
38          used[b] ? cout << "YES\n" : cout << "NO\n";
39          return 0;
40  }
```

Задача 3.3.5. (5 баллов)

На прямой отмечено n точек. Нужно соединить некоторые точки отрезками одинаковой длины, чтобы можно было по отрезкам дойти от первой точки до последней, причём можно использовать не более k отрезков. Среди всех таких длин отрезков требуется вывести минимальную.

Формат входных данных

В первой строке заданы целые числа n и k ($2 \leqslant n \leqslant 2500$, $1 \leqslant k \leqslant n - 1$).

На следующей строке в возрастающем порядке заданы n различных неотрицательных целых чисел, каждое из которых не превосходит 100 000.

Формат выходных данных

Выведите одно целое число — ответ на задачу.

Примеры

Пример №1

Стандартный ввод
5 3 1 2 3 4 5
Стандартный вывод
2

Пример №2

Стандартный ввод
9 3 1 3 4 5 7 8 9 10 13
Стандартный вывод
4

Решение

Перебираем все возможные длины отрезка: отрезок длиной равной расстоянию от первой точки до второй, от первой точки до третьей и т.д. Для каждого такого отрезка пробуем соединить некоторые точки: начинаем с первой точки и проверяем, существует ли точка на позиции, равной сумме координаты текущей точки и длины рассматриваемого отрезка. Если такой точки не существует, то соединить точки отрезком текущей длины невозможно, иначе переходим к точке, существование которой мы проверяли. Если таким образом мы смогли дойти до последней точки, и при этом было использовано не более k отрезков, то текущая длина и является ответом.

Пример программы

Ниже представлено решение на языке Python3

```
A = input().split();
n = int(A[0]);
k = int(A[1]);

x = input().split();

for i in range(n) :
    x[i] = int(x[i]);

for i in range(1, n) :
    len = int(x[i]) - int(x[0]);
    cnt = 0;
    now = x[0];
    while (now != x[n - 1]) :
        now += len;
        if now not in x:
            cnt = -1;
            break;
        cnt = cnt + 1;
    if cnt != -1 and cnt <= k :
        print(len);
        break;
```

Третья попытка

4.1. Задачи по математике (9 класс)

Задача 4.1.1. (2 балла)

Имеется 4 попарно различных по весу камней и двухчашечные весы без гирь. За одно взвешивание можно положить по камню на каждую из чаш и узнать какой из

камней более тяжелый. За какое наименьшее число взвешиваний можно упорядочить камни по возрастанию?

Решение

Пусть имеются камни весами a, b, c, d. Если взвешивать все пары камней, то получится 6 взвешиваний. Надо предсказать результат одного взвешивания и нам не придется его совершать. Взвешиваем a с b и c с d. Пусть a и c окажутся более тяжелыми. Еще за одно взвешивание из них выберем более тяжелый. Тогда этот камень не надо взвешивать с более легким из другой пары камней (очевидно он тяжелей).

Ответ: 5

Задача 4.1.2. (2 балла)

Чему равен наибольший общий делитель пары чисел 7808809 и 7182391?

Решение

Используем алгоритм Евклида.

Ответ: 2677

Задача 4.1.3. (3 балла)

Три генератора тактовой частоты начинают вырабатывать импульсы одновременно. Интервалы между импульсами составляют 4/3 секунды, 5/3 секунды и 2 секунды соответственно. Совпавшие во времени импульсы накладываются (обрабатываются компьютером за один). Сколько тактов будет обработано за 1 минуту? (Первый импульс также считается.)

Решение

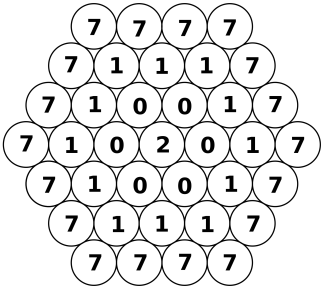
Первый генератор выработает 45 импульсов, второй — 36, третий — 30. Но мы посчитали совпадающие несколько раз: попарно совпадающие по два раза, все три совпадения по три раза. У первого и второго генератора совпадение импульсов каждые 20/3 секунды, у первого и третьего — каждые 4 секунды, у второго и третьего — каждые 10 секунд. Тогда попарное совпадение импульсов $9 + 15 + 6$ раз. В эту сумму тройное совпадение вошло 3 раза. Все три импульса совпадают $60 : \frac{60}{3} = 3$ раза. Общее количество импульсов за минуту равно:

$$45 + 36 + 30 - (9 + 15 + 6) + 3 = 84.$$

Ответ: 85

Задача 4.1.4. (4 балла)

Требуется выбрать 4 круга с цифрами 2, 0, 1 и 7 на рисунке ниже так, чтобы круг с цифрой 2 касался круга с 0, а круг с 0 касался круга с 1, круг с 1 касался круга с 7. Сколько способов это сделать?



Решение

Рассмотрим круги с 1. Если брать 1 из углового круга, то к нему от 2 ведет только 1 путь и от него можно пройти к 7 ровно 3 способами. Таких кругов с 1 ровно 6, поэтому всего путей через угловые 1 ровно 18. Если же брать круги с 1 на середине стороны шестиугольника, то к нему от 2 можно попасть через 2 нулика, а от него к 7 можно пройти тоже 2 способами. Таких путей всего $6 \cdot 2 \cdot 2 = 24$.

Ответ: 42

Задача 4.1.5. (4 балла)

Из одной точки окружности проведены две хорды длиной 9 и 17. Найдите радиус окружности, если расстояние между серединами хорд равно 5.

Решение

Расстояние между другими концами этих хорд равно 10 (по теореме о средней линии). Теперь надо выяснить радиус окружности, описанной вокруг треугольника со сторонами 9, 10, 17. Его можно выяснить по формулам $R = \dfrac{abc}{4S}$, и $S = \sqrt{p(p-a)(p-b)(p-c)}$.

Ответ: 10,625

4.2. Задачи по математике (10-11 класс)

Задача 4.2.1. (2 балла)

Имеется 5 попарно различных по весу камней и двухчашечные весы без гирь. За одно взвешивание можно положить по камню на каждую из чаш и узнать какой из камней более тяжелый. За какое наименьшее число взвешиваний можно упорядочить камни по возрастанию?

Решение

Пусть веса наших камней равны a, b, c, d, e. Первыми двумя взвешиваниями сравниваем пары (a, b) и (c, d). Не умаляя общности можно считать, чтобы $a > b$ и $c > d$. Третьим взвешиванием сравниваем пару (a, c). Не умаляя общности можно считать, что $a > c > d$ и $a > b$. За следующие 2 взвешивания вставляем камень e в цепочку $a > c > d$. А так как мы еще знаем, что $a > b$, то за последние 2 взвешивания вставляем b в нашу полученную цепочку.

Ответ: 7

Задача 4.2.2. (2 балла)

При каком наибольшем n на шахматной доске можно расставить n чёрных и n белых королей так, чтобы чёрные не били белых, а белые — чёрных?

Решение

Группа белых королей от группы черных королей должна быть отделена полосой пустых клеток. Если полоса в 8 пустых клеток, то в одной из частей не более 24 королей. Для полосы в 10 клеток пример строится несложно:

б	б	б	б		ч	ч	ч
б	б	б	б		ч	ч	ч
б	б	б	б		ч	ч	ч
б	б	б			ч	ч	ч
б	б	б			ч	ч	ч
б	б	б		ч	ч	ч	ч
б	б	б		ч	ч	ч	ч
б	б	б		ч	ч	ч	ч

Ответ: 27

Задача 4.2.3. (3 балла)

К числу 2017 припишите слева и справа по одной цифре так, чтобы полученное шестизначное число делилось на 31. Какие числа получились? В ответ напишите сумму таких чисел.

Решение

$\overline{a2017b} = 100000a + 20170 + b \equiv 25a + b - 11 \equiv 0 \pmod{31}$. Откуда $b \equiv 6a + 11 \pmod{31}$. Подставляя все цифры от 1 до 9 вместо a находим b. Нам подойдут $a = 4$, $b = 4$ и $a = 9$, $b = 3$.

Ответ: 1340347

Задача 4.2.4. (4 балла)

Рассматриваются всевозможные расстояния от точки $A(28; 13)$ до точек параболы $y = x^2$. Найдите среди них кратчайшее. В ответ запишите квадрат этого расстояния.

Решение

Пусть $M(x; x^2)$. Введем функцию $f(x) = AM^2 = (x - 28)^2 + (x^2 - 13)^2 = x^4 - 25x^2 - 56x + 28^2 + 13^2$. Требуется найти наименьшее значение $f(x)$. Она достигается в точках минимума. $f'(x) = 4x^3 - 50x - 56 = 2(x - 4)(2x^2 + 8x + 7)$ имеет единственную точку минимума $x = 4$. $f(4) = 576 + 9 = 585$.

Ответ: 585

Задача 4.2.5. (4 балла)

В окружности проведены хорды AC и BD, пересекающиеся в точке E, причем касательная к окружности, проходящая через точку A, параллельна BD. Известно, что $CD : ED = 4 : 3$, $S_{\triangle ABE} = 72$. Найдите площадь треугольника ABC.

Решение

Прямая, проходящая через A перпендикулярно касательной, проходит через центр окружности. Она же перпендикулярна хорде BD, следовательно, делит её пополам. Треугольник ABD равнобедренный. Используя теорему о вписанном угле выясняем справедливость следующих равенств:

$$\angle ACD = \angle ABD = \angle ADB = \angle ACB.$$

Из этих равенств очевидно следуют подобия треугольников:

$$\triangle ABC \sim \triangle AEB \sim \triangle DCE.$$

Откуда получаем $\dfrac{AB}{AE} = \dfrac{CD}{DE} = \dfrac{4}{3}$.

$$\frac{S_{\triangle ABC}}{S_{\triangle AEB}} = \left(\frac{AB}{AE}\right)^2 = \frac{16}{9}.$$

Ответ: 128

4.3. Задачи по информатике

Задача 4.3.1. (1 балл)

Юному Тимуру нравится рисовать четырехугольники с заданными длинами сторон. Длины для сторон он придумывает сам, и после этого начинает рисовать. Недавно на уроке математики он узнал, что не всегда из придуманных сторон можно получить четырехугольник. И теперь, чтобы не терять времени, Тимур хочет быть уверен, что по его задуманным длинам можно построить четырехугольник. За помощью в этом вопросе он обращается к вам.

Тимур даст вам длины четырех сторон. Необходимо определить, возможно ли построит из заданных сторон четырехугольник.

Формат входных данных

В единственной строке входных данных находится четыре целых положительных числа, разделенных пробелом. Каждое число не превосходит 10000.

Формат выходных данных

Если из заданных длин сторон можно построить четырехугольник, то необходимо вывести *Yes*, в ином случае — *No*.

Примеры

Пример №1

Стандартный ввод
`2 1 3 2`
Стандартный вывод
`Yes`

Пример №2

Стандартный ввод
`4 10 3 2`
Стандартный вывод
`No`

Решение

Необходимое и достаточное условие для построение четырехугольника с заданными сторонами – длина максимальной стороны должна быть строго меньше суммы трех остальных. Например, если a - максимальная сторона, то условие выглядит, как $a < b + c + d$. (b, c, d – остальные стороны четырехугольника).

Пример программы

Ниже представлено решение на языке C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
        vector <int> lines(4);
        for (int i = 0; i < 4; i++) {
                cin >> lines[i];
        }
        sort(lines.begin(), lines.end());
        if (lines[0] + lines[1] + lines[2] > lines[3]) {
                cout << "Yes\n";
        } else {
                cout << "No\n";
        }
}
```

Задача 4.3.2. (2 балла)

Из переписки двух друзей:

- Вася, привет! Не мог бы ты мне помочь?
- Привет, Настя! Да, конечно.
- Я готовлюсь к одной олимпиаде по программированию и решила немного подготовиться. Не мог бы ты дать мне какую-нибудь задачку?
- Без проблем! Вот условия: дана строка S_0. Каждая следующая строка получается добавлением в конец к предыдущей строке самого часто встречающего символа этой строки. Надо найти строку S_k. Ответов может быть несколько, поэтому надо вывести лексикографически минимальную строку из всех возможных.
- Хорошо, спасибо большое!

Помогите Насте решить задачу, которую ей дал Вася.

Формат входных данных

В первой строке заданы целые положительные числа n и k ($1 \leqslant n \leqslant 1000$, $1 \leqslant k \leqslant 1000$).

На следующей строке задана строка S_0, состоящая из n строчных букв латинского алфавита.

Формат выходных данных

Выведите лексикографически минимальную строку S_k.

Комментарии

Одна строка лексикографически меньше другой, если существует такая позиция m, что символ первой строки на позиции m меньше символа второй строки на той же позиции, а первые $m - 1$ символы двух строк совпадают.

Примеры

Пример №1

Стандартный ввод
5 1 abcdc
Стандартный вывод
abcdcc

Пример №2

Стандартный ввод
4 2 baab
Стандартный вывод
baabaa

Решение

Очевидно, что если мы в конец строки добавим самый часто встречающийся символ в нём, то и в следующих строках он также будет самым часто встречающимся. Поэтому ответом является исходная строка S_0, к которой в конец k раз приписан самый часто встречающийся символ в нём. Чтобы получить лексикографически минимальную строку в ответе, нужно среди самых часто встречающихся символов в S_0 выбрать тот символ, который раньше всего идёт в алфавите.

Пример программы

Ниже представлено решение на языке Python3

```
A = input().split();
n = int(A[0]);
k = int(A[1]);

s = input();

cnt = [0] * 26;

for i in range(n) :
    cnt[ord(s[i]) - 97] = cnt[ord(s[i]) - 97] + 1;

mx = 0;

```

```
for i in range(26) :
    if (cnt[i] > cnt[mx]) :
        mx = i;

for i in range(k) :
    s = s + chr(97 + mx);

print(s);
```

Задача 4.3.3. (3 балла)

Фермер по имени Артем сегодня решил озеленить свой сад. Для удобства представим, что сад Артема представляет собой декартову плоскость, где на целых координатах он посадил розы.

Чтобы цветок расцвел необходимо его поливать, поэтому он поставил две поливалки, которые поливают все розы в определенном радиусе.

После оказалось, что некоторые розы поливаются с обоих устройств. Артему стало интересно, сколько таких роз, но так как поле очень большое, то он попросил вас помочь ему.

Формат входных данных

Первая строка содержит три целых целых числа $x1$, $y1$, $r1$ ($-10^6 \leqslant x1, y1 \leqslant 10^6, 0 \leqslant r1 \leqslant 10^6$) — координаты центра первой поливалки и её радиус действия.

Во второй в аналогичном формате координаты и радиус второй поливалки.

Формат выходных данных

Выведите количество роз, которые поливаются с двух поливалок.

Примеры

Пример №1

Стандартный ввод
0 0 5 2 2 3
Стандартный вывод
26

Решение

Переберем одну из координат. Найдем целые координаты, которые принадлежат первой окружности на координате, которую мы перебираем. Обозначим их за l_1, r_1. Аналогично сделаем для второй окружности и обозначим за l_2, r_2. Прибавим к ответу количество целых координат в пересечении этих отрезков l_1, r_1 и l_2, r_2.

Чтобы найти координаты можно воспользоваться бинарным поиском или же решить уравнение окружности, где будет одна неизвестная, однако нужно будет хорошо округлить до целых координат. От этого и будет зависеть сложность вашего решения. O(N) — для уравнение и O(NlogN) — для бинпоиска, где N — разность между максимальной и минимальной координатой.

Пример программы

Ниже представлено решение на языке Python3

```
from math import sqrt

def calc(m, x0, x, r):
    return m * m <= r * r - (x - x0) * (x - x0)


def get_y(x0, y0, r, x):
    _l = -1
    _r = r + 2
    while _r - _l > 1:
        m = (_l + _r) // 2
        if m * m <= r * r - (x - x0) * (x - x0):
            _l = m
        else:
            _r = m
    return _l

x1, y1, r1 = map(int, input().split())
x2, y2, r2 = map(int, input().split())
ans = 0
for x in range(x1-r1, x1+r1+1):
    if x > x2 + r2 or x < x2 - r2:
        continue

    y1l = int(y1 - sqrt(r1 ** 2 - (x1 - x) ** 2)) - 10
    while ((y1l - y1) ** 2 + (x1 - x) ** 2) > r1 ** 2:
        y1l += 1

    y1h = int(y1 + sqrt(r1 ** 2 - (x1 - x) ** 2)) + 10
    while ((y1h - y1) ** 2 + (x1 - x) ** 2) > r1 ** 2:
        y1h -= 1

    y2l = int(y2 - sqrt(r2 ** 2 - (x2 - x) ** 2)) - 10
    while ((y2l - y2) ** 2 + (x2 - x) ** 2) > r2 ** 2:
        y2l += 1

    y2h = int(y2 + sqrt(r2 ** 2 - (x2 - x) ** 2)) + 10
    while ((y2h - y2) ** 2 + (x2 - x) ** 2) > r2 ** 2:
        y2h -= 1

    if not (y1h < y2l or y2h < y1l):
        # print((y1l, y1h), (y2l, y2h))
        # print(x, abs(max(y1l, y2l) - min(y1h, y2h)) + 1)
        ans += abs(max(y1l, y2l) - min(y1h, y2h)) + 1

print(ans)
```

Задача 4.3.4. (4 балла)

Гена очень послушный и умный маленький мальчик. Как и все дети он любит играть в кубики, но часто ему в голову приходят сложные математические формулы или непростые задачи. Вот и в этот раз, когда он раскладывал кубики, то придумал очень интересную задачу.

Он начал выкладывать кубики в ряд длинной n. Причем высота на i-ом месте ряда составляет a_i кубиков. Тут он и задался вопросом: какое минимальное количество кубиков надо переложить, чтобы все высоты стали равны. Кубики можно перекладывать только на соседние места, то есть, если мы взяли кубик с ряда номер j, то можно положить его на ряд номер $j+1$ или $j-1$, если такие существуют.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 10^5$) — количетсво рядов из кубиков, которые выложил Гена.

Во второй строке дано n чисел $a_i (0 \leqslant a_i \leqslant 10^6)$ — количество кубиков в i-м номере ряда.

Формат выходных данных

Выведите минимально количество перекладываний, которое необходимо сделать Гене, чтобы все высоты стали равны, если Гена не сможет выстроить такой ряд из кубиков, выведите -1.

Примеры

Пример №1

Стандартный ввод
4 2 1 2 3
Стандартный вывод
2

Пример №2

Стандартный ввод
5 2 3 4 5 6
Стандартный вывод
10

Решение

Посчитаем количество кубиков в каждом ряду: $h = \frac{1}{n} \cdot \sum_{i=1}^{n} a_i$. Пусть $df = 0$ — количество лишних кубиков на префиксе(отрицательное число, если кубиков не хватает). Теперь рассматриваем каждый ряд от 1 до n, $df = df + (a_i - h)$ — новая разница

кубиков. В каждой итерации к ответу прибавляется $|df|$ — если у нас переизбыток кубиков, в любом случае они перейдут дальше, иначе кубики с конца должны будут перейти в начало.

Пример программы

Ниже представлено решение на языке C++

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    int n;
    long long sum = 0;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        sum += a[i];
    }
    if (sum % n != 0) {
        cout << -1 << '\n';
        return 0;
    }
    long long k = sum / n;
    long long ans = 0;
    int j = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] >= k) {
            continue;
        }
        while (a[i] != k) {
            while (j < n && a[j] <= k) {
                j++;
            }
            int t = min(k - a[i], a[j] - k);
            a[j] -= t;
            a[i] += t;
            ans += t * abs(i - j);
        }
    }
    cout << ans << '\n';
}
```

Задача 4.3.5. (5 баллов)

В крупной компании по созданию программного обеспечения находится n серверов. Они соединяются сетью с помощью m проводов. Провод соединяет два сервера между собой.

Сервера A и B находятся в одной локальной сети, если сигнал от сервера A может по рабочим проводам дойти до сервера B, возможно проходя при этом через промежуточные сервера. Если сервер может соединиться только с собой, то считается, что он сам по себе представляет локальную сеть.

В датацентре компании появились грызуны, которые начали перегрызать провода. Пока ваш напарник поехал за отпугивателями грызунов, вам поручили посчитать полученный ущерб компании. Вам нужно ответить, сколько всего локальных сетей в компании возникало после выведения каждого провода из строя.

Формат входных данных

В первой строке вводится целое число n ($2 \leqslant n \leqslant 3 \cdot 10^5$) - количество серверов в компании.

Во второй строке вводится целое число m ($1 \leqslant m \leqslant 3 \cdot 10^5$) - количество проводов.

В следующих m строках вводятся пары различных чисел a, b ($1 \leqslant a, b \leqslant n$) номера серверов, которые соединяет i-ый провод.

В следующей строке вводится число q ($1 \leqslant q \leqslant m$) количество оборванных проводов.

В следующей строке вводится q различных чисел – номера оборванных кабелей. Все номера различны и идут в хронологическом порядке.

Формат выходных данных

Выведите q чисел, количество различных локальных сетей после выведения из строя следующего провода.

Комментарии

Первый пример: после удаления первого провода все компьютеры все еще находятся в одной сети. После удаления второго провода, сеть разбивается на две части: компьютеры 1,3 и компьютер 2.

Примеры

Пример №1

Стандартный ввод
3 3 1 2 2 3 1 3 2 1 2

Стандартный вывод
1 2

Пример №2

Стандартный ввод
4 3 1 2 1 4 4 2 1 3
Стандартный вывод
2

Решение

Давайте решать обратную задачу: провода не обрывают, а чинят(идем по запросам в обратном порядке). Так как запросов порядка 10^5, надо уметь быстро объединять две локальные сети. Для этого можно использовать структуру данных система непересекающихся множеств, которая позволяет объединить два множество за время $O(\log n)$. Пусть изначально ни один компьютер не соединен с другим, то есть всего n локальных сетей. Процесс соединения двух компьютеров: если два компьютера лежат в разных локальных сетях(разные множества), то количество сетей уменьшится и мы объединим два множества, иначе оно останется таким же.

Возьмем все целые провода(которые не подвергались хакерской атаке) и объединим ими наши компьютеры. Количество локальных сетей после объединения будет равно ответу после последней хакерской атаки. Далее идем в обратном порядке по атакам и объединяем сети.

Пример программы

Ниже представлено решение на языке C++

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

using namespace std;
typedef  long long ll;
typedef unsigned long long ull;
int inf_int=1e8;
ll inf_ll=1e18;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
const double pi=3.1415926535898;

const int MAXN=3e5+10;

int parent[MAXN];
int rank[MAXN];
int find_parent(int v) {
    if(v==parent[v]) {
        return v;
    }
    return parent[v]=find_parent(parent[v]);
}

bool union_set(int a,int b) {
```

```
    a=find_parent(a);
    b=find_parent(b);
    if(a!=b) {
        if(rank[a]<rank[b])
            swap(a,b);
        else if(rank[a]==rank[b])
            ++rank[a];
        parent[b]=a;
        return true;
    }
    return false;
}
int x[MAXN],y[MAXN];
char used[MAXN];
int a[MAXN];
int ans[MAXN];
void solve()
{
    int n,m,q;
    scanf("%d",&n);
    scanf("%d",&m);
    for(int i=1;i<=m;++i) {
        scanf("%d %d",&x[i],&y[i]);
    }
    scanf("%d",&q);
    for(int i=1;i<=q;++i) {
        scanf("%d",&a[i]);
        used[a[i]]=1;
    }
    for(int i=1;i<=n;++i)
        parent[i]=i;

    int cur=n;
    for(int i=1;i<=m;++i) {
        if(!used[i]) {
            cur-=union_set(x[i],y[i]);
        }
    }
    for(int i=q;i>=1;--i) {
        ans[i]=cur;
        cur-=union_set(x[a[i]],y[a[i]]);
    }
    for(int i=1;i<=q;++i) {
        printf("%d ",ans[i]);
    }


}

int main()
{
        int t=1;
        while(t--)
           solve();
        return 0;
}
```

Критерии отбора победителей и призеров первого этапа

Количество баллов, набранных при решении задач одной попытки, суммируется. Если участник решал задачи, как в первой, так и во второй попытке, то выбирается попытка с большей суммой баллов. Призерам первого отборочного этапа необходимо было набрать 16 баллов (для 9 класса) и 16 баллов (для 10-11 класса). Победители первого отборочного этапа должны были набрать 29 баллов.

2. ВТОРОЙ ОТБОРОЧНЫЙ ЭТАП

Второй отборочный этап проводится в командном формате в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Продолжительность второго этапа составляет 37 дней. Задачи условно разделены на задачи по математике и информатике, но носят междисциплинарный характер и в более простой форме воссоздают инженерную задачу заключительного этапа. Для каждой из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике общие — для всех участников. 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. В данном этапе можно получить суммарно от 0 до 68 баллов.

Все условия задач по математике доступны участникам с первого дня второго отборочного этапа. Задачи по программированию выкладывались двумя партиями: в начале второго этапа и через три недели после начала. Команды могут выполнять задачи в любом порядке. Задачи допускают неограниченное число попыток сдать решение.

Задачи второго этапа

6.1. Задачи по математике (9 класс)

Задача 6.1.1. Многочлены (7 баллов)

Многочленом с одной переменной называется выражение вида

$$G(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \qquad (a_n \neq 0)$$

Числа $a_0, a_1, \ldots, a_n$ – это коэффициенты многочлена, a_n – старший коэффициент, a_0 – свободный член.

Степенью многочлена называют наибольшую степень переменной, входящую в многочлен и обозначают $\deg G(x)$.

Два многочлена называются **равными**, если равны все их коэффициенты. Многочлен равен нулю, если все его коэффициенты равны нулю. Степень нулевого многочлена не определена.

Число α является **корнем** многочлена $G(x)$, если $G(\alpha) = 0$.

Основная теорема арифметики для многочленов: для любых двух многочленов $F(x)$ и $G(x)$ существует единственная пара многочленов $P(x)$ (частное) и $Q(x)$ (остаток) такая, что $F(x) = G(x) \cdot P(x) + Q(x)$, причём степень остатка $Q(x)$ меньше степени делителя $G(x)$, или $Q(x)$ нулевой многочлен.

Теорема Безу: остаток от деления многочлена $F(x)$ на $(x - \alpha)$ равен $F(\alpha)$.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 — неверному. (*Например 01111001*).

а) Чтобы однозначно задать многочлен степени n, достаточно знать значения этого многочлена в n различных точках числовой прямой.

б) Если $\deg G(x) = n$ и $\deg F(x) = m$, то $\deg F(G(x)) = m \cdot n$.

в) Многочлен $x^{10} - x^5 + 1$ делится на многочлен $2x^2 - 2x + 2$ без остатка.

г) Если всякий корень многочлена $G(x)$ является корнем многочлена $F(x)$, то многочлен $F(x)$ делится на многочлен $G(x)$.

д) У многочлена $F(x)$ сумма коэффициентов при четных степенях равна сумме коэффициентов при нечетных степенях. Тогда таким же свойством обладают все многочлены $F(x) \cdot G(x)$ при любых значениях коэффициентов $G(x)$.

е) Известно, что все коэффициенты произведения многочленов $F(x)$ и $G(x)$ с целыми коэффициентами делятся на простое число p. Тогда все коэффициенты $F(x)$ или $G(x)$ делятся на p.

ж) Известно, что многочлен $F(x)$ принимает целые значения для всех целых x. Тогда все коэффициенты $F(x)$ — целые.

з) Многочлен $G(x)$ третьей степени принимает значения $G(-1) = 22$, $G(0) = 14$, $G(1) = -4$, $G(2) = 4$. Тогда обязательно $G(3) = 74$.

Решение

Объясним только пункты, утверждения в которых неверны. а) неверно, т.к. многочлен 1 степени уже по одной точке не восстановить (существуют бесконечное количество прямых, проходящих через заданную точку на плоскости). Пункт г) $G(x) = x^2$, $F(x) = x^2 - x$. $x^2 - x \neq x^2 \cdot P(x)$. Пункт ж) многочлен $\frac{1}{2}x(x+1)$ принимает целые значения при целых x, но коэффициенты не целые.

Ответ: 01101101

Задача 6.1.2. Кривые на плоскости (7 баллов)

Все точки плоскости, координаты $(x; y)$, которых являются решениями уравнения $f(x; y) = 0$ называются графиком уравнения $f(x; y) = 0$.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 — неверному. (*Например 01111001*).

а) Уравнение $x^2 + y^2 = 2x + 2y - 1$ задает общие точки прямой и окружности.

б) Уравнение $y = \sqrt{6 - x - x^2}$ задает на плоскости одну ветвь параболы.

в) Площадь фигуры, заданной неравенством $|2x-3y|+|3x+2y| \leqslant 13$, равна 26.

г) Периметр фигуры, заданной уравнением $|x+15y-88|+|9x+9y-99|=63$, равен 38.

д) Если точка с координатами $(x;y)$ движется на плоскости по правилу:

$$\begin{cases} x = 13\cos t; \\ y = 31\sin t, \end{cases}$$

то траектория ее движения описывает эллипс.

е) Графиком уравнения $\sqrt{x^2+y^2+6x+8y+25}+\sqrt{x^2+y^2-4x-16y+68}=13$ является отрезок.

ж) Графиком уравнения $|4x-3y+5|-|2x+3y-5|=10$ является параллелограмм.

з) Графиком уравнения $ax^2+bxy+cy^2=1$ при $a,c \neq 0$ и $b^2-4ac>0$ является гипербола.

Решение

Верные утверждения:

а) Уравнение $x^2+y^2=2x+2y-1$ задает окружность.

б) Уравнение $y=\sqrt{6-x-x^2}$ задает на плоскости полуокружность.

в) Площадь фигуры, заданной неравенством $|2x-3y|+|3x+2y| \leqslant 13$, равна 26.

г) Периметр фигуры, заданной уравнением $|x+15y-88|+|9x+9y-99|=63$, равен 36.

д) Если точка с координатами $(x;y)$ движется на плоскости по правилу:

$$\begin{cases} x = 13\cos t; \\ y = 31\sin t, \end{cases}$$

то траектория ее движения описывает эллипс.

е) Графиком уравнения $\sqrt{x^2+y^2+6x+8y+25}+\sqrt{x^2+y^2-4x-16y+68}=13$ является отрезок.

ж) Графиком уравнения $|4x-3y+5|-|2x+3y-5|=10$ является пара непересекающихся углов с соответственно параллельными сторонами.

з) Графиком уравнения $ax^2+bxy+cy^2=1$ при $a,c \neq 0$ и $b^2-4ac>0$ является гипербола.

Ответ: 00101101

Задача 6.1.3. ГМТ (6 баллов)

Геометрическим местом точек, обладающих некоторым свойством называется множество всех точек (плоскости), обладающих этим свойством.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 — неверному. (*Например 01111001*).

а) Даны точки A и B. ГМТ X таких, что $AX^2 + BX^2 = r^2$, где $r > AB$ является окружностью.

б) Даны точки A и B. ГМТ X таких, что $AX^2 - BX^2 = r^2$, где $r > 0$ является окружностью.

в) Даны точки A и B. ГМТ X таких, что $AX + BX = AB$, является прямой AB.

г) Существует единственная точка, равноудаленная от трех пересекающихся прямых.

д) Дан треугольник ABC. Внутри него взяли точку M и соединили ее с вершинами. Получилось три треугольника. ГМТ M, для которых сумма площадей двух из этих треугольников будет равна площади третьего, — средние линии треугольника.

е) Даны точки A и B. ГМТ X таких, что $\angle AXB = 36^\circ$, является дугой окружности.

ж) Даны точки A и B. Множество всех точек C, таких, что ABC — равнобедренный треугольник, является прямой, перпендикулярной к AB.

з) На плоскости даны окружности S_1 и S_2. ГМТ центров окружностей, перпендикулярных S_1 и S_2 является радикальной осью этих окружностей.

Решение

Верные утверждения:

а) Не ограничивая общности введем систему координат так, чтобы точки A и B имели координаты $(-1; 0)$ и $(1; 0)$ соответственно. Пусть $X(x; y)$. Тогда уравнение $AX^2+BX^2 = r^2$, где $r > 2$, примет вид $(x+1)^2+y^2+(x-1)^2+y^2 = r^2$. Которое в свою очередь преобразуется к виду $x^2 + y^2 = \frac{r^2-2}{2}$.

б) В обозначениях пункта а) уравнение $AX^2 - BX^2 = r^2$ преобразуется к виду $4x = r^2$, что задает вертикальную прямую.

в) ГМТ X таких, что $AX + BX = AB$, является отрезок AB. Для остальных точек плоскости $AX + BX > AB$.

г) Существуют четыре точки, равноудаленных от трех пересекающихся прямых. Если взять треугольник, образованный этими прямыми, то искомые точки находятся в центрах вписанной и вневписанных окружностей этого треугольника.

д) Пусть $S_{AMB} = S_{AMC} + S_{BMC}$. Продолжим CM до пересечения с AB в точке D. Тогда $\dfrac{CM}{MD} = \dfrac{S_{AMC}}{S_{AMD}} = \dfrac{S_{BMC}}{S_{BMD}} = \dfrac{S_{AMC} + S_{BMC}}{S_{AMD} + S_{BMD}} = 1$.

е) ГМТ X таких, что $\angle AXB = 36^\circ$, является объединением двух дуг окружности.

ж) Даны точки A и B. Множество всех точек C, таких, что ABC — равнобедренный треугольник, является объединением серединного перпендикуляра к AB и двух окружностей с центрами A и B и радиусом AB.

з) На плоскости даны окружности S_1 и S_2. ГМТ центров окружностей, перпендикулярных S_1 и S_2 является радикальной осью этих окружностей.

Ответ: 10001001

6.2. Задачи по математике (10-11 класс)

Задача 6.2.1. Интерполяция (7 баллов)

Задача состоит в том, чтобы неизвестную функцию, у которой известны значения в нескольких точках, приблизить полиномами.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 — неверному. (*Например 01111001*).

а) Даны $n+1$ точек $(x_i; y_i)$, $i = \overline{0,n}$, причем все x_i различны. Тогда существует единственный многочлен $P_n(x)$ степени n такой, что $P(x_i) = y_i$.

б) Интерполяционный многочлен второй степени, построенный по узлам $x_0 = 0, x_1 = 1, x_2 = 2$ для функции $f(x) = x^3$ в точке 3 дает погрешность в 6 единиц.

в) Определим функции

$$L_{n,k}(x) = \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}.$$

Тогда $L_{n,k}(x_k) = 1$ и $L_{n,k}(x_i) = 0$ при $i \neq k$.

г) Многочлен $P(x)$ 4 степени проходит через точки $(1;2), (2;1), (3;5), (4;6), (5;1)$. Тогда $P(0) = 21$.

д) Заданы центры $x_0 = 1$, $x_1 = 3$, $x_2 = 4$, $x_3 = 4,5$ и коэффициенты $a_0 = 5$, $a_1 = -2$, $a_2 = 0,5$, $a_3 = -0,1$, $a_4 = 0,003$. По ним вычислили интерполяционный многочлен Ньютона $P_4(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \dots + a_4(x-x_0)(x-x_1)(x-x_2)(x-x_3)$. Тогда $P_4(2,5) = 1,5$.

е) Пусть $T_n(x) = \cos(n \arccos(x))$ для $x \in [-1;1]$. Тогда при всех $n \in \mathbb{N}$ выполнятся соотношение:

$$T_{n+2}(x) = 2x^2 T_{n+1}(x) - T_n(x).$$

ж) Функция $T_n(x)$, определенная в предыдущем пункте, является четной функцией.

з) Функция $T_n(x)$ является многочленом степени n и имеет n различных корней на промежутке $[-1;1]$.

Решение

Объясним только пункты, утверждения в которых неверны.

а) Неверно, т.к. степень многочлена $P_n(x)$ может быть ниже n. Например, если все y_i равны между собой, то степень равна нулю.

д) $P_4(2,5) = 1,51$.

е) Верное соотношение:

$$T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x).$$

ж) Функция $T_n(x)$ (см. многочлены Чебышева), является четной функцией при четных n и нечетной при нечетных n.

Ответ: 01110001

Задача 6.2.2. Параметризованная кривая (7 баллов)

Отображение $r(t) = (x(t); y(t))$, которое каждому значению t из интервала числовой прямой ставит в соответствие точку плоскости (или пространства) называется **параметризованной кривой**. Обычно параметр t определяет время, а точка $M(x(t); y(t))$ — материальную точку, которая движется на плоскости. Её вектор скорости в каждый момент времени находится по формуле $\vec{v}(t) = (x'(t); y'(t))$. Для определенности, время будем считать в секундах, а расстояние в метрах.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 — неверному. (*Например 01111001*).

а) Если точка с координатами $M(x; y)$ движется на плоскости по правилу:

$$\begin{cases} x = 3\cos t; \\ y = 3\sin t, \end{cases}$$

то траектория ее движения описывает окружность с радиусом 3.

б) Если точка с координатами $M(x; y)$ движется на плоскости по правилу:

$$\begin{cases} x = 2^t + 2^{-t}; \\ y = 2^t - 2^{-t}, \end{cases}$$

то эта точка движется по ветви гиперболы.

в) Скорость тела, движущегося на плоскости по закону

$$\begin{cases} x = t^3 - 3t; \\ y = t^2, \end{cases}$$

в момент времени $t = 2$ равна $\sqrt{97}$ (м/с).

г) Прямая $5x - 3y + 11 = 0$ касается кривой

$$\begin{cases} x = t^2 - t; \\ y = t^2 + t, \end{cases}$$

в точке $t = 2$.

д) Длина кривой $(x(t); y(t))$ в промежутке $a \leqslant t \leqslant b$ вычисляется по формуле:

$$\int_a^b \sqrt{(x'(t))^2 + (y'(t))^2}dt.$$

е) Площадь области, ограниченной кривой $(x(t); y(t))$, которую параметр $t \in [a; b]$ обходит против часовой стрелки, вычисляется по формуле

$$S = \int_a^b y(t)x'(t)dt.$$

ж) Тело движется на плоскости по правилу (единица измерения расстояния в метрах):

$$\begin{cases} x = 3\cos^3 \pi t; \\ y = 3\sin^3 \pi t. \end{cases}$$

Тогда каждые 2 секунды оно проходит по 18 метров.

з) Площадь области, ограниченной кривой

$$\begin{cases} x = t^3 - 4t; \\ y = t^2; \\ -2 \leqslant t \leqslant 2, \end{cases}$$

равна 15.

Решение

Объясним только пункты, утверждения в которых неверны. г) уравнение касательной в точке $t = 2$ имеет вид $5x - 3y + 8 = 0$. е) знак не правильный, эта формула считает площадь фигуры, которую параметр t обходит по часовой стрелке. з) $\int_{-2}^{2} t^2(3t^2 - 4)dt = \dfrac{256}{15}$.

Ответ: 11101010

Задача 6.2.3. ГМТ (6 баллов)

Геометрическим местом точек, обладающих некоторым свойством называется множество всех точек, обладающих этим свойством. (Например, ГМТ равноудаленных от данной точки называется сферой).

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 — неверному. (*Например 01111001*).

а) Даны точки A и B. ГМТ X пространства таких, что $AX^2 + BX^2 = r^2$, где $r > AB$ является сферой.

б) Даны точки A и B. ГМТ X пространства таких, что $AX^2 - BX^2 = r^2$, где $r > 0$ является прямой.

в) Даны точки A и B. ГМТ X пространства таких, что $AX+BX = AB$, является прямой AB.

г) Существует единственная точка плоскости, равноудаленная от трех пересекающихся прямых в этой плоскости.

д) В пространстве проведены скрещивающиеся перпендикулярные прямые на расстоянии h друг от друга. Рассматриваются всевозможные отрезки длины d, один конец у которых на первой прямой, а второй — на другой. Тогда середины всех таких отрезков образуют окружность для всех $d > h$.

е) Даны точки A и B. ГМТ X таких, что $\angle AXB = 36^\circ$, является дугой окружности.

ж) На плоскости дан квадрат со стороной 1. Тогда объём тела, состоящего из ГМТ (пространства) на расстоянии 1 от этого квадрата равно $\frac{4\pi}{3} + 3$.

з) На плоскости даны окружности S_1 и S_2. ГМТ центров окружностей плоскости, перпендикулярных S_1 и S_2 является радикальной осью этих окружностей.

Решение

Верные утверждения:

а) Не ограничивая общности введем систему координат так, чтобы точки A и B имели координаты $(-1;0;0)$ и $(1;0;0)$ соответственно. Пусть $X(x;y;z)$. Тогда уравнение $AX^2 + BX^2 = r^2$, где $r > 2$, примет вид $(x+1)^2 + y^2 + z^2 + (x-1)^2 + y^2 + z^2 = r^2$. Которое в свою очередь преобразуется к виду $x^2 + y^2 + z^2 = \frac{r^2-2}{2}$, что является уравнением сферы.

б) В обозначениях пункта а) уравнение $AX^2 - BX^2 = r^2$ в координатах приводится к виду $x = \dfrac{r^2}{4}$, которое задает плоскость в пространстве.

в) ГМТ X пространства таких, что $AX + BX = AB$, является **отрезком** AB. Для точек вне отрезка AB выполняется $AX + BX > AB$.

г) Существуют четыре точки, равноудаленных от трех пересекающихся прямых. Если взять треугольник, образованный этими прямыми, то искомые точки находятся в центрах вписанной и вневписанных окружностей этого треугольника.

д) В пространстве проведены скрещивающиеся перпендикулярные прямые на расстоянии h друг от друга. Рассматриваются всевозможные отрезки длины d, один конец у которых на первой прямой, а второй — на другой. Тогда середины всех таких отрезков образуют окружность для всех $d > h$.

е) ГМТ X таких, что $\angle AXB = 36^\circ$, является поверхностью вращения дуги окружности вокруг отрезка AB.

ж) На плоскости дан квадрат со стороной 1. Тогда объём тела, состоящего из ГМТ (пространства) на расстоянии 1 от этого квадрата равно $\frac{4\pi}{3} + 2\pi + 2$.

з) На плоскости даны окружности S_1 и S_2. ГМТ центров окружностей плоскости, перпендикулярных S_1 и S_2 является радикальной осью этих окружностей.

Ответ: 10001001

6.3. Задачи по информатике

Задача 6.3.1. Одометрия (3 балла)

Робот, собранный по дифференциальной схеме, перемещается вдоль чёрной линии. Перемещение робота задаётся парами линейных ν и угловых ω скоростей. Пары скоростей передаются через равные промежутки времени t с. Значения скоростей не изменяются между замерами, они могут измениться мгновенно в момент проведения

измерения.

Робот начинает своё движение в точке $(0, 0)$ и имеет направление движения сонаправленное с направлением оси X. Необходимо определить расстояние от точки, где черная линия пересекает саму себя, до начала координат. В случае если пересечений несколько, то необходимо найти координату первого из них.

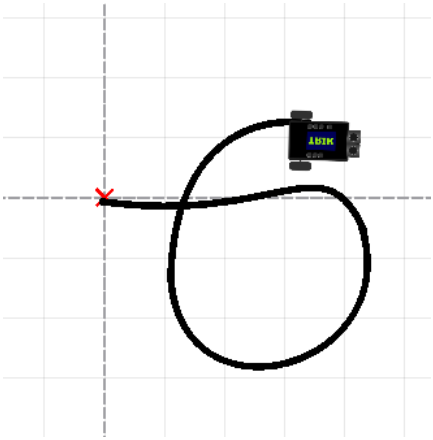


Рис. 6.1: Пример поля для тестирования робота

Формат входных данных

Первая строка входной файла содержит два числа N - количество замеров угловой и линейной скоростей ($0 \leqslant N \leqslant 10000$), и t - время между каждой парой замеров ($0 < t \leqslant 1$). N — целое число, t — вещественное число.

Далее идет N строк, в которых указана пара вещественных чисел ν мм/с и ω рад/с — линейная ($-1000 \leqslant \nu \leqslant 1000$) и угловая ($-100 \leqslant \omega \leqslant 100$) скорости через пробел .

Формат выходных данных

Расстояние в миллиметрах от начала координат до точки пересечения черной линии с самой собой. Допускается погрешность в 50 мм.

Примеры

Пример №1

Стандартный вход

```
33 0,5
133,902 0
137,323 -0,02
```

```
136,834 0
137,323 0,14
136,834 0,01
136,834 0,07
137,323 0,06
137,323 0,05
136,834 0
136,834 -0,2
137,323 -0,56
137,323 -0,66
136,834 -0,67
136,834 -0,65
137,323 -0,65
136,834 -0,57
137,323 -0,59
137,323 -0,51
136,834 -0,55
137,323 -0,58
136,834 -0,66
137,323 -0,69
136,834 -0,67
136,834 -0,56
137,323 -0,44
136,834 -0,35
137,323 -0,37
137,323 -0,34
136,834 -0,41
136,834 -0,47
137,323 -0,45
136,834 -0,45
137,323 -0,63
```

Стандартный выход

```
430
```

Решение

Рассмотрим движения робота между моментами изменения скоростей. В этот момент линейная и угловая скорость постоянны и мы можем рассматривать движение робота, как показано на рисунке (в начальный момент времени робот направлен вдоль оси X)

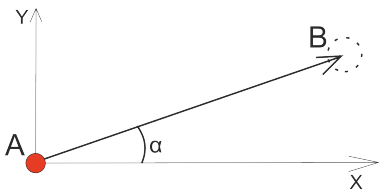


Рис. 6.2: Движение между измерениями скоростей из точки старта

Угол α мы можем вычислитть по формуле $\alpha = \omega t$. Длину отрезка AB можно высчитать по формул $l = \nu t$.

Далее мы узнаём координаты точки B по формуле $X_B = l\cos(\alpha)$ $Y_B = l\sin(\alpha)$

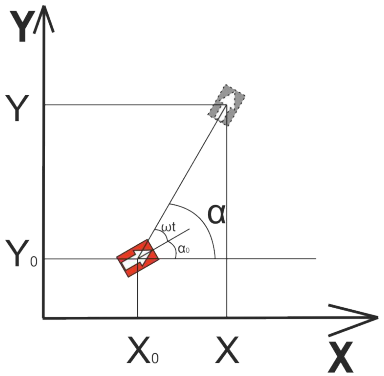


Рис. 6.3: Движение между измерениями скоростей

Теперь рассмотрим случай когда α_0 (начальный угол робота) $\neq 0$, то есть робот уже находится под каким то углов к оси X. В этом случае угол α будет вычисляться по формуле

$\alpha = \alpha_0 + \omega t$.

Дальше рассмотрим случай, когда наш робот уже находится в точке $(X_0; Y_0)$ при этом формула нахождения координат приобретёт вид

$X = X_0 + l\cos(\alpha)$

$Y = Y_0 + l\sin(\alpha)$

Используя данные формулы мы можем построить массив из отрезков

$(X_0; Y_0), (X_1; Y_1), (X_2; Y_2), (X_3; Y_3)...(X_N; Y_N)$,

Далее нам остаётся только проверить отрезки на пересечение друг с другом, для этого

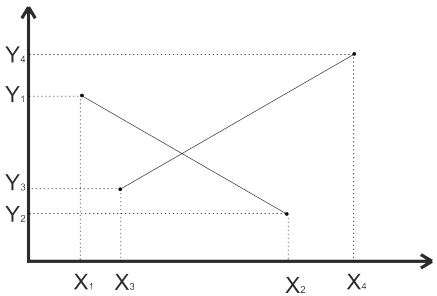


Рис. 6.4: Чертёж двух отрезков

Зная уравнения прямой:

$$y = kx + b$$

Найдём k и b для первого и второго отрезка

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

$$b = y - kx$$

(y_1 x_1 – координаты начала отрезка y_2 x_2 – координаты конца отрезка)

Далее найдём точку пересечения двух прямых

$$X = \frac{d_2 - b_1}{k_1 - k_2}$$

$$Y = k_1 x + b_1$$

Далее осталось проверить, лежит ли эта точка в пределах наших отрезков

$$X_{max} = Max(X_1, X_2, X_3, X_4)$$
$$Y_{min} = Min(Y_1, Y_2, Y_3, Y_4)$$

И если выполняется условие, то отрезки пересекаются

$$X < X_{max} and X > X_{min} and Y < Y_{max} and Y > Y_{min}$$

Пример программы

Ниже представлено решение на языке Java

```
1  import java.awt.geom.Line2D;
2  import java.io.File;
3  import java.io.FileNotFoundException;
4  import java.math.RoundingMode;
5  import java.text.DecimalFormat;
6  import java.util.ArrayList;
7  import java.util.Scanner;
8
9  public class Main {
10     private static double life_cycle = 0;
11     private static double time = 0;
12
13     public static void main(String[] args) throws FileNotFoundException {
14
15             // read input
16         Scanner scan = new Scanner(System.in);
17         ArrayList<Num> arrayOfSpeed = new ArrayList<>();
18         ArrayList<Cordinate> arrayOfPoints = new ArrayList<>();
19         life_cycle = Double.valueOf(scan.next().replace(',','.'));
20         time = Double.valueOf(scan.next().replace(',','.'));
21
22             //format control
23         DecimalFormat df = new DecimalFormat("#.#");
24         df.setRoundingMode(RoundingMode.HALF_DOWN);
25         df.setMinimumFractionDigits(0);
```

```
26
27          int counter = 0;
28          arrayOfSpeed.add(new Num(0, 0));
29          while (counter != life_cycle) {
30              arrayOfSpeed.add(new Num(Double.valueOf(scan.next().
31                                  replace(',','.')), Double.valueOf(scan.next().
32                                  replace(',','.'))));
33              counter++;
34          }
35
36          arrayOfPoints.add(new Cordinate(0, 0));
37
38          double angle = 0;
39          double A1;
40          double A2;
41          double b1;
42          double b2;
43          double xIntersect = 0;
44          double yIntersect = 0;
45          double distance = 0;
46          boolean key = false;
47          int interCount = 0;
48
49          for (int i = 1; i < arrayOfSpeed.size(); i++) {
50              angle = angle + arrayOfSpeed.get(i).angleSpeed * time;
51              arrayOfPoints.add(
52                                  new Cordinate(arrayOfPoints.get(i - 1).x +
53                                  arrayOfSpeed.get(i).lineSpeed * time *
54                                  Math.cos(angle),arrayOfPoints.get(i - 1).y +
55                                  arrayOfSpeed.get(i).lineSpeed * time *
56                                  Math.sin(angle)));
57
58              if (arrayOfPoints.size() > 2) {
59                  for (int k = 1; k < arrayOfPoints.size() - 4; k++) {
60                      for (int j = k + 2; j < arrayOfPoints.size() - 1; j++) {
61                          key = Line2D.linesIntersect(
62                                                      arrayOfPoints.get(k).x,
63                                                      arrayOfPoints.get(k).y,
64                                                      arrayOfPoints.get(k + 1).x,
65                                                      arrayOfPoints.get(k + 1).y,
66                                                      arrayOfPoints.get(j).x,
67                                                      arrayOfPoints.get(j).y,
68                                                      arrayOfPoints.get(j + 1).x,
69                                                      arrayOfPoints.get(j + 1).y);
70                          if (key == true) {
71                              interCount = k;
72                              A1 = (arrayOfPoints.get(k).y -
73                                  arrayOfPoints.get(k + 1).y) /
74                                  (arrayOfPoints.get(k).x -
75                                  arrayOfPoints.get(k + 1).x);
76                              A2 = (arrayOfPoints.get(j).y -
77                                  arrayOfPoints.get(j + 1).y) /
78                                  (arrayOfPoints.get(j).x -
79                                  arrayOfPoints.get(j + 1).x);
80                              b1 = arrayOfPoints.get(k).y -
81                                  A1 * arrayOfPoints.get(k).x;
82                              b2 = arrayOfPoints.get(j).y -
83                                  A2 * arrayOfPoints.get(j).x;
84                              xIntersect = (b2 - b1) / (A1 - A2);
85                              yIntersect = xIntersect * A1 + b1;
```

```
                            distance = Math.sqrt((xIntersect - 0) *
                                (xIntersect - 0) + (yIntersect - 0) *
                                (yIntersect - 0));
                            System.out.println((int) (distance));
                            return;
                        }
                    }
                }
            }
        }
    }
}

class Num {
    double lineSpeed;
    double angleSpeed;

    public Num(double lineSpeed, double angleSpeed) {
        this.angleSpeed = angleSpeed;
        this.lineSpeed = lineSpeed;
    }

    public double getLineSpeed() {
        return lineSpeed;
    }

    public double getAngleSpeed() {
        return angleSpeed;
    }
}

class Cordinate {
    double x;
    double y;

    public Cordinate(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }
}
```

Задача 6.3.2. Построение карты (10 баллов)

На роботе, собранному по дифференциальной схеме, установлен инфракрасный дальномер так, что он направлен вправо относительно курса движения робота, передняя плоскость датчика совпадает с осью симметрии робота и выдвинута на Y мм относительно оси вращения в сторону передней части робота.

Он движется вдоль чёрной линии и его перемещение робота задаётся парами ли-

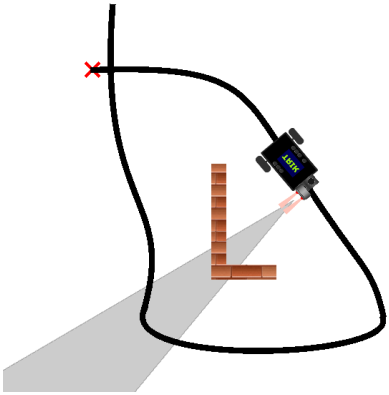


Рис. 6.5: Пример поля для входных данных №1

нейных ν и угловых ω скоростей. Пары скоростей и показания датчика передаются через равные промежутки времени t с. Диапазон работы датчика 0–100см. При этом если препятствие находится вне этого диапазона, то показания будут равны 100. Значения скоростей не изменяются между замерами, они могут измениться мгновенно в момент проведения измерения.

Робот начинает своё движение в точке $(0, 0)$ и имеет направление движения сонаправленное с направлением оси X. Необходимо определить является ли проекция препятствия замкнутым многоугольником, если известно, что на поле во время запуска робота точно не используются препятствия, чья площадь меньше Z мм^2.

Формат входных данных

Первая строка входной файла содержит четыре числа, разделённых пробелами: N — количество замеров угловой и линейной скоростей ($0 \leqslant N \leqslant 10000$), t — время между каждой парой замеров в секундах ($0 \leqslant t \leqslant 10000$), y — количество мм на сколько выдвинута передняя плоскость датчика расстояния относительно оси вращения ($0 \leqslant y \leqslant 10000$), и z — минимально возможная площадь препятствия в мм^2 ($0 \leqslant z \leqslant 100000$). N, y, z — целые числа, t — вещественное число.

Далее идет N строк, в которых указаны два вещественных числа и одно целое, разделённых пробелами: ν - линейная скорость ($-1000 \leqslant \nu \leqslant 1000$), ω - угловая скорость ($-100 \leqslant \omega \leqslant 100$), и d - показание дальномера ($0 \leqslant d \leqslant 100$).

Формат выходных данных

"True"или "False"в зависимости от того является или нет препятствие замкнутым многоугольником.

Примеры

Пример №1

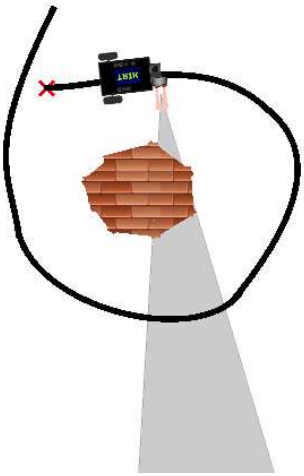


Рис. 6.6: Пример поля для входных данных №2

Стандартный вход

```
51 0,3 175 320523
202,807 0,12 100
221,54 -0,09 100
215,025 -0,13 100
190,59 -0,13 48
211,767 -0,15 23
199,549 -0,33 20
218,283 -0,41 17
209,323 -0,42 14
197,106 -0,5 17
209,323 -0,31 19
214,21 -0,33 22
211,767 -0,15 26
172,671 -0,04 28
216,654 -0,17 23
219,097 -0,07 18
223,169 -0,02 19
221,54 -0,02 100
221,54 -0,02 100
220,726 -0,1 100
215,025 -0,24 100
216,654 -0,3 100
180,002 -1,03 100
134,39 -1,63 42
175,115 -1,16 27
192,219 -0,74 22
183,26 -0,53 21
```

```
191,404 -0,4 20
196,291 -0,31 19
219,097 -0,34 18
208,509 -0,24 22
205,251 -0,2 49
201,178 -0,21 100
218,283 -0,35 100
196,291 -0,68 49
144,164 -1,54 23
182,445 -0,97 17
186,518 -0,44 15
219,097 -0,23 16
219,911 0,02 17
219,097 0,1 20
222,355 0,03 100
222,355 0,01 100
221,54 -0,06 100
222,355 -0,1 100
220,726 -0,16 100
217,468 -0,1 100
220,726 -0,1 100
214,21 -0,14 100
210,138 -0,19 100
218,283 -0,11 100
222,355 -0,01 100
```

Стандартный выход

```
False
```

Пример №2

Стандартный вход

```
47 0,3 175 117505
167,784 0,3 19
218,283 0,07 18
222,355 -0,01 17
217,468 0,01 16
219,911 -0,06 20
215,839 -0,12 24
216,654 -0,2 33
200,364 -0,24 100
207,694 -0,33 100
210,138 -0,41 100
204,436 -0,42 100
211,767 -0,32 100
210,138 -0,67 100
191,404 -0,66 100
197,106 -0,8 43
197,92 -0,72 29
196,291 -0,61 26
186,518 -0,38 27
```

```
209,323 -0,47 26
209,323 -0,36 27
205,251 -0,24 29
209,323 -0,38 32
191,404 -0,53 31
193,033 -0,57 29
193,848 -0,65 25
184,074 -0,45 24
210,138 -0,48 23
211,767 -0,5 20
197,92 -0,34 19
210,952 -0,38 22
215,839 -0,42 22
197,92 -0,25 25
181,631 -0,44 26
207,694 -0,46 24
210,952 -0,62 32
208,509 -0,67 28
209,323 -0,46 27
207,694 -0,34 100
209,323 -0,4 100
215,025 -0,35 100
217,468 -0,29 100
187,332 -0,34 100
209,323 -0,21 100
209,323 -0,14 100
219,097 -0,1 100
215,839 -0,18 100
216,654 -0,06 100
```

Стандартный выход

```
True
```

Решение

Рассмотрим движения робота между моментами изменения скоростей. В этот момент линейная и угловая скорость постоянны и мы можем рассматривать движение робота, как показано на рисунке (в начальный момент времени робот направлен вдоль оси X)

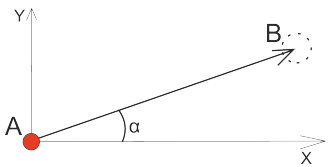


Рис. 6.7: Движение между измерениями скоростей из точки старта

Угол α мы можем вычислитть по формуле $\alpha = \omega t$. Длину отрезка AB можно высчитать по формул $l = \nu t$.

Далее мы узнаём координаты точки B по формуле $X_B = l\cos(\alpha)$ $Y_B = l\sin(\alpha)$

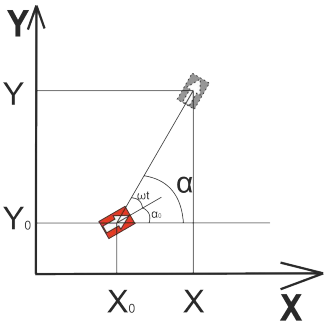


Рис. 6.8: Движение между измерениями скоростей

Теперь рассмотрим случай когда α_0 (начальный угол робота) $\neq 0$, то есть робот уже находится под каким то углов к оси X. В этом случае угол α будет вычисляться по формуле

$\alpha = \alpha_0 + \omega t.$

Дальше рассмотрим случай, когда наш робот уже находится в точке $(X_0; Y_0)$ при этом формула нахождения координат приобретёт вид

$X = X_0 + l\cos(\alpha)$

$Y = Y_0 + l\sin(\alpha)$

Далее нам нужно определить координаты препятствия

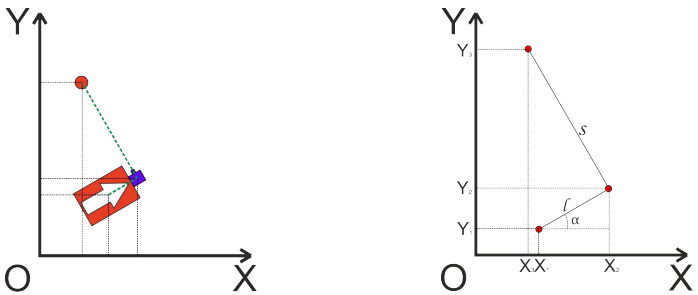


Рис. 6.9: Пересчёт координат робота в координаты препятствия

$$X_{obstacle} = X_{robot} + f\cos(\alpha) - s\sin(\alpha)$$
$$Y_{obstacle} = Y_{robot} + f\sin(\alpha) + s\cos(\alpha)$$

Далее находим площадь фигуры по формуле Гаусса

Где

- A - площадь многоугольника
- n — количество сторон многоугольника

$$\mathbf{A} = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right|$$

$$= \frac{1}{2}\left|x_1 y_2 + x_2 y_3 + \cdots + x_{n-1} y_n + x_n y_1 - x_2 y_1 - x_3 y_2 - \cdots - x_n y_{n-1} - x_1 y_n\right|$$

$$\mathbf{A} = \frac{1}{2}\left|\sum_{i=1}^{n} x_i (y_{i+1} - y_{i-1})\right| = \frac{1}{2}\left|\sum_{i=1}^{n} y_i (x_{i+1} - x_{i-1})\right| = \frac{1}{2}\left|\sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i\right| =$$

$$= \frac{1}{2}\left|\sum_{i=1}^{n} \det\begin{pmatrix} x_i & y_i \\ x_{i+1} & y_{i+1} \end{pmatrix}\right|$$

Рис. 6.10: Запись формулы Гаусса

- (x_i, y_i), i = 1, 2,..., n — координаты вершин многоугольника

Пример программы

Ниже представлено решение на языке Java

```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.math.RoundingMode;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    private static double life_cycle;
    private static double time;
    private static double naScolko;
    private static double minS;

    public static void main(String[] args) throws IOException {

        Scanner scan = new Scanner(System.in);

        life_cycle = Double.valueOf(scan.next().replace(',','.'));
        time = Double.valueOf(scan.next().replace(',','.'));
        naScolko = Double.valueOf(scan.next().replace(',','.'));
        minS = Double.valueOf(scan.next().replace(',','.'));

        //создаем массивы координат и входных данных
        ArrayList<Cordinates> arrayOfCoordinates = new ArrayList<>();
        ArrayList<Nums> arrayOfInput = new ArrayList<>();
        ArrayList<Double> arrayOfDalnomer = new ArrayList<>();
        ArrayList<Double> arrayOf_L = new ArrayList<>();
        ArrayList<CoordinatesOfPolygon> arrayPointsPolygon =
        new ArrayList<>();
        ArrayList<Double> array = new ArrayList<>();
        ArrayList<Double> arrayCor = new ArrayList<>();

        //заполняем массив входных данных
        int counter = 0;
        arrayOfInput.add(new Nums(0, 0, 0));
```

```
37          while (counter != life_cycle) {
38              arrayOfInput.add(new Nums(Double.valueOf(scan.next().
39                  replace(',', '.')), Double.valueOf(scan.next().replace(',', '.')),
40              Double.valueOf(scan.next().replace(',', '.'))));
41              counter++;
42          }
43
44          //добовляем координаты движения в массив кординат
45          arrayOfCoordinates.add(new Cordinates(0, 0));
46          double angle = 0;
47
48          for (int i = 1; i < arrayOfInput.size(); i++) {
49
50              angle = angle + arrayOfInput.get(i).angleSpeed * time;
51
52              arrayOfCoordinates.add(new Cordinates(arrayOfCoordinates.get(i - 1).x +
53                  arrayOfInput.get(i).lineSpeed * time * Math.cos(angle),
54                  arrayOfCoordinates.get(i - 1).y + arrayOfInput.get(i).lineSpeed *
55                  time * Math.sin(angle)));
56          }
57                  //отображаем координаты у
58          for (int i = 0; i < arrayOfCoordinates.size(); i++) {
59              arrayCor.add(i, arrayOfCoordinates.get(i).getY() * (-1));
60          }
61
62          angle = 0;
63          for (int i = 0; i < arrayOfInput.size(); i++) {
64              angle = angle + arrayOfInput.get(i).angleSpeed * time;
65              array.add(angle);
66          }
67
68          //заполним массив видимых расстояний и где соточка(ничего не видим) ставим -1
69          for (int i = 0; i < arrayOfInput.size(); i++) {
70              if (arrayOfInput.get(i).dalnomer != 100) {
71                  arrayOfDalnomer.add(arrayOfInput.get(i).dalnomer);
72              } else if (arrayOfInput.get(i).dalnomer == 100) {
73                  arrayOfDalnomer.add(-1.0);
74              }
75          }
76
77          //получение и заполнение массива с координатами полигона которое обьезжаем
78          //координаты полигона
79          double xP;
80          double yP;
81
82          for (int i = 1; i < arrayOfDalnomer.size(); i++) {
83
84              if (arrayOfDalnomer.get(i) != -1.0) {
85
86                  xP = arrayOfCoordinates.get(i).getX() + naScolko *
87                      Math.cos(array.get(i)) + Math.sin(array.get(i)) *
88                      arrayOfDalnomer.get(i) * 10;
89
90                  yP = arrayCor.get(i) - naScolko * Math.sin(array.get(i)) +
91                      Math.cos(array.get(i)) * arrayOfDalnomer.get(i) * 10;
92
93                  arrayPointsPolygon.add(new CoordinatesOfPolygon(xP, yP));
94
95              }
96
```

```
97          }
98
99                  //используем формулу Гаусса для подсчёта площади
100         double s = 0, s1 = 0;
101         for (int j = 0; j < arrayPointsPolygon.size() - 1; j++) {
102             s = s + (arrayPointsPolygon.get(j).getX() *
103                 arrayPointsPolygon.get(j + 1).getY());
104             s1 = s1 + (arrayPointsPolygon.get(j).getY() *
105                 arrayPointsPolygon.get(j + 1).getX());
106         }
107         s = s + arrayPointsPolygon.get(arrayPointsPolygon.size() - 1).getX() *
108             arrayPointsPolygon.get(0).y;
109         s1 = s1 + arrayPointsPolygon.get(arrayPointsPolygon.size() - 1).getY() *
110             arrayPointsPolygon.get(0).x;
111
112         DecimalFormat form = new DecimalFormat("#.##");
113         form.setRoundingMode(RoundingMode.HALF_DOWN);
114         form.setMinimumFractionDigits(2);
115         double space = (s1 - s) / 2;
116
117                 //проверка площади на знак
118         if (space < 0) {
119             space = space * (-1);
120             if (space >= minS) {
121                 System.out.println("True");
122             } else if (space < minS) {
123                 System.out.println("False");
124             }
125         } else if (space > 0) {
126             if (space >= minS) {
127                 System.out.println("True");
128             } else if (space < minS) {
129                 System.out.println("False");
130             }
131         }
132     }
133 }
134 class CoordinatesOfPolygon {
135     double x;
136     double y;
137
138     public CoordinatesOfPolygon(double x, double y) {
139         this.x = x;
140         this.y = y;
141     }
142
143     public double getX() {
144         return x;
145     }
146
147     public double getY() {
148         return y;
149     }
150 }
151 class Cordinates {
152     double x;
153     double y;
154
155
156     public Cordinates(double x, double y) {
```

```
157             this.x = x;
158             this.y = y;
159         }
160 
161         public double getX() {
162             return x;
163         }
164 
165         public double getY() {
166             return y;
167         }
168 
169     }
170     class Dalnomer {
171         double dalnomer;
172 
173         public void Dalnomer(double dalnomer) {
174             this.dalnomer = dalnomer;
175         }
176 
177         public double getDalnomer() {
178             return dalnomer;
179         }
180     }
181     class Nums {
182         double lineSpeed;
183         double angleSpeed;
184         double dalnomer;
185 
186         public Nums(double lineSpeed, double angleSpeed, double dalnomer) {
187             this.angleSpeed = angleSpeed;
188             this.lineSpeed = lineSpeed;
189             this.dalnomer = dalnomer;
190         }
191 
192         public double getLineSpeed() {
193             return lineSpeed;
194         }
195 
196         public double getAngleSpeed() {
197             return angleSpeed;
198         }
199 
200         public double getDalnomer() {
201             return dalnomer;
202         }
203     }
204 
205     class Smen {
206         double hy;
207         double lx;
208 
209         public Smen(double hy, double lx) {
210             this.hy = hy;
211             this.lx = lx;
212         }
213 
214         public double getHy() {
215             return hy;
216         }
```

```
217
218     public double getLx() {
219         return lx;
220     }
221 }
```

Задача 6.3.3. Исследование лабиринта (5 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. Последний — вправо.

Показания датчиков расстояния бинарные, т.е.

- 1 — данный датчик не увидел препятствие, проезд в данном направлении свободен
- 0 — данные датчик увидел препятствие, проезд в данном направлении перекрыт

Первоначальная структура лабиринта известна в виде списка достижимости секций лабиринта. Связь в списке двунаправленная. Первоначальный сектор старта и направление движения также известны.

Считается, что непосредственно перед стартом робота случилось обрушение каких-то секций лабиринта. Поэтому роботу необходимо в процессе обхода всего лабиринта, выяснить какие секции оказались недоступны в результате обрушения.

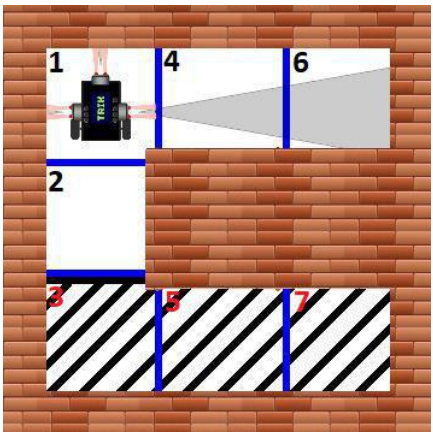


Рис. 6.11: Пример изображения для входных данных

Формат входных данных

Первая стока входного файла содержит три целых числа, разделённые пробелами: M — общее количество секций в лабиринте ($0 \leqslant M \leqslant 10000$), N — количество

соединений комнат ($0 \leqslant N \leqslant 10000$), и K — количество замеров ($0 \leqslant K \leqslant 10000$), произведенных датчиками, в процессе обхода всего лабиринта.

Далее идет N строк, в которых указаны 3 целых числа — номера секторов, между которые связаны между собой и направление связи 0–3 (0—вверх, 1—вправо, 2—вниз, 3—налево).

Потом идет строчка содержащая 2 целых числа: сектор и направление старта.

После указаны K строк, в которых указаны 3 цифры: первая — показание инфракрасного датчика, направленного влево относительно курса робота, вторая — показание инфракрасного датчика, направленного прямо по курсу робота, третья — показание инфракрасного датчика, направленного вправо относительно курса робота. При этом если в строке все числа 3, то это означает поворот робота вокруг своей оси по часовой стрелке на 90°, если 2 — против часовой стрелки.

Формат выходных данных

Вывести на экран номера всех секций, которые оказались недоступны, через пробел, в порядке возрастания их номеров.

Примеры

Пример №1

Стандартный вход

```
7 6 18
4 1 3
2 1 0
3 2 0
5 3 3
6 4 3
7 5 3
1 0
0 0 1
3 3 3
0 1 1
0 1 0
0 0 0
2 2 2
1 0 0
2 2 2
0 1 0
0 1 0
1 0 0
2 2 2
1 1 0
0 0 0
2 2 2
1 0 0
2 2 2
0 1 0
```

Стандартный выход

```
3 5 7
```

Комментарии

На рисунке 6.15 перечёркнутые сектора — недоступные сектора. Показания датчиков: 0 0 1. Нумерация секторов также представлена на рисунке.

Решение

По условию задачи, мы знаем, что роботу для составления полной картины лабиринта нужно будет проехать все доступные ему участки. По этому опираясь на начальную позицию и начальное направление робота, мы можем построить его маршрут. При этом зная предыдущий сектор и направление проезда робота, определить тот сектор в котором робот окажется после своего проезда вперёд, не составит труда (в условии задаче дан список всех переходов с их направлениями). Изменение направления же явно указано во входных данных. В конце этого алгоритма мы получаем список комнат в которых побывал робот, вычтя их из списка всех комнат мы получим ответ.

Пример программы

Ниже представлено решение на языке Python3

```
def turnRight(initialPosition):
    return (initialPosition + 1) % 4

def turnLeft(initialPosition):
    return (initialPosition - 1) % 4

def symmetricAdd(room, a, b, c):
    if c == 3:
        room[1][b - 1] = a
    elif c == 2:
        room[0][b - 1] = a
    elif c == 1:
        room[3][b - 1] = a
    else:
        room[2][b - 1] = a

n, k, l = map(int, input().strip().split())
rooms = []
for i in range(4):
    rooms.append([-1 for j in range(n)])
for i in range(k):
    a, b, c = map(int, input().strip().split())
    rooms[c][a-1] = b
    symmetricAdd(rooms, a, b, c)
initialPosition, initialDirection = map(int, input().strip().split())
newRooms = []
for i in range(4):
    newRooms.append([-1 for j in range(n)])

```

```
currentRoom = initialPosition
currentDirection = initialDirection
turnFlag = True

for i in range(l):
    left, middle, right = map(int, input().strip().split())
    if left == middle == right == 3:
        currentDirection = turnRight(currentDirection)
        turnFlag = True
    elif left == middle == right == 2:
        currentDirection = turnLeft(currentDirection)
        turnFlag = True
    else:
        if not turnFlag:
            currentRoom = newRooms[currentDirection][currentRoom - 1]
        if left == 1:
            newRooms[turnLeft(currentDirection)][currentRoom-1] = \
                rooms[turnLeft(currentDirection)][currentRoom-1]
            symmetricAdd(newRooms, currentRoom,
                    rooms[turnLeft(currentDirection)][currentRoom-1],
                         turnLeft(currentDirection))
        if right == 1:
            newRooms[turnRight(currentDirection)][currentRoom - 1] = \
                rooms[turnRight(currentDirection)][currentRoom - 1]
            symmetricAdd(newRooms, currentRoom,
                         rooms[turnRight(currentDirection)][currentRoom - 1],
                         turnRight(currentDirection))
        if middle == 1:
            newRooms[currentDirection][currentRoom - 1] = \
                rooms[currentDirection][currentRoom - 1]
            symmetricAdd(newRooms, currentRoom,
                    rooms[currentDirection][currentRoom - 1],
                    currentDirection)
        turnFlag = False

allRooms = set()
allNewRooms = set()
for i in range(4):
    for j in newRooms[i]:
        if j != -1:
            allNewRooms.add(j)
for i in range(4):
    for j in rooms[i]:
        if j != -1:
            allRooms.add(j)

hiddenRooms = allRooms.difference(allNewRooms)
sortedHiddenRooms = sorted(list(hiddenRooms))

for i in sortedHiddenRooms:
    print(i, end = " ")
```

Задача 6.3.4. Оптическая одометрия (5 баллов)

Робот, собранный по дифференциальной схеме, оборудован камерой и движется прямо вперёд.

Камера на роботе установлена так, что смотрит вниз перпендикулярно поверхности, по которой движется робот, известны характеристики камеры такие как раз-

решение ($M \times N$), угол обзора α, высота расположения над плоскостью движения. Верхний край кадра находится дальше от робота, чем нижний край кадра. Есть последовательность кадров, сделанных камерой через равные промежутки времени. Время между кадрами равно 1 секунде. Зная что робот движется только прямо и скорость его движения постоянна, определить расстояние на которое переместился робот.

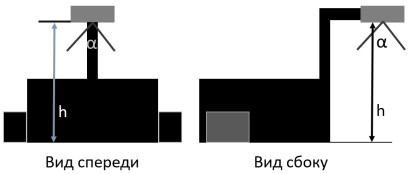


Рис. 6.12: Вид робота. Поясняющая картинка

Формат входных данных

Первая строка входных данных содержит 4 числа, целочисленные M, N и вещественные α, h — разрешение фотографии в пикселях, где M - длина, N - ширина ($2 \leqslant M, N \leqslant 60$), угол обзора камеры в градусах ($10^\circ \leqslant \alpha \leqslant 170^\circ$), высота расположения камеры над плоскостью движения в сантиметрах ($1 \leqslant h \leqslant 100$). Во второй строке задано целое число K — количество снимков ($2 \leqslant K \leqslant 8$). В следующих $K * M$ строках перечисляются N целых чисел — 0 или 1.

Формат выходных данных

Выведите единственное число — расстояние в см, на которое переместился робот с точностью до 2 символов.

Примеры

Пример №1

Стандартный вход

```
8 8 54 14
3
1 1 1 0 0 1 1 0
0 0 1 1 0 0 1 0
1 0 0 1 0 1 1 0
0 0 1 1 1 0 0 1
0 1 0 1 0 1 1 0
1 0 1 1 0 0 0 1
0 0 1 0 0 1 0 1
1 0 0 1 1 1 0 1
1 0 1 1 0 0 0 1
0 0 1 0 0 1 0 1
```

```
1 0 0 1 1 1 0 1
0 1 1 0 1 1 0 1
0 0 0 1 1 0 0 1
0 0 1 1 1 1 0 0
1 1 1 1 1 1 0 0
0 0 1 1 0 0 1 0
0 0 1 1 1 1 0 0
1 1 1 1 1 1 0 0
0 0 1 1 0 0 1 0
1 0 0 1 1 1 0 1
0 0 0 0 0 0 0 0
1 1 1 1 0 1 0 0
0 0 0 0 1 1 0 1
0 1 1 0 0 1 0 1
```

Стандартный выход

```
26.75
```

Решение

Для лучшего понимания представим нашу камеру в виде равнобедренного треугольника с опущенной высотой

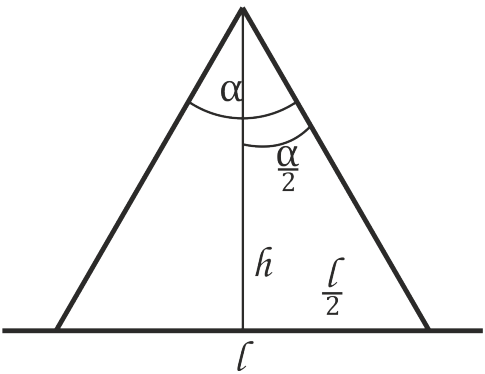


Рис. 6.13: Геометрическое представление камеры робота

Зная высоту h и угол α мы можем найти l по формуле $l = 2h\,\mathrm{tg}(\frac{\alpha}{2})$

Далее мы можем найти длину одного пикселя $l_p = \frac{l}{M} = \frac{2h\,\mathrm{tg}(\frac{\alpha}{2})}{M}$

Смещение камеры в пикселях(r) находится алгоритмом построчного сравнения кадров.

Далее подставляем в формулу и находи смещение в см $l_s = \frac{2rh\,\mathrm{tg}(\frac{\alpha}{2})}{M}$

Так как интервал между кадрами 1сек скорость в см/с равна смещению в см. А пройденный путь $S = \nu t$

$$S = \frac{2Krh\,\mathrm{tg}(\frac{\alpha}{2})}{M}$$

Пример программы

Ниже представлено решение на языке Python3

```
import math
data = input().split()

m = int(data[0])
n = int(data[1])
a = float(data[2])
h = float(data[3])
countFrame = int(input())

file = ''
for i in range(countFrame*m):
    file +=input() +'\n'

lines = file.split('\n')

res = -1
nRes = 0
for r in range(countFrame - 1):
    if res != -1: break
    nRes = 0
    for i in range(m):
        if lines[i + m*r ].split() == lines[ m + m*r].split():
            c = 0
            for j in range(m - i):
                if lines[ i + m*r +j ].split() == lines[m + m*r + j]\
                        .split():
                    c = j + 1
                else:break
            if c == m - i:
                res = i
                nRes += 1
        if nRes > 1: res = -1;
print(round(2*math.tan(a*math.pi/180/2)*h*res/m*countFrame,2))
```

Задача 6.3.5. Фильтрация значений (5 баллов)

Робот, собранный по дифференциальной схеме, имеет инфракрасный дальномер и перемещается прямо по одной прямой. Дальномер установлен так, что он направлен влево относительно курса движения робота.

Необходимо определить форму исследуемой стены. Она может одной из следующих:

- линейная (linear)
- модуль (module)
- квадратичная (quadratic)
- кубическая (cubic)
- синусоида (sinus)

Показания датчика не идеальные, существует некоторая погрешность измерений. Гарантируется, что стена всегда находится в поле зрения датчика.

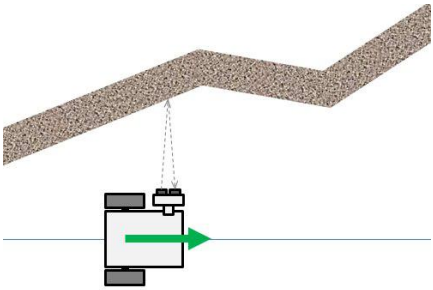


Рис. 6.14: Движение робота относительно стены

Формат входных данных

Первая строка входного файла содержит одно целое число: N — количество ($200 \leqslant N \leqslant 10000$) показаний датчика, снятых в процессе движения вдоль стены.

Последующие N строк содержат одно целое число — s — показание дальномера ($0 \leqslant s \leqslant 1200$).

Формат выходных данных

Одна стока, содержащая одно слово - форму стены. Слово должно быть на английском одним из списка, представленного при описании формы стены.

Примеры

Пример №1

Стандартный вход

```
206
13
17
17
18
18
21
21
22
23
22
22
21
26
25
26
27
31
```

32
30
35
33
35
37
37
40
41
41
44
42
45
46
46
49
50
48
50
52
53
53
54
60
58
61
61
61
62
63
61
65
63
65
66
66
67
70
73
72
71
74
75
76
77
78
82
79
81
79
81
80
83

82
84
83
83
82
81
79
79
79
80
77
77
78
74
74
73
74
71
72
71
70
67
66
66
63
61
62
59
62
57
56
54
54
49
48
48
50
49
45
45
43
45
41
41
37
39
37
35
36
37
36
34
35

33
32
30
32
33
28
30
30
31
28
30
31
29
31
28
27
30
29
29
27
29
29
27
28
29
29
31
31
29
30
29
30
30
31
33
33
34
33
33
38
33
37
36
37
35
36
39
39
36
40
42
44
43

```
46
47
49
49
51
53
55
54
56
59
60
59
65
62
63
66
67
67
69
69
72
77
76
79
82
83
83
82
85
84
```

Стандартный выход

```
cubic
```

Комментарии

Рис. 6.15: Пример графика по входным данным

На рис. 6.15 изображен график, построенный по входным значениям с некоторой фильтрацией. По графику видно, что исследуемая стена имеет вид кубической функции, следовательно необходимо вывести ”cubic”

Рекомендуется использовать теорему Лагранжа (http://www.math24.ru/%D1%82%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0-%D0%BB%D0%B0%D0%B3%D1%80%D0%B0%D0%BD%D0%B6%D0%B0.html)

Решение

Считаем значения и обработаем их медиальным и сглаживающим фильтрам.

Далее мы можем заметить, что у разных функций разное количество точек перегиба (линейная – 0, модуль и квадратичная – 1, кубическая – 2, синусоида – 3 и больше).

Точки перегиба – точки, где функция меняет свой характер (возрастающий переходит убывающий и наоборот).

Возрастающая функция – функция в которой выполняется условие, что каждое последующее значение больше предыдущего ($x_i > x_{i-1}$)

Убывающая функция – функция в которой выполняется условие, что каждое последующее значение меньше предыдущего ($x_i < x_{i-1}$)

Зная это мы можем определять количество точек перегиба, для этого достаточно найти разницу между x_i и x_{i-1} (если она положительная то функция на этом участке имеет возрастающий характер, если нет, то убывающий), если $x_i - x_{i-1}$ и $x_{i+1} - x_i$ имеют разные знаки, то x_i это точка перегиба.

Для того что бы мы могли различить функцию модуля и квадратичную функцию, давайте рассмотрим их внимательнее.

Квадратичная функция

Функция модуля

Рис. 6.16: Сравнение функций

Как мы видим, функция модуль всегда возрастает и убывает с одной той же скоростью, а квадратичная с разной, чем дальше от точки перегиба тем больше эта скорость. То есть $|x_i - x_{i-1}| = |x_{i+1} - x_i|$ то перед нами функция модуля, а если $|x_i - x_{i-1}| \neq |x_{i+1} - x_i|$, то квадратичная. Хочу заметить, что в нашем случае строгое равенство не подходит, так как мы имеем дело с неидеальными данными (данные, на которых присутствует шум, который мы хотя и практически полностью убрали фильтрами, но его следы всё таки присутствуют в данных), по этому в программе допускаем небольшую погрешность в моменте сравнения $|x_i - x_{i-1}|$ и $|x_{i+1} - x_i|$.

Пример программы

Ниже представлено решение на языке Python3

```
1 dataset = []
2 dataset.append(input())
3
```

```
for i in range(int(dataset[0])):
    dataset.append(input())

k = 4
d = 3

line = []

for i in range(int(dataset[0])//d):
    val = int(dataset [1+i*d])
    for j in range(d):
        if val < int(dataset[1+i*d+j]):
            val = int(dataset[1+i*d + j])
    line.append(val)

for i in range(1,int(dataset[0])//d):
    line[i]= (line[i-1]*k+line[i])/(k+1)

dif = []
for i in range(5,int(dataset[0])//d):
    dif.append(line[i]-line[i-1])

c = 0

for i in range(2,len(dif)):
    if dif[i-2]*dif[i-1] < 0 and dif[i-2]*dif[i]:
        c+=1

if c == 0:
    print('linear')
elif c == 1:
    if abs(dif[len(dif)//4*3] - dif[len(dif)-3])>1:
        print('quadratic')
    else: print('module')
elif c == 2:
    print('cubic')
else:
    print('sinus')
```

Задача 6.3.6. Всенаправленная тележка (5 баллов)

Робот, моторы которого расположены под углом в 120° друг к другу, движется в заданном направлении некоторое время t. На валах моторов закреплены омниколёса (`https://en.wikipedia.org/wiki/Omni_wheel`).

Необходимо определить новые координаты центра робота, если в момент начала движения он находился в точке $(0, 0)$ и один из двигателей находился на оси Y, в направлении положительной части (см. рис. 6.17). Расположение моторов является постоянным и соответствует расположению моторов на рисунке 6.17.

Считать что, мощность на все моторы подаётся одновременно, время их вращения одинаковое, при этом они достигают своей скорости мгновенно, колеса вращаются без проскальзывания.

Формат входных данных

Одна строчка, состоящая из 5ти чисел: d, w_1, w_2, w_3, t, где

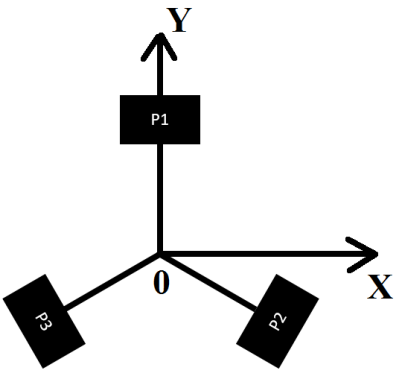


Рис. 6.17: Расположение моторов робота относительно центра глобальной системы отсчета в начальный момент времени

- d — диаметра колёс в мм ($d_1, d_2, d_3 = d; 30 \leqslant d \leqslant 100$);
- $P1, P2, P3$ — скорости вращения моторов (-2 рад/с $\leqslant P1, P2, P3 \leqslant 2$ рад/с);
- t — времени движения в с ($10 \leqslant t \leqslant 1000$).

Диаметр колёс и время движения — целые числа, скорости вращения — вещественные.

Формат выходных данных

Одна строка, содержащая два вещественных числа с точностью до одного знака — координаты робота, где он закончил своё движение, через пробел. Числа указывать с точностью до сотых. Допускается погрешность в 1 мм по каждой из координат.

Примеры

Пример №1

Стандартный вход

```
30 1.5 -1.0 -0.5 10
```

Стандартный выход

```
337.5 64.95
```

Пример №2

Стандартный вход

```
100 -0.29 -0.3 0.59 10
```

Стандартный выход

```
-217.5 385.37
```

Решение

Рассмотрим модель, у нас есть 3 жёстко соединённые точки которые вращаются под углом 120°

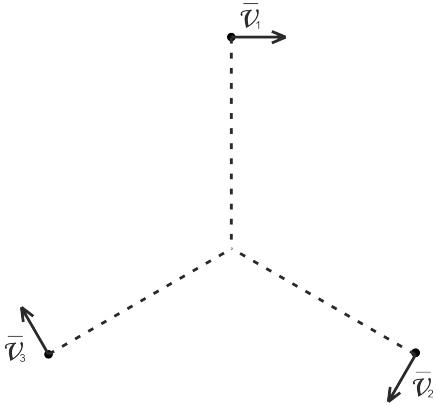


Рис. 6.18:

как можно заметить, что при при скоростях не выполняющих условие $\nu_1 + \nu_2 + \nu_3 = 0$ будет возникать вращение устройства, но так как в условии задачи не дано расстояние от центра устройства до каждого колеса (только с этим параметром возможно вычисление движение по дуге), мы рассмотрим случай, когда $\nu_1 + \nu_2 + \nu_3 = 0$, в такой ситуации движение устройства прямолинейно и скорость можно вычислить сложением трёх векторов.

$$\nu_x = \nu_1 - \nu_2 \cos(-30^\circ) - \nu_3 \cos(-150^\circ)$$
$$\nu_y = \nu_3 \sin(-150^\circ) - \nu_2 \sin(-30^\circ)$$

Далее вычисление конечных координат происходит по формулам

$$X = \nu_x t$$

$$Y = \nu_y t$$

Задача 6.3.7. Планирование маршрута (5 баллов)

Робот перемещается по полигону, на котором установлены препятствия. Проекции препятствий на поле имеют вид многоугольников, координаты которых задаются во входном файле.

В процессе движения робот может лишь двигаться прямо или поворачиваться на месте вокруг своей оси и не может выходить за пределы полигона.

Определите оптимальный путь перемещения робота из точки старта в точку финиша. Под оптимальным путем подразумевается траектория, состоящая из прямых

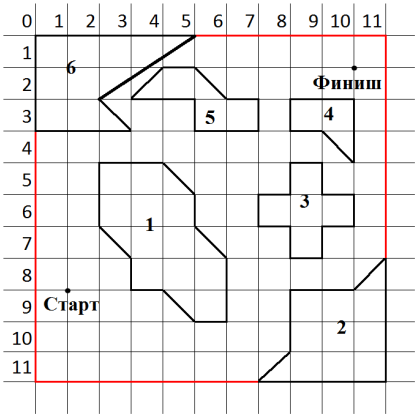


Рис. 6.19: Пример изображения для входных данных

отрезков, где количество отрезков минимально. Концами отрезков являются точки на полигоне, где роботу необходимо выполнить операцию поворота.

Размерами робота можно пренебречь, поэтому допускаются повороты и перемещения вдоль границ полигона и препятствий. На старте и на финише робот может иметь любое направление.

Формат входных данных

Первая строка содержит одно целое число — N — количество многоугольников на полигоне ($0 \leqslant N \leqslant 100$). Вторая строка содержит размер полигона — одно или два целых числа ($10 \leqslant A, B \leqslant 1000$). Третья стока содержит два числа — координаты старта (X_s и Y_s) через пробел. Четвёртая строка содержит два числа — координаты финиша (X_f и Y_f) через пробел.

Далее идёт N блоков. В первой строчке блока находится одно целое число — N_i — количество вершин в данном многоугольнике. После идет N_i строчек, в каждой находится два целых числа — координаты вершины (X_j, Y_j) через пробел. Координаты вершин каждого многоугольника задаются друг за другом по часовой стрелке. Координаты могут быть вещественными числами. Разделитель дробной части - точка.

Формат выходных данных

На первой строчке — количество поворотов K, которые должен совершить робот при перемещении от точки старта к точке финиша. Далее идет K строчек, в каждой из которой указаны координаты точек поворота. Координаты указываются в порядке прохождения точек роботом. Формат вывода координат: X, Y через пробел. В случае если вывести требуется вещественные координаты, то следует указать точность в 1 знак после запятой.

Примеры

Пример №1

Стандартный вход

```
6
11 11
1 8
10 1
11
2 4
4 4
5 5
5 6
6 7
6 9
5 9
4 8
3 8
3 7
2 6
6
11 7
11 11
7 11
8 10
8 8
10 8
12
8 4
9 4
9 5
10 5
10 6
9 6
9 7
8 7
8 6
7 6
7 5
8 5
5
8 2
10 2
10 4
9 3
8 3
8
3 2
4 1
5 1
6 2
```

```
7 2
7 3
6 3
5 2
5
0 0
5 0
2 2
3 3
0 3
```

Стандартный выход

```
3
1 3.5
7.5 3.5
8 1
```

Решение

Для начала мы строим граф по принципу достижимости. Дальше используя BFS мы можем очень просто найти путь с наименьшим числом поворотов, так как повороты происходят только в вершинах нашего графа.

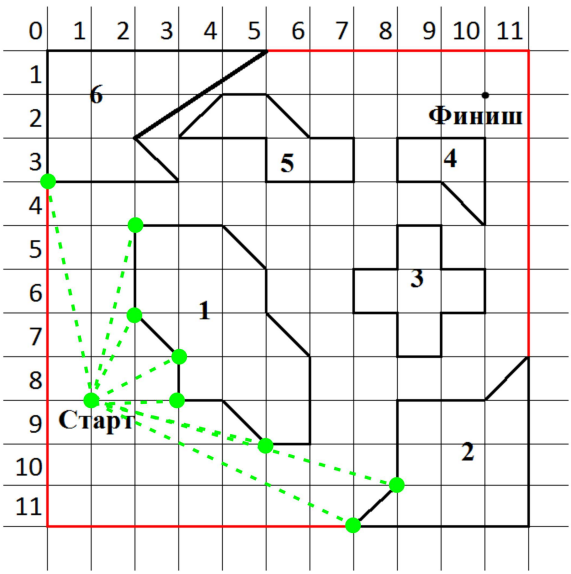


Рис. 6.20: Первая волна постройки графа. Добавляем все вершины в которые можно попасть из точки старта.

Пример программы

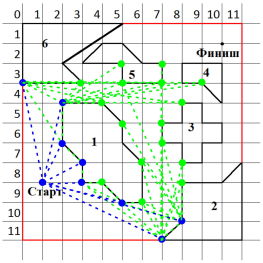
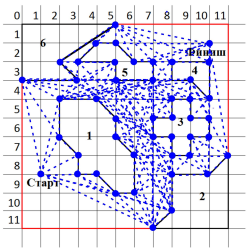
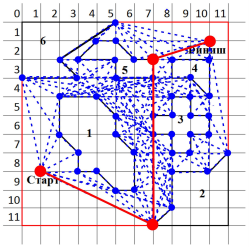


Рис. 6.21: Вторая волна постройки графа. Добавляем все вершины в которые можно попасть из вершин которые уже в графе. Заметим, что так как мы можем двигаться вдоль границ фигур, в нашем графе есть рёбра $(11;7) - (3;7)$ и $(11;7) - (2;7)$.



Конечный вид графа



Путь с наименьшим числом поворотов $(11;7) - (2;7)$

Рис. 6.22:

Ниже представлено решение на языке Java

```
import com.sun.org.apache.xpath.internal.SourceTree;

import java.awt.*;
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;
import java.io.*;
import java.util.*;
public class Main {
    public static Line2D.Double Povorot(Line2D.Double line, double grad){
        Line2D.Double new_line;
        double x1 = line.x1 , y1 = line.y1 ,x2 = line.x2, y2 = line.y2 , r ;
        grad = (Math.PI*grad)/180;
        r = Math.sqrt( (x2-x1) * (x2-x1) + (y2 - y1) * (y2 - y1));
        x2 = Math.cos(Math.acos((x2 - x1)/r) + grad) * r + x1;
        y2 = Math.sin(Math.asin((y2 - y1)/r) + grad) * r + y1;
        new_line = new Line2D.Double(x1,y1,x2,y2);
        return new_line;
    }
    public static double rastoyanie(Line2D.Double line, Point2D.Double point){
        return (Math.abs( (line.y2-line.y1) * point.x - (line.x2-line.x1) *
                                    point.y + line.x2*line.y1 - line.y2*line.x1)/
                (Math.sqrt( (line.y2-line.y1)*(line.y2-line.y1) + (line.x2 -
                                    line.x1)*(line.x2-line.x1))));
    }
```

```
25      public static Point2D.Double Free_point(Point2D.Double[] arr, int koef,
26                  Line2D.Double dotS, Line2D.Double dotE, Line2D.Double line){
27          int k = 0;
28          while(Intersection(arr,dotS,dotE,line)) {
29              line = Povorot(line, koef);
30          }
31          Point2D.Double min = null;
32          for (int i = 0; i < arr.length; i++)
33              if (!(arr[i].x == dotS.x1 && arr[i].y == dotS.y1) &&
34                                          !(arr[i].x == dotE.x1 && arr[i].y == dotE.y1))
35                  min = arr[i];
36          for (int i = 1; i < arr.length && min != null; i++)
37              if (rastoyanie(line,min) > rastoyanie(line,arr[i]) &&
38                                          !(arr[i].x == dotS.x1 && arr[i].y == dotS.y1) &&
39                                          !(arr[i].x == dotE.x1 && arr[i].y == dotE.y1))
40                  min = arr[i];
41          return min;
42      }
43      public static boolean Intersection(Point2D.Double[] arr, Line2D.Double dotS,
44                                  Line2D.Double dotE, Line2D.Double line) {
45          for (int i = 0; i < arr.length; i++) {
46              Line2D.Double edge = new Line2D.Double(arr[i].x, arr[i].y,
47                      arr[(i + 1) % arr.length].x, arr[(i + 1) % arr.length].y);
48              if (edge.intersectsLine(line) && !edge.intersectsLine(dotS)
49                                                  && !edge.intersectsLine(dotE)) {
50                  return true;
51              }
52          }
53          return false;
54      }
55      public static boolean IntersectionAll(Point2D.Double[] arr,
56              Line2D.Double dotS, Line2D.Double dotE, Line2D.Double line){
57          boolean flag = false;
58          for (int k = 0; k < arr.length && !flag; k++)
59              if (Intersection(arr[k], dotS, dotE, line))
60                  flag = true;
61          return flag;
62      }
63
64      public static double LenPath(String path){
65          if (path.equals("")) return Double.MAX_VALUE;
66          String[] s = path.split("\n");
67          Point2D.Double[] Points = new Point2D.Double[s.length];
68          double result = 0;
69          for (int i = 0; i < s.length; i++){
70              String[] s1 = s[i].split(" ");
71              Points[i] = new Point2D.Double(Double.parseDouble(s1[0]),
72                                          Double.parseDouble(s1[1]));
73          }
74          for (int i = 1; i < Points.length; i++){
75              result += Math.sqrt((Points[i-1].x - Points[i].x)*
76                                  (Points[i-1].x - Points[i].x) +
77                      (Points[i-1].y - Points[i].y) * (Points[i-1].y -
78                                      Points[i].y));
79          }
80          return result;
81      }
82      public static boolean containPart(String s, String p){
83          String[] arr = s.split("\n");
84          for (int i = 0 ; i < arr.length; i++){
```

```
            if (arr[i].equals(p)) return true;
        }
        return false;
    }
    public static boolean polygon_check(Point2D.Double[] arr,
                        Point2D.Double First, Point2D.Double Second){
        int j1 = 1000,j2 = 1000;
        boolean flag1 = false,flag2 = false;
        for (int i = 0; i < arr.length; i++){
            if (First.x == arr[i].x && First.y == arr[i].y) {
                j1 = i;
                flag1 = true;
            }
            if (Second.x == arr[i].x && Second.y == arr[i].y) {
                j2 = i;
                flag2 = true;
            }
        }
        if (flag1 && flag2){
            return Math.abs(j1-j2)==1 || Math.abs(j1-j2) == arr.length - 1;
        }
        return true;
    }
    public static void Path(Point2D.Double[][] arr,String path_lineS,
                            Point2D.Double lineS,Point2D.Double End) {
        String result = "";
       // System.out.println(lineS.x + " " + lineS.y );
        Line2D.Double dotS = new Line2D.Double(lineS.x,lineS.y,

        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr[i].length; j++){
                Point2D.Double lineE = arr[i][j];
                Line2D.Double line = new Line2D.Double(lineS.x, lineS.y,
                                                       lineE.x, lineE.y);
                Line2D.Double dotE = new Line2D.Double(lineE.x,lineE.y,
                                                       lineE.x, lineE.y);
                boolean flag = true;
                for (int k = 0; k < arr.length && flag; k++)
                    if (Intersection(arr[k], dotS, dotE, line))
                        flag = false;
                if ((flag && !containPart(path_lineS,lineS.x + " " + lineS.y))
                                && polygon_check(arr[i],lineS,lineE)) {
                    if (path_point[i][j].split("\n").length >
                            path_lineS.split("\n").length + 1 ||
                            (LenPath(path_point[i][j]) >
                            LenPath(path_lineS + lineS.x + " " +
                                                    lineS.y + '\n')
                            && path_point[i][j].split("\n").length ==
                            path_lineS.split("\n").length + 1  )
                            || path_lineS.equals("") ||
path_point[i][j].equals("")) {
                        path_point[i][j] = path_lineS + lineS.x + " " +
                                                    lineS.y + '\n';
                        Path(arr, path_point[i][j],
                                new Point2D.Double(dotE.x1, dotE.y1), End);
                    }
                }
            }
        }
    }
```

```
145
146     public static Point2D.Double IntersPoint(Line2D.Double line1,
147                                              Line2D.Double line2){
148         double k1,k2,b1,b2,y1,y2,x1,x2;
149         y1 = line1.y1;y2 = line1.y2; x1 = line1.x1; x2 = line1.x2;
150         k1 = (y2 - y1)/(x2 - x1);
151         b1 = y1 - k1 * x1;
152         System.out.println("k1: " + k1 + "   b1:" + b1);
153         y1 = line2.y1;y2 = line2.y2; x1 = line2.x1; x2 = line2.x2;
154         k2 = (y2 - y1)/(x2 - x1);
155         b2 = y1 - k1 * x1;
156         System.out.println("k2: " + k2 + "   b2:" + b2);
157         return new Point2D.Double((b2-b1)/(k1-k2),(b2-b1)/(k1-k2) * k2 + b2);
158     }
159
160     public static Point2D.Double Check(Point2D.Double[][] arr,
161                 Point2D.Double lineS, Point2D.Double lineE, double a, double b){
162         Line2D.Double lineStart = new Line2D.Double(lineS,lineE),
163                                   lineEnd = new Line2D.Double(lineE,lineS);
164         Point2D.Double pp;
165         for (double i = 0; i < b; i+=0.1) {
166             pp = new Point2D.Double(0,i);
167             lineStart = new Line2D.Double(lineS,pp);
168             lineEnd = new Line2D.Double(lineE, pp);
169             if (!IntersectionAll(arr, new Line2D.Double(lineS, lineS),
170                                  new Line2D.Double(pp,pp), lineStart)
171             && !IntersectionAll(arr, new Line2D.Double(lineEnd.getP1(),
172                     lineEnd.getP1()), new Line2D.Double(pp,pp), lineEnd)) {
173                 return pp;
174             }
175         }
176         for (double i = 0; i < a; i+=0.1){
177             pp = new Point2D.Double(i,0);
178             lineStart = new Line2D.Double(lineS,pp);
179             lineEnd = new Line2D.Double(lineE, pp);
180             if (!IntersectionAll(arr, new Line2D.Double(lineS, lineS),
181                                  new Line2D.Double(pp,pp), lineStart)
182                 && !IntersectionAll(arr, new Line2D.Double(lineEnd.getP1(),
183                     lineEnd.getP1()), new Line2D.Double(pp,pp), lineEnd)) {
184                 return pp;
185             }
186         }
187         lineStart = new Line2D.Double(lineS,lineE);lineEnd =
188                                   new Line2D.Double(lineE,lineS);
189         for (double i = 0; i < a; i+=0.1){
190             pp = new Point2D.Double(i,b);
191             lineStart = new Line2D.Double(lineS,pp);
192             lineEnd = new Line2D.Double(lineE, pp);
193             if (!IntersectionAll(arr, new Line2D.Double(lineS, lineS),
194                                  new Line2D.Double(pp,pp), lineStart)
195                     && !IntersectionAll(arr, new Line2D.Double(lineEnd.getP1(),
196                         lineEnd.getP1()), new Line2D.Double(pp,pp), lineEnd)) {
197                 return pp;
198             }
199         }
200         lineStart = new Line2D.Double(lineS,lineE);lineEnd =
201                                   new Line2D.Double(lineE,lineS);
202         for (double i = 0; i < b; i+=0.1){
203             pp = new Point2D.Double(a,i);
204             lineStart = new Line2D.Double(lineS,pp);
```

```
205             lineEnd = new Line2D.Double(lineE, pp);
206             if (!IntersectionAll(arr, new Line2D.Double(lineS, lineS),
207                                     new Line2D.Double(pp,pp), lineStart)
208                     && !IntersectionAll(arr, new Line2D.Double(lineEnd.getP1(),
209                         lineEnd.getP1()), new Line2D.Double(pp,pp), lineEnd)) {
210                 return pp;
211             }
212         }
213         return null;
214     }
215     public static String[][] path_point;
216     public static void main(String[] args) throws Exception {
217         Locale.setDefault(Locale.US);
218         Scanner in = new Scanner(System.in);
219         int n = in.nextInt();
220         int a = in.nextInt(), b = in.nextInt();
221         Point2D.Double Start = new Point2D.Double(in.nextDouble(),
222 
223         Point2D.Double End = new Point2D.Double(in.nextDouble(),
224 
225         Point2D.Double[][] arr = new Point2D.Double[n+1][];
226         path_point = new String[arr.length][];
227         for (int i = 0; i < n; i++){
228             int m = in.nextInt();
229             arr[i] = new Point2D.Double[m];
230             path_point[i] = new String[m];
231             for (int j = 0 ; j < m; j++) {
232                 arr[i][j] = new Point2D.Double(in.nextDouble(),
233 
234                 path_point[i][j] = "";
235             }
236         }
237         arr[n] = new Point2D.Double[1];
238         arr[n][0] = End;
239         path_point[n] = new String[1];
240         path_point[n][0] = "";
241         Path(arr,"",Start,End);
242         String del = path_point[n][0].split("\n")[0];
243         path_point[n][0] = path_point[n][0].replaceFirst(del + "\n","");
244         if (path_point[n][0].equals("")){
245             System.out.println(0);
246         }else {
247             Point2D.Double pointcheck = Check(arr,Start,End,a,b);
248             if (pointcheck == null || (pointcheck.x < 0 && pointcheck.x > a)
249                             || (pointcheck.y < 0 && pointcheck.y > b) ){
250                 System.out.println(path_point[n][0].split("\n").length);
251                 System.out.print(path_point[n][0]);
252             }else{
253                 System.out.println(1);
254                 System.out.println(pointcheck.x + " " + pointcheck.y);
255             }
256          }
257     }
258 }
```

Задача 6.3.8. Распознавание ARTag маркера (10 баллов)

Для определения своего местоположения квадрокоптер использует камеру, снимающую поверхность, над которой перемещается робот. Изображение с камеры при-

ходит в управляющую программу в видео набора $N \times M$ точек, где каждая точка закодирована в RGB-формате.

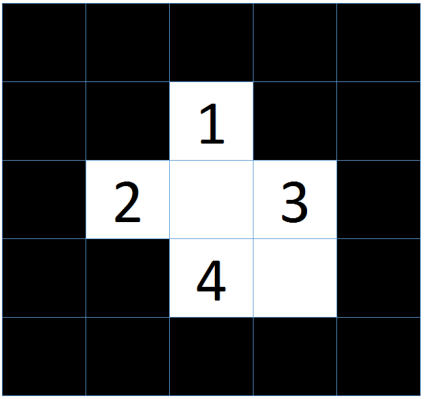


Рис. 6.23: Нумерация битов в маркера

На поверхность нанесены ARTag маркеры `https://inside.mines.edu/~whoff/courses/EENG512/lectures/other/ARTag.pdf`. Элементы маркера, 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 четное, то он черный. Оставшиеся 4 элемента маркера кодируют число по следующему правилу: если элемент черный, то в он обозначает 1, если белый, то 0 при этом самый первый элемент — старший бит закодированного числа. Элементы пронумерованы сверху вниз, слева направо.

Например, на маркере с рис. 6.24 закодировано число 0011_2, что эквивалентно 3_{10}.

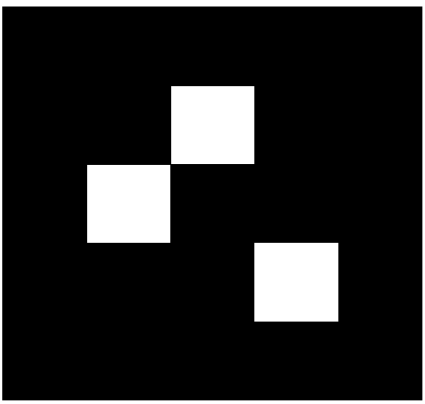


Рис. 6.24: Маркер с закодированным значением - 0011_2

Поскольку квадрокоптер не перемещается постоянно параллельно поверхности, то изображения ARTag маркера, получаемые с камеры, получаются в виде неправильного выпуклого четырехугольника, а непостоянные условия освещенности изменяют фокус и тон изображения.

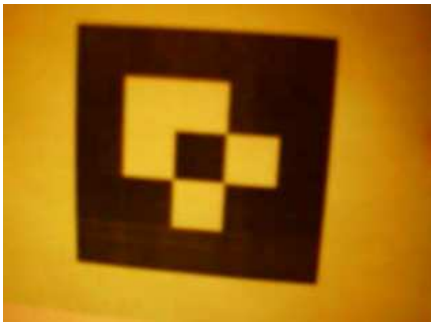


Рис. 6.25: Пример маркера

Поскольку направление запуска квадрокоптера заранее неизвестно, то ориентация маркеров заранее неизвестна, но его изображение таково, что оно по каждой из осей X, Y, Z относительно оптической оси камеры не превышает 25 градусов

Напишите программу для определения закодированного в маркере числа.

Формат входных данных

Первая строка входного файла содержит два целых числа, разделенные пробелом: N и M ($0 \leqslant N, M \leqslant 320$), определяющие размер кадра.

Далее идет M строк, содержащих N триплетов вида $\#RRGGBB$, где RR - шестнадцатеричное число R составляющей данного элемента матрицы, GG и BB соответственно шестнадцатеричные числа G и B составляющих.

Формат выходных данных

Одно целое число в десятеричной системе счисления — число, закодированное на маркере.

Примеры

Наборы входных данных доступны здесь: `https://drive.google.com/drive/folders/1UOIXYAyd9G_5CES-ePKW_2APlY1LRPzQ`

Решение

Для начала нам нужно перевести изображение в Ч/Б матрицу (бинарный двумерный массив, где 0 – белое, 1 – чёрное), для этого как вариант можно использовать один из самых простых способов, мы записываем в матрицу 1 если сумма всех трёх

компонентов(RGB) меньше какого-то среднего значение, например 120 подходит для всех примеров, следовательно можем сделать вывод, что с большой вероятностью оно подойдёт и для остальных тестов. Далее нам необходимо найти углы маркера, тем самым мы определяем границы маркера, после чего мы делим полученный четырёхугольник на 25 частей, как показано на рисунке

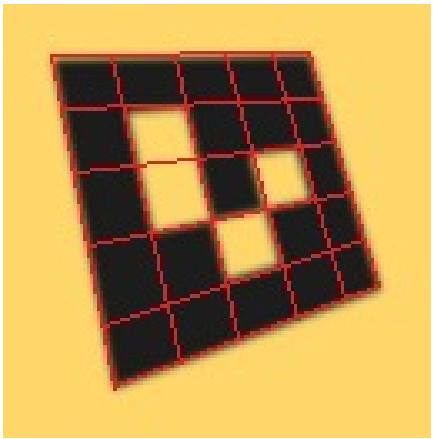


Рис. 6.26: Разметка маркера

Для разметки маркера нам нужнопосторить 12 прямых, каждая прямая задаётся функцией $y = kx + b$. Самыми первыми строятся контурные линии так как нам известны 2 точки на прямой мы можем вычесли k и b по формулам:

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

$$b = y - kx$$

После вычисления первых четырёх прямых, мы каждую из них делим на 5 равных частей и берём эти точки попарно, что бы найти оставшиеся 8 уравнений. После нахождения коэффициентов всех прямых, нам открывается возможность определять, лежит ли данная точка в данном секторе, если выполняется условие

$$f_l(x) < y < f_r(x)$$

$$f_d(x) < y < f_u(x)$$

Где

- y - координата y точки
- x - координата x точки
- $f_l(x)$ - функция девой границы
- $f_r(x)$ - функция правой границы
- $f_d(x)$ - функция нижний границы
- $f_u(x)$ - функция верхней границы

Далее мы суммируем все пиксили внутри красных четырёхугольниках, и записываем в новую матрицу 25 на 25 1 если чёрных пиксилей больше чем белых и 0 если это не так. После чего нам остаётся определить ориентацию путём нахождения специального бита (в маркере нижний правый угол всегда белого цвета, в то время как все остальные углы чёрного цвета), дальше просто расшифровываем по алгоритму указанному в условие задачи.

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <cmath>
#include <stdio.h>

using namespace std;

float PI = 3.14159265;
float PIdiv2 = PI / 2;
float a, blueColor, c;
float xi, yi;

char hexValueColor[8];
int redColor, greenColor, blueColor;
int grayColor = 100;
int N, M;

bool binaryImg[1000][1000];
int borders[1000][1000];
int bordersPoints[100000][2];
int positionCount = 0;
int xMin, yMin, xMax, yMax;
int xMin2, yMin2, xMax2, yMax2;
int xP, yP, xC, yC;
int cornersCoodinates[4][2];
int NumberOfCordes[9][2];
bool matrix[3][3];
int decodeValue = 0;

void RGBRead()
{
    char rhex[] = { hexValueColor[1],hexValueColor[2], };
    char ghex[] = { hexValueColor[3],hexValueColor[4], };
    char bhex[] = { hexValueColor[5],hexValueColor[6], };

    sscanf(rhex, "%x", &redColor);
    sscanf(ghex, "%x", &greenColor);
    sscanf(bhex, "%x", &blueColor);
}

void calculateCoefficients(float x1, float y1, float x2, float y2)
{
    a = y2 - y1;
    blueColor = x1 - x2;
    c = y1*x2 - y2*x1;
}

int solveEquation(float a2, float b2, float c2)
```

```
{
    if (a*b2 - a2*blueColor == 0)
        return 1;
    xi = (c*b2 - c2*blueColor) / (a*b2 - a2*blueColor)*-1;
    yi = (a*c2 - a2*c) / (a*b2 - a2*blueColor)*-1;
    return 0;
}

void findBoarder()
{
    for (int ia = 0; ia < N; ia++)
    {
        for (int ib = 0; ib < M; ib++)
        {
            if (binaryImg[ia][ib] == 1)
            {
                if (ia > yMax)
                {
                    yMax = ia;
                    yMax2 = ib;
                }
                if (ia < yMin)
                {
                    yMin = ia;
                    yMin2 = ib;
                }
                if (ib > xMax)
                {
                    xMax = ib;
                    xMax2 = ia;
                }
                if (ib < xMin)
                {
                    xMin = ib;
                    xMin2 = ia;
                }
                if (borders[ia][ib] != 1)
                {
                    borders[ia][ib] = 1;

                    bordersPoints[positionCount][0] = ib;
                    bordersPoints[positionCount][1] = ia;
                    positionCount++;
                }

                break;
            }
        }
    }

    for (int ia = 0; ia < N; ia++)
    {
        for (int ib = M - 1; ib >= 0; ib--)
        {
            if (binaryImg[ia][ib] == 1)
            {
                if (ia > yMax)
                {
                    yMax = ia;
                    yMax2 = ib;
```

```
                }
                if (ia < yMin)
                {
                    yMin = ia;
                    yMin2 = ib;
                }
                if (ib > xMax)
                {
                    xMax = ib;
                    xMax2 = ia;
                }
                if (ib < xMin)
                {
                    xMin = ib;
                    xMin2 = ia;
                }
                if (borders[ia][ib] != 1)
                {
                    borders[ia][ib] = 1;

                    bordersPoints[positionCount][0] = ib;
                    bordersPoints[positionCount][1] = ia;
                    positionCount++;
                }

                break;
            }
        }
    }

    for (int ib = 0; ib < N; ib++)
    {
        for (int ia = M - 1; ia >= 0; ia--)
        {
            if (binaryImg[ia][ib] == 1)
            {
                if (ia > yMax)
                {
                    yMax = ia;
                    yMax2 = ib;
                }
                if (ia < yMin)
                {
                    yMin = ia;
                    yMin2 = ib;
                }
                if (ib > xMax)
                {
                    xMax = ib;
                    xMax2 = ia;
                }
                if (ib < xMin)
                {
                    xMin = ib;
                    xMin2 = ia;
                }
                if (borders[ia][ib] != 1)
                {
                    borders[ia][ib] = 1;

```

```
                    bordersPoints[positionCount][0] = ib;
                    bordersPoints[positionCount][1] = ia;
                    positionCount++;
                }
                break;
            }
        }
    }

    for (int ib = 0; ib < N; ib++)
    {
        for (int ia = 0; ia < M; ia++)
        {
            if (binaryImg[ia][ib] == 1)
            {
                if (ia > yMax)
                {
                    yMax = ia;
                    yMax2 = ib;
                }
                if (ia < yMin)
                {
                    yMin = ia;
                    yMin2 = ib;
                }
                if (ib > xMax)
                {
                    xMax = ib;
                    xMax2 = ia;
                }
                if (ib < xMin)
                {
                    xMin = ib;
                    xMin2 = ia;
                }
                if (borders[ia][ib] != 1)
                {
                    borders[ia][ib] = 1;

                    bordersPoints[positionCount][0] = ib;
                    bordersPoints[positionCount][1] = ia;
                    positionCount++;
                }

                break;
            }
        }
    }
}

int calculetePoints()
{
    if (binaryImg[yMin][xMin] == 1)
    {
        xP = xMin;
        yP = yMin;
        return 0;
    }
    if (binaryImg[yMin][xMax] == 1)
    {
```

```
228             xP = xMax;
229             yP = yMin;
230             return 0;
231         }
232         if (binaryImg[yMax][xMin] == 1)
233         {
234             xP = xMin;
235             yP = yMax;
236             return 0;
237         }
238         if (binaryImg[yMax][xMax] == 1)
239         {
240             xP = xMax;
241             yP = yMax;
242             return 0;
243         }
244         if (binaryImg[yMax][yMax2] == 1)
245         {
246             xP = yMax2;
247             yP = yMax;
248             return 0;
249         }
250         if (binaryImg[yMin][yMin2] == 1)
251         {
252             xP = yMin2;
253             yP = yMin;
254             return 0;
255         }
256         if (binaryImg[xMax2][xMax] == 1)
257         {
258             xP = xMax;
259             yP = xMax2;
260             return 0;
261         }
262         if (binaryImg[xMin2][xMin] == 1)
263         {
264             xP = xMin;
265             yP = xMin2;
266             return 0;
267         }
268         return 1;
269     }
270
271     int findCorners()
272     {
273         cornersCoodinates[0][0] = xP;
274         cornersCoodinates[0][1] = yP;
275
276         float dist = sqrt(pow(xC - xP, 2) + pow(yC - yP, 2));
277
278         double cosx;
279         double sinx;
280         float a;
281
282         if (dist != 0)
283         {
284             cosx = (yC - xP) / dist;
285             sinx = (xC - yP) / dist;
286         }
287         else
```

```
288             return -2;
289
290         if (sinx >= 0)
291             a = acos(cosx);
292         else
293             a = 2 * PI - acos(cosx);
294
295         for (int i = 0; i < 4; i++)
296             {
297             float maxgyp = 0;
298             int mdotx = 0, mdoty = 0;
299             float x1, y1, x2, y2;
300             x1 = xC;
301             y1 = yC;
302             x2 = xC + 10 * cos(a);
303             y2 = yC + 10 * sin(a);
304
305             calculateCoefficients(x1, y1, x2, y2);
306
307             for (int ia = 0; ia < positionCount; ia++)
308             {
309                 int xd, yd;
310                 xd = bordersPoints[ia][0];
311                 yd = bordersPoints[ia][1];
312
313                 float dist = sqrt(pow(xC - xd, 2) + pow(yC - yd, 2));
314
315                 double cosx;
316                 double sinx;
317                 float ad;
318
319                 if (dist != 0)
320                 {
321                     cosx = (yd - yC) / dist;
322                     sinx = (xd - xC) / dist;
323                 }
324                 else
325                     return -2;
326
327                 if (sinx >= 0)
328                     ad = acos(cosx);
329                 else
330                     ad = 2 * PI - acos(cosx);
331
332                 float dx = ad - a;
333
334                 if (sin(dx) < 0)
335                 {
336                     solveEquation(0, 1, (yd*-1));
337                     float gyp = sqrt(pow(xi - xd, 2) + pow(yi - yd, 2));
338
339                     if (gyp > maxgyp)
340                     {
341                         maxgyp = gyp;
342                         mdotx = xd;
343                         mdoty = yd;
344                     }
345                 }
346             }
347
```

```
348             cornersCoodinates[i][0] = mdotx;
349             cornersCoodinates[i][1] = mdoty;
350 
351             a = a + PIdiv2;
352         }
353     }
354 
355     void GetNumberOfCoords()
356     {
357         int realcenterx, realcentery;
358         float a2, b2, c2;
359 
360         calculateCoefficients(cornersCoodinates[0][0],
361                 cornersCoodinates[0][1], cornersCoodinates[2][0],
362                 cornersCoodinates[2][1]);
363         a2 = a; b2 = blueColor; c2 = c;
364         calculateCoefficients(cornersCoodinates[1][0],
365                 cornersCoodinates[1][1], cornersCoodinates[3][0],
366                 cornersCoodinates[3][1]);
367 
368         solveEquation(a2, b2, c2);
369         realcenterx = xi; realcentery = yi;
370 
371         int CucQuad[4][2];
372         for (int i = 0; i < 4; i++)
373         {
374             CucQuad[i][0] = (cornersCoodinates[i][0] - realcenterx) / 2.4 + realcenterx;
375             CucQuad[i][1] = (cornersCoodinates[i][1] - realcentery) / 2.4 + realcentery;
376         }
377 
378         NumberOfCordes[2][0] = CucQuad[0][0];
379         NumberOfCordes[2][1] = CucQuad[0][1];
380         NumberOfCordes[1][0] = (CucQuad[0][0] + CucQuad[1][0]) / 2;
381         NumberOfCordes[1][1] = (CucQuad[0][1] + CucQuad[1][1]) / 2;
382 
383         NumberOfCordes[0][0] = CucQuad[1][0];
384         NumberOfCordes[0][1] = CucQuad[1][1];
385         NumberOfCordes[3][0] = (CucQuad[1][0] + CucQuad[2][0]) / 2;
386         NumberOfCordes[3][1] = (CucQuad[1][1] + CucQuad[2][1]) / 2;
387 
388         NumberOfCordes[6][0] = CucQuad[2][0];
389         NumberOfCordes[6][1] = CucQuad[2][1];
390         NumberOfCordes[7][0] = (CucQuad[2][0] + CucQuad[3][0]) / 2;
391         NumberOfCordes[7][1] = (CucQuad[2][1] + CucQuad[3][1]) / 2;
392 
393         NumberOfCordes[8][0] = CucQuad[3][0];
394         NumberOfCordes[8][1] = CucQuad[3][1];
395         NumberOfCordes[5][0] = (CucQuad[3][0] + CucQuad[0][0]) / 2;
396         NumberOfCordes[5][1] = (CucQuad[3][1] + CucQuad[0][1]) / 2;
397 
398         NumberOfCordes[4][0] = realcenterx;
399         NumberOfCordes[4][1] = realcentery;
400     }
401 
402 
403     void decode()
404     {
405         int icuc = 0;
406 
407         for (int iy = 0; iy<3; iy++)
```

```
408     {
409         for (int ix = 0; ix<3; ix++)
410         {
411             matrix[ix][iy] = binaryImg[NumberOfCordes[icuc][1]][NumberOfCordes[icuc][0]];
412             icuc++;
413         }
414     }
415 
416     if (matrix[2][2] == 0)
417     {
418     bool code[4] = { matrix[1][2],matrix[2][1],matrix[0][1],matrix[1][0], };
419         for (int i = 0; i<4; i++)
420         {
421             decodeValue += pow(2, i)*code[i];
422         }
423     }
424     if (matrix[0][2] == 0)
425     {
426     bool code[4] = { matrix[0][1],matrix[1][2],matrix[1][0],matrix[2][1], };
427         for (int i = 0; i<4; i++)
428         {
429             decodeValue += pow(2, i)*code[i];
430         }
431     }
432     if (matrix[0][0] == 0)
433     {
434     bool code[4] = { matrix[1][0],matrix[0][1],matrix[2][1],matrix[1][2], };
435         for (int i = 0; i<4; i++)
436         {
437             decodeValue += pow(2, i)*code[i];
438         }
439     }
440     if (matrix[2][0] == 0)
441     {
442     bool code[4] = { matrix[2][1],matrix[1][0],matrix[1][2],matrix[0][1], };
443         for (int i = 0; i<4; i++)
444         {
445             decodeValue += pow(2, i)*code[i];
446         }
447     }
448 }
449 
450 
451 int main()
452 {
453     cin >> M >> N;
454 
455     for (int ia = 0; ia < N; ia++)
456     {
457         for (int ib = 0; ib < M; ib++)
458         {
459         cin >> hexValueColor; RGBRead();
460             if (redColor < grayColor && greenColor < grayColor && blueColor < grayColor)
461             {
462             binaryImg[ia][ib] = 1;
463             xMin = ib; xMax = ib; xMin2 = ia; xMax2 = ia;
464             yMin = ia; yMax = ia; yMin2 = ib; yMax2 = ib;
465             }
466         else
467             binaryImg[ia][ib] = 0;
```

```
468             }
469         }
470
471         findBoarder();
472         if (calculetePoints() == 1) cout << "[ ! ] getPoints() failed\n";
473
474         xC = (xMax + xMin) / 2;
475         yC = (yMax + yMin) / 2;
476
477         findCorners();
478         GetNumberOfCoords();
479
480         decode();
481
482         cout << decodeValue << endl;
483     }
```

Критерии определения призеров и победителей второго этапа

Количество баллов, набранных при решении всех задач суммируется. Призерам второго отборочного этапа было необходимо набрать 35 баллов (для участников из 9 класса) и 45 (для участников из 10-11 класса). Победители второго отборочного этапа должны были набрать 63 балла и выше.

3. ФИНАЛЬНЫЙ ЭТАП

Заключительный этап олимпиады состоит из двух частей: индивидуальное решение задач по предметам (математика, информатика) и командное решение инженерной задачи.

Задачи индивидуального тура

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 каждая задача делилась на 3 подзадачи таким образом, что решение каждой подзадачи подводило к решению следующей. 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 группа 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.

8.1. Задачи по математике (9 класс)

Задача 8.1.1. (20 баллов)

Саша едет на велосипеде по ровному участку дороги и хочет измерит свою скорость. Он делает 56 оборотов в минуту педалями (легко измерить при помощи часов с секундомером). Еще он знает, что диаметры колес 66 сантиметров, большая звездочка имеет 45 зубьев, а маленькая — 16 зубьев. Найдите скорость Саши в км/час. Результат округлите до целого числа.

Решение

Один оборот заднего колеса соответствует одному обороту малой звездочки и $\frac{16}{45}$ оборота большой звездочки. За это время Саша сдвигается на расстояние, равное длине окружности колеса, т.е. 66π см. Угловая скорость заднего колеса равна $\omega = \frac{56 \cdot 45}{16}$ об/мин. Тогда скорость велосипеда равна $v = \frac{56 \cdot 45 \cdot 66\pi}{16}$ см/мин= $6,237\pi$ км/час$\approx$ 20 км/час.

Дополнительные критерии оценки

Отсутствуют.

Задача 8.1.2. (30 баллов)

Робот манипулятор состоит из двух стержней длиной 1 м и управляется двумя осями в точках A и B. Предмет D в начальный момент находится в точке C на расстоянии 1 м от A. Требуется поднять его строго вертикально вверх управляя только углами α и β.

а) **(5 балл)** На какую максимальную высоту можно поднять предмет D при помощи этого манипулятора?

б) **(10 балла)** Найдите значения α и β в момент, когда $CD = 0,5$ м.

в) **(15 баллов)** Выразите α и β в виде функций от времени t так, чтобы предмет D поднимался с постоянной скоростью и достиг максимальной высоты в конце первой минуты.

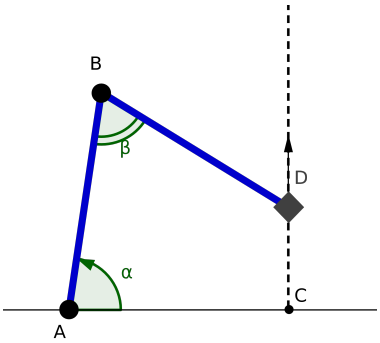
Решение

а) Пусть $h = CD$ высота предмета. По неравенству треугольника $AD \leqslant AB + BD = 2$, причем равенство достигается когда угол $\beta = 180°$. Тогда $h = \sqrt{AD^2 - 1^2} \leqslant \sqrt{3}$.

б) В решение пункта в) вместо $\sqrt{3}t$ поставьте 0,5.

в) Пусть $t \in [0; 1]$ время в минутах, $h(t) = \sqrt{3}t$. По теореме Пифагора $AD^2 = 3t^2 + 1$. Из треугольника ABD по теореме косинусов $AD^2 = AB^2 + BD^2 - 2AB \cdot BD \cos\beta$, откуда $\cos\beta = \frac{1-3t^2}{2}$, $\beta(t) = \arccos(\frac{1-3t^2}{2})$. $\alpha = \angle CAD + \angle DAB$, $\angle CAD = \operatorname{arctg}(\sqrt{3}t)$, $\angle DAB = \arccos\frac{AD}{2} = \arccos\frac{\sqrt{3t^2+1}}{2}$.

https://ggbm.at/wpMJhsXm



Дополнительные критерии оценки

Если пункт в) решен, то за б) тоже начисляются баллы.

Задача 8.1.3. (50 баллов)

Имеется квадратное поле $PQRS$ 100см×100см. В вершинах P и Q размещены датчики, которые каждую секунду измеряют квадрат расстояния до движущегося объекта в точке M_t. Известно, что объект движется прямолинейно с постоянным ускорением и не меняет направления движения. Обозначим $p_t = PM_t^2$ и $q_t = QM_t^2$. Время t измеряется в секундах. Результаты трех измерений приведены в таблице.

t	0	1	2
p_t	1625	3809	6929
q_t	3625	4409	6529

а) **(10 баллов)** Введите систему координат с началом в точке P, осью абсцисс по направлению луча PS и осью ординат по направлению луча PQ (единица измерения в сантиметрах). Найдите координаты точек M_0, M_1, M_2 в этой системе координат.

б) **(20 баллов)** Какие значения p_3 и q_3 покажут датчики в момент времени $t = 3$с ?

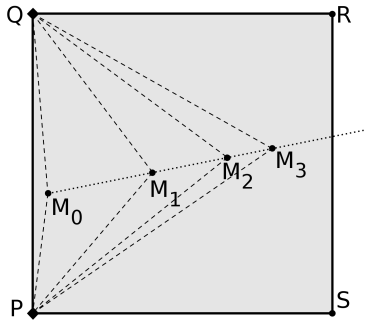
в) **(20 баллов)** Выясните, в какой момент времени остановится объект и покинет ли он поле $PQRS$.

Решение

Для точки $M_t(x_t; y_t)$ выполняются равенства (1) $\begin{cases} x_t^2 + y_t^2 = p_t; \\ (100 - x_t)^2 + y_t^2 = q_t. \end{cases}$

а) Подставляем в систему (1) вместо t значения 0, 1, 2 и, решая систему, находим координаты точек $M_0(5; 40)$, $M_1(40; 47)$, $M_2(65; 52)$.

б) По условию $\begin{cases} x_t = x_0 + v_x t + \frac{a_x}{2} t^2; \\ y_t = y_0 + v_y t + \frac{a_y}{2} t^2. \end{cases}$ Подставляем известные величины при $t =$ 1 и $t = 2$, решаем систему линейных уравнений относительно координат скорости и



ускорения тела. Получаем зависимость координат тела от времени (2) $\begin{cases} x_t = 5 + 40t - 5t^2; \\ y_t = 40 + 8t - t^2. \end{cases}$ Подстановкой $t = 3$ в (2) и (1) находим $M_3(80; 55)$, $p_3 = 9425$, $q_3 = 8425$.

в) Скорость тела, движущегося равноускоренно по правилам описанным в пункте б), можно описать формулами: $\vec{v}\{40 - 10t; 8 - 2t\}$. Направление движения не менялось, поэтому координаты вектора скорости не меняли знака. В момент $t = 4$ тело остановилось. Его координаты в этот момент $M_4(85; 56)$ — точка внутри поля.

Дополнительные критерии оценки

Отсутствуют.

8.2. Задачи по математике (10-11 класс)

Задача 8.2.1. (20 баллов)

Можно ли многочлен $x^4 + x^3 + 2x^2 + 9x + 15$ представить в виде произведения двух многочленов ненулевой степени с целыми коэффициентами?

Решение

Можно $(x^2-2x+5)(x^2+3x+3)$. Разложение ищем в виде $(x^2+ax+b)(x^2+cx+d)$, где неизвестные коэффициенты a, b, c, d — целые числа. $bd = 15$, перебираем различные варианты (всего 4 случая), для которых коэффициенты a и c находим из условий: $a + c = 1$, $ac + b + d = 2$, $ad + bc = 9$.

Дополнительные критерии оценки

За ответ «Можно» 0 баллов. Пример разложения 20 баллов.

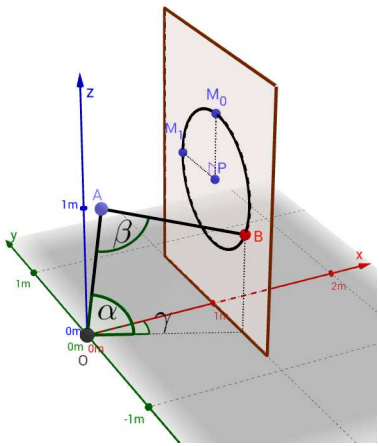
Задача 8.2.2. (30 баллов)

Роботу манипулятору предстоит выполнить плазменную резку в листе стали в виде окружности диаметром 1 м. Тело манипулятора состоит из двух стержней OA и AB длиной 1 м и управляется двумя осями в точке O и еще одной в точке A. Резак находится в конце B. Угол α отвечает за отклонение стержня OA от плоскости Oxy, β — угол между стержнями OA и AB, γ — поворот вокруг оси Oz (отсчитывается от оси Ox против хода часовой стрелки). Лист стали отстоит на расстоянии 1 м от точки O перпендикулярно оси Ox, центр окружности резки в точке $P(1;0;1)$. Резка начинается в точке M_0 и совершается против направления часовой стрелки.

а) **(5 балла)** Найдите значения углов α, β и γ в начальный момент резки, т.е. когда резак находится в точке $M_0(1;0;1,5)$.

б) **(10 балла)** Найдите значения углов α, β и γ в момент резки, когда резак находится в точке M_1.

в) **(15 баллов)** Выразите α, β и γ в виде функций от времени t так, чтобы манипулятор завершил резку за 4 минуты двигаясь по контуру с постоянной скоростью.



Решение

а) $\alpha = \arccos\dfrac{\sqrt{13}}{4} + \operatorname{arctg} 1,5$; $\beta = \arccos(-0,625)$; $\gamma = 0$.

б) $\alpha(1) = \dfrac{1}{2}\arccos\left(\dfrac{1}{8}\right) + \arcsin\left(\dfrac{2}{3}\right)$, $\beta(1) = \arccos\left(-\dfrac{1}{8}\right)$, $\gamma(1) = \operatorname{arctg}\left(\dfrac{1}{2}\right)$.

в) Зададим точку $M = M(t)$, которая равномерно вращается по окружности резки с периодом 4 минуты. Её можно задать уравнениями

$$M: \begin{cases} x(t) = 1; \\ y(t) = \dfrac{1}{2}\sin\dfrac{\pi t}{2}; \\ z(t) = 1 + \dfrac{1}{2}\cos\dfrac{\pi t}{2}. \end{cases}$$

Будем искать условия на α, β, γ такие, чтобы точка B совпадала с M. По определению $OM^2 = x^2 + y^2 + z^2 = 2,25 + \cos\dfrac{\pi t}{2}$. По теореме косинусов из треугольника AOB получаем условие на β: $OB^2 = 2 - 2\cos\beta = 2,25 + \cos\dfrac{\pi t}{2}$. Откуда $\cos\beta = -\dfrac{1}{8} - \dfrac{1}{2}\cos(\dfrac{\pi t}{2})$. Обозначим N проекцию точки M на плоскость Oxy. Тогда $\angle NOx = \gamma = \operatorname{arctg}\dfrac{y}{x} = \operatorname{arctg}\left(\dfrac{1}{2}\sin(\dfrac{\pi t}{2})\right)$. $\alpha = \angle MON + \angle AOB = \arcsin\dfrac{z}{OM} + \dfrac{1}{2}(180^\circ - \beta) = \arcsin\left(\dfrac{2 + \cos(\frac{\pi t}{2})}{\sqrt{9 + 4\cos(\frac{\pi t}{2})}}\right) + \dfrac{1}{2}\arccos\left(\dfrac{1}{8} + \dfrac{1}{2}\cos(\dfrac{\pi t}{2})\right)$.

Решение задачи по ссылке

https://ggbm.at/zZUZYByv

Дополнительные критерии оценки

Отсутствуют.

Задача 8.2.3. (50 баллов)

Имеется квадратное поле $ABCD$ 20м×20м. В вершинах A и B размещены датчики, которые каждую секунду выдают квадрат расстояния до движущегося объекта в точке M_t. Известно, что объект движется по окружности с постоянной угловой скоростью. Обозначим $p_t = AM_t^2$ и $q_t = BM_t^2$. Время t измеряется в секундах. Результаты трех измерений приведены в таблице.

t	0	1	2
p_t	320	346	317,6
q_t	80	146	221,6

а) **(10 баллов)** Введите систему координат с началом в точке A, осью абсцисс по направлению луча AB и осью ординат по направлению луча AD (единица измерения в метрах). Найдите координаты точек M_0, M_1, M_2 в этой системе координат.

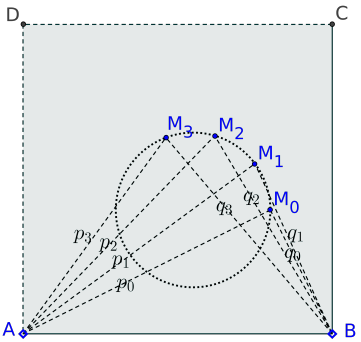
б) **(20 баллов)** Найдите уравнение окружности, по которой движется точка M_t.

в) **(20 баллов)** Какие значения p_3 и q_3 выдадут датчики в момент времени $t = 3$с ?

Решение

а) Для точки $M_t(x_t; y_t)$ выполняются равенства (1) $\begin{cases} x_t^2 + y_t^2 = p_t; \\ (20 - x_t)^2 + y_t^2 = q_t. \end{cases}$ Откуда получаем (2) $\begin{cases} x_t = 10 - \dfrac{p_t - q_t}{40}; \\ y_t = \sqrt{p_t - x_t^2}. \end{cases}$ Подставляя значения из таблицы приходим к ответу.

б) Общий вид уравнения окружности с центром в точке $O(a; b)$ и радиусом r: $(x - a)^2 + (y - b)^2 = r^2$. Вместо x и y подставляя координаты точек M_0, M_1, M_2 получаем



систему уравнений с тремя неизвестными, решением которой является тройка $a = 11$, $b = 8$, $r = 5$.

в) Пусть угловая скорость объекта равна α. В момент времени t угол M_0OM_t равняется αt, $\cos\alpha = \cos\angle M_0OM_1 = \frac{4}{5}$ по теореме косинусов. Тогда вектор $\overrightarrow{OM_3}$ имеет координаты $\{5\cos 3\alpha; 5\sin 3\alpha\}$; $\cos 3\alpha = 4\cos^3\alpha - 3\cos\alpha = -0,352$, $\sin 3\alpha = 3\sin\alpha - 4\sin^3\alpha = 0,936$. $M_3 = (11+5\cdot(-0,352); 8+5\cdot 0,936) = (9,24; 12,68)$ подставляя в формулы (1) получаем ответ.

Дополнительные критерии оценки

Отсутствуют.

8.3. Задачи по информатике

Задача 8.3.1. Траектории (15 баллов)

Уборка - дело важное, вот только никому не нравится ей заниматься. И Сереже тоже! Но ему повезло - недавно его мама подарила ему набор из большого количества роботов, которые могут делать за Сережу работу по дому.

Однако, радость Сережи была не долгой. Он совершенно не представляет, как ими пользоваться. Меню настроек - сложное, а инструкция - на английском.

Серёжа знает, что вы увлекаетесь робототехникой, и обратился к вам за помощью.

Через пару часов началось тестирование: роботы носились по всей квартире. Бесцельно, но весело.

Оказалось, что каждый робот двигается по заранее определенной (запрограммированной по умолчанию) траектории. Траектория движения каждого из роботов представляет собой ломаную. А сами роботы имеют цилиндрическую форму и убирают весь мусор, который окажется под ними в какой-либо момент времени.

Для начала Сережа поручил вам определить пересекаются ли траектории двух роботов.

Формат входных данных

В первой строке два целых числа:

- $2 \leqslant n \leqslant 500$ - количество точек в траектории первого робота;
- $2 \leqslant m \leqslant 500$ - количество точек в траектории второго робота.

В следующих двух строках заданы точки ($2 \cdot n$ целых чисел в первой строке и $2 \cdot m$ целых чисел во второй строке): $x_{i,j}$ $y_{i,j}$ — j-я точка траектории i-го робота ($|x_{i,j}| \leqslant 10^9$; $|y_{i,j}| \leqslant 10^9$).

Никакие две последовательные точки никакой из траекторий не совпадают.

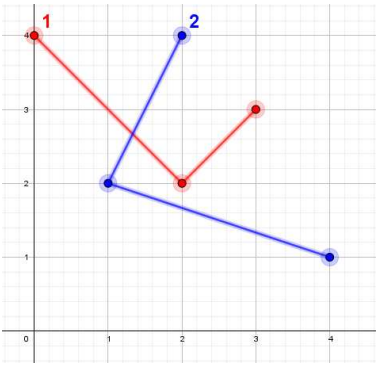


Рис. 8.1: Рисунок к первому примеру

Формат выходных данных

Выведите *Yes* , если траектории пересекаются, иначе - *No*.

Примеры

Пример №1

Стандартный ввод
3 3 0 4 2 2 3 3 2 4 1 2 4 1
Стандартный вывод
Yes

Пример №2

Стандартный ввод
2 2 0 0 1 1000000000 1 999999999 2 999999999
Стандартный вывод
No

Способ оценки работы

За решение задачи начислялось:

- **5 баллов**, если пройдена только первая группа тестов;
- **10 баллов**, если пройдена первая и вторая группы тестов;
- **15 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    return reply.strip() == clue.strip()

x = []
y = []

# triangle area
def area(a,b,c):
    s = (x[b] - x[a]) * (y[c] - y[a]) - (y[b] - y[a]) * (x[c] - x[a])
    return -1 if s < 0 else (1 if s > 0 else 0)

# bounding box
def box(a, b, c, d):
    return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))

# пересечение отрезков
def intersect(a, b, c, d):
    return box(x[a],x[b],x[c],x[d]) &
           box(y[a],y[b],y[c],y[d]) &
           (area(a, b, c) * area(a, b, d) <= 0) &
           (area(c, d, a) * area(c, d, b) <= 0)

def solve(dataset):
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    s = ds[1].split()
    for i in range(n):
        x.append(int(s[i*2]))
        y.append(int(s[i*2+1]))

    s = ds[2].split()
    for i in range(m):
        x.append(int(s[i * 2]))
        y.append(int(s[i * 2 + 1]))

    inter = False

    for i in range(n - 1):
        for j in range(m - 1):
            inter |= intersect(i, i + 1, n + j, n + j + 1)

    return "Yes" if inter else "No"
```

Решение

В данной задаче достаточно проверить пересекается ли хотя бы один отрезок первой траектории с хотя бы одним отрезком второй траектории.

Асимптотика: $O(n \cdot m)$

Пример программы

Ниже представлено решение на языке Python3

```
x = []
y = []

# triangle area
def area(a,b,c):
    s = (x[b] - x[a]) * (y[c] - y[a]) - (y[b] - y[a]) * (x[c] - x[a])
    return -1 if s < 0 else (1 if s > 0 else 0)

# bounding box
def box(a, b, c, d):
    return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))


# пересечение отрезков
def intersect(a, b, c, d):
    return box(x[a],x[b],x[c],x[d]) & box(y[a],y[b],y[c],y[d]) \
    & (area(a, b, c) * area(a, b, d) <= 0) \
    & (area(c, d, a) * area(c, d, b) <= 0)

def solve(dataset):
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    s = ds[1].split()
    for i in range(n):
        x.append(int(s[i * 2]))
        y.append(int(s[i * 2 + 1]))

    s = ds[2].split()
    for i in range(m):
        x.append(int(s[i * 2]))
        y.append(int(s[i * 2 + 1]))

    inter = False

    for i in range(n - 1):
        for j in range(m - 1):
            inter |= intersect(i, i + 1, n + j, n + j + 1)

    return "Yes" if inter else "No"

print(solve(input() + '\n' + input() + '\n' + input()))
```

Задача 8.3.2. Радиус робота-уборщика (20 баллов)

Пока вы занимались проверкой пересечения траекторий, Серёжа не терял времени даром. Он разработал идеальную (по его мнению) траекторию движения робота уборщика. Учёл (по его мнению) все факторы: расположение мебели, обуви и т. д.

И теперь он просит вас рассчитать, какой должен быть минимальный радиус робота, чтобы он смог убрать весь мусор. При этом робот должен двигаться по траектории, разработанной Серёжей, а мусор - это набор из m материальных точек.

Формат входных данных

В первой строке два целых числа:

- $2 \leqslant n \leqslant 500$ - количество точек в траектории робота;
- $1 \leqslant m \leqslant 500$ - количество мусор-точек.

В следующей строке заданы точки ($2 \cdot n$ целых чисел): $x_i\ y_i$ — i-я точка траектории робота ($1 \leqslant i \leqslant n$; $|x_i| \leqslant 10^9$; $|y_i| \leqslant 10^9$).

В следующей строке заданы точки ($2 \cdot m$ целых чисел): $x_j\ y_j$ — j-я точка мусора ($1 \leqslant j \leqslant m$; $|x_j| \leqslant 10^9$; $|y_j| \leqslant 10^9$).

Никакие две последовательные точки траекторий не совпадают.

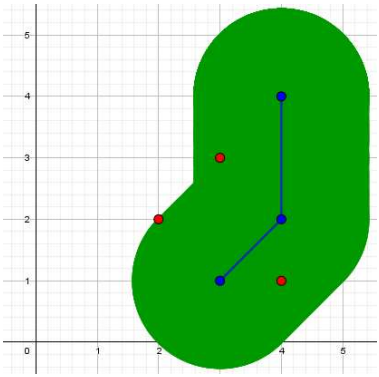


Рис. 8.2: Рисунок к первому примеру

Формат выходных данных

Одно неотрицательное число – минимальный радиус.

Ваш ответ будет засчитан, если его абсолютная или относительная ошибка не превосходит 10^{-6}. Формально, пусть ваш ответ равен a, а ответ жюри равен b. Ваш ответ будет засчитан, если $\frac{|a-b|}{\max(1,b)} \leqslant 10^{-6}$.

Примеры

Пример №1

Стандартный ввод
3 3 4 4 4 2 3 1 4 1 2 2 3 3
Стандартный вывод
1.4142135623730951

Пример №2

Стандартный ввод
2 1 0 0 1 1 1 1
Стандартный вывод
0.0

Способ оценки работы

За решение задачи начислялось:

- **5 баллов**, если пройдена только первая группа тестов;
- **10 баллов**, если пройдена первая и вторая группы тестов;
- **20 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    a = float(reply)
    b = float(clue)
    return abs(a - b) / max(1.0, b) <= 1e-6

from collections import namedtuple
from math import sqrt

Point = namedtuple("Point", ["x", "y"])

# расстояние между точками a и b
def dist(a, b):
        return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))

# расстояние между точкой c и отрезком ab
def dist2(a, b, c):
        A = a.y - b.y
        B = b.x - a.x
        C = b.y * a.x - a.y * b.x
        cc = B * c.x - A * c.y
        d = A * A + B * B
        p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)

        return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7 else
        min(dist(c, a), dist(c, b))
def solve(dataset):
```

```
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    s = ds[1].split()
    p = []
    for i in range(n):
        p.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))

    s = ds[2].split()
    pm = []
    for i in range(m):
        pm.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))

    dist = [1e10] * m

    for i in range(n - 1):
        for j in range(m):
            dist[j] = min(dist[j], dist2(p[i], p[i + 1], pm[j]))

    ans = dist[m-1]
    for i in range(m - 1):
        ans = max(ans, dist[i])

    return str(ans)
```

Решение

В данной задаче достаточно найти для каждой мусор-точки расстояние до ближайшего к ней отрезка, после чего найти максимум среди этих значений – это и будет ответ.

Асимптотика: $O(n \cdot m)$

Пример программы

Ниже представлено решение на языке Python3

```
from collections import namedtuple
from math import sqrt

Point = namedtuple("Point", ["x", "y"])


# расстояние между точками a и b
def dist(a, b):
        return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))


# расстояние между точкой c и отрезком ab
def dist2(a, b, c):
        A = a.y - b.y
        B = b.x - a.x
        C = b.y * a.x - a.y * b.x
        cc = B * c.x - A * c.y
```

```
        d = A * A + B * B
        p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)

        return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7
                          else min(dist(c, a), dist(c, b))


def solve(dataset):
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    s = ds[1].split()
    p = []
    for i in range(n):
        p.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))

    s = ds[2].split()
    pm = []
    for i in range(m):
        pm.append(Point(int(s[i * 2]), int(s[i * 2 + 1])))

    dist = [1e10] * m

    for i in range(n - 1):
        for j in range(m):
            dist[j] = min(dist[j], dist2(p[i], p[i + 1], pm[j]))

    ans = dist[m-1]
    for i in range(m - 1):
        ans = max(ans, dist[i])

    return ans


print(solve(input() + '\n' + input() + '\n' + input()))
```

Задача 8.3.3. Радиус роботов-уборщиков (25 баллов)

Не успели вы закончить с предыдущей задачей, как Серёжа уже нарисовал какие-то ломаные на полу и сказал, что роботы должны двигаться по ним, не сталкиваться между собой и при этом должны быть одинаковых размеров (по каждой из траекторий движется только один робот; касание не считается столкновением).

Рассчитайте максимальный радиус, при котором роботы не столкнутся ни при какой разнице времён запуска роботов.

Формат входных данных

В первой строке два целых числа:

- $2 \leqslant n \leqslant 500$ - количество роботов;
- $2 \leqslant m \leqslant 500$ - количество точек в траектории каждого робота.

При этом $4 \leqslant n \cdot m \leqslant 500$.

В следующих n строках задано по m точек ($2 \cdot m$ целых числа):$x_{i,j}$, $y_{i,j}$ — j-я точка траектории i-го робота ($1 \leqslant i \leqslant n$; $1 \leqslant j \leqslant m$; $|x_{i,j}| \leqslant 10^9$; $|y_{i,j}| \leqslant 10^9$).

Никакие две последовательные точки никакой из траекторий не совпадают.

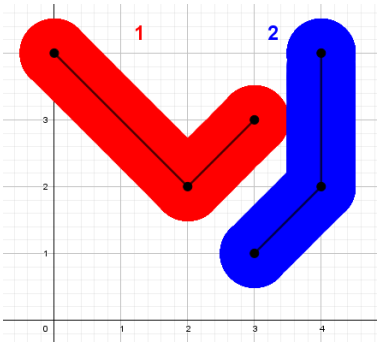


Рис. 8.3: Рисунок к первому примеру

Формат выходных данных

Если есть пересекающиеся траектории, выведите -1, иначе одно неотрицательное число – максимальный радиус.

Ваш ответ будет засчитан, если его абсолютная или относительная ошибка не превосходит 10^{-6}. Формально, пусть ваш ответ равен a, а ответ жюри равен b. Ваш ответ будет засчитан, если $\frac{|a-b|}{\max(1,b)} \leqslant 10^{-6}$.

Примеры

Пример №1

Стандартный ввод
2 3 0 4 2 2 3 3 4 4 4 2 3 1
Стандартный вывод
0.5

Способ оценки работы

За решение задачи начислялось:

- **10 баллов**, если пройдена только первая группа тестов;
- **25 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    a = float(reply)
    b = float(clue)
    return abs(a - b) / max(1.0, b) <= 1e-6

from collections import namedtuple
from math import sqrt

Point = namedtuple("Point", ["x", "y"])

# triangle area
def area(a, b, c):
    s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x)
    return -1 if s < 0 else (1 if s > 0 else 0)

# bounding box
def box(a, b, c, d):
    return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))

# пересечение отрезков
def intersect(a, b, c, d):
    return box(a.x, b.x, c.x, d.x) & box(a.y, b.y, c.y, d.y) \
           & (area(a, b, c) * area(a, b, d) <= 0) & (area(c, d, a) * area(c, d, b) <= 0)

# расстояние между точками a и b
def dist(a, b):
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))

# расстояние между точкой c и отрезком ab
def dist2(a, b, c):
    A = a.y - b.y
    B = b.x - a.x
    C = b.y * a.x - a.y * b.x
    cc = B * c.x - A * c.y
    d = A * A + B * B
    p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)

    return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7 else min(dist(c, a),
    dist(c, b))
# расстояние между отрезками ab и cd
def dist3(a, b, c, d):
    return min(min(dist2(c, d, a), dist2(c, d, b)), min(dist2(a, b, c), dist2(a, b, d)))

def solve(dataset):
    ds = dataset.splitlines()

    s = ds[0].split()
    n = int(s[0])
    m = int(s[1])

    p = [[]]
    for i in range(n):
        s = ds[i + 1].split()
        p.append([])
        for j in range(m):
            p[i].append(Point(int(s[j * 2]), int(s[j * 2 + 1])))
```

```
    inter = False

    for i in range(n):
        for j in range(i + 1, n):
            for ii in range(m - 1):
                for jj in range(m - 1):
                    inter |= intersect(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1])

    if inter:
        return str(-1)

    ans = 1e15

    for i in range(n):
        for j in range(i + 1, n):
            for ii in range(m - 1):
                for jj in range(m - 1):
                    ans = min(ans, dist3(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1]))

    return str(ans / 2.0)
```

Решение

В данной задаче достаточно проверить есть ли пересекающиеся траектории (аналогично тому, как это делается в задаче A) и, если такие найдутся, вывести -1 или, если таких траекторий нет, найти два отрезка, которые принадлежат разным траекториям и расстояние между которыми минимально, и вывести поделенное на два расстояние между этими отрезками (тут пригодится решение задачи B).

Асимптотика: $O((n \cdot m)^2)$

Пример программы

Ниже представлено решение на языке Python3

```python
from collections import namedtuple
from math import sqrt

Point = namedtuple("Point", ["x", "y"])


# triangle area
def area(a, b, c):
    s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x)
    return -1 if s < 0 else (1 if s > 0 else 0)


# bounding box
def box(a, b, c, d):
    return max(min(a, b), min(c, d)) <= min(max(a, b), max(c, d))


# пересечение отрезков
def intersect(a, b, c, d):
    return box(a.x, b.x, c.x, d.x) & box(a.y, b.y, c.y, d.y) \
           & (area(a, b, c) * area(a, b, d) <= 0) & (area(c, d, a) * area(c, d, b) <= 0)

```

```
23
24  # расстояние между точками a и b
25  def dist(a, b):
26      return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y))
27
28
29  # расстояние между точкой c и отрезком ab
30  def dist2(a, b, c):
31      A = a.y - b.y
32      B = b.x - a.x
33      C = b.y * a.x - a.y * b.x
34      cc = B * c.x - A * c.y
35      d = A * A + B * B
36      p = Point((cc * B - C * A) / d, -(cc * A + C * B) / d)
37
38      return dist(c, p) if dist(a, p) + dist(p, b) - dist(a, b) < 1e-7
39                        else min(dist(c, a), dist(c, b))
40
41
42  # расстояние между отрезками ab и cd
43  def dist3(a, b, c, d):
44      return min(min(dist2(c, d, a), dist2(c, d, b)), min(dist2(a, b, c), dist2(a, b, d)))
45
46
47  def solve(dataset):
48      ds = dataset.splitlines()
49
50      s = ds[0].split()
51      n = int(s[0])
52      m = int(s[1])
53
54      p = [[]]
55      for i in range(n):
56          s = ds[i + 1].split()
57          p.append([])
58          for j in range(m):
59              p[i].append(Point(int(s[j * 2]), int(s[j * 2 + 1])))
60
61      inter = False
62
63      for i in range(n):
64          for j in range(i + 1, n):
65              for ii in range(m - 1):
66                  for jj in range(m - 1):
67                      inter |= intersect(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1])
68
69      if inter:
70          return -1
71
72      ans = 1e15
73
74      for i in range(n):
75          for j in range(i + 1, n):
76              for ii in range(m - 1):
77                  for jj in range(m - 1):
78                      ans = min(ans, dist3(p[i][ii], p[i][ii + 1], p[j][jj], p[j][jj + 1]))
79
80      return ans / 2.0
81
82
```

```
pds = input()
n = int(pds.split()[0])
for i in range(n):
    pds += '\n'
    pds += input()

print(solve(pds))
```

Задача 8.3.4. Уборка (15 баллов)

Определить возможность столкновения - задача довольно сложная, поэтому для начала вы договорились с Серёжей, что перепрограммируете роботов (зададите им новые траектории) и будете запускать их по очереди.

Происходит это следующим образом: очередного робота Серёжа ставит на первую точку траектории и запускает. После достижения конечной точки Серёжа моментально убирает робота и переходит к следующему.

Ваша задача определить каких роботов нужно запустить, чтобы убрать весь мусор. При этом нужно минимизировать количество запущенных роботов.

Допускается погрешность не более 10^{-6} при расчёте расстояния от центра робота в любой момент времени до любой из мусор-точек.

Формат входных данных

В первой строке три целых числа:

- $1 \leqslant n \leqslant 20$ - количество роботов;
- $2 \leqslant m \leqslant 500$ - количество точек в траектории каждого робота;
- $1 \leqslant k \leqslant 200$ - количество мусор-точек.

В следующих n строках по $2 \cdot m + 1$ целых чисел: $1 \leqslant radius_i \leqslant 10^9$ - радиус i-го робота и m точек ($2 \cdot m$ целых числа):$x_{i,j}$, $y_{i,j}$ — j-я точка траектории i-го робота ($1 \leqslant i \leqslant n$; $1 \leqslant j \leqslant m$; $|x_{i,j}| \leqslant 10^9$; $|y_{i,j}| \leqslant 10^9$).

В последней строке заданы мусор-точки ($2 \cdot k$ целых чисел): x_i y_i — i-я точка мусора ($1 \leqslant i \leqslant k$; $|x_i| \leqslant 10^9$; $|y_i| \leqslant 10^9$).

Никакие две последовательные точки никакой из траекторий не совпадают.

Формат выходных данных

Если весь мусор невозможно убрать, выведите -1, иначе одно положительное целое число — минимальное необходимое для уборки всего мусора количество роботов.

Примеры

Пример №1

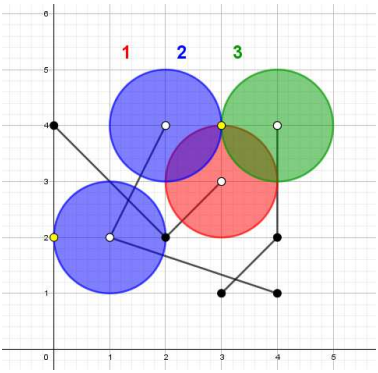


Рис. 8.4: Рисунок к первому примеру

Стандартный ввод
3 3 2 1 0 4 2 2 3 3 1 2 4 1 2 4 1 1 4 4 4 2 3 1 0 2 3 4
Стандартный вывод
1

Способ оценки работы

За решение задачи начислялось:

- **15 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    return reply.strip() == clue.strip()
```

Решение

С помощью функции нахождения расстояния от точки до отрезка можно определить для каждого робота какие мусор-точки он может собрать.

Далее достаточно сделать полный перебор (битмасок) вариантов очереди запуска роботов и найти тот вариант, в котором используется минимум роботов и весь мусор убран.

Асимптотика: $O(n \cdot k \cdot (2^n + m))$

Пример программы

Ниже представлено решение на языке Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.StringTokenizer;

import static java.lang.Math.min;
import static java.lang.Math.sqrt;

public class Main {
    // Уборка
    private FastScanner in;
    private PrintWriter out;

    class Point {
        double x, y;

        Point(double x, double y) {
            this.x = x;
            this.y = y;
        }
    }

    private double eps = 1e-7;

    // расстояние между точками a и b
    private double dist(Point a, Point b) {
        return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
    }

    // расстояние между точкой c и отрезком ab
    private double dist(Point a, Point b, Point c) {
        double A = a.y - b.y, B = b.x - a.x, C = b.y * a.x - a.y * b.x;
        double cc = B * c.x - A * c.y, d = A * A + B * B;
        Point p = new Point((cc * B - C * A) / d, -(cc * A + C * B) / d);

        return dist(a, p) + dist(p, b) - dist(a, b) < eps ?
                dist(c, p) :
                min(dist(c, a), dist(c, b));
    }

    // траектории и радиусы роботов
    private int n, m; // количество роботов и количество точек в траектории каждого робота
    private double[] radius; // радиусы роботов
    private Point[][] p; // все точки всех траекторий

    // мусор
    private int k;
    private Point[] pk;

    // инициализация
    private void init() throws IOException {
        n = in.nextInt();
        m = in.nextInt();
        k = in.nextInt();

        radius = new double[n];
```

```
59          p = new Point[n][m];
60          pk = new Point[k];
61
62          for (int i = 0; i < n; i++) {
63              radius[i] = in.nextInt();
64
65              for (int j = 0; j < m; j++)
66                  p[i][j] = new Point(in.nextInt(), in.nextInt());
67          }
68
69          for (int i = 0; i < k; i++)
70              pk[i] = new Point(in.nextInt(), in.nextInt());
71      }
72
73      // Основа решения
74      private void solve() throws IOException {
75          boolean[][] can = new boolean[n][k];
76
77          for (int i = 0; i < n; i++)
78              for (int ki = 0; ki < k; ki++)
79                  for (int j = 0; j + 1 < m && !can[i][ki]; j++)
80                      can[i][ki] = dist(p[i][j], p[i][j + 1], pk[ki]) - radius[i] < eps;
81
82          int full = 1 << n, ans = n + 1;
83          boolean ok;
84          for (int f = 1; f < full; f++) {
85              ok = true;
86              for (int ki = 0; ki < k && ok; ki++) {
87                  ok = false;
88                  for (int i = 0; i < n && !ok; i++)
89                      ok = ((1 << i) & f) > 0 && can[i][ki];
90              }
91              if (ok)
92                  ans = min(ans, cnt(f));
93          }
94
95          out.println(ans > n ? -1 : ans);
96      }
97
98      private int cnt(int f) {
99          int cnt = 0;
100         for (char c : Integer.toBinaryString(f).toCharArray())
101             cnt += c == '1' ? 1 : 0;
102         return cnt;
103     }
104
105     class FastScanner {
106         StringTokenizer st;
107         BufferedReader br;
108
109         FastScanner(InputStream s) {
110             br = new BufferedReader(new InputStreamReader(s));
111         }
112
113         String next() throws IOException {
114             while (st == null || !st.hasMoreTokens())
115                 st = new StringTokenizer(br.readLine());
116             return st.nextToken();
117         }
118
```

```
119         int nextInt() throws IOException {
120             return Integer.parseInt(next());
121         }
122     }
123
124     private void run() throws IOException {
125         in = new FastScanner(System.in);
126         out = new PrintWriter(System.out);
127
128         init();
129         solve();
130
131         out.flush();
132         out.close();
133     }
134
135     public static void main(String[] args) throws IOException {
136         new Main().run();
137     }
138 }
```

Задача 8.3.5. Столкновения (15 баллов)

Поскольку Серёжа не может перепрограммировать роботов, Серёжа решил переставить их так, чтобы они собрали больше мусора, тем самым всё же изменив траектории их движения. Сейчас он готовится запустить всех роботов и посмотреть, сколько мусора они уберут. Но вы-то знаете, что теперь эти роботы не только могут убрать мусор, но и врезаться друг в друга!

Вам нужно срочно указать Серёже какие пары роботов могут столкнуться, если их запустить одновременно.

После достижения конечной точки Серёжа моментально убирает робота.

Касание (расстояние меньше 10^{-6}) считайте столкновением.

Формат входных данных

В первой строке два целых числа:

- $2 \leqslant n \leqslant 100$ - количество роботов;
- $2 \leqslant m \leqslant 100$ - количество точек в траектории каждого робота.

В следующих n строках по $2 \cdot m + 2$ целых чисел: $1 \leqslant radius_i \leqslant 10^9$ - радиус i-го робота, $1 \leqslant speed_i \leqslant 10^9$ - скорость i-го робота и m точек $x_{i,j}$, $y_{i,j}$ — j-я точка траектории i-го робота ($1 \leqslant i \leqslant n$; $1 \leqslant j \leqslant m$; $|x_{i,j}| \leqslant 10^9$; $|y_{i,j}| \leqslant 10^9$).

Формат выходных данных

В первой строке одно целое число: k – число пар роботов, которые столкнутся, если запустить одновременно.

В следующих k строках по одной паре целых чисел: i j - i-й робот столкнётся с j-м ($1 \leqslant i \leqslant n$; $1 \leqslant j \leqslant n$; $i < j$)

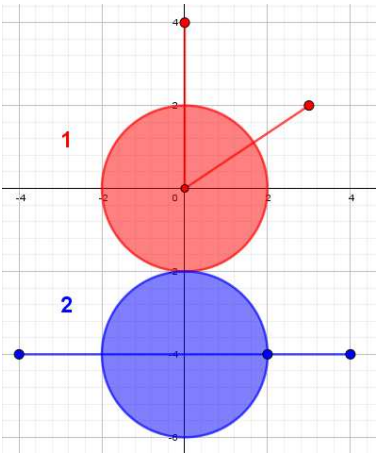


Рис. 8.5: Рисунок к первому примеру. Столкновение происходит примерно через 4 секунды после запуска.

Примеры

Пример №1

Стандартный ввод
2 3 2 1 0 4 0 0 3 2 2 1 -4 -4 2 -4 4 -4
Стандартный вывод
1 1 2

Способ оценки работы

За решение задачи начислялось:

- **15 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    return reply.strip() == clue.strip()
```

Решение

Для решения данной задачи нужно написать проверку на столкновение двух одновременно запущенных роботов.

Можно заметить, что в промежутки времени между достижениями роботами точек роботы движутся равномерно и прямолинейно. А это значит, что с помощью тернарного поиска на каждом из таких промежутков времени, мы можем найти минимальное расстояние между центрами роботов, достижимое в некоторый текущий промежуток времени. Если хотя бы одно из этих минимальных расстояний не превышает суммы радиусов роботов, то столкновения не избежать.

Асимптотика: $O(n^2 \cdot m)$

Пример программы

Ниже представлено решение на языке Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.StringTokenizer;

import static java.lang.Math.min;
import static java.lang.Math.sqrt;

public class Main {
    // Столкновения
    private FastScanner in;
    private PrintWriter out;

    class Point {
        double x, y;

        Point(double x, double y) {
            this.x = x;
            this.y = y;
        }
    }

    private double eps = 1e-7;

    // расстояние между точками a и b
    private double dist(Point a, Point b) {
        return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
    }

    // точка, в которой находился бы робот, если бы
    // прошёл растояние s из точки a в направлении точки b
    private Point goTo(Point a, Point b, double s) {
        double S = dist(a, b);
        return new Point(a.x + (b.x - a.x) / S * s, a.y + (b.y - a.y) / S * s);
    }

    // траектории и радиусы роботов
    private int n, m; // количество роботов и количество точек в траектории каждого робота
    private double[] radius, speed; // радиусы и скорости роботов
    private Point[][] p; // все точки всех траекторий
    private double[][] time;

    // инициализация
```

```
46      private void init() throws IOException {
47          n = in.nextInt();
48          m = in.nextInt();
49
50          radius = new double[n];
51          speed = new double[n];
52          p = new Point[n][m];
53          time = new double[n][m];
54
55          for (int i = 0; i < n; i++) {
56              radius[i] = in.nextInt();
57              speed[i] = in.nextInt();
58
59              for (int j = 0; j < m; j++)
60                  p[i][j] = new Point(in.nextInt(), in.nextInt());
61          }
62      }
63
64      // заполняем таблицу времени
65      private void fillTime() {
66          for (int i = 0; i < n; i++) {
67              time[i][0] = 0.0;
68
69              for (int j = 0; j + 1 < m; j++)
70                  time[i][j + 1] = time[i][j] + dist(p[i][j], p[i][j + 1]) / speed[i];
71          }
72      }
73
74      // Основа решения
75      private void solve() throws IOException {
76          StringBuilder ans = new StringBuilder();
77          int cnt = 0;
78
79          for (int i = 0; i < n; i++)
80              for (int j = i + 1; j < n; j++)
81                  if (crash(i, j)) {
82                      ans.append(i + 1).append(' ').append(j + 1).append('\n');
83                      cnt++;
84                  }
85
86          out.print(cnt + "\n" + ans);
87      }
88
89      // проверяем на столкновение i-ого и j-ого роботов
90      private boolean crash(int i, int j) {
91          double pt = 0.0, nt;
92          double l, r, t;
93          double tl, tr, dl, dr;
94          Point pil, pir, pjl, pjr;
95
96          for (int ii = 1, jj = 1; ii < m && jj < m; pt = nt) {
97              nt = min(time[i][ii], time[j][jj]);
98
99              if (nt - pt > eps) {
100
101                 // тернарный поиск
102                 l = pt;
103                 r = nt;
104                 while (r - l > eps) {
105
```

```
                    tl = l + (r - l) / 3;
                    tr = tl + (r - l) / 3;

                    pil = goTo(p[i][ii - 1], p[i][ii], (tl - time[i][ii - 1]) * speed[i]);
                    pir = goTo(p[i][ii - 1], p[i][ii], (tr - time[i][ii - 1]) * speed[i]);

                    pjl = goTo(p[j][jj - 1], p[j][jj], (tl - time[j][jj - 1]) * speed[j]);
                    pjr = goTo(p[j][jj - 1], p[j][jj], (tr - time[j][jj - 1]) * speed[j]);

                    dl = dist(pil, pjl);
                    dr = dist(pir, pjr);

                    if (dl > dr)
                        l = tl;
                    else
                        r = tr;
                }

                t = (l + r) / 2.0;
                pil = goTo(p[i][ii - 1], p[i][ii], (t - time[i][ii - 1]) * speed[i]);
                pjl = goTo(p[j][jj - 1], p[j][jj], (t - time[j][jj - 1]) * speed[j]);

                if (dist(pil, pjl) - radius[i] - radius[j] < eps)
                    return true;
            }

            if (time[i][ii] - nt < eps)
                ii++;
            if (time[j][jj] - nt < eps)
                jj++;
        }

        return false;
    }

    class FastScanner {
        StringTokenizer st;
        BufferedReader br;

        FastScanner(InputStream s) {
            br = new BufferedReader(new InputStreamReader(s));
        }

        String next() throws IOException {
            while (st == null || !st.hasMoreTokens())
                st = new StringTokenizer(br.readLine());
            return st.nextToken();
        }

        int nextInt() throws IOException {
            return Integer.parseInt(next());
        }
    }

    private void run() throws IOException {
        in = new FastScanner(System.in);
        out = new PrintWriter(System.out);

        init();
        fillTime();
```

```
166             solve();
167
168             out.flush();
169             out.close();
170         }
171
172         public static void main(String[] args) throws IOException {
173             new Main().run();
174         }
175     }
```

Задача 8.3.6. Complete cleaning (10 баллов)

Возможно вам надоело решать вспомогательные задачи. Обрадуем: теперь вы готовы решить изначальную задачу.

У вас есть n роботов, они двигаются по траекториям состоящим из m точек каждая. Вам нужно убрать как можно больше мусора используя роботов, но вы очень сильно дорожите своими роботами. Поэтому вы не хотите, чтобы во время движения они столкнулись.

Таким образом, вам нужно вывести минимальное количество роботов, при котором можно убрать максимально возможное количество мусора. При этом выбранные роботы запускаются одновременно и не сталкиваются друг с другом.

После достижения конечной точки робот моментально исчезает.

Касание (расстояние меньше 10^{-6}) считайте столкновением.

Формат входных данных

В первой строке три целых числа:

- $1 \leqslant n \leqslant 20$ - количество роботов;
- $2 \leqslant m \leqslant 500$ - количество точек в траектории каждого робота;
- $1 \leqslant k \leqslant 200$ - количество мусор-точек.

В следующих n строках по $2 \cdot m + 2$ целых чисел: $1 \leqslant radius_i \leqslant 10^9$ - радиус i-го робота, $1 \leqslant speed_i \leqslant 10^9$ - скорость i-го робота и m точек $x_{i,j}$, $y_{i,j}$ — j-я точка траектории i-го робота ($1 \leqslant i \leqslant n$; $1 \leqslant j \leqslant m$; $|x_{i,j}| \leqslant 10^9$; $|y_{i,j}| \leqslant 10^9$).

В последней строке заданы мусор-точки ($2 \cdot k$ целых чисел): x_i y_i — i-я точка мусора ($1 \leqslant i \leqslant k$; $|x_i| \leqslant 10^9$; $|y_i| \leqslant 10^9$).

Формат выходных данных

Два неотрицательных числа - максимальное количество убранных мусор-точек и минимальное количество задействованных для этого роботов.

Примеры

Пример №1

Стандартный ввод
2 3 2 2 1 0 4 0 0 3 2 2 1 -4 -4 2 -4 4 -4 3 3 4 -6
Стандартный вывод
1 1

Пример №2

Стандартный ввод
2 3 2 2 1 0 4 0 0 3 2 2 1 -4 -4 2 -4 4 -4 0 -3 4 -6
Стандартный вывод
2 1

Способ оценки работы

За решение задачи начислялось:

- **10 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке Python3

```
def generate():
    return []

def check(reply, clue):
    return reply.strip() == clue.strip()
```

Решение

Для решения последней задачи нужно оптимизировать фрагменты кода из прошлых задач.

Пример программы

Ниже представлено решение на языке Java

```
1  import java.io.BufferedReader;
2  import java.io.InputStream;
3  import java.io.InputStreamReader;
4  import java.io.IOException;
5  import java.io.PrintWriter;
6  import java.util.ArrayList;
7  import java.util.StringTokenizer;
8
9  import static java.lang.Math.min;
10 import static java.lang.Math.sqrt;
11
```

```java
public class Main {
    // Complete cleaning
    private FastScanner in;
    private PrintWriter out;

    class Point {
        double x, y;

        Point(double x, double y) {
            this.x = x;
            this.y = y;
        }
    }

    private double eps = 1e-7;

    // расстояние между точками a и b
    private double dist(Point a, Point b) {
        return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
    }

    // расстояние между точкой c и отрезком ab
    private double dist(Point a, Point b, Point c) {
        double A = a.y - b.y, B = b.x - a.x, C = b.y * a.x - a.y * b.x;
        double cc = B * c.x - A * c.y, d = A * A + B * B;
        Point p = new Point((cc * B - C * A) / d, -(cc * A + C * B) / d);

        return dist(a, p) + dist(p, b) - dist(a, b) < eps ?
                    dist(c, p) :
                    min(dist(c, a), dist(c, b));
    }

    // точка, в которой находился бы робот, если бы
    // прошёл растояние s из точки a в направлении точки b
    private Point goTo(Point a, Point b, double s) {
        double S = dist(a, b);
        return new Point(a.x + (b.x - a.x) / S * s, a.y + (b.y - a.y) / S * s);
    }

    // траектории и радиусы роботов
    private int n, m; // количество роботов и количество точек в траектории каждого робота
    private double[] radius, speed; // радиусы и скорости роботов
    private Point[][] p; // все точки всех траекторий
    private double[][] time;

    // мусор
    private int k;
    private Point[] pk;

    // инициализация
    private void init() throws IOException {
        n = in.nextInt();
        m = in.nextInt();
        k = in.nextInt();

        radius = new double[n];
        speed = new double[n];
        p = new Point[n][m];
        pk = new Point[k];

```

```
72          for (int i = 0; i < n; i++) {
73              radius[i] = in.nextInt();
74              speed[i] = in.nextInt();
75
76              for (int j = 0; j < m; j++)
77                  p[i][j] = new Point(in.nextInt(), in.nextInt());
78          }
79
80          for (int i = 0; i < k; i++)
81              pk[i] = new Point(in.nextInt(), in.nextInt());
82      }
83
84      // заполняем таблицу времени
85      private void fillTime() {
86          time = new double[n][m];
87
88          for (int i = 0; i < n; i++) {
89              time[i][0] = 0.0;
90
91              for (int j = 0; j + 1 < m; j++)
92                  time[i][j + 1] = time[i][j] + dist(p[i][j], p[i][j + 1]) / speed[i];
93          }
94      }
95
96      // Основа решения
97      private void solve() throws IOException {
98
99          // создаём матрицу столкновений
100         boolean[][] crash = new boolean[n][n];
101         for (int i = 0; i < n; i++)
102             for (int j = i + 1; j < n; j++)
103                 crash[i][j] = crash[j][i] = crash(i, j);
104
105         // создаём матрицу для проверки захвата роботами мусора
106         boolean[][] can = new boolean[n][k];
107         for (int i = 0; i < n; i++)
108             for (int ki = 0; ki < k; ki++)
109                 for (int j = 0; j + 1 < m && !can[i][ki]; j++)
110                     can[i][ki] = dist(p[i][j], p[i][j + 1], pk[ki]) - radius[i] < eps;
111
112         int full = 1 << n, ans = n;
113         boolean ok;
114         ArrayList<Integer> ids = new ArrayList<>();
115
116         int max = 0, min = 0;
117         // делаем полный перебор (битмасок) вариантов запуска роботов
118         for (int f = 1; f < full; f++) {
119             ids.clear();
120
121             // создаём список роботов, которые запускаются
122             for (int i = 0; i < n; i++)
123                 if (((1 << i) & f) > 0)
124                     ids.add(i);
125
126             // проверяем роботов из списка на столкновения
127             ok = true;
128             for (int i : ids)
129                 for (int j : ids)
130                     ok &= !crash[i][j];
131
```

```
            // в случае столкновения, сразу переходим к следующему варианту
            if (!ok)
                continue;

            // считаем количество убираемых мусор-точек
            int cnt = 0;
            for (int ki = 0; ki < k; ki++) {
                ok = false;
                for (int i : ids)
                    if (can[i][ki]) {
                        ok = true;
                        break;
                    }

                if (ok)
                    cnt++;
            }

            // улучшаем текущий ответ
            if (cnt > max) {
                max = cnt;
                min = ids.size();
            } else if (cnt == max)
                if (ids.size() < min)
                    min = ids.size();
        }

        // выводим ответ: max - количество мусора, min - кол-во запущенных роботов
        out.println(max + " " + min);
    }

    // проверяем на столкновение i-ого и j-ого роботов
    private boolean crash(int i, int j) {
        double pt = 0.0, nt;
        double l, r, t;
        double tl, tr, dl, dr;
        Point pil, pir, pjl, pjr;

        for (int ii = 1, jj = 1; ii < m && jj < m; pt = nt) {
            nt = min(time[i][ii], time[j][jj]);

            if (nt - pt > eps) {

                // тернарный поиск
                l = pt;
                r = nt;
                while (r - l > eps) {

                    tl = l + (r - l) / 3;
                    tr = tl + (r - l) / 3;

                    pil = goTo(p[i][ii - 1], p[i][ii], (tl - time[i][ii - 1]) * speed[i]);
                    pir = goTo(p[i][ii - 1], p[i][ii], (tr - time[i][ii - 1]) * speed[i]);

                    pjl = goTo(p[j][jj - 1], p[j][jj], (tl - time[j][jj - 1]) * speed[j]);
                    pjr = goTo(p[j][jj - 1], p[j][jj], (tr - time[j][jj - 1]) * speed[j]);

                    dl = dist(pil, pjl);
                    dr = dist(pir, pjr);

```

```
                    if (dl > dr)
                        l = tl;
                    else
                        r = tr;
                }

                t = (l + r) / 2.0;
                pil = goTo(p[i][ii - 1], p[i][ii], (t - time[i][ii - 1]) * speed[i]);
                pjl = goTo(p[j][jj - 1], p[j][jj], (t - time[j][jj - 1]) * speed[j]);

                if (dist(pil, pjl) - radius[i] - radius[j] < eps)
                    return true;
            }
            if (time[i][ii] - nt < eps)
                ii++;
            if (time[j][jj] - nt < eps)
                jj++;
        }

        return false;
    }

    class FastScanner {
        StringTokenizer st;
        BufferedReader br;

        FastScanner(InputStream s) {
            br = new BufferedReader(new InputStreamReader(s));
        }

        String next() throws IOException {
            while (st == null || !st.hasMoreTokens())
                st = new StringTokenizer(br.readLine());
            return st.nextToken();
        }

        int nextInt() throws IOException {
            return Integer.parseInt(next());
        }
    }

    private void run() throws IOException {
        in = new FastScanner(System.in);
        out = new PrintWriter(System.out);

        init();
        fillTime();
        solve();

        out.flush();
        out.close();
    }

    public static void main(String[] args) throws IOException {
        new Main().run();
    }
}
```

Задача командного тура

В командной части заключительного этапа участники должны спроектировать автономную систему управления роботом-погрузчиком для решения задач навигации на модели логистического центра с использованием алгоритмов компьютерного зрения.

Командная часть заключительного этапа имеет продолжительность 3,5 дня (всего 24 астрономических часа), которые включают работу по оснащению роботов необходимыми датчиками, программированию, пробные заезды на макете логистического центра, зачетные попытки.

9.1. Легенда

В автоматических логистических центрах практически не будет людей, всю работу будут выполнять роботы-погрузчики, которыми будет управлять интеллектуальное ПО для распределения задач.

Несмотря на отсутствие человеческого фактора, не следует думать, что в таких центрах никогда не будет никаких проблем. Форс-мажор может произойти 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 произошел инцидент с обрушением стеллажей, завалы привели к изменению структуры проездов по помещениям центра, поэтому робот-погрузчик должен уметь перемещаться в условиях измененного окружения.

В начале выполнения задания считается, что робототехническое устройство активируется в одном из секторов логистического центра, координаты сектора активации и структура логистического центра известны заранее. Робот-погрузчик должен переместиться в сектор распределения заданий, где должен считать ARTag маркеры - координаты сектора забора груза - следующей точки своего пути. При перемещении робот должен обнаруживать заваленные проходы и перестраивать свой маршрут без нанесения вреда упавшим стеллажам.

Задача участников Олимпиады - разработать программу управления робототехническим устройством для выполнения задания описанного выше.

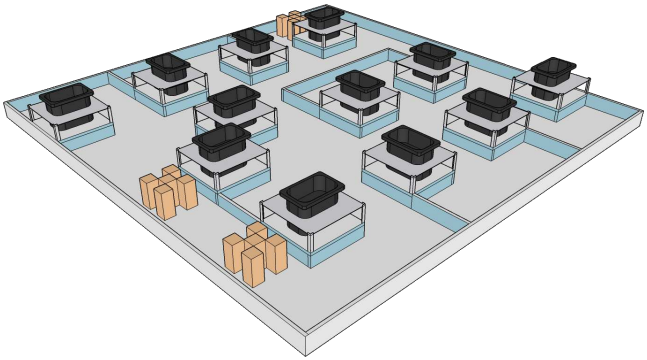


Рис. 9.1: Внешний вид полигона с тремя завалами

Рис. 9.2: Полигон, использовавшийся на финале Олимпиады НТИ

9.2. Набор заданий

Решение командной задачи было разбито на 3 этапа. Первые два этапа итеративно подводили участников к решению полной финальной задачи, осуществляемому во время последнего третьего этапа. На каждом этапе в проверку решения заданий данного этапа входили:

- способность проверить гипотезу о работоспособности алгоритма через демонстрацию решения в симуляторе;
- полнота решения задания конкретного этапа;
- воспроизводимость результатов — робототехническое устройство участников должно было неоднократно выполнить требуемые действия.

Первый этап

Задача: робототехническое устройство должно проехать по модели логистического центра из сектора старта в сектор финиша. Путь перемещения робота будет заранее выбран так, чтобы сектор старта не будет доступен при движении по правилу

правой или левой руки. Устройство должно заехать в сектор финиша, остановиться и вывести на экран количество секторов, которые находились справа от робота, но проезд к ним был перекрыт препятствием.

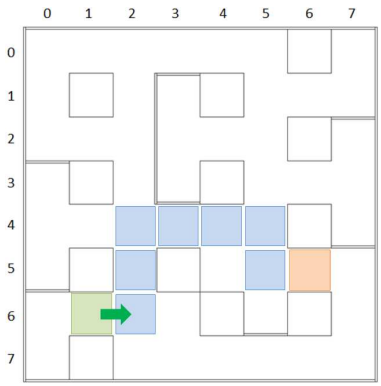


Рис. 9.3: Один из возможных оптимальных путей перемещения

Пример: Если путь перемещения робота, как на рис. 9.3, то во время движения робота были 3 сектора смежные с правой стороны с посещенными секторами, и при этом между ними располагалось препятствие.

Включая содержательные задачи:

- Сборка робототехнического устройства
- Реализация навигации
  - Выравнивание, используя синхронизацию моторов, и физические свойства среды;
  - Перемещение по азимуту с использованием показаний датчиков гироскопа или акселерометра.
  - Поворот на заданный угол с использованием показаний датчиков гироскопа или акселерометра.
- Реализация планирования маршрута
  - Реализация алгоритмов построения оптимальных маршрутов от одного сектора до другого;
  - Реализация алгоритмов автоматического планирования перемещения.
- Реализация сбора и обработки информации с датчиков
- Реализация счисления пути

Второй этап

Задача: робототехническое устройство должно проехать по модели логистического центра из сектора старта в сектор финиша через сектор распределения заданий. Координаты сектора финиша робот должен определить самостоятельно по ARTag меткам, расположенным на стеллаже, прилегающем к сектору распределения заданий.

Включая содержательные задачи:

- Калибровка камеры робототехнического устройства;
- Реализация алгоритмов компьютерного зрения: считывание ARTag меток без использования дополнительных библиотек;
- Декодирование бинарного кода, с использованием алгоритма обнаружения ошибок в кодах с битом четности.

Третий этап

Задача: робототехническое устройство должно проехать по модели логистического центра из точки старта сектор распределения заданий в сектор финиша, выявив все зоны, которые стали недоступны из-за обрушения стеллажей. Координаты сектора финиша должны будут определены роботом самостоятельно посредством декодирования ARTag маркеров.

Пример: На рис. 9.12 изображено поле, где недоступно 11 секций из-за обрушения стеллажей.

Включая содержательные задачи:

- Реализация алгоритмов построения карты;
- Реализация сопоставления шаблонов для выявления различий;
- Реализация алгоритмов обнаружения препятствий и перепланирования маршрута движения;
- Реализация алгоритмов оценки полноты исследования заранее известной карты.

9.3. Описание модели логистического центра

Полигон - квадратное поле 3200x3200 мм., разделенное на квадратные сектора 400x400 мм. Некоторые сектора отделены друг от друга перегородкой высотой 100 мм. Некоторые сектора недоступны для посещения робототехническим устройством и представляют из себя модель стеллажа высотой 210 мм. На каждой полке стеллажа располагается черный контейнер с грузом (рис. 9.4) размером $200 \times 300 \times 100$ мм (груз - 2-3 коробки (рис. 9.5) размером $180 \times 80 \times 85$ мм).

Полигон окружен бортом высотой 100 мм. Конфигурация полигона определяется в первый день финального этапа и объявляется участникам. Данная конфигурация будет использоваться все дни финального этапа.

На нижней полке стеллажа, прилегающего одной из четырех сторон к сектору распределения заданий, на высоте от 100-150 мм от уровня поверхности поля закреплены два ARTag маркера (`https://goo.gl/WaTFMB`), определяющие координаты сектора забора груза (сектор финиша). Размер каждого маркера - 30×30 мм. Расстояние между маркерами — $(70–150 \pm 5)$ мм. Маркеры обращены лицевой стороной внутрь сектора старта. Конкретная высота расположения маркеров определяется в первый день финала и остается постоянной во все дни финального этапа. При этом допустимая погрешность установки маркеров ± 5 мм. Пример расположения маркеров на стелаже представлен на рисунке 9.6

Рис. 9.4: Внешний вид контейнера

Рис. 9.5: Внешний вид коробки

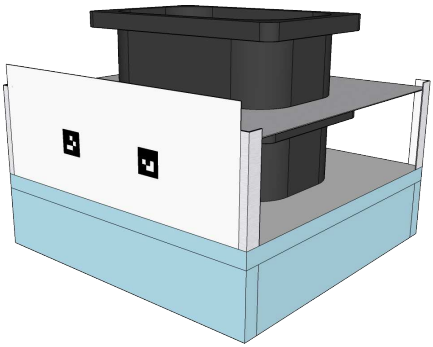


Рис. 9.6: Стеллаж с установленными ARTag маркерами

Маркер состоит из 5×5 элементов одинакового размера. Элементы маркера, 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 в он обозначает 1, если белый, то 0 при этом самый первый элемент — старший бит закодированного числа. Элементы пронумерованы сверху вниз, слева направо.

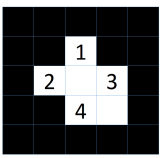


Рис. 9.7: Нумерация битов в маркера

Например, на маркере с рис. 9.8 закодировано число 0011_2, что эквивалентно 3_{10}.

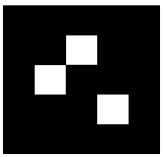


Рис. 9.8: Маркер с закодированным значением - 0011_2

Если закодированное на маркере двоичное число находится в диапазоне от 0_{10} до 7_{10}, то оно кодирует координату X сектора финиша. Если двоичное число — в диапазоне от 8_{10} до 15_{10}, то оно кодирует Y координату сектора финиша , при этом координата определяется вычитанием 8 из закодированного числа.

Рис. 9.9: Пример кодирования сектора

Левый маркер на рисунке 9.9 задает число $110_2 - 6_{10}$. Правый штрих-код задает число $1101_2 - 13_{10}$. Необходимый сектор находится в позиции с координатами $(6, 5)$ — см. рисунок 9.10.

Гарантируется, что сектор распределения заданий, будет располагаться в таких местах полигона, где к сторонам сектора прилегает, как минимум, один стеллаж.

Сектор активации робота, сектор распределения заданий и сектор забора груза никак не обозначаются на поле и определяются непосредственно перед каждым заездом робота.

Также на полигоне может присутствовать несколько завалов. Данные завалы состоят из 5 коробок, представленных на рисунке 9.5. Коробки располагаются вертикально. Пример завалов можно увидеть на рисунке 9.1.

9.4. Описание конструктора

В первый день финального тура каждой команде выдаются:

- Мобильная наземная платформа на базе конструктора TRIK в сборе (блок управления TRIK, аккумулятор, два мотора с энкодерами на датчиках Холла, колеса), но без установленных датчиков. 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 дополнительных деталей из конструктора TRIK и набор датчиков:

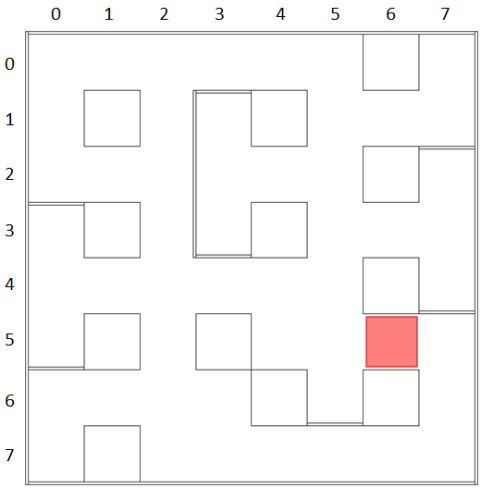


Рис. 9.10: Расположение сектора финиша для пары маркеров из рисунка 9.10

1 датчик освещенности, 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 по одному ноутбуку. При этом участники при этом могут пользоваться своими ноутбуками.

9.5. Условия проведения

- Из полученного набора датчиков команды могут выбирать те, с помощью которых, на их взгляд, можно решить задачу наиболее эффективным способом.
- Команды могут вносить любые изменения в мобильную наземную платформу.
- Участники во время командного этапа финального тура могут использовать интернет и заранее подготовленные библиотеки для решения задачи.
- Участники не могут использовать помощь тренера, сопровождающего лица или привлекать третьих лиц для решения задачи.
- 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, закрепленный за конкретным этапом.
- Некоторые подзадачи строго требуют выполнение каких-то предыдущих подзадач. Выполнение данных подзадач, без выполнения предыдущих допускается, однако данная попытка будет оценена в 0 баллов.

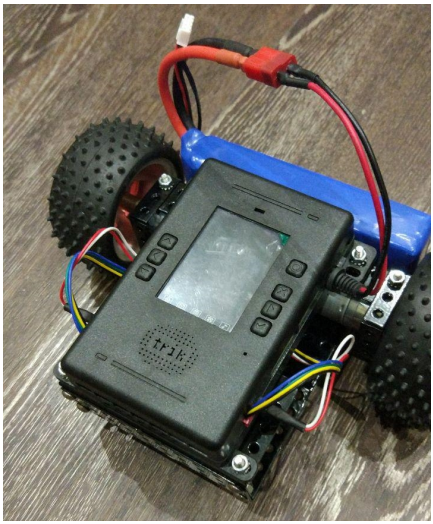


Рис. 9.11: Мобильная платформа TRIK в сборе

- При выполнении подзадачи в виртуальной среде полное количество баллов можно получить лишь при первой попытке. При второй попытке сдать подзадачу может быть начислено лишь половина баллов за данную задачу. При последующих попытках баллы не начисляются.
- 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.
- Испытания на полигоне должны осуществляться так, чтобы не мешать другим командам, проводящим в это время испытания на полигоне. Для осуществления 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.
- 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.
- У команды есть не более двух попыток для сдачи решения подзадач в симуляторе:
  - Решения принимаются на проверку до истечения первых 4 часов работы в соответствующий соревновательный день.
  - До истечения четырех часов, команда должна загрузить свое решение на *Google Drive* в каталог, доступ к которому участники получат в

начале дня. Участники команды ответственны за то, что ссылка на каталог с их решениями не попадет участникам других команд.

  - В директории должен быть только один файл с решением.
  - До истечения указанного времени, команды могут изменять файл с решением сколько угодно раз. Проверяться будет всегда только последняя доступная версия.
  - Если решение отправлено на проверку в течение первых 3.5 часов работы в в соответствующий соревновательный день, то команда имеет право на вторую попытку, если результаты проверки решения ее не устраивают.
  - Если команда хочет воспользоваться правом проверки решения до истечения 3.5 часов, то она должна загрузить в каталог на *Google Drive* программу со своим решением и сообщить об этом судьям.

- Часть подзадач для реального робота могут быть запрещены к приемке без решения соответствующей подзадачи в симуляторе.
- 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.
- Перед сдачей робота в карантин команда должна загрузить на робота управляющую программу, подготовленную для демонстрации решения задачи, а также ее копию в *Google Drive* в каталог, доступ к которому участники получат в начале соревновательного дня. Без программы, загруженной в каталог *Google Drive*, команды не допускаются до проверки решения на реальном роботе.
- После момента, когда все роботы сданы в карантин, судьи вызывают команды по одной для приемки решения подзадач, закрепленных за этапом конкретного дня финального тура.
- Может быть предусмотрено до двух попыток сдачи решения одной и той же подзадачи на реальном роботе. Конкретное количество попыток определяется в конкретных подзадачах.
- 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.
- Робот должен выполнять задание полностью автономно. Удаленное управление не допускается. Касание робота участником команды после его старта во время приемочных запусков не допускается. Алгоритм, реализующий систему управления робота должен планировать свое выполнение, полагаясь только на информацию с датчиков.
- Если подзадача подразумевает введение данных в программу (например, координат робота) до старта устройства. Такое разрешается только для тех задач, где это явно прописано. Во всех других случаях, введение данных в программу робота перед запуском запрещено.
- Если какая-то подзадача подразумевает считывание информации с элемен-

тов, расположенных на полигоне, запрещается при запуске робота вводить информацию о положении этих элементов или значениях, которые данные элементы определяют.

- Если во время приемочных запусков у судьи возникли сомнения о том, что задачи подэтапа решены корректно (робот не выполняет задачу полностью автономно, участник вводит значения в робота перед запуском), то он вправе провести инспекцию кода. По результатам инспекции, судья вправе снять с команды баллы, набранные за данный этап.
- 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.
- Команда вправе обсуждать с судьей результаты приемочных запусков до вызова следующей команды, но финальное решение остается о начислении баллов остается за судьей.

9.6. Процедура проведения приемочных запусков и критерии оценки

Первый этап

1. В качестве задачи для симулятора участникам необходимо решить следующую задачу:
    - Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Задача робота проехать из точки старта в точку финиша. Гарантируется, что точка старта и точка финиша не будут находиться на пути перемещения робота по алгоритму правой или левой руки.

      Входные данные: через файл `input.txt` управляющей программе передается координаты сектора старта X_s, Y_s и направление старта D_s (направление робота от 0 до 3 начиная с направления вверх и дальше по часовой стрелке) - первая строка, сектор финиша X_f, Y_f - вторая строка, $0 \leqslant X_s, Y_s, X_f, Y_f \leqslant 7$. Три числа в первой строке разделены пробелом. Для чисел во второй строке разделитель - также пробел.

      Ожидаемый результат: после запуска программы в консоль выводится оптимальный путь перемещения робота от сектора старта в сектор финиша. После этого робот перемещается в сектор финиша. После остановки на экран робота выведено `finish`. Оптимальным является маршрут, при описании которого будет использоваться наименьшее количество символом, при использовании следующих обозначений: F - одна секция прямо, R- поворот направо на 90°, L - поворот налево на 90°: *например*, для маршрута, представленном на рис. 9.3, запись оптимального маршрута "`FLFFRFFFRFLF`".
    - Имя файла с управляющей программой для проверки решения в симуляторе: `sim-day1.js`.
2. Командам будет предоставлено две попытки на демонстрацию решения задачи на реальном роботе.

3. Первая попытка будет осуществляться 24 февраля, вторая - 25 февраля.
4. За 15 минут до перед каждой попыткой судья определяет сектора старта и направление робота в секторе старта, также определяется сектор финиша. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
5. Правила именования файлов с программой управления:
    - для первой попытки: `day1-1.js`.
    - для второй попытки: `day1-2.js`.
6. Максимальное время выполнения одной попытки - 2 минуты.
7. Баллы за решение задач этапа:
    (a) **Симулятор:**
        i. Программа управления вывела оптимальный маршрут верно для всех проверочных полигонах — 4 баллов
        ii. Робот проехал из точки старта в точку финиша на всех проверочных полигонах — 8 баллов
    (b) **Реальный робот:**
        i. Робот покинул сектор старта — 4 баллов
        ii. Робот проехал половину от всего количества секторов, находящихся на оптимальном пути от сектора старта до сектора финиша — 8 баллов
        iii. Робот проехал от старта до финиша, остановился и вывел на экран `finish` — 16 баллов
        iv. Маршрут, по которому проехал робот от старта до финиша является оптимальным — 12 баллов
        v. Вместо `finish` на экран робота выведено верное количество перекрытых справа от робота секторов — 4 баллов
8. Баллы за подзадачи *iii*, *iv* и *v* не начисляются, если не было засчитано решение в симуляторе.
9. Баллы за две попытки суммируются.
10. Выполнение всех критериев в каждой из двух попыток дает дополнительные 32 баллов.
11. Максимальное количество баллов за этап — 132.

Второй этап

1. В качестве задачи для симулятора участникам необходимо решить следующую задачу:
    - Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Задача робота проехать из точки старта в точку финиша, чьи координаты заданы изображениями ARTag маркеров, заранее считанным с камеры реального устройства.

      Входные данные: через файл `input.txt` управляющей программе передаются:

– В первой строке через пробел — координаты сектора старта X_s, Y_s и направление старта D_s (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке), $0 \leqslant X_s, Y_s, \leqslant 7$;

– Во второй строке 19200, разделенных пробелом, целых чисел $P_{1,i}$ $(0 \leqslant P_{1,i} \leqslant 2^{32})$ — изображение первого ARTag маркера;

– Во третьей строке 19200, разделенных пробелом, целых чисел $P_{2,j}$ $(0 \leqslant P_{1,j} \leqslant 2^{32})$ — изображение второго ARTag маркера.

Каждое число в макере - точка, закодированная в формате RGB, т.е. строка с изображением маркера эквивалентна снимку разрешением 160×120 точек. Один маркер кодирует координату X финиша, второй - Y. Порядок маркеров не определяет порядок координат.

Ожидаемый результат: После запуска программы робот перемещается в сектор финиша. После остановки на экран робота выведено `finish`.

- Имя файла с управляющей программой для проверки решения в симуляторе: `sim-day2.js`.

2. Команде необходимо будет подготовить решения для трех разных подзадач для реального робота. На демонстрацию каждого решения предоставляется 2 попытки.
3. Все попытки осуществляются 26 февраля.
4. За 15 минут до времени сдачи роботов в карантин для 2ой и 3ей подзадач судья определяет сектора старта и направление робота в секторе старта, а также — сектор распределения задач (сектор, из которого необходимо сканировать ARTag метки) и направление расположения ARTag меток (0 - маркеры находятся на "верхнем" стеллаже, 1 - на "правом" стеллаже и т.д. по часовой стрелке).
   - Для второй подзадачи сектор старта и сектора распределения задач будет один и тот же для каждой попытки;
   - Для третьей подзадачи сектор старта и сектор распределения задач будут разные для разных попыток.

   Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
5. Правила именования файлов с программой управления:
   - для первой подзадачи: `day2-1.js`;
   - для второй подзадачи: `day2-2.js`.
   - для первой попытки третьей подзадачи: `day2-3-1.js`.
   - для второй попытки третьей подзадачи: `day2-3-2.js`.
6. Максимальное время выполнения одной попытки - 3 минуты.
7. Баллы за решение задач этапа:
   (a) **Симулятор**: робот проехал из точки старта в точку финиша на всех проверочных полигонах — 12 баллов

   (b) **Первая подзадача на реальном роботе:** Робот распознал верно ARTag метку, которая установлена прямо перед камерой, и вывел на экран верное значение, закодированное на метке — 4 балла

(c) **Вторая подзадача на реальном роботе:** Робот располагается за два сектора до сектора распределения задач, так что сектор старта расположен перпендикулярно сектору распределения задач.

i. Робот доехал до сектора распределения задач, остановился, издал звуковой сигнал, вывел на экран верное значение первой по ходу движения ARTag метки и продолжил движение через 10 секунд — 4 балла

ii. Робот вывел на экран верное значение второй по ходу движения ARTag метки и продолжил движение через 10 секунд — 4 балла

(d) **Третья подзадача на реальном роботе:** *полное выполнение задание второго этапа*

i. Робот проехал от старта до сектора распредения задач, остановился, издал звуковой сигнал, вывел на экран верные координаты сектора финиша и продолжил движение через 10 секунд — 28 баллов

ii. Робот остановился в секторе финиша и вывел `finish` — 4 балла

8. Баллы за третью подзадачу не начисляются, если не было засчитано решение в симуляторе.
9. Баллы за все попытки в каждой подзадаче суммируются.
10. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 32 балла.
11. Максимальное количество баллов за этап — 132.

Третий этап

1. В качестве задачи для симулятора участникам необходимо решить следующую задачу:
   - Робот устанавливается в модели логистического центра в заранее неизвестном секторе. При этом структура логистического центра известна заранее, но из-за завалов в нем образовалось N новых препятствий, заграждая путь, по которому может перемещаться робот. Задача робота проехать из точки старта в точку финиша, чьи координаты заданы изображениями ARTag маркеров, заранее считанным с камеры реального устройства. На финише необходимо вывести максимальное количество секторов во всем логистическом центре, которые стали недоступно для посещения роботом из-за появившихся завалов.

     Входные данные: через файл `input.txt` управляющей программе передаются:

     – В первой строке через пробел — координаты сектора старта X_s, Y_s и направление старта D_s (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке), $0 \leqslant X_s, Y_s, \leqslant 7$;

     – Во второй строке 19200, разделенных пробелом, целых чисел $P_{1,i}$ ($0 \leqslant P_{1,i} \leqslant 2^{32}$) — изображение первого ARTag маркера;

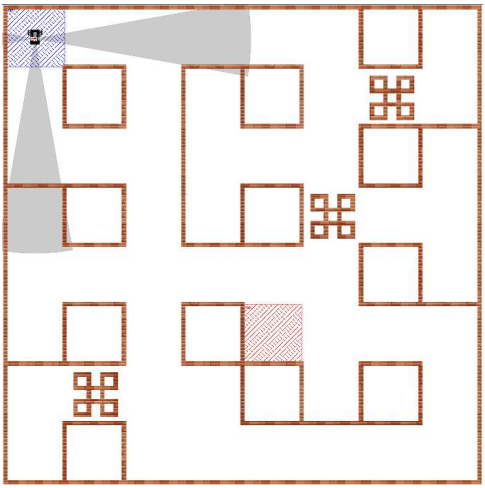


Рис. 9.12: 3 обрушения заблокировали 11 секторов

- Во третьей строке 19200, разделенных пробелом, целых чисел $P_{2,j}$ ($0 \leqslant P_{1,j} \leqslant 2^{32}$) — изображение второго ARTag маркера;
- В четвертой строке — количество завалов ($1 \leqslant N \leqslant 5$).

Каждое число в маркере - точка, закодированная в формате RGB, т.е. строка с изображением маркера эквивалентна снимку разрешением 160×120 точек. Один маркер кодирует координату X финиша, второй - Y. Порядок маркеров не определяет порядок координат.

Ожидаемый результат первой подзадачи: После запуска программы робот перемещается в сектор финиша. После остановки на экран робота выведено `finish`.

Ожидаемый результат второй подзадачи: После запуска программы робот перемещается в сектор финиша. После остановки на экран робота выведено одно число количество секций недоступных для посещения роботом из-за завалов.

- Имя файла с управляющей программой для проверки решения в симуляторе: `sim-day3.js`.

2. Командам будет предоставлено две попытки на демонстрацию решения задачи на реальном роботе.
3. Первая попытка будет осуществляться 27 февраля, вторая - 28 февраля.
4. За 15 минут до перед каждой попыткой судья определяет сектора старта и направление робота в секторе старта, также определяется сектор распределения задач (сектор, из которого необходимо сканировать ARTag метки) и направление расположения ARTag меток (0 - маркеры находятся на "верхнем" стеллаже, 1 - на "правом" стеллаже и т.д. по часовой стрелке). Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
5. Правила именования файлов с программой управления:

- для первой попытки: `day3-1.js`.
- для второй попытки: `day3-2.js`.

6. Максимальное время выполнения одной попытки - 5 минут.
7. Баллы за решение задач этапа:
   (a) **Симулятор:**
      i. Робот проехал из точки старта в точку финиша и вывел `finish` на всех проверочных полигонах — 8 баллов
      ii. Робот проехал из точки старта в точку финиша и вывел количество недоступных секторов на всех проверочных полигонах — 24 балла

   Баллы не суммируются: принимается либо первая либо вторая подзадача.

   (b) **Реальный робот:** для проверки решения задачи этапа на реальном роботе на макете будет **ВСЕГДА** выставлено только 2 завала.
      i. Робот выехал со старта и остановился в секторе соседнем от сектора с первым по ходу движения завалом и воспроизвел звук, робот не касался элементов завалов до момента остановки рядом с данным завалом — 4 балла
      ii. После остановки робота в секторе с первым по ходу движения завалом на экран верно выведено минимально определяемое количество секторов недоступных для посещения, согласно изменениям на всей карте, исследованным к моменту остановки — 4 балла
      iii. Робот остановился в секторе соседнем от сектора со вторым по ходу движения завалом и воспроизвел звук, робот не касался элементов завалов во время выполнения задания до момента остановки рядом с данным завалом — 4 балла
      iv. После остановки робота возле сектора со вторым по ходу движения завалом на экран верно выведено минимально определяемое количество секторов недоступных для посещения, согласно изменениям на всей карте, исследованным к моменту остановки — 4 балла
      v. Робот доехал до сектора распределения задач, остановился, издал звуковой сигнал, вывел на экран верные координаты сектора финиша и продолжил движение через 10 секунд, до момента остановки в секторе распределения задач робот не касался элементов любых завалов — 4 балла
      vi. Робот остановился в секторе финиша и вывел `finish`, до момента остановки в секторе финиша робот не касался элементов любых завалов — 12 баллов
      vii. Вместо `finish` на экран робота верно выведено количество недоступных для посещения секторов — 8 баллов
8. Баллы за подзадачи *v*, *vi* и *vii* не начисляются, если не было засчитано решение любой из подзадач в симуляторе.
9. Баллы за все попытки в каждой подзадаче суммируются.

10. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 32 балла.
11. Максимальное количество баллов за этап — 136.

9.7. Решение

```
// Параметры робота
var pi = 3.141592653589793;
var d = 5.6 // Диаметр колеса, см
var l = 17.5 // База, см
var x = 0 // Начальные координаты робота
var y = 0
var a = 0

// Моторы
var mLeft = brick.motor(M4).setPower;
var mRight = brick.motor(M3).setPower;
var cpr = 360 // Показания энкодера за оборот

// Энкодеры
var eLeft = brick.encoder(E4);
var eRight = brick.encoder(E3);

// Длина клетки
var cellLength = 40 * cpr / (pi * d);

// Датчики расстояния
var svFront = brick.sensor(D1).read;
var svLeft = brick.sensor(A1).read;
var svRigth = brick.sensor(A2).read;

var readGyro = brick.gyroscope().read

function readYaw() {
  return -readGyro()[6];
}

var direction = 0; // absolute angle of direction movement
var directionOld = 0;
var azimut = 0; // we should go on azimut or turn to it
print("-----------------------------------------");

eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);

//инициализация и калибровка гироскопа
brick.gyroscope().calibrate(12000);
script.wait(13000);
print("gyro inited");
brick.display().addLabel(30, 10, "Start!");
brick.display().redraw();
script.wait(1000);
```

```
52
53  var v = 50; // velocity
54
55  var ex = 0;
56  var ey = 0;
57  var encLeftOld = 0;
58  var encRightOld = 0;
59  var encLeft = 0;
60  var encRight = 0;
61  var n = 0;
62
63  // вычисление абсолютного угла относительно начального положения
64  function angle() {
65    var sgn = 0;
66    var _direction = readYaw(); // mgrad
67    var dtDirection = _direction - directionOld;
68
69    sgn = directionOld == 0 ? 0 : directionOld / Math.abs(directionOld);
70    n += sgn * Math.floor(Math.abs(dtDirection / 320000));
71    direction = _direction + n * 360000;
72    directionOld = _direction;
73  }
74
75  // делаем прерывание основной программы с частотой 200Гц,
76  // чтобы посчитать абсолютный угол с гироскопа
77  var mtimer = script.timer(50);
78  mtimer.timeout.connect(angle);
79
80  //поворот на угол по гироскопу _angle - относительный угол на который необходимо повернуться
81  function turnDirection(_angle, _v) {
82    _angle = azimut + _angle;
83    azimut = _angle;
84    _angle = _angle * 1000; //toMGrad
85
86    eLeft.reset();
87    eRight.reset();
88
89    var _vel = _v == undefined & 40: _v; // скорость по умолчанию
90    var angleOfRotate = _angle - direction;
91    var sgn = angleOfRotate == 0 ? 0 : angleOfRotate / Math.abs(angleOfRotate);
92    mLeft(-_vel * sgn);
93    mRight(_vel * sgn);
94    eLeftOld = eLeft.read();
95    eRightOld = eRight.read();
96    target = 211;
97
98    while (Math.abs(eRight.read() - eLeft.read()) / 2 < target) {
99      if ((eLeft.read() - eLeftOld) == 0) && (eRight.read() - eRightOld == 0) {
100       _vel += 5;
101       mLeft(-_vel * sgn);
102       mRight(_vel * sgn);
103     }
104     eLeftOld = eLeft.read();
105     eRightOld = eRight.read();
106     script.wait(20);
107   }
108
109   brick.motor(M3).powerOff(500);
110   brick.motor(M4).powerOff(500);
111   script.wait(500);
```

```
}

// проезд в перед на количество ячеек _kceel со скоростью _v
// выравниваясь по гироскопу на угол azimut
function forward(_v, _kcell) {
  print("azimut = " + azimut);
  _alpha = azimut;
  var _vel = _v == undefined ? 40 : _v; // скорость по умолчанию
  var u = 0;
  eLeft.reset();
  eRight.reset();
  var el = Math.abs(eLeft.readRawData());
  while (Math.abs(eLeft.readRawData()) < (el + (_kcell * cellLength))) {
    u = 1.5 * (_alpha - direction / 1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
  }
  brick.motor(M3).powerOff();
  brick.motor(M4).powerOff();
  script.wait(300);
}

min = function(a, b) {
  return a < b ? a : b;
}
max = function(a, b) {
  return a > b ? a : b;
}

//============================================================================
// массив для изображения
var pic = [];

var marker_size = 5;
var total_width = 320 / 2;
var total_height = 240 / 2;
var prob = 25 * 5 * 5;

function getColor(pic, x, y) {
  return pic[y * total_width + x];
}

function squareAverage(pic, x, y, diam) {
  var sum = 0;
  var start_row = y * total_width;
  var end_row = start_row + diam * total_width;

  for (var index = start_row + x; index < end_row; index += total_width)
    for (var j = 0; j < diam; j += 1)
      sum += pic[index + j];

  return sum;
}

// lu - left-up corner. Coordinates: (x, y)
// ld - left-down corner
// ru - right-up corner
// rd - right-down corner
function getCenterColor(pic, lu, ld, ru, rd, diam) {
```

```
  var x = (lu[0] + ld[0] + ru[0] + rd[0]) >> 2;
  var y = (lu[1] + ld[1] + ru[1] + rd[1]) >> 2;
  var color = squareAverage(pic, x - diam, y - diam, diam << 1);
  return color;
}

function findGridCorners(corners, marker_size) {
  var grid_corners = [];

  var vertical_lines = [];
  var upper_line_x1 = corners[0][0];
  var upper_line_y1 = corners[0][1];
  var upper_line_x2 = corners[2][0];
  var upper_line_y2 = corners[2][1];
  var down_line_x1 = corners[1][0];
  var down_line_y1 = corners[1][1];
  var down_line_x2 = corners[3][0];
  var down_line_y2 = corners[3][1];

  var mks = 1.0 / marker_size;
  var k_ux = (upper_line_x2 - upper_line_x1) * mks;
  var k_uy = (upper_line_y2 - upper_line_y1) * mks;
  var k_dx = (down_line_x2 - down_line_x1) * mks;
  var k_dy = (down_line_y2 - down_line_y1) * mks;

  for (var i = 0; i < marker_size + 1; i += 1) {

    var up_x = upper_line_x1 + k_ux * i;
    var up_y = upper_line_y1 + k_uy * i;

    var down_x = down_line_x1 + k_dx * i;
    var down_y = down_line_y1 + k_dy * i;

    var k_x = (down_x - up_x) * mks;
    var k_y = (down_y - up_y) * mks;

    for (j = 0; j <= marker_size; j += 1) {

      var point_x = up_x + k_x * j;
      var point_y = up_y + k_y * j;

      grid_corners.push([Math.floor(point_x), Math.floor(point_y)]);
    }
  }
  return grid_corners;
}

function detectCode(pic, grid_corners, diam) {
  var calculated_colors = []
  var markerSizePlusOne = marker_size + 1;
  var shiftedDiam = diam << 8;

  for (var i = 0; i < marker_size; i += 1) {
    for (var j = 0; j < marker_size; j += 1) {
      lu_index = i * (markerSizePlusOne) + j;
      ld_index = i * (markerSizePlusOne) + j + 1;
      ru_index = (i + 1) * (markerSizePlusOne) + j;
      rd_index = (i + 1) * (markerSizePlusOne) + j + 1;

      var lu = grid_corners[lu_index];
```

```
      var ld = grid_corners[ld_index];
      var ru = grid_corners[ru_index];
      var rd = grid_corners[rd_index];

      grid_color = getCenterColor(pic, lu, ld, ru, rd, diam);
      if (grid_color < shiftedDiam) {
        calculated_colors.push(0);
      } else {
        calculated_colors.push(1);
      }
    }
  }
  return calculated_colors;
}

function findULCorner(pic, diam) {
  var color = 1;
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = j;
      var y = i - j;
      if (getColor(pic, x, y) == 0) {
        color = squareAverage(pic, x, y, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
}

function findDLCorner(pic, diam) {
  var color = 1;
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = j;
      var y = total_height - (i - j);
      if (getColor(pic, x, y) == 0) {
        color = squareAverage(pic, x, y - diam + 1, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
}

function findURCorner(pic, diam) {
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = total_width - j;
      var y = i - j;
      if (getColor(pic, x, y) == 0) {
        var color = squareAverage(pic, x - diam + 1, y, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
```

```
}

function findDRCorner(pic, diam) {
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = total_width - j;
      var y = total_height - (i - j);
      if (getColor(pic, x, y) == 0) {
        var color = squareAverage(pic, x - diam + 1, y - diam + 1, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
}

function findCorners(pic, diam) {
  return [findULCorner(pic, diam), findDLCorner(pic, diam), findURCorner(pic,
    diam), findDRCorner(pic, diam)];
}

function threshold2(level, pic, height, width) {
  var length = pic.length;
  for (var i = 0; i < length; i += 1) {
    var color = pic[i];
    if (color < level) {
      pic[i] = 0;
    } else {
      pic[i] = 255;
    }
  }

  return pic;
}

// ------------------------------------------------------------------------------
var scale = 1;
var histogram = [];
var histSize = 256;

function calculateHistogram() {
  for (var i = 0; i < histSize; i += 1)
    histogram[i] = 0;

  var curPixelLine = 0;
  for (var i = 0; i < total_height; i += 1) {
    curPixelLine = i * total_width;
    for (var j = 0; j < total_width; j += 1)
      histogram[Math.floor(pic[curPixelLine + j])] += 1;
  }
}

// binarization using 2 elems in grayscale
var grayscale = "@#ao|-. ";
var numOfBins = grayscale.length;
var rangeBins = [];
var binCapacity = total_height * total_width / numOfBins;

function getRange() {
```

```
  for (var i = 0; i < numOfBins; i += 1)
    rangeBins[i] = 0;

  var curBin = 0;
  var curSum = 0;
  var i = 0;
  var lastIndexBin = numOfBins - 1;

  for (;
    (i < histSize) && (curBin < lastIndexBin); i += 1) {
    var diff = binCapacity - curSum;

    if (Math.abs(diff) < Math.abs(diff - histogram[i])) {
      curBin++;
      curSum = 0;
    }

    curSum += histogram[i];
    rangeBins[curBin] = i;
  }

  for (; curBin <= lastIndexBin; curBin += 1)
    rangeBins[curBin] = histSize;
}

var mapColorToLetter = [];

function initMapColorToLetter() {
  var curBin = 0;
  for (var i = 0; i < histSize; i += 1) {
    if (rangeBins[curBin] <= i) {
      curBin += 1;
    }
    mapColorToLetter[i] = grayscale[curBin];
  }
}

// Возвращает значение ARTag
function getARTagValue() {
  source_pic = getPhoto();

  // init pic, grayscale mode
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j < total_width; j += 1) {
      var x = (j + i * scale * total_width) * scale;
      var p = source_pic[x];
      p = (((p & 0xff0000) >> 18) + ((p & 0xff00) >> 10) + ((p & 0xff) >> 2));
      pic[j + i * total_width] = p;
    }
  }
  calculateHistogram();
  getRange();
  initMapColorToLetter();

  var thresh = threshold2(rangeBins[1], pic, total_height, total_width);
  var corners = findCorners(thresh, 9);
  var grid_corners = findGridCorners(corners, marker_size)
  var values = detectCode(thresh, grid_corners, 3);

  var ans = 0;
```

```
412   if (values[1][1] == 0)
413     ans = 8 * values[3][2] + 4 * values[2][3] + 2 * values[2][1] + values[1][2];
414   else if (values[1][3] == 0)
415     ans = 8 * values[2][1] + 4 * values[3][2] + 2 * values[1][2] + values[2][3];
416   else if (values[3][3] == 0)
417     ans = 8 * values[1][2] + 4 * values[2][1] + 2 * values[2][3] + values[3][2];
418   else if (values[3][1] == 0)
419     ans = 8 * values[2][3] + 4 * values[2][3] + 2 * values[3][2] + values[2][1];
420   else
421     print("Error: Incorrect ARTag");
422   return ans;
423 };
424 //=================================================================
425
426 // Местонахождение робота
427 var positionOfRobot = 0;
428 var directionOfRobot = 0;
429
430 // карта
431 lab = [
432   [-1, 1, 8, -1],
433   [-1, 2, -1, 0],
434   [-1, 3, 10, 1],
435   [-1, 4, -1, 2],
436   [-1, 5, -1, 3],
437   [-1, -1, 13, 4],
438   [-1, -1, -1, -1],
439   [-1, -1, 15, -1],
440   [0, -1, 16, -1],
441   [-1, -1, -1, -1],
442   [2, -1, 18, -1],
443   [-1, -1, 19, -1],
444   [-1, -1, -1, -1],
445   [5, 14, 21, -1],
446   [-1, 15, -1, 13],
447   [7, -1, -1, 14],
448   [8, 17, -1, -1],
449   [-1, 18, -1, 16],
450   [10, -1, 26, 17],
451   [11, 20, 27, -1],
452   [-1, 21, -1, 19],
453   [13, -1, 29, 20],
454   [-1, -1, -1, -1],
455   [-1, -1, 31, -1],
456   [-1, -1, 32, -1],
457   [-1, -1, -1, -1],
458   [18, -1, 34, -1],
459   [19, -1, -1, -1],
460   [-1, -1, -1, -1],
461   [21, 30, 37, -1],
462   [-1, 31, -1, 29],
463   [23, -1, 39, 30],
464   [24, 33, 40, -1],
465   [-1, 34, -1, 32],
466   [26, 35, 42, 33],
467   [-1, 36, -1, 34],
468   [-1, 37, 44, 35],
469   [29, -1, 45, 36],
470   [-1, -1, -1, -1],
471   [31, -1, -1, -1],
```

```
    [32, -1, -1, -1],
    [-1, -1, -1, -1],
    [34, -1, 50, -1],
    [-1, -1, -1, -1],
    [36, 45, -1, -1],
    [37, 46, 53, 44],
    [-1, 47, -1, 45],
    [-1, -1, 55, 46],
    [-1, 49, 56, -1],
    [-1, 50, -1, 48],
    [42, 51, 58, 49],
    [-1, -1, 59, 50],
    [-1, -1, -1, -1],
    [45, -1, -1, -1],
    [-1, -1, -1, -1],
    [47, -1, 63, -1],
    [48, -1, -1, -1],
    [-1, -1, -1, -1],
    [50, 59, -1, -1],
    [51, 60, -1, 58],
    [-1, 61, -1, 59],
    [-1, 62, -1, 60],
    [-1, 63, -1, 61],
    [55, -1, -1, 62]
];

// double cycle for traveling in maze
cycle = [0, 2, 4, 13, 15, 21, 19, 29, 31, 37, 45, 46, 55, 62, 60, 50, 48, 34,
  32, 18, 16, 0, 2, 4, 13, 15, 21, 19, 29, 31, 37, 45, 46, 55, 62, 60, 50, 48,
  34, 32, 18, 16
];

var foundedDestroyedSectors = 0;
var destroyedSectors = 0;

// получаем маршрут перемещения до необходимой точки из текущей
// в нотации F, L, R
function getPath(finishSector) {
  from = [];
  for (var i = 0; i < 64; i++) {
    from[i] = [];
    for (var j = 0; j < 4; j++)
      from[i][j] = "N";

  }

  var queue = [];
  queue.push([positionOfRobot, directionOfRobot]);
  var finishDir = 0;
  while (queue.length > 0) {
    temp = queue.shift();
    currentSector = temp[0], currentDir = temp[1];
    if (currentSector == finishSector) {
      finishDir = currentDir;
      break;
    }
    // Вперед
    if (lab[currentSector][currentDir] > -1) {
      adjSector = lab[currentSector][currentDir];
      adjDir = currentDir;
```

```
532         if (from[adjSector][adjDir] == "N") {
533           from[adjSector][adjDir] = "F";
534           queue.push([adjSector, adjDir]);
535         }
536       }
537       // Направо
538       if (from[currentSector][(currentDir + 1) % 4] == "N") {
539         adjSector = currentSector;
540         adjDir = (currentDir + 1) % 4;
541         from[adjSector][adjDir] = "R";
542         queue.push([adjSector, adjDir]);
543       }
544       // Налево
545       if (from[currentSector][(currentDir + 3) % 4] == "N") {
546         adjSector = currentSector;
547         adjDir = (currentDir + 3) % 4;
548         from[adjSector][adjDir] = "L";
549         queue.push([adjSector, adjDir]);
550       }
551     }
552
553     path = "";
554     // Восстанавливаем путь
555     if (from[finishSector][finishDir] == "N")
556       print("No way");
557     else {
558       currentSector = finishSector;
559       currentDir = finishDir;
560       while (currentSector != positionOfRobot || currentDir != directionOfRobot) {
561         action = from[currentSector][currentDir];
562         if (action == "F") {
563           path = "F" + path;
564           currentSector = lab[currentSector][(currentDir + 2) % 4];
565         } else if (action == "R") {
566           path = "R" + path;
567           currentDir = (currentDir + 3) % 4;
568         } else if (action == "L") {
569           path = "L" + path;
570           currentDir = (currentDir + 1) % 4;
571         }
572       }
573     }
574     return path;
575   }
576
577   // Внести информацию о недоступности сектора
578   function isolateSector(sector) {
579     foundedDestroyedSectors++;
580     for (dir = 0; dir < 4; dir++)
581       if (lab[sector][dir] > -1) {
582         lab[lab[sector][dir]][(dir + 2) % 4] = -2;
583         lab[sector][dir] = -2;
584       }
585     calcAvailable();
586     print("countUnavailable=", countUnavailable);
587   }
588
589   // Проверка окружающих секторов
590   function checkAdjacentSectors() {
591     if (!(svFront() > 25) &&
```

```
592           lab[positionOfRobot][directionOfRobot] > -1)
593       isolateSector(lab[positionOfRobot][directionOfRobot]);
594     if (!(svLeft() > 25) &&
595           lab[positionOfRobot][(directionOfRobot + 3) % 4] > -1)
596       isolateSector(lab[positionOfRobot][(directionOfRobot + 3) % 4]);
597     if (!(svRigth() > 25) &&
598           lab[positionOfRobot][(directionOfRobot + 1) % 4] > - 1)
599       isolateSector(lab[positionOfRobot][(directionOfRobot + 1) % 4]);
600   }
601
602   // Перемещение по кратчайшему пути до finishSector
603   function follow_path(finishSector) {
604     path = getPath(finishSector);
605     while (positionOfRobot != finishSector && path != "") {
606       if (foundedDestroyedSectors < destroyedSectors)
607         checkAdjacentSectors();
608       for (var i = 0; i < path.length; i++) {
609         if (path[i] == "R") {
610           turnDirection(-90, v);
611           directionOfRobot = (directionOfRobot + 1) % 4;
612         } else if (path[i] == "L") {
613           turnDirection(90, v);
614           directionOfRobot = (directionOfRobot + 3) % 4;
615         } else if (path[i] == "F") {
616           if (lab[positionOfRobot][directionOfRobot] < 0) {
617             print("The path is blocked. No way!");
618             break;
619           }
620           forward(v, 1);
621           positionOfRobot = lab[positionOfRobot][directionOfRobot];
622         }
623         if (foundedDestroyedSectors < destroyedSectors)
624           checkAdjacentSectors();
625         else if (foundedDestroyedSectors == destroyedSectors)
626           break;
627       }
628       if (foundedDestroyedSectors == destroyedSectors) {
629         foundedDestroyedSectors++;
630         break;
631       }
632       path = getPath(finishSector);
633     }
634   }
635
636   // Поворот робота на заданное направление
637   function turnTo(targetDir) {
638     var dDir = (targetDir - directionOfRobot + 4) % 4;
639     if (dDir == 1) {
640       turnDirection(-90, v);
641     } else if (dDir == 3) {
642       turnDirection(90, v);
643     } else if (dDir == 2) {
644       turnDirection(-180, v);
645     }
646     directionOfRobot = targetDir;
647   }
648
649   // Проход по лабиринту
650   function travelThroughoutMaze() {
651     var firstSector = 0;
```

```
652   for (firstSector = 0; firstSector < cycle.length / 2; ++firstSector) {
653     if (positionOfRobot == cycle[firstSector] || lab[positionOfRobot][0] ==
654       cycle[firstSector] || lab[positionOfRobot][1] == cycle[firstSector] ||
655       lab[positionOfRobot][2] == cycle[firstSector] ||
656       lab[positionOfRobot][3] == cycle[firstSector])
657       break;
658   }
659   for (i = firstSector; i < firstSector + cycle.length / 2; i++) {
660     follow_path(cycle[i]);
661     if (foundedDestroyedSectors >= destroyedSectors) {
662       break;
663     }
664   }
665 }
666
667 function dfs(sector) {
668   used[sector] = true;
669   countAvailable++;
670   for (var dir = 0; dir < 4; ++dir) {
671     nextSector = lab[sector][dir];
672     if (nextSector > -1 && !used[nextSector])
673       dfs(nextSector);
674   }
675 }
676
677 // Вычисление доступных/недоступных секторов
678 function calcAvailable() {
679   used = [];
680   for (var i = 0; i < 64; i++)
681     used = [];
682   // Сколько доступных секторов
683   countAvailable = 0;
684   dfs(positionOfRobot);
685   // Сколько недоступных секторов
686   countUnavailable = 64 - countAvailable - 12;
687 }
688
689 var value1, value2;
690 // Считывание двух ARTag маркеров
691 function readARTag(dist) {
692   forward(-v, dist);
693   value1 = getARTagValue();
694
695   forward(v, dist);
696   value2 = getARTagValue();
697   print(value1 + " " + value2);
698 }
699
700 //============================================================================
701 //=================================M A I N ==================================
702 var main = function() {
703
704   var xStart = 7;
705   var yStart = 1;
706   directionOfRobot = 3;
707   var xARTag = 5;
708   var yARTag = 6;
709   // С какой стороны от сектора распредления решений располагается ARTag
710   var directionOfARTag = 3;
711
```

```
  // Количество обрушенных секций
  destroyedSectors = 2;
  positionOfRobot = xStart + yStart * 8;

  travelThroughoutMaze();
  follow_path(xARTag + yARTag * 8);
  turnTo((directionOfARTag + 3) % 4);

  //detecting ARTag markers until it isn't successful
  value1 = 0;
  value2 = 0;
  dist = 0.3;
  while (value1 < 8 && value2 < 8 || value1 > 7 && value2 > 7) {
    readARTag(dist);
    dist += 0.05;
  }

  // Координаты финиша
  xFinish = 0;
  yFinish = 0;
  if (value1 < 8) {
    xFinish = value1;
    yFinish = value2 - 8;
  } else {
    xFinish = value2;
    yFinish = value1 - 8;
  }

  print(xFinish + " " + yFinish);
  finishSector = xFinish + yFinish * 8;
  follow_path(finishSector);
  print(countUnavailable);
  return;
}
```

Критерий определения победителей и призеров заключительного этапа

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.

$$S = S_1 + S_2$$

где S_1 — количество баллов, набранное в рамках индивидуальной части заключительного этапа (максимум 200 баллов); S_2 — количество баллов, набранное в рамках командной части заключительного этапа (максимум 400 баллов).

Критерий определения победителей и призеров:

	Призеры	**Победители**
Набранные баллы	от 100 до 270 баллов	от 270 баллов и выше