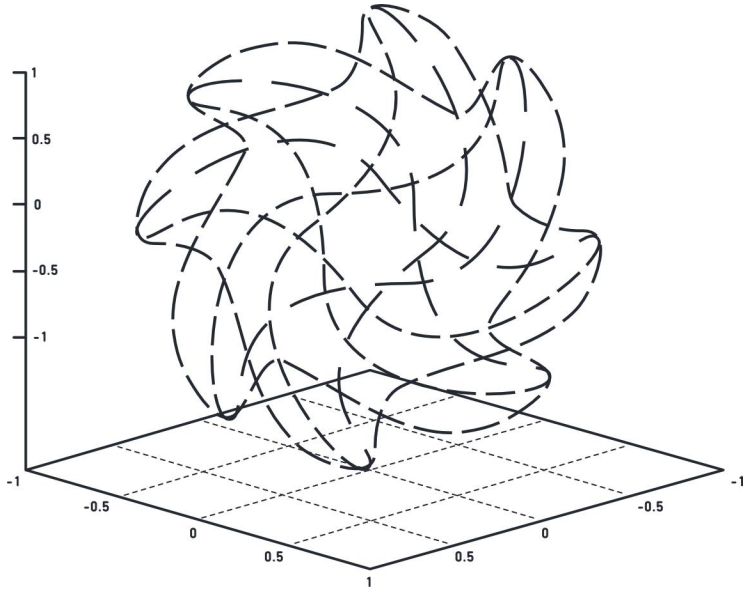


МАТЕРИАЛЫ ЗАДАНИЙ

командной инженерной олимпиады школьников «Олимпиада Национальной технологической инициативы»

по профилю «Интеллектуальные робототехнические системы»

2018/19 учебный год

http://nti-contest.ru

Оглавление

1 Введение **5**

2 Профиль «Интеллектуальные робототехнические системы» **27**

I Первый этап **29**

3 Задачи первого этапа. Математика. **31**

3.1 Первая попытка. Задачи 9 класса. 31

3.2 Первая попытка. Задачи 10-11 класса. 34

3.3 Вторая попытка. Задачи 9 класса. 39

3.4 Вторая попытка. Задачи 10-11 класса. 42

3.5 Третья попытка. Задачи 9 класса. 47

3.6 Третья попытка. Задачи 10-11 класса. 50

4 Задачи первого этапа. Углубленная информатика. **57**

4.1 Первая попытка. 57

4.2 Вторая попытка. 67

4.3 Третья попытка. 77

II Второй этап **90**

5 Задачи второго этапа **92**

5.1 Задачи 92

III Заключительный этап **180**

6 Предметный тур **181**

6.1 Математика. 9 класс 181

6.2 Математика. 10-11 класс 184

6.3 Информатика 186

7 Командный тур **198**

7.1 Легенда 198

7.2 Набор заданий 199

7.3 Описание модели логистического центра 201

7.4 Описание конструктора 203

7.5 Условия проведения 204

7.6 Процедура проведения приемочных запусков и критерии оценки 207

7.7 Решение 215

7.8 Критерии определения команды-победителя командного тура 228

IV Критерии **229**

8 Критерии определения победителей и призеров заключительного этапа **230**

8.1 Первый отборочный этап 230

8.2 Второй отборочный этап 230

8.3 Заключительный этап 230

Введение

Олимпиада Национальной технологической инициативы (далее – Олимпиада НТИ)[1] – 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», «Нейротехнологии», «Инженерные биологические системы: агробиотехнологии и геномное редактирование», «Интеллектуальные робототехнические системы», «Беспилотные авиационные системы», «Композитные технологии», «Когнитивные технологии», «Аэрокосмические системы», «Наносистемы и наноинженерия», «Технологии беспроводной связи», «Умный город», «Передовые производственные технологии», «Виртуальная и дополненная реальность», «Анализ космических снимков и геопространственных данных», «Водные робототехнические системы» и «Программная инженерия финансовых технологий».

Цель Олимпиады НТИ: поддержка школьников в стремлении решать технологические вызовы XXI века (что подразумевает включение их в решение технологических задач переднего края и, одновременно, повышение социальной значимости такой работы старшеклассников через льготы к поступлению). Эта цель лежит в рамках большей цели Кружкового движения: формирование и подготовка команд, способных запускать глобальные технологические проекты, менять мир, создавая новые общественные практики. Именно участники этих команд должны будут через 10-15 лет «перезапустить» НТИ: создать собственные рынки и глобальные прорывные компании. Важной особенностью олимпиады является то, что в части отборочного и в заключительном этапах участники выполняют задания в командах по 2-4 человека. Умение работать в команде - важный навык человека 21 века. Команды формируются на основе компетентностного принципа, различные компетенции участников в одной команде позволяют найти оригинальное нестандартное решение задачи. В командах участники планируют свою работу, обсуждают, ищут совместно решения, распределяют роли - часто один участник выполняет несколько ролей. Комплекс-

[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.

ные инженерные задачи, которые решают участники, не под силу решить отдельно взятому школьнику. Задачи разработаны таким образом, что декомпозируются на несколько подзадач, за решение, которых берутся участники согласно своей роли в команде. Каждый участник несет ответственность за результат работы команды. Поэтому, при подведении итогов олимпиады определяются не только победители в личном зачете, но и команда-победитель.

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 2015/2016 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 2015/2016 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.

Второй год проведения олимпиады

В 2016/2017 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 Образовательного центра «Сириус», в лабораториях и помещениях Парка Наук и Искусств. Вечером проходили лекции и неформальные встречи с представителями технологических компаний.

В 2016/2017 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 (в зависимости 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 2017/2018 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 Образовательный центр «Сириус», МАИ, МИФИ, ТПУ, Университет Иннополис, СПбПУ, ДВФУ, УрФУ. В 2017/2018 учебном году девять из 14 профилей Олимпиады НТИ включены 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 110, 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/`.

Четвертый год проведения олимпиады

В отборе на Олимпиаду 2018/2019 учебного года приняло участие более 36 тыс. школьников, подавших более 70 тыс. заявок на различные профили, число которых увеличилось до 19. В финал вышли 1053 участника Олимпиады из 60 регионов РФ.

Финал стал распределенным и проходил с марта по апрель 2019 года: Олимпиаду приняли МФТИ, МАИ, МИФИ, ТПУ, Университет Иннополис, СПбПУ, ДВФУ, НГУ, НовГУ, Московский Политех, ИГУ, ИРНИТУ и ряд других площадок. В 2018/2019 учебном году 13 из 19 профилей Олимпиады НТИ включены в Перечень олимпиад школьников (приказ №32н от 28 августа 2018 года Министерства науки и высшего образования Российской Федерации) и дают льготы при поступлении в вузы.

В олимпиаде в 2018/2019 учебном году впервые были проведены синхронные по времени распределенные финалы на площадках в разных городах в рамках одного профиля: Нейротехнологии (ДВФУ, НГУ, МФТИ, НовГУ), ИЭС (ИГУ, МИФИ), Нанотехнологии (Школа Летово, НГУ), АТС (Московский политех, НовГУ). Участники распределенных финалов имели одинаковые задания, критерии оценивания и единый рейтинг участников.

График проведения заключительных этапов Олимпиады НТИ 2018/2019 гг.

Площадка проведения	**Даты проведения**	**Перечень профилей Олимпиады НТИ**
Университет Иннополис (г. Иннополис)	3-11 марта 2019 г.	Интеллектуальные робототехнические системы
Университет Иннополис (г. Иннополис)	6-11 марта 2019 г.	Программная инженерия финансовых технологий
Школа Летово (г. Москва) **Новосибирский государственный университет** (г. Новосибирск)	11-16 марта 2019 г.	Наносистемы и наноинженерия
Московский политехнический университет (г. Москва)	11-16 марта 2019 г.	Инженерные биологические системы: Агробиотехнологии
Московский авиационный институт (г. Москва)	11-16 марта 2019 г.	Беспилотные авиационные системы
Томский политехнический университет (г. Томск)	12-17 марта 2019 г.	Умный город
Иркутский национальный исследовательский технический университет (г. Иркутск)	13-19 марта 2019 г.	Технологии беспроводной связи
Иркутский государственный университет (г. Иркутск) **Национальный исследовательский ядерный университет «МИФИ»** (г. Москва)	13-19 марта 2019 г.	Интеллектуальные энергетические системы
Иркутский государственный университет (г. Иркутск)	13-19 марта 2019 г.	Технологии виртуальной и дополненной реальности: Дополненная реальность

Дальневосточный федеральный университет (г. Владивосток)	18-23 марта 2019 г.	Виртуальная и дополненная реальность: Виртуальная реальность
Дальневосточный федеральный университет (г. Владивосток)	18-23 марта 2019 г.	Водные робототехнические системы
Московский физико-технический институт (г. Москва) **Новосибирский государственный университет** (г. Новосибирск) **Новгородский государственный университет имения Ярослава Мудрого** (г. Великий Новгород) **Дальневосточный федеральный университет** (г. Владивосток)	18-23 марта 2019 г.	Нейротехнологии
Московский физико-технический институт (г. Москва), **Новосибирский государственный университет** (г. Новосибирск)	18-23 марта 2019 г.	Инженерные биологические системы: Геномное редактирование
Московский физико-технический институт (г. Москва)	18-23 марта 2019 г.	Большие данные и машинное обучение
АО «ИПК Машприбор» ГК Роскосмос (г. Королев) **Детский технопарк «Кванториум»** (г. Королев)	26-31 марта 2019 г.	Системы связи и дистанционного зондирования Земли
АО «ИПК Машприбор» ГК Роскосмос (г. Королев) **Детский технопарк «Кванториум»** (г. Королев)	26-31 марта 2019 г.	Аэрокосмические технологии
АО «ИПК Машприбор» ГК Роскосмос (г. Королев) **Детский технопарк «Кванториум»** (г. Королев)	26-31 марта 2019 г.	Анализ космических снимков и пространственных геоданных Земли
Санкт-Петербургский университет Петра Великого, **Академия цифровых технологий** (г. Санкт-Петербург)	01-06 апреля 2019 г.	Передовые производственные технологии
Московский государственный психолого-педагогический университет (г. Москва)	02-06 апреля 2019 г.	Когнитивные технологии

Московский государственный технический университет им. Н.Э. Баумана (г. Москва)	07-12 апреля 2019 г.	Композитные технологии
Московский политехнический университет (г. Москва) **Новгородский государственный университет имения Ярослава Мудрого** (г. Великий Новгород)	08-14 апреля 2019 г.	Автономные транспортные системы

Организаторы и партнеры Олимпиады НТИ

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

Вузы-соучредители олимпиады:

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

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 выступают: Аэрофлот, ПАО «Сухой», ОАК, Роскосмос, ФИОП Роснано, МТС, Газпром нефть, Фонд новых форм развития образования, сеть детских технопарков «Кванториум», Спутникс, Полюс-НТ, BiTronicsLab, КРОК, Инфосистемы Джет, Лоретт, Коптер Экспресс, АсРоботикс, Образование будущего 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:

- АНО ВО «Университет Иннополис»;
- ФГАОУ ВО «Национальный исследовательский ядерный университет «МИФИ»;
- ФГАОУ ВО «Московский физико-технический институт (государственный университет)»;
- ФГБОУ ВО «Московский авиационный институт (национальный исследовательский университет)»;
- ФГБОУ ВО «Сибирский государственный университет науки и технологий

имени академика М.Ф. Решетнева»;

- ФГАОУ ВО «Новосибирский национальный исследовательский государственный университет»;
- АНО ВО «Сколковский институт науки и технологий»;
- ФГБОУ ВО «Московский государственный психолого-педагогический университет»;
- ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»;
- ФГБОУ ВО «Новгородский государственный университет имени Ярослава Мудрого»;
- ФГБОУ ВО «Иркутский государственный университет»;
- ФГБОУ ВО «Новосибирский государственный технический университет»;
- ФГАОУ ВО «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»;
- ФГБОУ ВО «Московский технический университет связи и информатики».

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.

Университет Иннополис:

Университет Иннополис – интеллектуальное ядро нового города и российский университет, который специализируется на образовании и научных исследованиях в области современных информационных технологий. Основная цель создания университета – подготовка высококвалифицированных кадров по ИТ -специальностям. В рамках работы над этой целью в университете организовано подразделение по подготовке школьников. Основное направление работы подразделения – это выявление и развития талантливых школьников, а также развитие IT компетенций в регионе и стране.

В течение года университет проводит большое количество образовательных мероприятий и интеллектуальных конкурсов по различным направлениям.

Работа по выявлению и развитию талантов школьников проходит по шести направлениям:

- Регулярные занятия
- Региональные образовательные смены
- Федеральные образовательные смены
- Инженерные, научно-технические конкурсы, соревнования и олимпиады
- Учебно-тренировочные сборы для финалистов, представителей республики и страны
- Курсы повышения квалификации для педагогов

Регулярные занятия проводятся для учащихся 4 – 11 классов на площадках школ

города Казань, города Иннополис и Университета Иннополис по направлениям «программирование», «олимпиадное программирование», «математика», «олимпиадная математика», «олимпиадная робототехника», «прототипирование и 3D моделирование» и «компьютерное зрение».

Региональные образовательные смены проводятся в каникулярное время только для учащихся 1 – 11 классов из Республики Татарстан. Основные направления подготовки в рамках таких смен:

- «программирование»,
- «математика»,
- «робототехника»

Федеральные образовательные смены также проводятся исключительно в каникулярное время, для учащихся 7 – 11 классов российских школ. Для старшей возрастной группы смены проходят по общим предметам:

- Информатика
- Математика

И по узким направлениям подготовки:

- Введение в олимпиадное программирование
- Интеллектуальные робототехнические системы
- Информационная безопасность сетей и систем
- Программная инженерия финансовых технологий
-

А также проектная школа, которая проводится по направлениям, над которыми работают лаборатории университета и индустрия города. Участники – ученики 9 – 11 классов российских общеобразовательных учреждений.

Ежегодно в Университете проводятся олимпиады, конкурсы и соревнования по всем вышеперечисленным направлениям подготовки.

Среди них Олимпиады, которые входят в перечень РСОШ:

- Олимпиада Университета Иннополис (Innopolis Open) по математике, для учеников 7 – 11 классов
- Олимпиада Университета Иннополис (Innopolis Open) по информатике, для учеников 7 – 11 классов

Оба профиля с 2016 входят в перечень РСОШ. Олимпиаде по информатике присвоен первый уровень, по математике – третий.

- Олимпиада НТИ, профиль «Интеллектуальные робототехнические системы», проводится Университетом для учеников 8 – 11 классов с 2016 года, последние 2 сезона присвоен 3 уровень в перечне РСОШ.
- Олимпиада НТИ, профиль «Программная инженерия финансовых технологий», в 2017 году проводилась ФинТех олимпиада Университета Иннополис, с 2018 года выступает как часть Олимпиады НТИ, в 2019 году проходила как Олимпиада третьего уровня перечня РСОШ.

Помимо этого, проходят Олимпиады, не входящие в перечень РСОШ:

- Всероссийская робототехническая олимпиада, с 2014 года Университет явля-

ется национальным организатором World Robot Olympiad, отвечает за проведение российского этапа и формирует сборную России для участия в международном этапе. По соглашению с университетом в 53 регионах России проводятся региональные отборочные этапы, суммарно в них принимают участие более 10 000 школьников 4 – 11 классов

- Олимпиада Университета Иннополис (Innopolis Open) по информационной безопасности для учеников 9 – 11 классов, новое направление, в 2018 году мероприятие проходило в формате CTF.
- STEM Олимпиада по информатике, для учеников 5 – 7 классов
- STEM Олимпиада по математике, для учеников 5 – 7 классов
- Научно-технический конкурс работ школьников РОСТ. Конкурс проводится Университетом с 2018 года, до 2018 года оказывал методическую поддержку конкурса. В конкурсе могут принимать участие ученики 7 – 11 классов, среди которых отбирается 4 команды – представители сборной России на международном соревновании Intel isef.

На площадке Университета также проходят мероприятия регионального и федерального уровня, различные смены по математике и информатике, проводимые партнерскими организациями и олимпиады, такие как Высшая проба и ВсОШ.

Для успешного выступления участникам и командам на соревнованиях необходимо хорошая подготовка. Университет проводит учебно-тренировочные сборы для финалистов конкурсов и олимпиад, для членов сборной республики Татарстан на федеральных конкурсах и для членов сборной России на международных конкурсах.

- Ежегодно проходят сборы для финалистов ко всем олимпиадам, в формате 2-3х дневных хакатонов и 7 дневных школ.
- Сборы для членов сборной республики Татарстан к различным федеральным Олимпиадам и соревнованиям по математике, информатике и робототехнике, например: ВРО, ВсОШ по информатике, Турнир двух Столиц.
- Сборы для членов сборной России и Татарстана к международным соревнованиям, например: WRO, WRO FIT, WRO ARC, eJOI, IATI, Intel isef.

Университет активно работает со школьными учителями и педагогами дополнительного образования, наставниками и руководителями команд, организуются курсы подготовки для педагогов по олимпиадной робототехнике (профили, которые представлены на олимпиаде ВРО) и программированию. Помимо этого, проводятся конференции и семинары для педагогов.

Ежегодно в образовательных сменах принимают участия более 1000 участников, а через олимпиады проходят более 20 000 школьников.

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 проекты Департамента образования г. Москвы «Центр технологической поддержки образования» (с 2013 года) и «Инженерный класс в мос-

ковской школе» (с 2015 года) 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 41 школе города Москвы (более 3000 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 года открыла кружки для старшеклассников (8-11 класс):

- Космическая инженерия;
- Кружок схемотехники и микроэлектроники;
- Техника низких температур;
- Инфракрасные технологии и радиоэлектроника;
- Программирование на C++;
- Аквапонные системы;
- Автомобилестроение;
- Беспилотный транспорт;
- Кружок 3D моделирования и прототипирования и другие.

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 года 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 года факультет организует участие Московского Политеха в ежегодном форуме талантливых детей «Проектория» в г. Ярославле. Форум проводится под руководством аппарата полномочного представителя Президента Российской Федерации в ЦФО и Министерства образования и науки Российской Федерации. В форуме принимают участие до 500 школьников со всей страны. В ноябре 2016 года и сентябре 2017 года Московский Политех выступил партнером образовательной программы форума в рамках направления «Технологии движения». В ноябре 2018 года Московский Политех принимал участие в фестивале “Билет в будущее” в рамках

направлений “Космос”, “Транспорт”, “Новые материалы”, “Умная среда”.

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

В январе 2018 года, июне 2018 года, октябре 2018 года и феврале 2019 года совместно с Центром педагогического мастерства (Департамент образования гор. Москвы) факультет провел выездную инженерную школу Московского Политеха для учащихся инженерных классов города Москвы, приняли участие 160 человек.

В марте 2018 года поддержке Департамента науки, промышленной политики и предпринимательства города Москвы в Московском Политехе открылся детский технопарк Центра развития инжиниринга - инженерно-технологический комплекс, на базе которого проводятся углубленные технико-ориентированные курсы дополнительного образования для школьников. 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:

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 соорганизатора) в следующих мероприятиях: инженерно-конструкторское направление предпрофессиональной олимпиады Московской олимпиады школьников 2016-19 гг., предпрофессиональный экзамен для инженерных классов в Москве 2016-2019 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

В 2010 году получил статус национального исследовательского университета, что явилось признанием его роли и возможностей как в области подготовки кадров, так и в мультидисциплинарных научных исследованиях и разработках. В рейтинге технических университетов России Политехнический неизменно занимает ведущие позиции.

СПбПУ активно работает со школьниками со всей страны. В ВУЗе работает Центр профориентации и довузовской подготовки, где учащиеся могут получить дополнительные знания по школьным предметам основного образования. Также активную работу со школьниками ведет Центр научно-технического творчества молодежи

Фаблаб Политех.

В рамках довузовской подготовки в Университете успешно функционируют такие проекты, как «Вызов Политехника», «Мой город цифровой» и проектные интенсивы для школьников от Фаблаб Политех, где более 3 000 учащихся проходят обучение по передовым направлениям дополнительного образования. Подшефные школьники ежегодно демонстрируют высокие результаты на Всероссийских и международных конкурсах для молодых профессионалов: WorldSkills, Реактор, Олимпиада НТИ, Техномейкер, Шустрик, Кубок ЦНИИ РТК. Университет активно работает со школьными учителями и педагогами дополнительного образования, организуются курсы повышения квалификации для педагогов по программам дополнительного образования, проводятся собственные конкурсы для учащихся и педагогические конференции.

Томский политехнический университет

Сегодня ТПУ – опорный вуз для крупнейших государственных корпораций, среди которых «Газпром», «Росатом», АО «"Информационные спутниковые системы" имени академика М.Ф. Решетнева», «Микроген», «Системный оператор ЕЭС», «РАО Энергетические системы Востока».

В 2009 году в ТПУ был запущен Интернет-лицей, позволяющий школьникам подготовиться к вступительным испытаниям в режиме онлайн.

Для школьников и их учителей, занимающихся исследовательской работой, мы проводим ежегодные конференции. Это - Всероссийская конференция-конкурс исследовательских работ старшеклассников «Юные исследователи - российской науке и технике», Межрегиональная научно-практическая конференция для учителей «Организация исследовательской деятельности детей и молодежи: проблемы, поиск, решения», конкурс учителей физики «От школьной физики – к высоким технологиям» и конкурс учителей химии «Мой выбор – химия».

Лицей при Томском политехническом университете создан в 1992 г. Лицей имеет физико-математический профиль и полностью располагается на площадях университета. В 2015 г. в Лицее при ТПУ открыт первый в Сибири профильный класс компании «Газпром». По результатам итоговой аттестации в форме ЕГЭ лицей занимает лидирующие позиции в регионе, демонстрируя при этом постоянную положительную динамику. Лицей при ТПУ входит в ТОП-10 рейтинга лучших школ по качеству подготовки к поступлению в ведущие высшие учебные заведения России и топ-500 лучших школ России по результатам рейтинга, составленного Московским центром непрерывного математического образования. Все выпускники лицея ТПУ поступают в вузы. Лицеисты – постоянные участники и дипломанты Международных научно-технических конференций школьников, проводимых МГУ, МФТИ, НИЯУ МИФИ и др.

Дальневосточный федеральный университет (г. Владивосток) – один из крупнейших университетов на Дальнем Востоке России.

Дальневосточный федеральный университет уделяет большое внимание привлечению абитуриентов со всей страны, выявлению и поддержке талантливых школьников в области инженерных и естественных наук.

1. Олимпиады. Дальневосточный федеральный университет проводит три собственные олимпиады:
    - «Океан знаний». Предметы: математика, физика, химия, биология,

география, русский язык, литература, история и обществознание.

- «Ближе к Дальнему». Предметы: история (включая культурологию), география (включая экономику), биология, филология (включая литературу и лингвистику), международные отношения и политология.
- «Турнир юных программистов». Предметы: программирование.

В 2018/2019 году ДВФУ стал площадкой для Всероссийских олимпиад:

- Олимпиада Национальной технологической инициативы (НТИ)
- Всероссийская олимпиада школьников (региональный этап)
- Евразийская лингвистическая олимпиада
- Физико-математическая олимпиада «Физтех»
- Всероссийская командная школьная олимпиада по программированию
- Северо-Восточная олимпиада школьников
- Объединенная межвузовская олимпиада по математике
- Открытая олимпиада по экономике
- Олимпиада для школьников «Ломоносов»
- Объединённая межвузовская математическая олимпиада
- Олимпиада СПбГУ
- Математическая олимпиада им. В.Б. Осипова

С 2012 года в рамках смены «Российский интеллект» реализуется совместная образовательная программа Дальневосточного федерального университета и Всероссийского детского центра «Океан» для победителей и призеров региональных этапов Всероссийской олимпиады школьников и победителей/призеров отборочного этапа олимпиады школьников «Океан знаний».

2. Проекты дополнительного образования для талантливых школьников:
   - «Тихоокеанские Школы ДВФУ»: учебно-тренировочные сборы. Проводятся по предметам: математика, программирование, английский язык, китайский язык. В год на базе ДВФУ проходит 4 сессии, когда школьники на неделю погружаются в интенсивное изучение предмета. В 2017/2018 годах в «Тихоокеанских школах ДВФУ» приняли участие более 700 школьников, 140 преподавателей прошли курсы повышения квалификации.
   - «Тихоокеанская проектная школа». Совместный образовательный проект Дальневосточного федерального университета и Всероссийского детского центра «Океан».Для участия в конкурсном отборе необходимо подать заявку с идеей по развитию Дальнего Востока.Участниками «Тихоокеанской проектной школы» в июне-июле 2017 года стали 100 старшеклассников в возрасте 15-17 лет. За три недели они подготовили проекты по четырем направлениям:инженерное,естественнонаучное, социально-гуманитарное и современные информационные технологии. Авторы лучших проектов представили свои разработки на III Восточном экономическом форуме, который прошел в ДВФУ в сентябре 2017 г.
   - Образовательная программа «Яндекс.Лицей». Проект стартовал на

базе ДВФУ в 2017 году. Ученики 8–9 классов дважды в неделю осваивают программирование на языке Python. Для лицеистов ДВФУ проводит «Хакатон» – трехдневное интенсивное погружение в предмет. Обучение бесплатное. Поступление на конкурсной основе. В 2018/2019 учебном году ДВФУ планирует вдвое увеличить количество учеников «Яндекс.Лицея».

- Роснефть классы. Роснефть-класс – профильный класс, сформированный в результате конкурсного отбора учащихся, обучающихся по углубленным программам физики, математики, информатики, ориентированный на выбор профессий, связанных с судоремонтом и судостроением.
- JUNIOR POCT. Программа бизнес-школы «JUNIOR POCT» направлена на развитие способностей предпринимательства у школьников от зарождения предпринимательской идеи до реализации, организации совместных проектных групп из обучающихся и представителей бизнес-среды, получению дополнительных знаний из других областей, нахождению единомышленников. Данное направление апробировано в 2018 году в рамках Русско-азиатской бизнес-школы «РОСТ».
- Взаимодействие с Образовательным центром «Сириус». «Социальный лифт» – организация и проведение региональных треков Всероссийских мероприятий Образовательного центра «Сириус». Обеспечение раннего выявления, развития и дальнейшая профессиональная поддержка одаренных детей, проявивших выдающиеся способности в к проектной, научной (научно-исследовательской), инженерно-технической, изобретательской, творческой деятельности.
- Кадры будущего. Проект направлен на поддержку талантливых школьников и студентов, неравнодушных к судьбе Приморского края и готовых включиться в реализацию проектов в важных для региона социально-экономических направлениях развития.В рамках проекта участникам будет предоставлена возможность разработать проект, а также пройти стажировку на предприятиях в разных отраслях экономики: транспорт и логистика, машиностроение и судоремонт, рыбная промышленность, сельское хозяйство и пищевая промышленность, сервис и туризм, строительство и умный город, экология и марикультура.

Структура отбора участников Олимпиады НТИ

Соревнование проходит в три этапа. Первый и второй отборочные этапы проходили с октября по декабрь 2018 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, создана специальная интерактивная форма формирования и подбора членов команд на платформе Олимпиады НТИ), мероприятия, направлен-

ные 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.

Информация о профиле

Участники профиля «Интеллектуальные робототехнические системы» в 2017 году решали задачу навигации и локализации робототехнического устройства в помещении, особенность которого – отсутствие направляющих линий или визуально различимых ориентиров. Т.е. участникам нужно будет продумать способы навигации, основанные на определении расстояний до препятствий, а также пройденных расстояний.

На фото: участники профиля «Интеллектуальные робототехнические системы» вокруг полигона в финале в 2017 году

Задачей 2018 года было решить задачу распознавания графических кодов и навигации робототехнического устройства в помещении, особенность которого – отсутствие направляющих линий или визуально различимых ориентиров.

На фото: Участники профиля «Интеллектуальные робототехнические системы» в 2018 году наблюдают за работой робототехнического устройства.

В 2019 году задача заключительного этапа требовала от участников навыков реализации алгоритмов передачи информации и принятия решений в многоагентных системах. Участники пытались запрограммировать группу робототехнических устройств, которые моделируют роботов-погрузчиков, так чтобы те перемещались по модели логистического центра без направляющих линий или визуально различимых ориентиров и координировали свои действия с другими устройствами в группе, избегая столкновений и передавая информацию об обнаруженных ARTag метках.

На фото: Участники профиля «Интеллектуальные робототехнические системы» в 2019 году запускают робота на полигоне.

В состав команды-разработчиков задач 2019 года входят финалисты, призеры и победители профиля за прошедшие сезоны.

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 2018/2019 учебного года в качестве волонтеров приняли участие победители и призеры Олимпиады НТИ 2017/2018 учебного года, студенты первых курсов из различных регионов России. Участники заключительного этапа 2018/2019 учебного года из числа учеников одиннадцатого класса также выразили желание принять участие в организации олимпиады и подготовке участников в качестве волонтеров.

В 2018/2019 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, члены команд-победителей финалов Олимпиады были приглашены на образовательный интенсив «Остров 10-22», проходящий в Сколково летом 2019 года. На авиационную смену в «Артеке» получили приглашение лучшие участники профиля «Беспилотные авиационные системы». Отбор на июльскую проектную смену в Образовательный центр «Сириус» предполагает дополнительные баллы для призеров и победителей Олимпиады НТИ.

Партнерство с инженерными соревнованиями

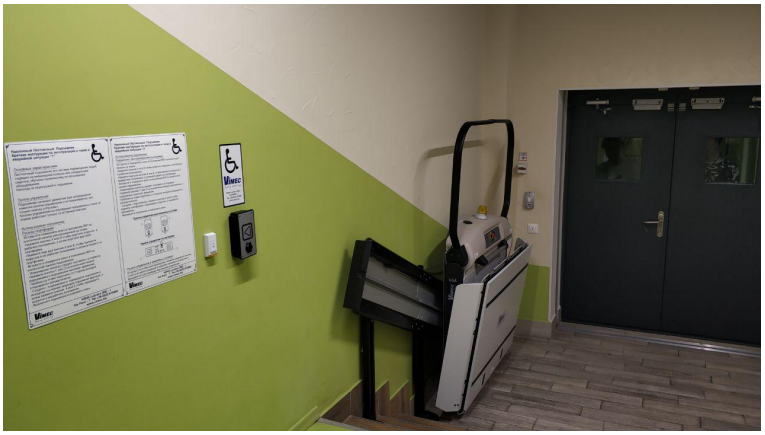
Оргкомитет Олимпиады НТИ, в свою очередь, ежегодно утверждает перечень инженерных мероприятий и конкурсов, победители которых, могут принять участие в заключительном этапе Олимпиады, минуя отборочные. В 2016/2017 учебном году таковыми мероприятиями являлись: IT-хакатон GoTo, инженерно-конструкторские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

В 2017/2018 учебном году льготы предоставлялись победителям мероприятий: всероссийский форум «Будущие интеллектуальные лидеры России», чемпионат «WorldSkills Abu Dhabi» и «World Skills High Tech (Junior)», Воздушно-инженерная школа МГУ, региональный этап международных соревнований по подводной робототехнике «Russia Far-East MATE ROV Competition», Всероссийская Робототехническая Олимпиада.

В 2018/2019 году напрямую во второй этап Олимпиады получили доступ победители Региональных чемпионатов WorldSkills Junior Russia, Всероссийской робототехнической олимпиады, Олимпиады «Шаг в Будущее», Russia Far-East MATE ROV Competition, Russian Self-Driving Challenge, трека «Микробный топливный элемент» конкурса icet2018.ru, конкурса «Энергопрорыв» 2017/2018, Всероссийской олимпиады по 3D технологиям «Робофинист», Олимпиада «Кибервызов» компании Ростелеком, проектных смен Образовательного центра «Сириус» и всероссийских олимпиад школьников 1-3 уровней.

Равные возможности для участников с ограниченными возможностями здоровья

Организаторы Олимпиады НТИ соблюдают принцип равных возможностей и доступности участия школьников с ограниченными возможностями здоровья. В Олимпиаде беспрепятственно могут участвовать школьники с с ограниченными возможностями здоровья, обучающиеся дети-инвалиды, а также те, кто обучался по состоянию здоровья на дому. Важное условие для участия в олимпиаде детей с ОВЗ и инвалидностью - способность выполнять инженерные работы и работать в команде.



Отборочные этапы олимпиады проводятся дистанционно, это позволяет детям 7-11 классов с ОВЗ и инвалидам решать задания в домашних условиях или в образовательной организации, оборудованной с учетом их индивидуальных особенностей.

Для проведения заключительного этапа соревнований были выбраны площадки с соответствующими материально-техническими условиями, которые обеспечивают: возможность беспрепятственного доступа участников олимпиады в аудитории, туалетные и иные помещения, а также их пребывания в указанных помещениях; наличие пандусов, поручней, расширенных дверных проемов, лифтов, при отсутствии лифтов аудитория располагается на первом этаже наличие специальных кресел и других приспособлений.

Большинство площадок проведения финалов олимпиады оснащены паспортами доступности для инвалидов объектов и предоставляемых в этих объектах услуг.

При проведении олимпиады в случае необходимости возможно сопровождение детей с ОВЗ ассистентами (сопровождающие лица или родители), оказывающими участникам с ОВЗ, детям-инвалидам и инвалидам необходимую техническую помощь с учетом их индивидуальных особенностей, помогающими им занять рабочее место, передвигаться, прочитать задания.

В заключительных этапах олимпиады 2018/2019 учебного года приняло участие 9 школьников с ОВЗ, для которых были созданы все необходимые условия для полноценной работы и своевременного оказания необходимой медицинской и иной помощи. Во время проведения олимпиады сопровождающие могли сделать экспресс-анализ крови, дать при необходимости лекарство, сделать укол. На каждой площадке проведения олимпиады находился дежурный врач, который оказывал необходимую медицинскую помощь в т.ч. и детям с ОВЗ.

Подготовка участников

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

«Урок НТИ» (`http://nti-contest.ru/ntilessonteacher/`) – это инициатива Олимпиады НТИ, проходившая в сентябре 2018 года и направленная на распространение информации об НТИ среди школьников и привлечение их к Олимпиаде НТИ через проведение уроков и занятий в школах и учреждениях дополнительного образования. Учебный материал для проведения «Урока НТИ» сформирован в виде конструктора, с помощью которого учителя могли собрать урок по теме НТИ. Урок позволяет познакомить учащихся НТИ и с профилями Олимпиады НТИ, организовать практическую работу по решению задач в рамках выбранного профиля. Для

участия в проекте «Урок НТИ» зарегистрировалось 2185 педагогов.

Демо-этап Олимпиады НТИ (`https://stepik.org/course/24389/syllabus`) – это публикация задач олимпиады в открытом доступе. Демо-этап создан для знакомства с задачами по профилям олимпиады, тренировки и испытания собственных знаний и умений решать непростые инженерные задачи. Прежде чем определиться с участием в олимпиаде и выбором профиля, потенциальные участники и их наставники могут познакомиться с задачами и выбрать наиболее интересный для себя профиль.

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, совместно с площадками подготовки проводят хакатоны для участников с возможностью попробовать на практике фрагменты финальной задачи.

Команды разработчиков профилей с целью эффективной подготовки к Олимпиаде НТИ создали видео разборы задач 2 этапа, которые доступны на канале Олимпиады НТИ, `https://www.youtube.com/channel/UCZV1CNpOrDNj7tuWuf35lgw/playlists` в 2018 году разработан курс (веб-сайт: `https://stepik.org/course/15697/syllabus`) по подготовке школьников к Олимпиаде НТИ на основе контента олимпиады 2017/2018 учебного года. Курс содержит задачи предметных треков 1 и 3 этапа по предметам: математика, физика, информатика, химия и биология и задачи 2 этапа по профилям олимпиады. Курс может использоваться наставниками и самими участниками для подготовки к олимпиаде следующего года. Формат курса максимально приближает участников к реальным условиям олимпиады.

Все указанные материалы находятся в свободном доступе и размещены на официальном сайте олимпиады, на страницах профилей в разделе «Материалы для участников».

Материалы по профилю «Интеллектуальные робототехнические системы»: `http://nti-contest.ru/profiles/irs/`

Олимпиада НТИ 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 широко освещается в различных средствах массовой информации (телевидение, печатные издания, электронные издания). В период с 15 августа 2018 года (начало подготовки к регистрациям) до 3 апреля 2019 года, по данным Медиалогии, Олимпиада НТИ упоминалась в СМИ 2 301 раз, из них 633 раза на федеральном уровне, 1655 на региональном уровне, 13 на зарубежном уровне.

Во время проведения отборочных этапов Олимпиада НТИ освещалась в федеральных, массовых, родительских, образовательных и иных медиа («ИТАР-ТАСС», «РИА-Новости», «Интерфакс», «Такие Дела», «Летидор», «Дети.Мэйл.ру», «Индикатор», «Занимательная робототехника», «Чердак.Ру», «Habrahabr», «Rusbase», «Учёба.Ру»), официальных образовательных порталах и порталах органов государственности власти в регионах (Новосибирск, Санкт-Петербург, Великий Новгород, Иннополис, Томск, Владивосток, Калининград, Тюмень, Курск, Курган, Тамбов, Мурманск, Новогород, Вологда и т.д.), в печатных изданиях («Российская газета», «Известия»). Радио «Медиаметрикс», программа «Выбор Родителей» под руководством автора самого большого блога для родителей в России. Кампания по привлечению шла также в научно-популярных группах и группах вузов и площадок партнеров (МАИ, НГУ, Абитуриент НГУ, ДВФУ, Школьники ДВФУ, Абитуриенты ДВФУ, Мурманский Арктический государственный университет, СибГУ им. М.Ф. Решетнева, Кванториум, АНО ДТ Красноярский кванториум, НовГУ, ТПУ, абитуриенты ТПУ, МГППУ, Московский Политех, школа Летово, технопарк Академгородка, Сколтех, Иннополис, группы довузовского управления университета Иннополис, Политех Петра, ИТМО, ИРНИТУ, студенты ИРНИТУ, АУ УР РЦИиОКО, Детский технопарк INGENERIKA, Инкубатор Профи, Центр компетенций для детей Поколение 2035, Лаборатория НГУ Инжевика, Чеченский государственный университет, Айти школа Орбита, Фонд Книту, Фонд Золотое сечение, ЦМИТЫ Коптер, Ноосфера, Рекорд, Уникум, Stem-Байкал, Роболатория, Академия Технолаб, Образовательный проект для подростков Tula Teens , Проектория, ЯКласс, Фаблаб Политех и другие). Заключительный этап Олимпиады НТИ в 2018/2019 учебном году проходил при участии журналистов таких печатных изданий, как «Российская газета», «Известия»; федеральных телевизионных каналов («Россия 24» (6 репортажей), «Общественное телевидение России»), федеральных новостных агентств («РИА-Новости», «ИТАР-ТАСС», «Интерфакс») научно-популярных порталов «Rusbase», «Habrahabr», «Индикатор», «Такие Дела», радиостанций («Радио России», «МедиаМетрикс») Профильные издания освещали соответствующие направления Олимпиады НТИ («Крылья Родины» – «Беспилотные авиационные системы»). В ходе финалов Олимпиады НТИ были инициированы события, вызывающие дополнительный интерес как со стороны участников, так и со стороны СМИ. Так, например в рамках финала в Новосибирском государственном университете, участники встретились с Нобелевским лауреатом Хироси Амано, информация об этом событии была распространена ведущими федеральными агентствами и телеканалами. Разработка победителей профиля «Нейротехнологии», привлекла внимание известной актрисы Екатерины Варнавы, которая написала о своих впечатлениях в блоге с аудиторией 5 млн. 400 тысяч человек, позитивную реакцию на ее пост о победителях олимпиады продемонстрировали больше 75 тысяч пользователей.

Широкое освещение мероприятий заключительного этапа имеет своей целью распространение информации среди потенциальных участников Олимпиады НТИ будущего года – учеников 7-11 классов и направлено на привлечение талантливых школьников со всей России и активное участие их родителей. В минувшем году была проведена большая работа с целевой аудиторией родителей, чьи дети учатся в 7-11 классах (появилась собственная передача «Выше среднего» на радио Медиаметрикс, регулярно выходят материалы на портале для активных родителей «Летидор», были инициированы эфиры в передача автора самого большого блога для родителей в России (1,6 млн.человек).

Для привлечения внимания участников к конкретным профилям Олимпиады

НТИ ведется точечная работа по освещению их разработок и задач. Инициированы эфиры на радио «Медиаметрикс», тексты в таких медиа как «Rusbase», «Понедельник», «Executive», «БОСС».

В отдельное направление выделена работа с финалистами Олимпиады НТИ с особенными достижениями. Регулярно, а не только в период проведения финалов, инициируются и выходят публикации в таких медиа как: «Российская Газета», «Известия», «Такие Дела», «Индикатор», «RusBase», «Летидор», «Дети Мэйл ру», радио «Медиаметрикс», запущен сервис подкасты в социальной сети ВК, его героями становятся как финалисты, так и разработчики профилей, партнеры, учредители и организаторы Олимпиады НТИ. Список лучших материалов об Олимпиаде: `http://nti-contest.ru/publications/`.

Профиль «Интеллектуальные робототехнические системы»

Профиль "Интеллектуальные робототехнические системы", проводимый в рамках Олимпиады НТИ 2018-2019 учебного года, был посвящен изучению алгоритмов компьютерного зрения и межагентрому взаимодействию. Необходимость высокого уровня подготовки участников для решения данных задач диктовала логику проведения отборочных этапов: необходимо было не только выявить школьников, заинтересованных в решении сложной финальной задачи, но и дать необходимые знания для ее решения.

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

Задачи второго отборочного этапа были разработаны таким образом, что их было бы сложно решить индивидуальному участнику, поэтому школьники должны были объединиться в команды для успешного прохождения в финал. Задачи требовали от участников погружения в такие робототехнические темы, как кинематика и навигация робототехнических устройств, планирование маршрутов перемещения, построение карты с использованием экстероцептивных сенсоров, компьютерное зрение, обработка сетевой информации. 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, реализовывали простейшие алгоритмы обработки изображения, захваченного с камеры роботототехнического устройства, а также организовывали передачу информации между несколькими роботами по беспроводному каналу данных.

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 покрывали следующие области математики: оптимизация, комбинаторика, алгебра и геометрия. А темы задач по информатике перекликались с классическими темами всероссийской олимпиады школьников.

1. ПЕРВЫЙ ЭТАП

Описание этапа

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

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

Задачи первого этапа. Математика.

3.1. Первая попытка. Задачи 9 класса.

Задача 3.1.1. (20 баллов)

Гоша взял у друга 11 гаек М6 (ГОСТ 5916-70) и положил в карман рюкзака. Согласно ГОСТу 1 гайка М6 весит 1.254 грамма. И вот незадача, придя домой, Гоша насчитал в кармане 12 внешне одинаковых гаек! Одна из них была из того набора, что когда-то был куплен на блошином рынке, и, по его личному опыту, такие гайки имеют меньший вес, около грамма, а также сами по себе более низкого качества менее прочные.

У Гошиного папы есть весы, состоящие из двух больших чаш на двух концах рычага. За какое минимальное количество взвешиваний можно найти ту самую низкокачесвенную гайку?

Решение

Если мы имеем n внешне одинаковых объектов, после одного взвешивания останется в худшем случае $\lceil n \rceil$ объектов, среди которых есть отличающийся по весу. Таким образом, если $n = 12$, то, в худшем случае, понадобятся 3 взвешивания $12 \Rightarrow 4 \Rightarrow 2 \Rightarrow 1$.

Ответ: 3.

Задача 3.1.2. (20 баллов)

Витя приклеивал цифры номера квартиры на дверь. Этот номер состоит из трех цифр. В процессе приклеивания Вите пришла необычная мысль, что если в номере квартиры поменять местами две последние цифры и сложить получившееся число с исходным, то получится номер его школы! Юноша учился в школе 1187. Найдите все такие номера квартир, и если Витя живет в квартире с наименьшим из них, то в какой квартире он живет?

Решение

Пусть номер квартиры равен $\overline{abc} = 100 \cdot a + 10 \cdot b + c$, где a, b и c – цифры числа. Число с перевернутыми двумя последними цифрами при этом будет равно

$\overline{acb} = 100 \cdot a + 10 \cdot c + b$. Затем решим уравнение в целых числах

$$100 \cdot a + 10 \cdot b + c + 100 \cdot a + 10 \cdot c + b = 1187,$$

$$200 \cdot a + 11 \cdot b + 11 \cdot c = 1187$$

с учетом ограничений на то, что a, b и c – цифры в десятичной системе счисления.

$$200 \cdot a + 11 \cdot (b + c) = 200 \cdot 5 + 11 \cdot 17.$$

Таким образом, возможные номера квартир 589 и 598. Выбираем наименьший – 589.

Ответ: 589.

Задача 3.1.3. (20 баллов)

Маша и Андрей, будущие математики, развлекались на перемене. Маша написала на доске 4 различных натуральных числа. Андрей выписал значения наибольших общих делителей для каждой из шести пар чисел. Получилось, что для одной из пар НОД равен 1, для другой — 2, для третьей — 3, для четвертой — 4, для пятой — 5, а для шестой — X. Найдите наименьшее возможное значение X?

Решение

Пусть на доске записаны 4 числа a, b, c и d. Пусть НОД$(a, b) = 2$, тогда НОД$(c, d) = 4$ быть не может, так как НОД всех пар будет кратным 2. Значит, пусть НОД$(a, c) = 4$ и тогда d будет нечётным числом.

Исходя из записанных равенств, можно выписать следующие:

$$a = 4 \cdot a_4 = 2 \cdot a_2$$

$$b = 2 \cdot b_2$$

$$c = 4 \cdot c_4$$

При этом НОД$(a_2, b_2) = 1$ и НОД$(a_4, c_4) = 1$. Очевидно также, что НОД$(b, c) = 2 \cdot x$ – то есть это последняя искомая пара. Попробуем подобрать значения a, b, c и d так, чтобы они удовлетворяли всем равенствам и из НОД оставшихся пар равнялись указанным значениям. Например, $a = 4$, $b = 10$, $c = 12$ и $d = 15$. Таким образом, наименьшее возможное значение равно 2.

Ответ: 2.

Задача 3.1.4. (20 баллов)

Женя загадал некоторое число: его наименьший делитель (не равный 1) на 77 меньше наибольшего делителя (не равного самому числу). Чему равно это число? Укажите наименьшее из возможных.

Решение

Пусть загаданное число $X = A \cdot B$, где A – наименьший делитель, B – наибольший. Тогда $A{+}77 = B$. $X = A{\cdot}(A{+}77) \to min$. Так как $A\neg 1$, то минимум достигается при $A = 2$. Таким образом, $X = 2 \cdot 79 = 158$.

Ответ: 158.

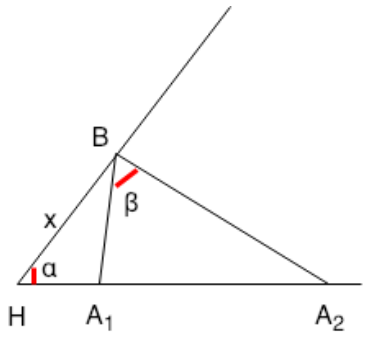
Задача 3.1.5. (20 баллов)

Дан некоторый острый угол $\alpha = 60°$. На одной из его сторон отмечены точки A_1 и A_2, на другой стороне отмечена точка B.

Вершина угла — H. Известно, что $HA_1 = 2$, $A_1A_2 = 8$. При какой величине отрезка HB величина острого угла между прямыми A_1B и A_2B будет максимальна? Ответ введите с точностью до десятитысячных.

Решение

Обозначим $\alpha = \angle BAA_2 = 60°$, $\beta = \angle A_1BA_2$, $x = BH$.



По теореме косинусов выразим A_1B и A_2B и подставим значения $A_1 = 2B$ и $A_1A_2 = 8$ из условия:

$$A_1B^2 = x^2 + A_1H^2 - 2xA_1H \cdot \cos\alpha = x^2 + 4 - 2x,$$
$$A_2B^2 = x^2 + A_2H^2 - 2xA_2H \cdot \cos\alpha = x^2 + 100 - 10x.$$

Рассмотрим $\triangle A_1BA_2$, по теореме косинусов:

$$A_1{A_2}^2 = A_1B^2 + A_2B^2 - 2A_1B \cdot A_2B\cos\beta.$$

Выразим $\cos\beta$:

$$\cos\beta = \frac{A_1B^2 + A_2B^2 - A_1{A_2}^2}{2A_1B \cdot A_2B} = \frac{x^2 + 4 - 2x + x^2 + 100 - 10x - 64}{2\sqrt{x^2 + 4 - 2x} \cdot \sqrt{x^2 + 100 - 10x}} =$$
$$= \frac{x^2 - 6x + 20}{\sqrt{x^2 + 4 - 2x} \cdot \sqrt{x^2 + 100 - 10x}}.$$

Для максимизации острого угла A_1BA_2 требуется, найти x, при котором достигается $min(\cos\beta)$. Решим уравнение $(\cos\beta)'_x = 0$ и найдем точку минимума. $x = 2\sqrt{5} \approx 4.472135955$.

Ответ: 4.472135955.

3.2. Первая попытка. Задачи 10-11 класса.

Задача 3.2.1. (10 баллов)

Гоша взял у друга 11 гаек М6 (ГОСТ 5916-70) и положил в карман рюкзака. Согласно ГОСТу 1 гайка М6 весит 1.254 грамма. И вот незадача, придя домой, Гоша насчитал в кармане 12 внешне одинаковых гаек! Одна из них была из того набора, что когда-то был куплен на блошином рынке, и, по его личному опыту, такие гайки имеют меньший вес, около грамма, а также сами по себе более низкого качества, менее прочные.

У Гошиного папы есть весы, состоящие из двух больших чаш на двух концах рычага. За какое минимальное количество взвешиваний можно найти ту самую низкокачесвенную гайку?

Решение

Если мы имеем n внешне одинаковых объектов, после одного взвешивания останется в худшем случае $\lceil n \rceil$ объектов, среди которых есть отличающийся по весу. Таким образом, если $n = 12$, то, в худшем случае, понадобятся 3 взвешивания $12 \Rightarrow 4 \Rightarrow 2 \Rightarrow 1$.

Ответ: 3.

Задача 3.2.2. (10 баллов)

Витя приклеивал цифры номера квартиры на дверь. Этот номер состоит из трех цифр. В процессе приклеивания Вите пришла необычная мысль, что если в номере квартиры поменять местами две последние цифры и сложить получившееся число с исходным, то получится номер его школы! Юноша учился в школе 1187. Найдите все такие номера квартир, и если Витя живет в квартире с наименьшим из них, то в какой квартире он живет?

Решение

Пусть номер квартиры равен $\overline{abc} = 100 \cdot a + 10 \cdot b + c$, где a, b и c – цифры числа. Число с перевернутыми двумя последними цифрами при этом будет равно $\overline{acb} = 100 \cdot a + 10 \cdot c + b$. Затем решим уравнение в целых числах

$$100 \cdot a + 10 \cdot b + c + 100 \cdot a + 10 \cdot c + b = 1187,$$

$$200 \cdot a + 11 \cdot b + 11 \cdot c = 1187$$

с учетом ограничений на то, что a, b и c – цифры в десятичной системе счисления.

$$200 \cdot a + 11 \cdot (b + c) = 200 \cdot 5 + 11 \cdot 17.$$

Таким образом, возможные номера квартир 589 и 598. Выбираем наименьший – 589.

Ответ: 589.

Задача 3.2.3. (10 баллов)

В химико-биологическом классе 25 учащихся. Для дежурства по школе всегда наугад выбирают двоих. Вероятность того, что оба дежурных окажутся мальчиками, равна 3/25. Сколько в классе девочек?

Решение

Пусть N – количество учащихся в классе, A – количество юношей в классе. Вероятность того, что оба дежурных окажутся мальчиками составляет

$$P = \frac{A}{N} \cdot \frac{A-1}{N-1}.$$

Подставим из условия $N = 25$, $P = 3/25$ и получим следующее уравнение:

$$\frac{A}{25} \cdot \frac{A-1}{24} = \frac{3}{25},$$

$$A^2 - A - 3 \cdot 24 = 0.$$

Уравнение имеет только один положительный корень $A = 9$. Значит, девочек в классе $25 - 9 = 16$.

Ответ: 16.

Задача 3.2.4. (10 баллов)

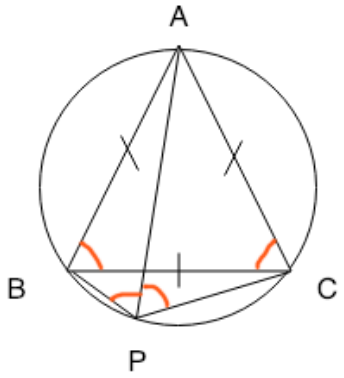
На уроке геометрии нарисовали окружность. На дуге BC этой окружности, описанной около равностороннего треугольника ABC, взята произвольная точка P. Выразите отрезок AP через отрезки и .

Укажите длину AP, если $BP = 3$, $CP = 4$. Ответ введите с точностью до десятитысячных.

Решение

Так как $\triangle ABC$ – равносторонний, то $\angle BAC = \angle ABC = \angle ACB = 60^\circ$ и $AB = AC$. $\angle ABC = \angle APC$, так как опираются на $\smile AC$. $\angle ACB = \angle APB$, так как опираются на $\smile AB$. Таким образом, $\angle APB = \angle APC = 60^\circ$. По теореме косинусов

$$\begin{cases} AB^2 = AP^2 + BP^2 - 2 \cdot AP \cdot BP \cdot \cos(\angle APB), \\ AC^2 = AP^2 + CP^2 - 2 \cdot AP \cdot CP \cdot \cos(\angle APC). \end{cases}$$



Так как $AB = AC$ и $\angle APC = \angle APB = 60°$, получим:

$$AP^2 + BP^2 - AP \cdot BP\cdot = AP^2 + CP^2 - AP \cdot CP,$$
$$BP^2 - CP^2 = AP \cdot (BP - CP),$$
$$AP = BP - CP.$$

Подставим значения $BP = 3$ и $CP = 4$ и получим $AP = 7$.

Ответ: 7.

Задача 3.2.5. (10 баллов)

Маша и Андрей, будущие математики, развлекались на перемене. Маша написала на доске 4 различных натуральных числа. Андрей выписал значения наибольших общих делителей для каждой из шести пар чисел. Получилось, что для одной из пар НОД равен 1, для другой — 2, для третьей — 3, для четвертой — 4, для пятой — 5, а для шестой — X. Найдите наименьшее возможное значение X?

Решение

Пусть на доске записаны 4 числа a, b, c и d. Пусть НОД$(a, b) = 2$, тогда НОД$(c, d) = 4$ быть не может, так как НОД всех пар будет кратным 2. Значит, пусть НОД$(a, c) = 4$ и тогда d будет нечётным числом.

Исходя из записанных равенств, можно выписать следующие:

$$a = 4 \cdot a_4 = 2 \cdot a_2$$
$$b = 2 \cdot b_2$$
$$c = 4 \cdot c_4$$

При этом НОД$(a_2, b_2) = 1$ и НОД $(a_4, c_4) = 1$.

Очевидно также, что НОД$(b, c) = 2 \cdot x$ – то есть это последняя искомая пара. Попробуем подобрать значения a, b, c и d так, чтобы они удовлетворяли всем равенствам и из НОД оставшихся пар равнялись указанным значениям. Например, $a = 4$, $b = 10$, $c = 12$ и $d = 15$. Таким образом, наименьшее возможное значение равно 2.

Ответ: 2.

Задача 3.2.6. (10 баллов)

Женя загадал некоторое число: его наименьший делитель (не равный 1) на 77 меньше наибольшего делителя (не равного самому числу). Чему равно это число? Укажите наименьшее из возможных.

Решение

Пусть загаданное число $X = A \cdot B$, где A – наименьший делитель, B – наибольший. Тогда $A+77 = B$. $X = A \cdot (A+77) \to min$. Так как $A \neg 1$, то минимум достигается при $A = 2$. Таким образом, $X = 2 \cdot 79 = 158$.

Ответ: 158.

Задача 3.2.7. (10 баллов)

У Лады на прикроватной тумбочке стоят часы с циферблатом. Они показывают текущее время суток от 00.00.00 до 23.59.59. Однако, сосед Дима решил перепрошить часы, и теперь, если на часах должны загореться ровно четыре цифры 3, циферблат перестает гореть. Сколько времени в течение суток часы не показывают время, если всё остальное время они работают корректно? Ответ укажите в секундах.

Решение

Каждая комбинация цифр на циферблате отображается ровно 1 секунду, следовательно в задаче требуется найти количество соответствующих комбинаций. В группе цифр, отображающей часы может встретиться только 0 или 1 тройка. В остальных – 0, 1 или 2. Четыре тройки можно получить из следующих комбинаций: $X_1 = Q([0] : [2] : [2])$, $X_2 = Q([1] : [1] : [2])$, $X_3 = Q([1] : [2] : [1])$. Q – количество комбинаций в соответствии с количеством троек в группах цифр циферблата, соответствующих часам, минутам, секундам. $X_1 = 21 \cdot 1 \cdot 1$. $X_2 = 3 \cdot 14 \cdot 1$. $X_3 = 3 \cdot 1 \cdot 14$. $X = X_1 + X_2 + X_3 = 21 + 42 + 42 = 105$.

Ответ: 105.

Задача 3.2.8. (10 баллов)

Дано уравнение $x^3 + y^3 = 4(x^2y + xy^2 + 1)$. Сколько различных решений в целых числах оно имеет?

Если решений бесконечное множество, введите -1.

Решение

$$x^3 + y^3 = 4(x^2y + xy^2 + 1),$$
$$(x + y)(x^2 - xy + y^2) - 4xy(x + y) = 4,$$

$$(x+y)(x^2-5xy+y^2)=4.$$

В левой части множители могут принимать только следующие пары значений: (-4, 1), (-2, -2), (-1, -4), (1, 4), (2, 2), (4, 1). Все 6 систем уравнений не дают целых корней.

Ответ: 0.

Задача 3.2.9. (10 баллов)

Дан некоторый острый угол $\alpha = 60°$. На одной из его сторон отмечены точки A_1 и A_2, на другой стороне отмечена точка B.

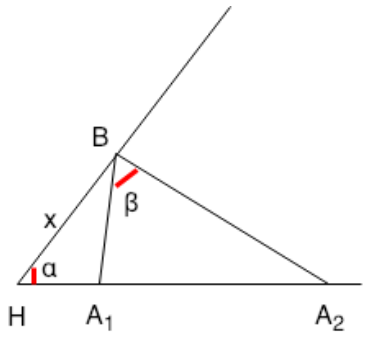
Вершина угла — H. Известно, что $HA_1 = 2$, $A_1A_2 = 8$.

При какой величине отрезка HB величина острого угла между прямыми A_1B и A_2B будет максимальна?

Ответ введите с точностью до десятитысячных.

Решение

Обозначим $\alpha = \angle BAA_2 = 60°$, $\beta = \angle A_1BA_2$, $x = BH$.



По теореме косинусов выразим A_1B и A_2B и подставим значения $A_1 = 2B$ и $A_1A_2 = 8$ из условия:

$$A_1B^2 = x^2 + A_1H^2 - 2xA_1H \cdot \cos\alpha = x^2 + 4 - 2x,$$

$$A_2B^2 = x^2 + A_2H^2 - 2xA_2H \cdot \cos\alpha = x^2 + 100 - 10x.$$

Рассмотрим $\triangle A_1BA_2$, по теореме косинусов:

$$A_1{A_2}^2 = A_1B^2 + A_2B^2 - 2A_1B \cdot A_2B\cos\beta.$$

Выразим $\cos\beta$:

$$\cos\beta = \frac{A_1B^2 + A_2B^2 - A_1{A_2}^2}{2A_1B \cdot A_2B} = \frac{x^2+4-2x+x^2+100-10x-64}{2\sqrt{x^2+4-2x}\cdot\sqrt{x^2+100-10x}} =$$

$$= \frac{x^2-6x+20}{\sqrt{x^2+4-2x}\cdot\sqrt{x^2+100-10x}}.$$

Для максимизации острого угла A_1BA_2 требуется, найти x, при котором достигается $min(\cos\beta)$. Решим уравнение $(\cos\beta)'_x = 0$ и найдем точку минимума. $x = 2\sqrt{5} \approx 4.472135955$.

Ответ: 4.472135955.

Задача 3.2.10. (10 баллов)

Коптер летит над поверхностью огромного поля.

В какой-то момент времени он оказывается в точке $(0, 3, 6)$ в заданной ортогональной системе координат с осями Ox, Oy и Oz.

Найдите расстояние от коптера до земли, если в той же системе координат поле можно считать плоскостью, заданной уравнением $2x+4y-4z-6 = 0$. Ответ укажите с точностью до десятитысячных.

Решение

Расстояние между точкой с координатами (x_0, y_0, z_0) и плоскостью, задаваемой уравнением $Ax + By + Cx + D = 0$ вычисляется по формуле:

$$S = \frac{|Ax_0 + By_0 + Cz_0 + D|}{\sqrt{A^2 + B^2 + C^2}}$$

Для заданных точки и плоскости

$$S = \frac{|2 \cdot 0 + 4 \cdot 3 - 4 \cdot 6 - 6|}{\sqrt{2^2 + 4^2 + 4^2}} = 3.$$

Ответ: 3.

3.3. Вторая попытка. Задачи 9 класса.

Задача 3.3.1. (20 баллов)

Полина прячет в кулаке от 1 до 4 гвоздей. Валера пытается угадать, сколько их. Для этого он задаёт вопросы, на которые Полина может отвечать "да"и "нет". За какое минимальное число вопросов Валера может угадать количество спрятанных гвоздей.

Решение

За один вопрос можно сократить перебираемое множество вдвое. Таким образом, достаточно двух вопросов.

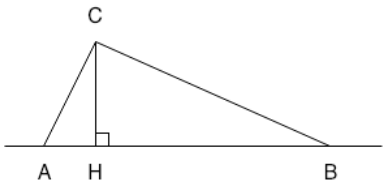
Ответ: 2.

Задача 3.3.2. (20 баллов)

В стране X есть три города – A, B, C. Известно, что расстояние между $AB = 25$, $CB = 24$, $AC = 7$. Города A и B лежат на прямолинейной границе страны, а все остальные участки границы страны пренебрежительно далеки. Найдите кратчайший путь из города до границы страны X. Ответ укажите с точностью до десятитысячных.

Решение

Наикратчайший путь равен длине перпендикуляра, опущенного из вершины C к прямой AB.



По длинам сторон очевидно, что $\triangle ABC$ – прямойгольный, с прямым углом в вершине $C(7^2 + 24^2 = 25^2)$. Так, согласно свойству прямоугольного треугольника, $CH = AC \cdot BC/AB = 6.72$.

Ответ: 6.72.

Задача 3.3.3. (20 баллов)

Коля – очень любознательный юноша. Он решил провести исследование. Для различных действительных чисел a он решил найти такое наибольшее целое число x, чтобы выполнялось следующее:

1. a лежит в интервале $(1, 2)$,
2. a^2 лежит в $(2, 3)$,
3. a^3 лежит в $(3, 4)$,
4. и так далее до показателя степени x.

Помогите Коле выяснить, каким же может быть максимальное значение числа x, при котором существует хотя бы одно значение a, удовлетворяющее условиям?

Решение

В данной задаче необходимо найти такое натуральное максимальное n, при котором $\exists a \forall x \in N : x < n \Rightarrow x < a^x < x + 1$. Выпишем границы интервалов $(\sqrt[x]{x}, \sqrt[x]{x+1})$ для нескольких первых значений x.

x	Левая граница a	Правая граница a
1	1	2
2	$\sqrt[2]{2} = 1.41421356$	$\sqrt[2]{3} = 1.73205081$
3	$\sqrt[3]{3} = 1.44224957$	$\sqrt[3]{4} = 1.58740105$
4	$\sqrt[4]{4} = 1.41421356$	$\sqrt[4]{5} = 1.49534878$
5	$\sqrt[5]{5} = 1.37972966$	$\sqrt[5]{6} = 1.43096908$
...	...	...

По таблице видно, что $\sqrt[5]{6} < \sqrt[3]{3}$. Таким образом, $n = 4$ – наибольшее значение, удовлетворяющее условию задачи.

Ответ: 4.

Задача 3.3.4. (20 баллов)

Найдите сумму коэффициентов после раскрытия скобок у выражения

$$(x^2 - 3x + 1)^{100}.$$

Решение

Постепенно раскрывая скобки найдем сумму коэффициентов полинома $(x^2 - 3x + 1)^n$ для n, начиная с 1.

n	Сумма коэффициентов
1	-1
2	1
3	-1
4	1
...	...

Подметим, что при нечётном n сумма коэффициентов равна -1, а при чётном – 1.

Ответ: 1.

Задача 3.3.5. (20 баллов)

На доске записаны 5 чисел: сначала некоторое рациональное $a = n/y$ (n и y натуральные взаимно простые числа), затем x и далее $x+2$, $x+3$ и $x+4$. При каком наименьшем значении a произведение всех пяти чисел всегда будет натуральным для любого натурального x? В ответе напишите целое число y.

Решение

В данной задаче надо проверить делимость произведения чисел. $x \cdot (x+2) \cdot (x+3) \cdot \cdot (x+4)$ гарантированно делится на 2 и на 3, так как содержит произведение последовательно идущих 3 чисел. Делимость на остальные числа при любом x гарантировать нельзя.

Ответ: 6.

3.4. Вторая попытка. Задачи 10-11 класса.

Задача 3.4.1. (10 баллов)

Полина прячет в кулаке от 1 до 4 гвоздей. Валера пытается угадать, сколько их. Для этого он задаёт вопросы, на которые Полина может отвечать "да"и "нет". За какое минимальное число вопросов Валера может угадать количество спрятанных гвоздей.

Решение

За один вопрос можно сократить перебираемое множество вдвое. Таким образом, достаточно двух вопросов.

Ответ: 2.

Задача 3.4.2. (10 баллов)

Мотоциклист поднимается на холм. Его движение в ортогональной системе координат xOy можно описать законом $y = ax^2+bx+c$, где a и b – некоторые неизвестные постоянные коэффициенты. Известно, что во время своего движения мотоциклист побывал в точках с координатами $(0, 1)$, $(1, 2)$, $(2, 1)$. Найдите координаты вершины холма. Ответ укажите в формате "(x, y)" где x и — значения абсциссы и ординаты с точностью до десятитысячных.

Решение

Подставим точки в уравнение и решим систему:

$$\begin{cases} 1 = a \cdot 0^2 + b \cdot 0 + c, \\ 2 = a \cdot 1^2 + b \cdot 1 + c, \\ 1 = a \cdot 2^2 + b \cdot 2 + c; \end{cases}$$

$$\begin{cases} c = 1, \\ a + b = 1, \\ 2a + b = 0; \end{cases}$$

$$\begin{cases} a = -1, \\ b = 2, \\ c = 1. \end{cases}$$

Уравнение параболы с вычисленными коэффициентами $y = -x^2+2x+1$. Найдем абсциссу точки перегиба:

$$y'_x = 0 \Leftrightarrow -2x + 2 = 0 \Leftrightarrow x = 1.$$

При x = 1 y = 2.

Ответ: $(1, 2)$.

Задача 3.4.3. (10 баллов)

Для тестирования новой программы компьютер выбирает случайное действительное число A из отрезка $[1, 2]$ и заставляет программу решать уравнение $3x + A = 0$. Учтите, что распределение случайной величины равномерное. Найдите вероятность того, что корень этого уравнения меньше, чем -0.4. Ответ укажите с точностью до десятитысячных.

Решение

Выразим корень уравнения: $x = -A/3$. Таким образом, $x \in [-2/3, -1/3]$. Значит

$$P(x \in [-2/3, -0.4]) = \frac{-0.4 + 2/3}{1/3} = 2 - 1.2 = 0.8.$$

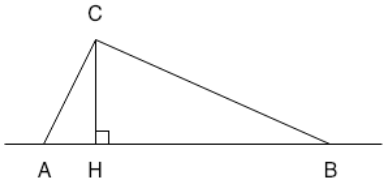
Ответ: 0.8.

Задача 3.4.4. (10 баллов)

В стране X есть три города – A, B, C. Известно, что расстояние между $AB = 25$, $CB = 24$, $AC = 7$. Города A и B лежат на прямолинейной границе страны, а все остальные участки границы страны пренебрежительно далеки. Найдите кратчайший путь из города до границы страны X. Ответ укажите с точностью до десятитысячных.

Решение

Наикратчайший путь равен длине перпендикуляра, опущенного из вершины C к прямой AB.



По длинам сторон очевидно, что $\triangle ABC$ – прямойгольный, с прямым углом в вершине $C(7^2 + 24^2 = 25^2)$. Так, согласно свойству прямоугольного треугольника, $CH = AC \cdot BC/AB = 6.72$.

Ответ: 6.72.

Задача 3.4.5. (10 баллов)

Коля – очень любознательный юноша. Он решил провести исследование. Для различных действительных чисел a он решил найти такое наибольшее целое число x, чтобы выполнялось следующее:

1. a лежит в интервале $(1, 2)$,
2. a^2 лежит в $(2, 3)$,
3. a^3 лежит в $(3, 4)$,
4. и так далее до показателя степени x.

Помогите Коле выяснить, каким же может быть максимальное значение числа x, при котором существует хотя бы одно значение a, удовлетворяющее условиям?

Решение

В данной задаче необходимо найти такое натуральное максимальное n, при котором $\exists a \forall x \in N : x < n \Rightarrow x < a^x < x + 1$. Выпишем границы интервалов $(\sqrt[x]{x}, \sqrt[x]{x+1})$ для нескольких первых значений x.

x	Левая граница a	Правая граница a
1	1	2
2	$\sqrt[2]{2} = 1.41421356$	$\sqrt[2]{3} = 1.73205081$
3	$\sqrt[3]{3} = 1.44224957$	$\sqrt[3]{4} = 1.58740105$
4	$\sqrt[4]{4} = 1.41421356$	$\sqrt[4]{5} = 1.49534878$
5	$\sqrt[5]{5} = 1.37972966$	$\sqrt[5]{6} = 1.43096908$
...	...	...

По таблице видно, что $\sqrt[5]{6} < \sqrt[3]{3}$. Таким образом, $n = 4$ – наибольшее значение, удовлетворяющее условию задачи.

Ответ: 4.

Задача 3.4.6. (10 баллов)

На доске записаны 5 чисел: сначала некоторое рациональное $a = n/y$ (n и y натуральные взаимно простые числа), затем x и далее $x+2$, $x+3$ и $x+4$. При каком наименьшем значении a произведение всех пяти чисел всегда будет натуральным для любого натурального x? В ответе напишите целое число y.

Решение

В данной задаче надо проверить делимость произведения чисел. $x \cdot (x+2) \cdot (x+3) \cdot \cdot (x+4)$ гарантированно делится на 2 и на 3, так как содержит произведение последовательно идущих 3 чисел. Делимость на остальные числа при любом x гарантировать нельзя.

Ответ: 6.

Задача 3.4.7. (10 баллов)

Двое бросают монету: один бросил её 8 раз, другой – 9 раз. Чему равна вероятность того, что у второго монета упала орлом большее число раз, чем у первого?

Монета никогда не падает на ребро.

Ответ укажите с точностью до десятитысячных.

Решение

Первый может выбросить 9 различных наборов (последовательность не учитывается). Если в наборе x орлов, то количество возможных вариантов второго игрока, удовлетворяющих условию $(9 - x)$. Всего возможных способов выкинуть наборы у обоих игроков: $9 \cdot 10 = 90$.

$$P = \frac{\sum\limits_{x=0}^{8} 9 - x}{90} = 0.5.$$

Ответ: 0.5.

Задача 3.4.8. (10 баллов)

Найдите сумму коэффициентов после раскрытия скобок у выражения

$$(x^2 - 3x + 1)^{100}.$$

Решение

Постепенно раскрывая скобки найдем сумму коэффициентов полинома $(x^2 - 3x + 1)^n$ для n, начиная с 1.

n	Сумма коэффициентов
1	-1
2	1
3	-1
4	1
...	...

Подметим, что при нечётном n сумма коэффициентов равна -1, а при чётном – 1.

Ответ: 1.

Задача 3.4.9. (10 баллов)

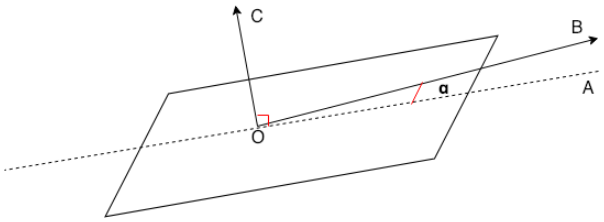
Самолет взлетает с авианосца. Вектор нормали к поверхности взлетной полосы имеет координаты $(4, 0, 3)$. Направляющий вектор траектории полета

самолета – $(5, 12, 0)$ в той же ортогональной системе координат. Найдите СИНУС угла, под которым взлетел самолет относительно взлетной полосы.

Введите ответ с точностью до десятитысячных.

Решение

$$\cos\alpha = \frac{|(\vec{OB}, \vec{OC})|}{|\vec{OB}| \cdot |\vec{OC}|} = \frac{20}{\sqrt{25+144}\sqrt{16+9}} = \frac{4}{13},$$
$$\sin\alpha = \sqrt{1-\cos^2\alpha} = 0.9514859.$$



Ответ: 0.9514859136.

Задача 3.4.10. (10 баллов)

Соня, Андрей и Егор живут в домах A, B, C соответственно. Эти дома соединены прямыми улицами – Садовой, Огородной и Персиковой. Известно, что Садовая и Персиковая улицы пересекаются у дома Сони под углом 45 градусов, Персиковая и Огородная – под углом 60 градусов у дома Егора и, наконец, Огородная и Садовая пересекаются под окнами у Андрея. Равноудаленно от домов B и C внутри треугольника ABC построили магазин. Известно, что прямая улица, которая соединяет магазин и дом Егора, пересекается с Персиковой под углом 15 градусов. Между домом Сони и магазином также есть прямая улица. Под каким углом она пересекается с Садовой?

Ответ укажите в градусах с точностью до десятитысячных.

Решение

Обозначим искомый угол буквой α.

$\angle BAC = 45^\circ \Rightarrow \angle OAC = 45^\circ - \alpha$. $\angle OCA = 15^\circ \Rightarrow \angle AOC = 120^\circ + \alpha$. $\angle OCB = \angle OBC = 60^\circ - 15^\circ = 45^\circ \Rightarrow \angle BOC = 90^\circ$. $\angle ABO = 30^\circ$. По теореме синусов:

$$\frac{\sin\alpha}{OB} = \frac{\sin 30^\circ}{AO} \Rightarrow AO = \frac{OB}{2\sin\alpha}$$

$$\frac{\sin 15^\circ}{OA} = \frac{\sin(45^\circ - \alpha)}{OC} \Rightarrow AO = \frac{OC\sin 15^\circ}{\sin(45^\circ - \alpha)}$$

Так как $OB = OC$,

$$\sin(45^\circ - \alpha) = 2\sin\alpha\sin 15^\circ \Rightarrow \alpha = 30^\circ.$$

Ответ: 30.

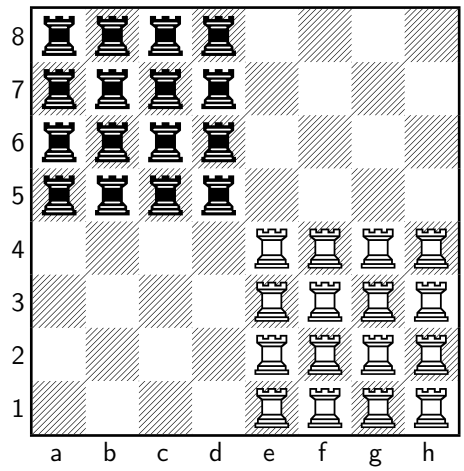
3.5. Третья попытка. Задачи 9 класса.

Задача 3.5.1. (20 баллов)

Пусть на шахматной доске размера 8×8 расставили n белых и столько же черных ладей так, что они не бьют друг друга. Найдите такое максимальное n, при котором это возможно.

Решение

Необходимо расставить ладьи одного цвета так, что на одной горизонтали с каждой из них было еще 3 и на одной вертикали 3 другие. Таким образом, в каждой строке или столбце ровно 4 ладьи одного цвета. Например:



Ответ: 16.

Задача 3.5.2. (20 баллов)

На планете Y все жители говорят на языке, состоящем из трех букв. При этом все слова в этом языке не длиннее 6 символов и не короче 3. Сколько слов можно составить в языке планеты Y?

Решение

Количество слов из 3 букв длиной n $W_n = 3^N$. Так $W = 3^3 + 3^4 + 3^5 + 3^6 = 1080$.

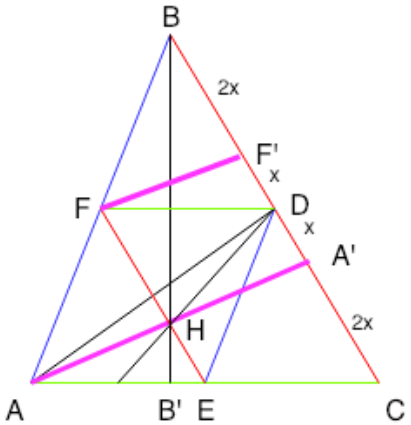
Ответ: 1080.

Задача 3.5.3. (20 баллов)

Дан треугольник ABC и H — точка пересечения высот этого треугольника. Пусть D — середина отрезка BC, E — середина отрезка . Кроме того, медианы треугольника AED пересекаются в точке H. Найдите градусную меру угла $\angle ABC$.

Ответ укажите с точностью до десятитысячных.

Решение



Рассмотрим $\triangle ABC$: DE, EF, DF – медианы, AA', BB' – высоты. $BD = DC$. В $\triangle ADE$: медианы пересекаются в точке H, которая разделяет их в отношении $2:1$. Из подобия треугольников очевидно, что $DA' : A'C = 1:2$.

Пусть $AD = x$, $A'C = 2x$. $FH = 2 \cdot (HE + HE/2) - HE = 2HE = 2x$ По свойству паралелограмма, образованного медианами FE и BC, перпендикулярами AA' и FF': $FH = F'A' = 2x$ и $FF' = HA'$, $F'D = x$. Также из подобия треугольников $BF' = 2x$, $FF' = HA' = AH$.

Рассмотрим $\triangle BA'H \sim \triangle AHE$(по 2 углам). $A'B : A'H = A'H : HE \Rightarrow A'H = \sqrt{A'B \cdot HE} = \sqrt{4x^2} = 2x = EF' = BF'$.

Так, $\triangle BF'F$ – прямоугольный равнобедренный, и угол при основании $\angle FBF' = 45°$.

Ответ: 45.

Задача 3.5.4. (20 баллов)

Артур и Саша играли в игру — по очереди выписывали натуральные числа на бумагу. В итоге оказалось, что на бумаге выписано 15 чисел, причем, наименьшее из чисел можно представить как $x + 1$, $x > 1$, а все остальные числа — последовательность $(1 + x^n)$, где n — натуральный показатель степени, изменяющийся от 2 до 15. Артуру показалось, что выписанных чисел слишком много и он зачеркнул часть из них таким образом, чтобы все оставшиеся на бумаге числа были взаимно простыми. Какое наименьшее количество чисел мог зачеркнуть Артур?

Решение

Все натуральные числа вида $(1 + x^n)$, где n – нечетное число, делятся нацело на $(1 + x)$.

Таким образом, вычеркнем все числа, где $n = 1$, 3, 5, 7, 9, 11 и 13. 15 оставляем.

Далее заменим x^2 на y. По аналогии вычеркнем числа, где n = 2, 6, 10. 14 оставляем.

Далее заменим x^3 на z. $n = 3$, 9 – вычеркнуты. 15 оставляем.

Далее заменим x^4 на w. По аналогии вычеркнем число, где $n = 4$. 12 оставляем.

Далее заменим x^5 на v. $n = 5$ вычеркнуто. 15 оставляем.

Далее нет смысла перебирать, так как $3n > 15$. Оставшиеся числа взаимнопростые. Таким образом, минимально мы вычеркнули 11 чисел (1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13).

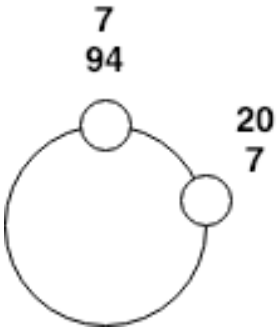
Ответ: 11.

Задача 3.5.5. (20 баллов)

Ира и Паша расставляют стулья вокруг круглого стола. После того, как все стулья были расставлены, ребята решили их пересчитать — они начали ходить по кругу

в одном направлении, но начиная с разных стульев. Известно, что стул, который Паша посчитал седьмым, у Иры оказался под двадцатым номером, а тот стул, который Ира посчитала седьмым, у Паши был 94 м. Сколько стульев было расставлено вокруг стола?

Решение



Считаем по обеим дугам количество стульев, включая один из двух крайних: $20 - 7 + 94 - 7 = 100$.

Ответ: 100.

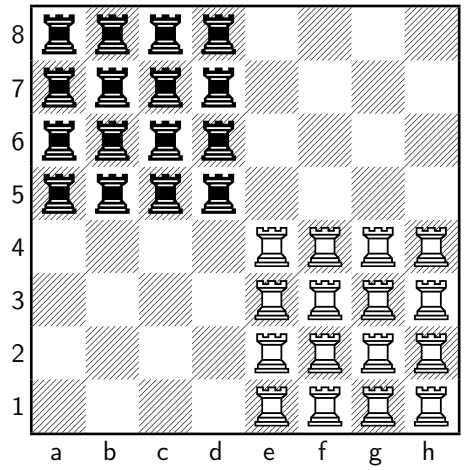
3.6. Третья попытка. Задачи 10-11 класса.

Задача 3.6.1. (10 баллов)

Пусть на шахматной доске размера 8×8 расставили n белых и столько же черных ладей так, что они не бьют друг друга. Найдите такое максимальное n, при котором это возможно.

Решение

Необходимо расставить ладьи одного цвета так, что на одной горизонтали с каждой из них было еще 3 и на одной вертикали 3 другие. Таким образом, в каждой строке или столбце ровно 4 ладьи одного цвета. Например:



Ответ: 16.

Задача 3.6.2. (10 баллов)

Найдите такое значение $a > 1$, при котором уравнение $a^x = log_a x$ имеет ровно один корень. Ответ укажите с точностью до десятитысячных.

Решение

Обе функции в левой и правой частях уравнения выпуклые, единственный корень у уравнения возникает тогда, когда $y = a^x$ касается прямой $y = x$, то есть $f(x_0) = x_0$ и $f'(x_0) = a^{x_0} \ln a = 1$. Отсюда $x_0 = \frac{1}{\ln a}$, т.е. $e = a^{\frac{1}{\ln a}} = \frac{1}{\ln a}$, $\ln a = \frac{1}{e}$, $a = e^{\frac{1}{e}} = 1.44466786$.

Ответ: 1.44466786101.

Задача 3.6.3. (10 баллов)

На планете Y все жители говорят на языке, состоящем из трех букв. При этом все слова в этом языке не длиннее 6 символов и не короче 3. Сколько слов можно составить в языке планеты Y?

Решение

Количество слов из 3 букв длиной n $W_n = 3^N$. Так $W = 3^3 + 3^4 + 3^5 + 3^6 = 1080$.

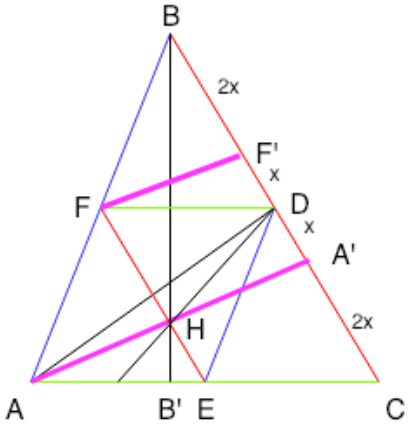
Ответ: 1080.

Задача 3.6.4. (10 баллов)

Дан треугольник ABC и H — точка пересечения высот этого треугольника. Пусть D — середина отрезка BC, E — середина отрезка . Кроме того, медианы треугольника AED пересекаются в точке H. Найдите градусную меру угла $\angle ABC$.

Ответ укажите с точностью до десятитысячных.

Решение



Рассмотрим $\triangle ABC$: DE, EF, DF – медианы, AA', BB' – высоты. $BD = DC$. В $\triangle ADE$: медианы пересекаются в точке H, которая разделяет их в отношении $2:1$. Из подобия треугольников очевидно, что $DA' : A'C = 1:2$.

Пусть $AD = x$, $A'C = 2x$. $FH = 2 \cdot (HE + HE/2) - HE = 2HE = 2x$ По свойству паралелограмма, образованного медианами FE и BC, перпендикулярами AA' и FF': $FH = F'A' = 2x$ и $FF' = HA'$, $F'D = x$. Также из подобия треугольников $BF' = 2x$, $FF' = HA' = AH$.

Рассмотрим $\triangle BA'H \sim \triangle AHE$(по 2 углам). $A'B : A'H = A'H : HE \Rightarrow A'H = \sqrt{A'B \cdot HE} = \sqrt{4x^2} = 2x = EF' = BF'$.

Так, $\triangle BF'F$ – прямоугольный равнобедренный, и угол при основании $\angle FBF' = 45°$.

Ответ: 45.

Задача 3.6.5. (10 баллов)

Артур и Саша играли в игру — по очереди выписывали натуральные числа на бумагу. В итоге оказалось, что на бумаге выписано 15 чисел, причем, наименьшее из чисел можно представить как $x + 1$, $x > 1$, а все остальные числа — последовательность $(1 + x^n)$, где n — натуральный показатель степени, изменяющийся от 2 до 15. Артуру показалось, что выписанных чисел слишком много и он зачеркнул часть из них таким образом, чтобы все оставшиеся на бумаге числа были взаимно простыми. Какое наименьшее количество чисел мог зачеркнуть Артур?

Решение

Все натуральные числа вида $(1 + x^n)$, где n – нечетное число, делятся нацело на $(1 + x)$.

Таким образом, вычеркнем все числа, где $n = 1, 3, 5, 7, 9, 11$ и 13. 15 оставляем.

Далее заменим x^2 на y. По аналогии вычеркнем числа, где n = 2, 6, 10. 14 оставляем.

Далее заменим x^3 на z. $n = 3$, 9 – вычеркнуты. 15 оставляем.

Далее заменим x^4 на w. По аналогии вычеркнем число, где $n = 4$. 12 оставляем.

Далее заменим x^5 на v. $n = 5$ вычеркнуто. 15 оставляем.

Далее нет смысла перебирать, так как $3n > 15$. Оставшиеся числа взаимнопростые. Таким образом, минимально мы вычеркнули 11 чисел (1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13).

Ответ: 11.

Задача 3.6.6. (10 баллов)

Паша загадал число x: это неправильная дробь с натуральным числителем и со знаменателем, равным 9. Далее он вычислил еще три числа умножил x на 5, на 2 и на 4. Затем округлил эти три числа по правилам округления до целого и сложил между собой, в итоге он получил 120. Каким был числитель у неправильной дроби?

Решение

Найдем первое значение для перебора: $\frac{5x+2x+4x}{9} \approx 120 \Leftrightarrow x \approx 98$. Проверим значение:

$$\left[\frac{5 \cdot 98}{9}\right] + \left[\frac{2 \cdot 98}{9}\right] + \left[\frac{4 \cdot 98}{9}\right] = 54 + 22 + 44 = 120.$$

Ответ: 98.

Задача 3.6.7. (10 баллов)

Если из резервуара выливают воду, уровень воды H в нём меняется в зависимости от времени t следующим образом: $H(t) = at^2 + bt + c$. Пусть t_0 — момент окончания слива. Известно, что в этот момент выполнены равенства $H(t_0) = H'(t_0) = 0$. В течение какого времени вода из резервуара будет полностью вылита, если за первый час слилась половина уровня?

Округлите ответ до ближайшего целого.

Решение

$$H(t_0) = at_0^2 + bt_0 + c = 0,$$
$$H'(t_0) = 2at_0 + b = 0.$$

Выразим b и c через a и t_0:

$$b = -2at_0,$$
$$c = at_0^2.$$

Пусть τ – время достижения половинного уровня.

$$2H(\tau) = H(0),$$

$$2(a\tau^2 + (-2at_0)\tau + at_0^2) = at_0^2,$$
$$2\tau^2 - 4t_0\tau + t_0^2 = 0,$$
$$D = 16t_0^2 - 4 \cdot 2t_0^2 = 8t_0^2,$$
$$\tau = \frac{4t_0 \pm 2\sqrt{2}t_0}{2 \cdot 2} = t_0 \pm t_0/\sqrt{2}.$$

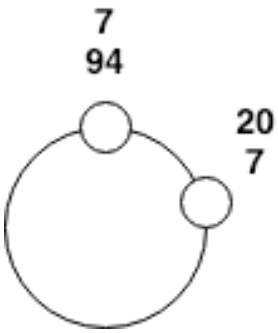
Так как $\tau < t_0$ из условия, то $\tau = t_0 \cdot (2 - \sqrt{2})/2$, то есть $t_0 = \tau(2 + \sqrt{2})$, то есть $[t_0] = [1 \cdot (2 + \sqrt{2})] = 3$.

Ответ: 3.

Задача 3.6.8. (10 баллов)

Ира и Паша расставляют стулья вокруг круглого стола. После того, как все стулья были расставлены, ребята решили их пересчитать — они начали ходить по кругу в одном направлении, но начиная с разных стульев. Известно, что стул, который Паша посчитал седьмым, у Иры оказался под двадцатым номером, а тот стул, который Ира посчитала седьмым, у Паши был 94 м. Сколько стульев было расставлено вокруг стола?

Решение



Считаем по обеим дугам количество стульев, включая один из двух крайних: $20 - 7 + 94 - 7 = 100$.

Ответ: 100.

Задача 3.6.9. (10 баллов)

В течение пяти часов 1 сентября Женя наблюдал за воздушными шариками в небе. По мере того, как утренние линейки проходили, шаров в небе становилось все меньше – так, с каждым часом за час пролетало не больше шаров, чем в предыдущий час. Суммарно Женя насчитал 100 шаров, пролетевших в небе мимо его окна. Причем, суммарно за второй и четвертый час Женя увидел не больше шаров, чем за первый и третий. Какое минимальное число шаров Женя мог увидеть суммарно за 1, 3 и 5 часы?

Решение

Пусть a, b, c, d, e — количество шаров в 1, 2, 3, 4 и 5 часы соответственно. $a \geqslant b \geqslant c \geqslant d \geqslant e$, $a+b+c+d+e = 100$, $b+d \leqslant a+c$, требуется найти $min(a+c+e)$. Данная задача сводится к поиску $max(b+d)$. Так как $b+d \leqslant a+c$, а $a+b+c+d+e = 100$, то $b+d \leqslant 100/2 = 50$. Такой случай легко придумать: $a, b, c, d, e = 25, 25, 25, 25, 0$. Таким образом, $min(a+c+e) = 100 - max(b+d) = 50$.

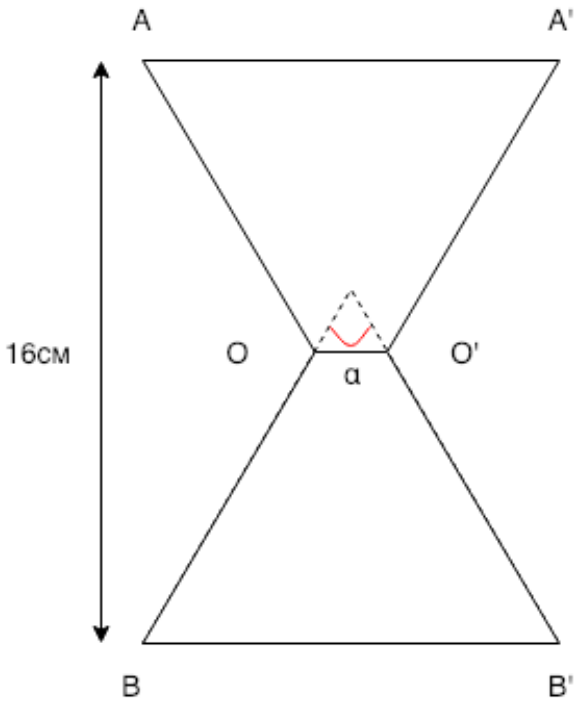
Ответ: 50.

Задача 3.6.10. (10 баллов)

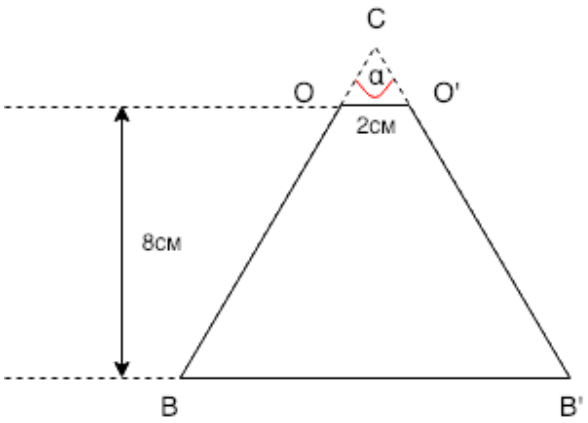
На столе стоят песочные часы высоты 16 см, представляющие собой два соединенных усеченных одинаковых конуса. Радиус горлышка (отверстия, через которое сыпется песок) равен 1 см. Тангенс угла раствора конусов равен 4/3. Чему равен объем песочных часов в см^3?

Ответ округлите до ближайшего целого.

Решение



Рассмотрим один из конусов.

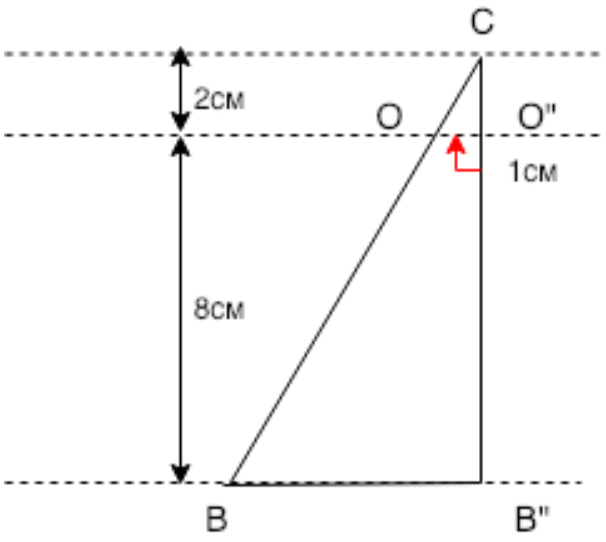


$$\operatorname{tg}^2\alpha + 1 = \frac{1}{cos^2\alpha},$$
$$cos\alpha = \frac{3}{5}.$$

Пусть $OC = O'C = x$, по теореме косинусов: $OO'^2 = x^2 + x^2 - 2x \cdot x\cos\alpha = \frac{4x^2}{5}$, $x = \sqrt{5}$.

Опустим перпендикуляр CO'' к OO''.

$$CO'' = \sqrt{x^2 - OO''^2} = \sqrt{5-1} = 2$$



Из подобия треугольников $BB'' = 5$. Объем песочных часов, т.е. удвоенный объем усеченного конуса, равен

$$V = \frac{2}{3}\pi O''B''(OO''^2 + OO''BB'' + BB''^2) = \frac{2\pi}{3} \cdot 8 \cdot (1 + 5 + 25) = 519.409 \approx 519.$$

Ответ: 519.

Задачи первого этапа. Углубленная информатика.

4.1. Первая попытка.

Задача 4.1.1. Башня (15 баллов)

Девочка Аня построила на пляже песчаный замок и хочет украсить главную башню замка лентой следующим образом:

Она хочет сделать k витков вокруг башни, а оставшуюся часть ленты разделить на ленточки равной длины и прикрепить получившиеся ленточки к вершине башни.

Аня посчитает башню красивой, если она сможет сделать желаемые k витков, затем разделить оставшуюся часть ленты на ленточки равной длины, а потом прикрепить к вершине хотя бы одну ленточку с длиной, равной высоте башни.

Родители помогли измерить Ане высоту башни, длину одного витка и длину имеющейся у неё ленты. Но Аня ещё не научилась считать, поэтому просит вас помочь определить, сможет ли она сделать свою башню красивой, используя имеющуюся у неё ленту.

Формат входных данных

В первой и последней строке четыре целых числа:

$1 \leqslant n \leqslant 1000$ – длина ленты;

$1 \leqslant m \leqslant 1000$ – длина одного витка;

$1 \leqslant k \leqslant 1000$ – количество витков;

$1 \leqslant h \leqslant 1000$ – высота башни.

Формат выходных данных

Выведите Yes, если лента подходит, иначе – No.

Примеры

Пример №1

Стандартный ввод
13 2 3 7

Стандартный вывод
Yes

Решение

Суммарная длина витков равна $k \cdot m$.

Если длины ленты не хватает даже для витков ($n < k \cdot m$), то сделать башню красивой точно не получится.

Если длины ленты хватает только на то, чтобы сделать эти k витков ($n = k \cdot m$), то сделать башню красивой не получится, так как Аня не сможет прикрепить к вершине башни ни одной ленты.

Если же можно сделать желаемые k витков и от ленты ещё что-то останется ($n > k \cdot m$), на ленточки останется ($n - k \cdot m$) единиц длины. Далее необходимо оставшуюся часть ленты разделить на ленточки равной длины и прикрепить получившиеся ленточки к вершине башни. Причём длина хотя бы одной из ленточек должна быть равна высоте башни. Следовательно длина всех ленточек должна быть равна высоте башни. А чтобы было возможно оставшуюся часть ленты разделить на ленточки с длиной, равной высоте башни, длина оставшейся части ленты должна нацело делиться на высоту башни.

Таким образом, нужно было вывести Yes, если ($n - k \cdot m$) положительно и делится нацело на h, иначе – No.

Пример программы-решения

Ниже представлено решение на языке Python3

```
1 n, m, k, h = [int(i) for i in input().split()]
2 print("Yes" if n - m * k > 0 and (n - m * k) % h == 0 else "No")
```

Задача 4.1.2. Классики (15 баллов)

Шёл 2048-ой год. Робот Вася шёл по улице и увидел играющих детей. Они прыгали то на одной ноге, то на двух. Вася подошёл и спросил, во что играют дети. Ему объяснили, что эта игра называется классики и дети прыгают по написанным на асфальте числам. Вася пропрыгал разок и пошёл дальше по своим делам.

Через несколько дней идя по улице он увидел написанные на асфальте ряды чисел и задался вопросом, классики ли это. Ещё через пару дней он снова увидел написанные на асфальте ряды чисел и вновь задался вопросом, классики ли это. Но ответить на этот вопрос он не мог.

Классиками в понимании Васи (ну, так ему объяснили) являются ряды, которые обладают следующим свойством:

Каждое целое число от 1 до $s_1 + s_2 + \ldots + s_i$ встречается в первых i рядах ровно один раз, где s_i – количество чисел в i-ом ряду.

Напишите для Васи программу, которая будет помогать ему определять, являются ли те или иные ряды чисел классиками (в понимании Васи).

Формат входных данных

В первой строке одно целое число:

$1 \leqslant t \leqslant 500$ – количество раз, которое Вася встречал написанные на асфальте ряды чисел в последнее время.

Каждый набор рядов описан следующим образом:

В первой строке одно целое число:

$1 \leqslant n \leqslant 500$ – количество рядов.

Во второй строке n целых чисел:

s_i – количество чисел в i-ом ряду

$(1 \leqslant i \leqslant n; 1 \leqslant s_i \leqslant 500)$.

Далее идёт ещё n строк. В $(i+2)$-ой строке s_i целых чисел:

$a_{i,j}$ – j-ое число в i-ом ряду

$(|a_{i,j}| \leqslant 10^9)$.

Гарантируется, что суммарное количество чисел во входных данных не превышает 10^5.

Формат выходных данных

Для каждого набора рядов в отдельной строке выведите:

Yes, если данные ряды чисел являются классиками в понимании Васи, иначе – No.

Примеры

Пример №1

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

Стандартный вывод
Yes Yes No

Решение

Поработаем с данным в условии задачи свойством. Какие числа находятся в i-ом ряду?

Если $i = 1$, то – числа от 1 до s_1.

Если $i > 1$, то – числа от $s_1 + s_2 + \ldots + s_{i-1} + 1$ до $s_1 + s_2 + \ldots + s_i$, так как числа от 1 до $s_1 + s_2 + \ldots + s_{i-1}$ записаны в первых $(i - 1)$ рядах.

Таким образом, для нахождения ответа для отдельно взятого набора рядов можно было, например, отсортировать числа в каждом ряду и проверить, что j-ое (после сортировки) число в i-ом ряду равно $s_1 + s_2 + \ldots + s_{i-1} + j$ для всех возможных пар (i, j).

Пример программы-решения

Ниже представлено решение на языке Python3

```
def solve():
    n = int(input())
    s = [int(i) for i in input().split()]

    ok = True
    sum = 0
    for i in range(n):
```

```
        ai = sorted([int(i) for i in input().split()])
        for j in range(s[i]):
            ok &= (ai[j] == sum + j + 1)
        sum += s[i]

    return "Yes" if ok else "No"


t = int(input())

for i in range(t):
    print(solve())
```

Задача 4.1.3. Треугольник (20 баллов)

Плоскость задана тремя точками. Определите площадь треугольника, образованного этими тремя точками.

Формат входных данных

Входные данные состоят из трёх строк. В каждой из трёх строк по три целых числа:

x_i y_i z_i – координаты i-ой точки

$(1 \leqslant i \leqslant 3; |x_i| \leqslant 10^9; |y_i| \leqslant 10^9; |z_i| \leqslant 10^9)$.

Гарантируется, что данные три точки не лежат на одной прямой.

Формат выходных данных

Одно положительное число – площадь треугольника.

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
0 4 0 0 0 0 3 0 0

Стандартный вывод
6.0000000000

Решение

Для нахождения площади треугольника можно, например, воспользоваться формулой Герона, которая позволяет вычислить площадь треугольника по длинам его

сторон. Но точности вычислений, которой позволяют достичь стандартные типы данных для работы с действительными числами, не достаточно для прохождения всех тестов. Чтобы повысить точность, нужно либо воспользоваться типом данных для работы с длинными действительными числами, либо написать свой вариант работы с длинными действительными числами.

Пример программы-решения

Ниже представлено решение на языке Python3

```
from decimal import Decimal

x1, y1, z1 = [Decimal(i) for i in input().split()]
x2, y2, z2 = [Decimal(i) for i in input().split()]
x3, y3, z3 = [Decimal(i) for i in input().split()]
a = ((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) + (z2 - z1) * (z2 - z1)).sqrt()
b = ((x3 - x2) * (x3 - x2) + (y3 - y2) * (y3 - y2) + (z3 - z2) * (z3 - z2)).sqrt()
c = ((x3 - x1) * (x3 - x1) + (y3 - y1) * (y3 - y1) + (z3 - z1) * (z3 - z1)).sqrt()
p = (a + b + c) / 2
print((p * (p - a) * (p - b) * (p - c)).sqrt())
```

Задача 4.1.4. R2D2 (25 баллов)

Робот R2D2 случайно оказался на Имперском корабле. Он хочет покинуть его как можно скорее. Для этого ему надо добраться до спасательной капсулы.

Для упрощения задачи корабль представляет собой прямоугольную таблицу высотой n и шириной m. Ячейка может быть либо пустой, либо представлять собой препятствие. Помогите за минимальное время добраться R2D2 из своей начальной точки до спасательной капсулы.

При этом известно, что робот может передвигаться только в клетки, соседние по стороне. То есть двигаться только вверх, вниз, влево и вправо. Также у робота есть текущее направление.

Движение вперед занимает у робота 1 секунду и поворот на 90° также занимает 1 секунду.

Зная начальное расположение робота и его направление. Выясните за какое минимальное время он сможет покинуть корабль. При этом, если робот оказался в ячейке со спасательной капсулой, его текущее направление не имеет значения.

Изначально робот всегда смотрит вниз.

Формат входных данных

В первой строке вводятся два целых числа n и m ($1 \leqslant n, m \leqslant 1000$) – высота и ширина.

В следующих n строках вводятся m символов $a_{i,j}$. Значения ячейки $a_{i,j}$ могут быть # – препятствие, . – пустая клетка, s – начальная позиция робота, f – спасательная капсула.

Гарантируется, что ровно одна клетка в таблице имеет значение s.

Гарантируется, что ровно одна клетка в таблице имеет значение f.

Формат выходных данных

Выведите минимальное количество секунд, нужное чтобы добраться роботу до спасательной капсулы или -1, если это сделать невозможно.

Примеры

Пример №1

Стандартный ввод
3 3 s.. ... ..f
Стандартный вывод
5

Пример №2

Стандартный ввод
3 3 s.. ### ..f
Стандартный вывод
-1

Пример №3

Стандартный ввод
3 3 s.. ..f ...
Стандартный вывод
4

Решение

Давайте перейдем к графам. Вершиной будем считать пару (клетка, направление).

Ребра будут двух типов:

- перейти в соседнюю клетку в соответствии с текущим направлением;
- повернуться на 90°, то есть сменить направление.

Тогда задача сводится к нахождению кратчайшего расстояния в получившемся графе. Задачу можно решить с помощью алгоритма поиска в ширину (BFS), так как вес всех ребер одинаков и равен 1.

Пример программы-решения

Ниже представлено решение на языке Python3

```python
n, m = map(int, input().split())

s = []
for i in range(n):
    s.append(input())

dist = [[[2000000000 for j in range(4)] for i in range(m)] for k in range(n)]
qi = [0] * (n * m * 4)
qj = [0] * (n * m * 4)
qd = [0] * (n * m * 4)
ql = 0
qr = 0

for i in range(n):
    for j in range(m):
        if (s[i][j] == 's'):
            dist[i][j][0] = 0
            qi[qr] = i
            qj[qr] = j
            qd[qr] = 0
            qr += 1
            break
    if qr > 0:
        break

ans = -1
while ql < qr:
    i = qi[ql]
    j = qj[ql]
    d = qd[ql]
    ql += 1

    if s[i][j] == 'f':
        ans = dist[i][j][d]
        break

    if d == 0:
        if i + 1 < n and s[i + 1][j] != '#' and dist[i + 1][j][d] > dist[i][j][d] + 1:
            dist[i + 1][j][d] = dist[i][j][d] + 1
            qi[qr] = i + 1
            qj[qr] = j
            qd[qr] = d
            qr += 1
    elif d == 1:
        if j > 0 and s[i][j - 1] != '#' and dist[i][j - 1][d] > dist[i][j][d] + 1:
            dist[i][j - 1][d] = dist[i][j][d] + 1
            qi[qr] = i
            qj[qr] = j - 1
            qd[qr] = d
            qr += 1
    elif d == 2:
        if i > 0 and s[i - 1][j] != '#' and dist[i - 1][j][d] > dist[i][j][d] + 1:
            dist[i - 1][j][d] = dist[i][j][d] + 1
            qi[qr] = i - 1
            qj[qr] = j
            qd[qr] = d
```

```
            qr += 1
    elif d == 3:
        if j + 1 < m and s[i][j + 1] != '#' and dist[i][j + 1][d] > dist[i][j][d] + 1:
            dist[i][j + 1][d] = dist[i][j][d] + 1
            qi[qr] = i
            qj[qr] = j + 1
            qd[qr] = d
            qr += 1

    if dist[i][j][(d + 1) % 4] > dist[i][j][d] + 1:
        dist[i][j][(d + 1) % 4] = dist[i][j][d] + 1
        qi[qr] = i
        qj[qr] = j
        qd[qr] = (d + 1) % 4
        qr += 1

    if dist[i][j][(d + 3) % 4] > dist[i][j][d] + 1:
        dist[i][j][(d + 3) % 4] = dist[i][j][d] + 1
        qi[qr] = i
        qj[qr] = j
        qd[qr] = (d + 3) % 4
        qr += 1

print(ans)
```

Задача 4.1.5. Робот и камушки (25 баллов)

Ильнар в одной из комнат увидел странного робота. Во время выполнения алгоритма, он доставал из мешка разные камушки. Причем он никогда не доставал один и тот же камень два раза. И говорил сколько камушков в мешке такого же цвета.

Разработчик робота рассказал Ильнару, что из-за ошибки в коде робот ровно один раз всегда ошибается.

Теперь Ильнару интересно, а сколько минимально может быть камушков в мешке.

Примечание

Робот говорит количество камушков того же цвета, что в руке. При этом камень, который у него в руках, он тоже учитывает. После данной операции камень возвращается обратно в мешок.

Формат входных данных

В первой строке содержится единственное целое число n ($1 \leqslant n \leqslant 10^5$) – количество выбранных камушков.

Во второй строке находятся n целых чисел a_i ($1 \leqslant a_i \leqslant 10^9$) – значения, названные роботом.

Формат выходных данных

Выведите одно положительное целое число – минимальное возможное количество камушков в мешке.

Примеры

Пример №1

Стандартный ввод
4 2 2 2 2
Стандартный вывод
5

Решение

Сгруппируем все одинаковые значения.

Пусть cnt_x – количество камней, про которые сказано, что такого же цвета x камней.

Если бы все n высказываний были правдивы, ответом было бы $\sum_{x \in \mathbb{N}} ceil(\frac{cnt_x}{x}) \cdot x$, где ceil – округление вверх.

Так как одно высказывание ложно, то некоторое значение cnt_{x_1} должно увеличиться на 1, а некоторое cnt_{x_2} должно уменьшиться на 1.

Давайте для каждого значения x посчитаем, как изменится ответ, если мы прибавим 1 к cnt_x и если отнимем 1 от cnt_x.

Дальше нужно найти такую пару, которая минимизирует суммарный результат. Это сделать можно, например, с помощью динамического программирования.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n = int(input())

a = list(map(int, input().split()))
a.sort()

sum = 0
hide = []
hide.append(1)
toHide = 0
l = 0
r = 0

while l < n:
    while r < n and a[l] == a[r]:
        r += 1
    sum += (r - l + a[l] - 1) // a[l] * a[l]
    if (r - l + a[l] - 2) // a[l] < (r - l + a[l] - 1) // a[l]:
        toHide = a[l]
    if (r - l + a[l]) // a[l] == (r - l + a[l] - 1) // a[l]:
        hide.append(a[l])
    l = r

```

```
hide.reverse()

if toHide == 0:
    if len(hide) == 1 or (a[0] == hide[0] and a[n - 1] == hide[0]):
        sum += 1
elif toHide == 1:
    if len(hide) > 1:
        sum -= 1
    else:
        sum += 1
else:
    for i in hide:
        if i != toHide:
            sum = sum - toHide + (1 if i == 1 else 0)
            break

print(sum)
```

4.2. Вторая попытка.

Задача 4.2.1. Призы (10 баллов)

Организаторы олимпиады по робототехнике решили, что у них будет ровно n участников финального этапа. Они хотят подарить каждому участнику батончики KitKat и уже нашли, что можно купить по низкой цене пачки, в которых будет ровно по m батончиков.

Организаторы решили, что все участники должны получить поровну батончиков, иначе кто-то обидится. Также они решили, что не должно остаться лишних батончиков. Поэтому им нужно определить минимальное количество пачек, которые они должны купить.

Формат входных данных

Вводится два целых числа n и m ($1 \leqslant n, m \leqslant 10^9$) – количество участников и количество батончиков в пачке.

Формат выходных данных

Выведите единственное число – ответ на задачу.

Примеры

Пример №1

Стандартный ввод
8 4

Стандартный вывод
2

Решение

Задача сводится к тому, что нужно найти такое минимальное число k, что $k \cdot m$ делится на n без остатка.

Получается, что $k \cdot m = lcm(m, n)$, где lcm – это наименьшее общее кратное.

Так как $lcm(a, b) \cdot gcd(a, b) = a \cdot b$, где lcm – это наименьшее общее кратное, а gcd – это наибольший общий делитель, получаем:

$lcm(m, n) \cdot gcd(m, n) = m \cdot n \Rightarrow k \cdot m = lcm(m, n) = \frac{m \cdot n}{gcd(m,n)} \Rightarrow k = \frac{n}{gcd(m,n)}$

Таким образом, ответом будет $\frac{n}{gcd(m,n)}$.

Пример программы-решения

Ниже представлено решение на языке Python3

```
from fractions import gcd

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

Задача 4.2.2. k-экстремум (15 баллов)

Дан массив a из n элементов.

Элемент массива a_i является k-экстремумом, если он является k-максимумом или k-минимумом.

Элемент массива a_i $(1 < i < n)$ является k-максимумом тогда и только тогда, когда $a_{i-1} + k \leqslant a_i$ и $a_{i+1} + k \leqslant a_i$.

Элемент массива a_i $(1 < i < n)$ является k-минимумом тогда и только тогда, когда $a_{i-1} - k \geqslant a_i$ и $a_{i+1} - k \geqslant a_i$.

Если a_i – один из крайних ($i = 1$ или $i = n$) элементов массива, то он не может быть k-экстремумом.

За одну единицу времени вы можете увеличить или уменьшить на единицу любой элемент массива.

Какое минимальное количество времени нужно потратить, чтобы в массиве появился хотя бы один k-экстремум?

Формат входных данных

В первой строке вводятся два целых числа n и k $(3 \leqslant n \leqslant 10^5, 1 \leqslant k \leqslant 10^9)$.

Во второй строке вводятся n чисел a_i $(1 \leqslant a_i \leqslant 10^9)$.

Формат выходных данных

Выведите единственное число – ответ на задачу.

Примеры

Пример №1

Стандартный ввод
3 2 4 3 4
Стандартный вывод
1

Решение

Заметим, что тратить время стоит только на увеличение или только на уменьшение одного конкретного элемента массива a_i $(1 < i < n)$.

Давайте для каждого элемента посчитаем, сколько нужно потратить времени, чтобы сделать его k-экстремумом.

Чтобы сделать i-ый $(1 < i < n)$ элемент k-максимумом, его значение должно стать не менее $\max(a_{i-1}+k, a_{i+1}+k)$, а на это нужно потратить $\max(a_i, a_{i-1}+k, a_{i+1}+k) - a_i$ единиц времени.

Чтобы сделать i-ый $(1 < i < n)$ элемент k-минимумом, его значение должно стать не более $\min(a_{i-1}-k, a_{i+1}-k)$, а на это нужно потратить $a_i - \min(a_i, a_{i-1}-k, a_{i+1}-k)$ единиц времени.

Тогда время, которое необходимо потратить, чтобы сделать i-ый элемент k-экстремумом, будет равно минимуму из этих двух значений. А ответом на всю задачу – минимальное значение по всем элементам.

Пример программы-решения

Ниже представлено решение на языке C++

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

using namespace std;

const int MAXN = 1e5 + 1;
int a[MAXN];

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    int n;
    cin >> n;
    int k;
    cin >> k;
    int ans = 1e9 + 500;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
    }
    for (int i = 2; i <= n - 1; ++i) {
```

```
22          //bigger
23          int val = a[i];
24          val = max(val, a[i - 1] + k);
25          val = max(val, a[i + 1] + k);
26          ans = min(ans, val - a[i]);
27          //smaller
28          val = a[i];
29          val = min(val, a[i - 1] - k);
30          val = min(val, a[i + 1] - k);
31          ans = min(ans, a[i] - val);
32
33      }
34      cout << ans << endl;
35
36      return 0;
37  }
```

Задача 4.2.3. Шпионаж (20 баллов)

Наше агентство осуществило перехват нескольких предположительно шпионских сообщений. Однако возникли проблемы при декодировании.

Нам удалось узнать, что:

- каждый символ изначального сообщения закодировали последовательностью из нулей и единиц;
- длина каждой из этих последовательностей равна k ;
- каждому символу поставлена в соответствие ровно одна последовательность из k нулей и единиц;
- каждой последовательности из k нулей и единиц поставлен в соответствие ровно один символ;
- экземпляры таблицы декодирования испорчены и не подлежат восстановлению.

Большего вам знать не нужно.

Для первичного отделения шпионских сообщений от сообщений, попавших в рассмотрение случайно, нам нужна программа, подсчитывающая количество различных символов, используемых в сообщении, представленном в виде строки.

Берётесь за эту работу?

Формат входных данных

В первой строке входных данных два целых числа:

$1 \leqslant n \leqslant 10^5$ – длина строки;

$1 \leqslant k \leqslant n$ – длина последовательностей, которыми были закодированы символы.

Во второй строке дано сообщение в виде строки s .

Гарантируется, что число n кратно k и закодированная строка s состоит из n символов, каждый из которых равен 0 или 1 .

Формат выходных данных

Выведите одно положительное число – количество различных символов в строке.

Примеры

Пример №1

Стандартный ввод
9 3 001000100
Стандартный вывод
3

Решение

В данной задаче нужно было разбить данную строку на строки длины k и определить количество различных строк в получившемся наборе строк.

Для этого можно было, например, отсортировать строки и посчитать количество строк, каждая из которых не равна предшествующей ей строке в получившемся отсортированном массиве строк. А можно было добавить все строки набора в множество (set) и посмотреть на размер получившегося множества.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n, k = (int(i) for i in input().split())
s = input()
print(len(set(s[i * k : (i + 1) * k] for i in range(n // k))))
```

Задача 4.2.4. Цепь (25 баллов)

Герои книги "Автостопом по галактике" в своем путешествии случайно попали в параллельную вселенную. В этой вселенной все по-другому. Можно перемещаться не только по пространству, но и во времени. Для этого они используют свой звездолет.

Законы физики и всех других наук здесь работают не так.

В этой вселенной есть n планет, для каждой планеты известно ее расстояние от центра вселенной r_i и коэффициент искажения пространства k_i.

Чтобы сделать эту вселенную более интересной для туристов из других вселенных, наши герои решили соединить все планеты с помощью пар звездных врат. Между звездными вратами можно перемещаться в обе стороны. При этом для каждой планеты герои выбрали год y_i, когда эта планета наиболее интересна туристам. Герои хотят, чтобы после этого построения звездных врат можно было посетить все планеты.

При этом для построения звездных врат нужно будет потратить часть энергии звездолета.

Чтобы построить портал между i-ой и j-ой планетами, нужно потратить

$3 \cdot (r_i - r_j)^2 + 2 \cdot |(2 \cdot k_i - 2 \cdot k_j) \cdot (2 \cdot k_i + 2 \cdot k_j)| + 5 \cdot |y_i - y_j|$ единиц энергии.

Герои хотят минимизировать суммарное количество энергии, которое нужно потратить на построение звездных врат, чтобы как можно больше топлива осталось им для остальных путешествий. Помогите им это сделать.

Формат входных данных

В первой строке вводится целое число n ($1 \leqslant n \leqslant 10^4$) – количество планет.

В следующих n строках вводятся по три целых числа r_i, k_i, y_i ($0 \leqslant r_i, k_i, y_i \leqslant 10^6$)
.

Формат выходных данных

Выведите единственное число – минимальное суммарное количество энергии.

Примеры

Пример №1

Стандартный ввод
2 1 1 1 2 2 2

Стандартный вывод
32

Решение

Переведём задачу на язык графов: планеты – это вершины графа, а звездные врата между планетами – рёбра.

Необходимо обеспечить существование пути между любой парой вершин в графе, причём суммарный вес всех рёбер в графе должен быть минимален. То есть задача сводится к нахождению минимального остовного дерева.

Поскольку в задаче дан полный граф с достаточно большим количеством вершин, для решения данной задачи выгоднее выбрать алгоритм Прима для плотных графов с асимптотикой работы $O(n^2 + m)$, нежели алгоритм Краскала или алгоритм Прима для разреженных графов.

Пример программы-решения

Ниже представлено решение на языке C++

```
1  #include <iostream>
2
3  using namespace std;
4  typedef long long ll;
5
```

```cpp
const int MAXN = 10000 + 1;
const ll inf = 1e18;
ll r[MAXN], k[MAXN], y[MAXN];

inline ll my_abs(ll x) {
    if (x > 0)
        return x;
    return -x;
}

inline ll cost(int i, int j) {
    return 3 * (r[i] - r[j]) * (r[i] - r[j]) +
           2 * my_abs(2 * k[i] - 2 * k[j]) * my_abs(2 * k[i] + 2 * k[j]) +
           5 * my_abs(y[i] - y[j]);
}

ll mn[MAXN];
bool used[MAXN];

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> r[i] >> k[i] >> y[i];
        mn[i] = inf;
    }

    ll ans = 0;

    for (int step = 1; step <= n; ++step) {
        int ver = -1;
        for (int i = 1; i <= n; ++i) {
            if (used[i])
                continue;
            if (ver == -1 || mn[i] < mn[ver]) {
                ver = i;
            }
        }
        used[ver] = 1;
        if (step > 1)
            ans += mn[ver];
        for (int j = 1; j <= n; ++j) {
            if (used[j])
                continue;
            mn[j] = min(mn[j], cost(ver, j));
        }

    }
    cout << ans << endl;
    return 0;
}
```

Задача 4.2.5. Сжали (30 баллов)

Строка t является результатом сжатия строки s, если (должны выполняться все пункты):

- Строка t является подпоследовательностью строки s .

  Строка t длины k является подпоследовательностью строки s длины n, если существует такой набор чисел p, что i-ый символ строки t равен p_i-ому символу строки s и выполняется неравенство $1 \leqslant p_1 < p_2 < \ldots < p_{k-1} < p_k \leqslant n$.
- Первый символ строки t равен первому символу строки s (то есть $p_1 = 1$).
- Последний символ строки t равен последнему символу строки s (то есть $p_k = n$).
- Если i-ый $(1 \leqslant i < k)$ символ строки t является m-ой буквой латинского алфавита, то в удовлетворяющем строке t наборе чисел p между p_i-ым и p_{i+1}-ым символами в строке s не больше $bonus_m$ символов (то есть $p_{i+1} - p_i - 1 \leqslant bonus_m$) .

Стоимость строки определяется как сумма стоимостей всех ее символов. Стоимость i-ой буквы латинского алфавита равна $cost_i$.

Определите стоимость самой дешёвой из сжатых строк s .

Формат входных данных

В первой строке входных данных одно целое число:

$3 \leqslant n \leqslant 10^6$ – длина строки s .

Во второй строке – сама строка s , состоящая из строчных латинских букв.

В третьей строке – 26 целых чисел:

$0 \leqslant cost_i \leqslant 10^9$ – стоимость добавления в сжатую строку i-го $(1 \leqslant i \leqslant 26)$ символа латинского алфавита.

В четвертой строке – 26 целых чисел:

$0 \leqslant bonus_i \leqslant 10^9$ – сколько символов можно пропустить в строке s после добавления в сжатую строку i-го $(1 \leqslant i \leqslant 26)$ символа латинского алфавита.

Формат выходных данных

Выведите одно неотрицательное число – стоимость самой дешёвой из полученных описанным сжатием строк.

Примеры

Пример №1

Стандартный ввод
8 abcdebha 0 2 2 3 0 0 1 1 2 1 0 3 2 0 2 2 3 1 0 1 2 2 0 1 2 3 1 2 0 2 1 1 2 0 1 1 1 1 0 1 2 0 0 1 2 2 3 0 1 3 1 2
Стандартный вывод
3

Решение

У данной задачи множество различных решений. Перечислим некоторые из них:

- Решение с использованием динамического программирования и дерева отрезков.
  Асимптотика: $O(n \cdot \log(n))$
- Решение с использованием динамического программирования и множества (set).
  Асимптотика: $O(n \cdot \log(n))$
- Решение с использованием динамического программирования.
  Асимптотика: $O(n \cdot 26)$
- Решение с использованием динамического программирования и стека на массиве.
  Асимптотика: $O(n \cdot \log(n))$

Мы подробно разберём только одно решение – решение с использованием динамического программирования и стека на массиве. И начнём с части про динамическое программирование, которая присутствует в каждом решении в том или ином виде.

Ответ для префикса строки длины i равен:

$$dp_i = \begin{cases} cost_{s_i}, & \text{если } i = 1; \\ min(dp_j) + cost_{s_i}, & \text{если } i > 1\text{; ограничения на } j\text{: } 1 \leqslant j < i \text{ и } i - j - 1 \leqslant bonus_{s_j}. \end{cases}$$

Тогда ответом на задачу является dp_n.

Посчитать такую динамику можно за $O(n \cdot \log(n))$, если умело воспользоваться множеством (set), но мы пойдём другим путём.

Развернём строку. Теперь ответ для префикса (который на самом деле является суффиксом) строки длины i равен:

$$dp_i = \begin{cases} cost_{s_i}, & \text{если } i = 1; \\ min(dp_j) + cost_{s_i}, & \text{если } i > 1\text{; ограничения на } j\text{: } \max(1, i - 1 - bonus_{s_i}) \leqslant j < i. \end{cases}$$

Ответом на задачу по-прежнему является dp_n.

Такую динамику достаточно быстро (за $O(n \cdot \log(n))$ или $O(n \cdot 26)$) можно посчитать разными способами. Мы для нахождения dp_n воспользуемся стеком на массиве.

Будем поддерживать стек на массиве: первый элемент массива – дно стека, а последний элемент – вершина стека. На элементы стека будет наложено следующее условие:

Если i лежит в стеке и лежит не на дне, то предшествующий ему в стеке элемент – это такое j, что $dp_j < dp_i$ и $j < i$; причём из всех j, удовлетворяющих этим условиям, предшествует i в стеке наибольшее j. Если i лежит на дне стека, то такого j, что $dp_j < dp_i$ и $j < i$, не существует.

Добавлять элементы в стек будем в порядке возрастания i. При $i = 1$ стек пуст и мы просто добавляем в стек i. При $i > 1$ сначала удалим с вершины стека элементы, которым соответствуют значения динамики, которые не меньше dp_i, и только потом добавим в стек i. Таким образом мы получаем новый стек, который не содержит ни одного элемента, относительно которого не соблюдалось бы описанное выше условие, и на вершине стека лежит i.

Для вычисления dp_i $(i > 1)$ воспользуемся стеком, полученным описанным способом и на вершине которого лежит $(i-1)$. Не сложно заметить, что j, при котором достигается значение $min(dp_j)$ в рекуррентной формуле для dp_i, равен наименьшему элементу стека, который не меньше $\max(1, i-1-bonus_{s_i})$. Поскольку элементы стека строго возрастают от дна стека к его вершине, такой элемент можно найти бинарным поиском (за $O(\log(n))$).

Так мы вычислим значение dp_n за $O(n \cdot \log(n))$.

Пример программы-решения

Ниже представлено решение на языке Python3

```
n = int(input())

s = input().strip()
s = s[::-1]

alphabet = 26
cost = list(map(int, input().split()))
bonus = list(map(int, input().split()))

ans = [0] * n
ans[0] = cost[ord(s[0]) - ord('a')]

st = [0] * n
ss = 1

for i in range(n - 1):
    l = -1
    r = ss - 1
    while l + 1 < r:
        mid = (l + r) // 2
        if st[mid] < i - bonus[ord(s[i + 1]) - ord('a')]:
            l = mid
        else:
            r = mid

    ans[i + 1] = ans[st[r]] + cost[ord(s[i + 1]) - ord('a')]

    while ss > 0 and ans[st[ss - 1]] >= ans[i + 1]:
```

```
29          ss -= 1
30      st[ss] = i + 1
31      ss += 1
32
33  print(ans[n - 1])
```

4.3. Третья попытка.

Задача 4.3.1. Печеньки (10 баллов)

У Ильнара есть три коробки с печеньем, в которых a, b, c печенек соответственно. К Ильнару в гости пришло $(n-1)$ человек. Ильнар хочет, чтобы всем гостям и ему досталось одинаковое количество печенек. Поэтому он хочет узнать сколько печенек достанется каждому, если он откроет несколько (возможно ни одной, возможно все три) коробок с печеньем, при этом все печеньки из открытых коробок должны быть розданы поровну (сами печенья нельзя ломать на части).

Определите максимальное количество печенек, которое получит каждый гость и Ильнар.

Формат входных данных

В первой строке вводится 4 целых числа a, b, c, n $(1 \leqslant a, b, c, n \leqslant 10^8)$.

Формат выходных данных

Выведите единственное число – ответ на задачу.

Примеры

Пример №1

Стандартный ввод
1 2 3 4
Стандартный вывод
1

Пример №2

Стандартный ввод
3 4 5 19
Стандартный вывод
0

Решение

Переберем все возможные варианты выбора коробок. Получаем 7 вариантов количества печенек в открытых коробках:

$a, b, c, a + b, a + c, b + c, a + b + c$

Из них возьмем максимальное значение, которое делится на n без остатка. Если все 7 значений не делятся на n без остатка, то ответ – 0.

Пример программы-решения

Ниже представлено решение на языке Python3

```
a, b, c, n = [int(x) for x in input().split()]
for i in sorted([a, b, c, a + b, a + c, b + c, a + b + c, 0], reverse=True):
    if i % n == 0:
        print(i // n)
        break
```

Задача 4.3.2. Круглый стол (20 баллов)

Организаторы одного из турниров по одной из самых известных карточных игр заказали круглый стол, за которым будет сидеть несколько (минимум 2) игроков во время каждой из игр.

Профсоюз игроков этой карточной игры выдвинул условие: каждый игрок на время игры должен иметь личное пространство как минимум длины d.

Если посчитать стол кругом, то длина пространства игрока определяется как длина хорды, которая отделяет сегмент круга, на котором игрок может положить свои карты и руки, от остальной части стола.

Помогите организаторам определить, какое максимальное количество игроков может одновременно сидеть за этим столом.

Формат входных данных

Вводится два целых числа d и r ($1 \leqslant d, r \leqslant 2 \cdot 10^9$) – минимальная длина личного пространства и радиус круглого стола.

Формат выходных данных

Если нельзя усадить за стол даже двух игроков, выведите -1, иначе выведите единственное положительное число – максимальное количество игроков, которых можно разместить за столом.

Примеры

Пример №1

Стандартный ввод
6 3
Стандартный вывод
2

Пример №2

Стандартный ввод
7 2
Стандартный вывод
-1

Пример №3

Стандартный ввод
12 1000
Стандартный вывод
523

Пример №4

Стандартный ввод
3 3
Стандартный вывод
6

Решение

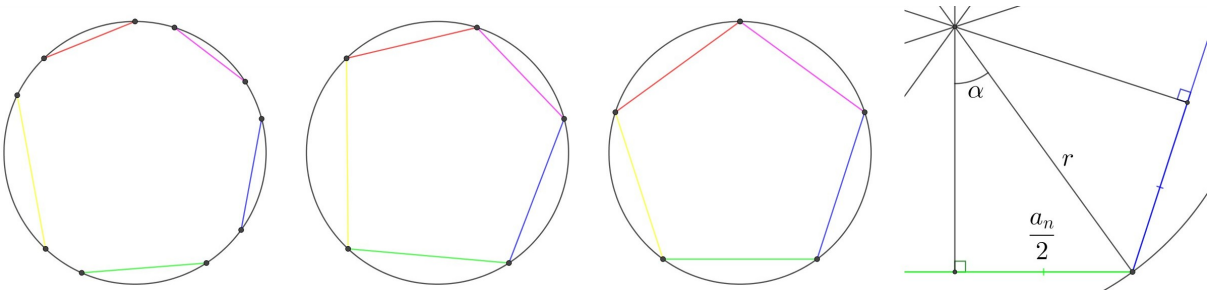
Очевидно, что если $d > 2 \cdot r$, то за стол даже двух игроков не посадить и в качестве ответа нужно выводить -1. Далее будем решать задачу с дополнительным ограничением $d \leqslant 2 \cdot r$.

Заметим, что при фиксированном количестве игроков, которых организаторы хотят усадить за стол, мы можем рассчитать наибольшую длину личного пространства, которое можно предоставить каждому из игроков.

Пусть организаторы хотят усадить за стол $n = 5$ игроков. Любым способом, который не противоречит условию, проведём хорды, отделяющие личное пространство игроков от остальной части стола (см. рис. 1).

Заметим, что могут остаться дуги, "не покрытые" ни одной из хорд. "Покроем" каждую из таких дуг одной из близлежащих хорд (см. рис. 2), что позволит не только не уменьшить длину наименьшего из предоставленных личных пространств, но и, возможно, увеличить эту длину.

Таким образом, получившиеся хорды стали образовывать n-угольник, около которого описана окружность.



Также заметим, что если существует пара хорд, длины которых различны, то длину меньшей из них можно увеличить за счёт уменьшения длины большей.

А теперь заметим, что для максимизации длины личного пространства необходимо сделать все хорды равными по длине, поскольку если длина меньшей хорды не равна длине большей, то меньшую хорду можно увеличить за счёт большей, таким образом увеличив длину личного пространства.

В итоге хорды стали образовывать правильный n-угольник (см. рис. 3).

Используя свойства правильных многоугольников, можно вывести формулу, выражающую длину стороны правильного n-угольника через радиус описанной около n-угольника окружности (см. рис. 4):

$$\sin(\alpha) = \sin(\tfrac{2\cdot\pi}{2\cdot n}) = \tfrac{a_n}{2\cdot r} \Leftrightarrow a_n = 2 \cdot r \cdot \sin(\tfrac{\pi}{n}) \ ,$$

где a_n – длина стороны правильного n-угольника, которая в свою очередь равна наибольшей длине личного пространства, которое мы можем предоставить каждому из n игроков на столе радиуса r.

Таким образом, остаётся определить ответ, который равен такому наибольшему n, что $a_n \geqslant d$. Искомое n можно найти бинарным поиском по ответу, так как последовательность, заданная формулой $a_i = 2 \cdot r \cdot \sin(\frac{\pi}{i})$ для $i \geqslant 2$, является убывающей. Но можно пойти дальше и доказать, что ответом на задачу является $n = \left\lfloor \frac{\pi}{\arcsin(\frac{d}{2\cdot r})} \right\rfloor$

Пример программы-решения

Ниже представлено решение на языке C++

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

#define EPS 1e-9

using namespace std;

int main() {
    double d, r;
    cin >> d >> r;
    double sine = d / 2 / r;
    if (sine > 1)
        cout << -1;
    else {
        double angle = asin(sine);
        cout << (long long) floor(M_PI / angle + EPS);
    }
    return 0;
}
```

Задача 4.3.3. Анна и красивая строка (20 баллов)

Анна написала генератор красивых строк. Она считает строку красивой, если она одинаково читается как слева направо, так и справа налево. Например, *rrr* и *anna* – красивые строки, а *abc* и *adba* – нет.

Но она допустила ошибку в коде и генератор выводит не красивые строки, а строки, которые можно сделать красивыми, если из каждой удалить ровно один символ. По крайней мере, она так думает.

Она просит вас помочь определить, верно ли её предположение.

Формат входных данных

В первой строке вводится строка s состоящая из маленьких латинских букв ($4 \leqslant |s| \leqslant 10^5$, где $|s|$ – длина строки).

Формат выходных данных

Выведите *YES* , если можно удалить один символ из строки так, чтобы она стала красивой; иначе – *NO* .

Примеры

Пример №1

Стандартный ввод
`abca`
Стандартный вывод
`YES`

Пример №2

Стандартный ввод
`abcd`
Стандартный вывод
`NO`

Решение

В данной задаче нас просят определить, может ли строка s длины n стать палиндромом, если из неё удалить ровно один символ. Если может, то ответ *YES* , иначе – *NO* .

Заметим, что если строка s является палиндромом, то можно удалить из неё $\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)$-ый символ и строка s останется палиндромом, то есть в качестве ответа нужно вывести *YES* .

Теперь рассмотрим случай, когда строка s не является палиндромом, то есть существует такое k ($k \geqslant 1$), что $s_1 = s_n$, $s_2 = s_{n-1}$, ..., $s_{k-1} = s_{n-k+2}$, $s_k \neq s_{n-k+1}$ ($k < n - k + 1$), где s_i – i-ый символ строки.

Докажем, что если $s_1 = s_n$, то ответ для строки s, из которой предварительно удалили первый и последний символы, будет такой же, как и для самой строки s.

Докажем от противного. Если утверждение не верно, то строка s может стать палиндромом после удаления i-го символа ($1 \leqslant i \leqslant n$), только если $i = 1$ или $i = n$. Но если после удаления первого символа получился палиндром, то получается, что $s_2 = s_n = s_1$, то есть не важно, удалим мы первый или второй символ, а если палиндром получился после удаления последнего символа, то $s_{n-1} = s_1 = s_n$, то есть не важно, удалим мы последний или предпоследний символ. Пришли к противоречию.

Следовательно, если после удаления одного символа строка s может стать палиндромом, то существует такое i, что соблюдается двойное неравенство $k \leqslant i \leqslant n-k+1$ и строка s станет палиндромом, если удалить i-ый символ. Также заметим, что после удаления такого i-го символа, что соблюдается двойное неравенство $k < i < n-k+1$, строка s точно не станет палиндромом, так как $s_k \neq s_{n-k+1}$, что противоречит определению палиндрома.

Таким образом, для определения ответа достаточно посмотреть, станет ли строка s палиндромом после удаления k-го символа и станет ли строка s палиндромом после удаления $(n-k+1)$-го символа. Если хотя бы в одном из этих двух случаев строка s станет палиндромом, то ответ *YES* , иначе – *NO* .

Пример программы-решения

Ниже представлено решение на языке Python3

```
s = input().strip()
n = len(s)
for i in range((n - 1) // 2 + 1):
    if s[i] != s[n - 1 - i] or i == (n - 1) // 2:
        t = "".join([('' if n - 1 - i == j else s[j]) for j in range(n)])
        s = "".join([('' if i == j else s[j]) for j in range(n)])
        break
print('YES' if t == t[::-1] or s == s[::-1] else 'NO')
```

Задача 4.3.4. Робот (25 баллов)

Ильнар решил научить робота перемещаться по плоскости. Но у него оставалось мало времени и он решил немного схитрить. Он научил робота перемещаться правильно только на дистанцию не более чем $\sqrt{k}$. Чтобы робот смог при этом добраться от начальной точки до конечной, он задал в конфигурации еще дополнительные точки на плоскости. Теперь робот может перемещаться только между этими точками.

Чтобы добраться от точки i до точки j, робот тратит $(x_i - x_j)^2 + (y_i - y_j)^2$ единиц времени.

Формат входных данных

В первой строке заданы числа n и k ($1 \leqslant n \leqslant 5000$, $1 \leqslant k \leqslant 10^9$) .

В следующих n строках заданы координаты точек x_i, y_i ($1 \leqslant x_i, y_i \leqslant 20000$) .

В последней строке заданы два числа s и t ($1 \leqslant s, t \leqslant n$, $s \neq t$) – начальная и конечная точки пути робота.

Формат выходных данных

Выведите единственное число – минимальное время за которое робот доберется из s в t или -1, если робот не может добраться.

Примеры

Пример №1

Стандартный ввод
2 32 1 1 5 5 1 2
Стандартный вывод
32

Пример №2

Стандартный ввод
2 31 1 1 5 5 1 2
Стандартный вывод
-1

Решение

Переведём задачу на язык графов: точки – это вершины графа, а отрезки, соединяющие две точки, квадрат расстояния между которыми не превышает k – рёбра графа.

Задача сводится к тому, что нужно найти кратчайшее расстояние от вершины, соответствующей точке s, до вершины, соответствующей точке t, во взвешенном графе, веса рёбер которого неотрицательны. А для решения такой задачи можно воспользоваться алгоритмом Дейкстры.

Также стоит отметить, что поскольку количество вершин в графе не превышает 5000 и описанный в тесте граф может быть достаточно плотным, то стоит отдать предпочтение реализации алгоритма Дейкстры для плотных графов с асимптотикой работы $O(n^2 + m)$.

Пример программы-решения

Ниже представлено решение на языке C++

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
```

```
4  typedef long long ll;
5  const ll inf_ll = 1e16;
6  const int MAXN = 1e5 + 100;
7
8  ll x[MAXN], y[MAXN];
9  ll dis[MAXN];
10
11 ll dist(int a, int b) {
12     return (x[a] - x[b]) * (x[a] - x[b]) + (y[a] - y[b]) * (y[a] - y[b]);
13 }
14
15 bool used[MAXN];
16
17 int main() {
18     int n;
19     cin >> n;
20     ll k;
21     cin >> k;
22     for (int i = 1; i <= n; ++i) {
23         cin >> x[i] >> y[i];
24     }
25     int s, t;
26     cin >> s >> t;
27     for (int i = 1; i <= n; ++i) {
28         dis[i] = inf_ll;
29     }
30     dis[s] = 0;
31     for (int i = 1; i <= n; ++i) {
32         int ver = -1;
33         for (int e = 1; e <= n; ++e) {
34             if (!used[e] && (ver == -1 || dis[e] < dis[ver])) {
35                 ver = e;
36             }
37         }
38
39         int v = ver;
40         used[v] = 1;
41         for (int to = 1; to <= n; ++to) {
42             if (dist(v, to) > k)
43                 continue;
44             if (dis[to] > dis[v] + dist(v, to)) {
45                 dis[to] = dis[v] + dist(v, to);
46             }
47         }
48     }
49
50     if (dis[t] == inf_ll) {
51         cout << -1 << endl;
52     } else {
53         cout << dis[t] << endl;
54     }
55 }
```

Задача 4.3.5. ЗАГС (25 баллов)

В стране M есть город N. А в этом городе есть ЗАГС. И сегодня там случилась поломка ПО.

Были потеряны данные о заявлениях на регистрацию брака. Однако остался

журнал прихода и вызова.

В журнале в хронологическом порядке записаны события двух видов:

- Приход человека, желающего подать заявление. В журнале записана фамилия этого человека.
- Объявление о вызове для утверждения заявления и проверки документов. В этом случае в журнале записано слово $next$.

В стране M принято, что супруги должны иметь одинаковую фамилию и начиная с прихода в ЗАГС для подачи заявления на регистрацию брака должны называть фамилию, которую будут носить после вступления в брак.

В городе N заявление на регистрацию брака принимается только в присутствии обоих будущих супругов.

После прихода в ЗАГС и внесения в журнал прихода и вызова соответствующей записи нужно встать в очередь. В случае, если один из будущих супругов пришёл раньше, то другой или другая присоединяются к будущей супруге или будущему супругу.

Когда вызывают следующую пару для утверждения заявления, может оказаться так, что первым в очереди стоит человек, будущая супруга или будущий супруг которого ещё не пришёл в ЗАГС. В этом случае на утверждение заявления идут те уже пришедшие в ЗАГС будущие супруги, которые находятся ближе всего к началу очереди. В случае, если в очереди нет ни одной пары, на вызов пойдёт первая появившаяся в очереди пара сразу после прихода второго супруга или супруги.

Ваша задача – восстановить по данной информации события, произошедшие в ЗАГСе города N за сегодня.

Формат входных данных

В первой строке дано число n ($n \leqslant 10^5$) – количество записей в журнале.

В каждой из следующих n строк находится либо слово $next$, обозначающее вызов следующей пары, либо фамилия, состоящая из латинских букв и длиной не менее одного и не более двадцати символов.

Каждая из фамилий встречается в записях журнала не менее одного и не более двух раз.

Формат выходных данных

В случае прихода человека, супруг или супруга которого ещё не пришёл или не пришла в ЗАГС, выведите перед фамилией этого человека $1st$. Таким образом мы опишем событие становления в конец очереди нового пришедшего.

В случае прихода человека, супруг или супруга которого уже в ЗАГСе, выведите перед фамилией этого человека $2nd$ без кавычек. Таким образом мы опишем событие появления в очереди пары на месте, которое занял пришедший ранее супруг или супруга.

В случае объявления о вызове, выведите фамилию будущих супругов, чьё заявление будет утверждаться следующим. В случае, если на момент вызова в очереди

нет ни одной пары будущих супругов, выводить фамилию будущих супругов, чьё заявление будет утверждаться следующим, следует только после появления этой пары в очереди. Если же до конца дня в ЗАГС не придёт ни одной пары, ничего выводить не нужно.

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

Для лучшего понимания формата выходных данных ознакомьтесь с примерами ниже.

Примеры

Пример №1

Стандартный ввод
5 Pit Wait Pit Wait next

Стандартный вывод
1st Pit 1st Wait 2nd Pit 2nd Wait Pit

Пример №2

Стандартный ввод
5 Pit Wait Wait Pit next

Стандартный вывод
1st Pit 1st Wait 2nd Wait 2nd Pit Pit

Пример №3

Стандартный ввод
5 next Pit Wait Wait Pit

Стандартный вывод
1st Pit 1st Wait 2nd Wait Wait 2nd Pit

Решение

Разобьём задачу на подзадачи. Что нам нужно уметь делать?

- Проверять, есть ли в очереди человек с определённой фамилией. И если есть, то помечать этого человека.
- Добавлять человека в конец очереди.
- Находить в очереди первого помеченного человека и удалять его из очереди.

Смоделируем очередь с помощью структуры данных очередь (queue). Добавлять в конец очереди новый элемент можно за $O(1)$, но находить помеченный или соответствующий определённой строке-фамилии элемент можно только за $O(n)$, что слишком медленно.

Чтобы быстро (за $O(\log(n))$ или быстрее) находить соответствующий определённой строке-фамилии элемент, будем хранить данные о находящихся сейчас в очереди элементах в словаре (map). Ключами в таком словаре будут строки-фамилии, а значение по определённой строке-фамилии – это положение элемента, соответствующего данной строке-фамилии, в очереди.

Как быстро находить первый помеченный элемент? Вместо того, чтобы в одной очереди хранить и помеченные, и непомеченные элементы, будем в одной очереди хранить помеченные, а в другой – непомеченные элементы. В изначальной очереди будем хранить непомеченные элементы, а вместо того, чтобы помечать некоторый элемент в изначальной очереди, будем переносить этот элемент во вторую очередь.

Но когда нужно будет найти первый помеченный элемент во второй очереди, мы уже не сможем просто взять первый элемент второй очереди. Нам нужно будет найти во второй очереди элемент, который был добавлен в изначальную очередь раньше всего – именно такой элемент должен считаться первым во второй очереди. Для быстрого нахождения и удаления такого элемента заменим структуру данных очередь (queue) на структуру данных очередь с приоритетом (priority queue), в которой в качестве приоритета в пару к элементу поставим целое число, которое будет соответствовать порядковому номеру строки, в которой во входных данных впервые появилась строка-фамилия, которой соответствует элемент. Напомним, что добавление в очередь с приоритетом, как и удаление первого элемента из неё, работает за $O(\log(n)))$.

Также отметим, что для отложенных операций удаления первого помеченного

элемента из второй очереди нужно поддерживать переменную-счётчик таких отложенных операций и, в случае появления нового элемента во второй очереди, совершать отложенную операцию, если хотя бы одна такая есть.

Таким образом, каждую из операций мы сможем сделать не более чем за $O(\log(n))$ и итоговая асимптотика работы программы будет $O(n \cdot \log(n))$.

Разумеется, существуют и другие способы решения данной задачи.

Пример программы-решения

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.PriorityQueue;
7  import java.util.StringTokenizer;
8  import java.util.TreeMap;
9
10 public class Main {
11     FastScanner in;
12     PrintWriter out;
13
14     private void solve() throws IOException {
15         int n = in.nextInt();
16
17         String[] q = new String[n];
18         int qs = 0, wait = 0;
19         TreeMap<String, Integer> map = new TreeMap<>();
20         PriorityQueue<Integer> pq = new PriorityQueue<>();
21
22         for (int i = 0; i < n; i++) {
23             String s = in.next();
24             if (s.equals("next")) {
25                 if (pq.isEmpty())
26                     wait++;
27                 else
28                     out.println(q[pq.remove()]);
29             } else {
30                 if (map.containsKey(s)) {
31                     out.println("2nd " + s);
32                     pq.add(map.get(s));
33                     map.remove(s);
34                     if (wait > 0) {
35                         out.println(q[pq.remove()]);
36                         wait--;
37                     }
38                 } else {
39                     out.println("1st " + s);
40                     map.put(s, qs);
41                     q[qs++] = s;
42                 }
43             }
44         }
45     }
46
```

```
47      class FastScanner {
48          StringTokenizer st;
49          BufferedReader br;
50
51          FastScanner(InputStream s) {
52              br = new BufferedReader(new InputStreamReader(s));
53          }
54
55          String next() throws IOException {
56              while (st == null || !st.hasMoreTokens())
57                  st = new StringTokenizer(br.readLine());
58              return st.nextToken();
59          }
60
61          boolean hasNext() throws IOException {
62              return br.ready() || (st != null && st.hasMoreTokens());
63          }
64
65          int nextInt() throws IOException {
66              return Integer.parseInt(next());
67          }
68
69          long nextLong() throws IOException {
70              return Long.parseLong(next());
71          }
72
73          double nextDouble() throws IOException {
74              return Double.parseDouble(next());
75          }
76
77          String nextLine() throws IOException {
78              return br.readLine();
79          }
80
81          boolean hasNextLine() throws IOException {
82              return br.ready();
83          }
84      }
85
86      private void run() throws IOException {
87          in = new FastScanner(System.in);
88          out = new PrintWriter(System.out);
89
90          solve();
91
92          out.flush();
93          out.close();
94      }
95
96      public static void main(String[] args) throws IOException {
97          new Main().run();
98      }
99  }
```

2. ВТОРОЙ ЭТАП

Описание этапа

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

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

Объем и сложность задач этого этапа подобраны таким образом, чтобы решение всех задач одним человеком было маловероятно. Это призвано обеспечить включение командной работы и распределения обязанностей. Решение каждой задачи дает определенное количество баллов. Баллы зачисляются в полном объеме за правильное решение задачи. Также существуют задачи, где допускается частичное решение. В данном этапе можно получить суммарно от 0 до 138 баллов.

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

Задачи второго этапа

5.1. Задачи

Задача 5.1.1. Запуск всенаправленной тележки (5 баллов)

Робот, моторы которого расположены под углом в 120° друг к другу, движется в некотором направлении, пока на моторы подается мощность. Каждый мотор работает некоторое время: t_1, t_2, t_3, соответственно. На валах моторов закреплены омниколёса (`https://en.wikipedia.org/wiki/Omni_wheel`). С некоторой кинематической моделью робота можно познакомиться по ссылке: `https://bharat-robotics.github.io/blog/kinematic-analysis-of-holonomic-robot/`.

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. 5.1). Расположение моторов является постоянным и соответствует рисунку 5.1.

Считать, что мощность на все моторы подаётся одновременно и достигается мгновенно. Также считать что происходит движение без поворотов, иначе говоря робот двигается только прямолинейно в любом направлении. Колёса вращаются без проскальзывания.

Данные в тестах подобраны таким образом, что нет вариата движения, когда робот движется вокруг какой-то точки.

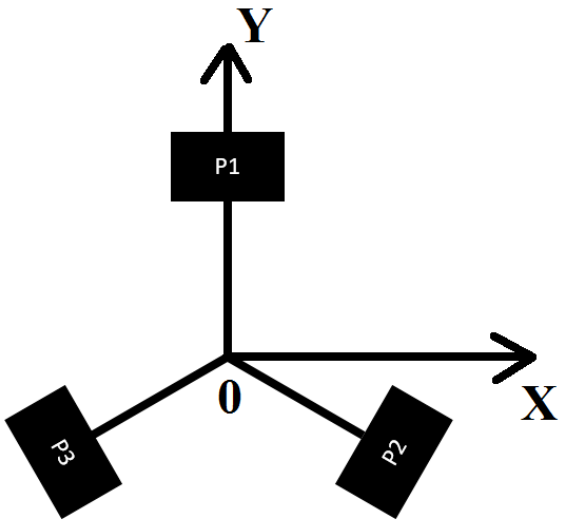


Рис. 5.1: Расположение моторов робота относительно центра глобальной системы отсчета в начальный момент времени

Формат входных данных

Одна строчка, состоящая из 7ми чисел: d, w_1, t_1, w_2, t_2, w_3, t_3, разделёнными пробелами, где

- d — диаметр колёс в мм ($30 \leqslant d \leqslant 100$);
- w_1, w_2, w_3 — скорости вращения моторов в рад/с ($-2 \leqslant w_1, w_2, w_3 \leqslant 2$);
- t_1, t_2, t_3 — время работы каждого мотора в с ($10 \leqslant t_1, t_2, t_3 \leqslant 1000$).

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

Комментарии

Дополнительные наборы входных данных доступны по `http://bit.ly/2RdizA3`ссылке.

Формат выходных данных

Одна строка, содержащая два целых числа через пробел — координаты центра робота в мм, где он закончил своё движение. Допускается погрешность в 1 мм по каждой из координат.

Примеры

Пример №1

Стандартный ввод
`35 1 15 0 0 -1 15`
Стандартный вывод
`196.875 -113.666`

Пример №2

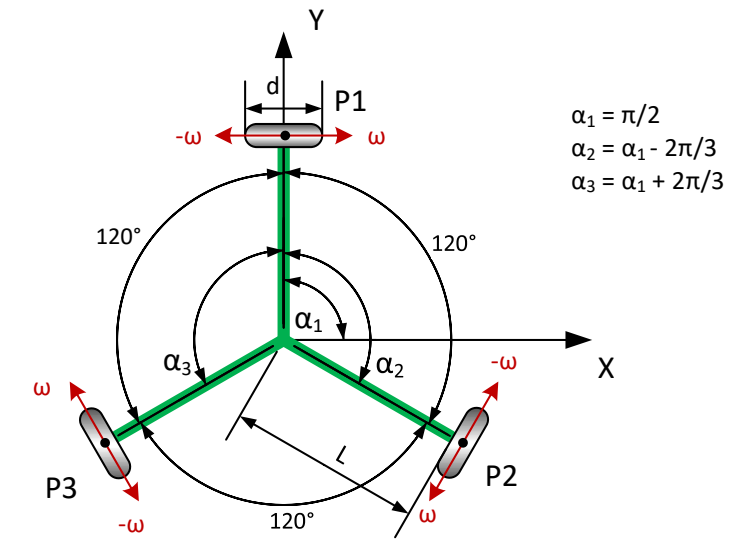
Стандартный ввод
`40 1.4 100 -1.4 100 1.4 100`
Стандартный вывод
`2800 2424.87`

Решение

Рассмотрим схему всенаправленной тележки (рис. 5.2).

У каждого мотора может быть 3 основных состояния: 'вперед', 'назад', 'выключен'.

Определим направления вращения моторов: пусть при угловой скорости $\omega_i < 0$ колеса вращаются "вперед"или по часовой стрелке (если смотреть со стороны колес), если $\omega_i > 0$, то колеса вращаются "назад"или против часовой стрелки.



d – диаметр колеса, $d_1 = d_2 = d_3$

L – длина оси от центра робота до колеса, $L_1 = L_2 = L_3$

Рис. 5.2: Схема всенаправленной тележки

По условиям задачи движение происходит без вращения робота, т.е. он двигается только прямолинейно в любом направлении. Рассмотрим комбинации угловых скоростей (ω), при которых всенаправленная тележка едет только прямо (таблица 5.1), при этом угловые скорости моторов (ω) должны быть или равны по модулю или у одного из моторов $\omega = 0$:

P1	↑	↑	↑	↑	0	0	↓	↓	↓	↓
P2	↓	↓	0	↓	↓	↑	↑	0	↑	↑
P3	↑	0	↓	↓	↑	↓	↑	↑	0	↓

где:
↑ - включение мотора 'вперед',
↓ - включение мотора 'назад',
0 - мотор выключен

Таблица 5.1: Варианты включения моторов для прямолинейного движения всенаправленной тележки

Исходя из этих условий, мы можем использовать упрощенную кинематическую модель, в которой получаем следующие уравнения прямой кинематики (формулы 5.1) и (5.2).

$$x = x_0 + \pi D \left(\omega_1 \cdot sin(\alpha) + \omega_2 \cdot sin\left(\alpha - \frac{2\pi}{3}\right) + \omega_3 \cdot sin\left(\alpha + \frac{2\pi}{3}\right)\right) \cdot t \qquad (5.1)$$

$$y = y_0 + \pi D \left(\omega_1 \cdot cos(\alpha) + \omega_2 \cdot cos\left(\alpha - \frac{2\pi}{3}\right) + \omega_3 \cdot cos\left(\alpha + \frac{2\pi}{3}\right)\right) \cdot t \qquad (5.2)$$

Поскольку в задаче сказано, что ось первого колеса совпадает с осью Y, то в нашем случае $\alpha = \pi/2$, тогда формулы принимают вид (формулы 5.3 и 5.4):

$$x = x_0 + \pi D \left(\omega_1 \cdot t_1 \cdot sin\left(\frac{\pi}{2}\right) + \omega_2 \cdot t_2 \cdot sin\left(\frac{\pi}{2} - \frac{2\pi}{3}\right) + \omega_3 \cdot t_3 \cdot sin\left(\frac{\pi}{2} + \frac{2\pi}{3}\right)\right) \quad (5.3)$$

$$y = y_0 + \pi D \left(\omega_1 \cdot t_1 \cdot cos\left(\frac{\pi}{2}\right) + \omega_2 \cdot t_2 \cdot cos\left(\frac{\pi}{2} - \frac{2\pi}{3}\right) + \omega_3 \cdot t_3 \cdot cos\left(\frac{\pi}{2} + \frac{2\pi}{3}\right)\right) \quad (5.4)$$

где x_0 и y_0 – координаты робота на плоскости в начальный момент времени

После упрощения формула принимают вид:

$$x = x_0 + \pi D \left(\omega_1 \cdot t_1 + \omega_2 \cdot t_2 \cdot cos\left(-\frac{2\pi}{3}\right) + \omega_3 \cdot t_3 \cdot cos\left(\frac{2\pi}{3}\right)\right) \quad (5.5)$$

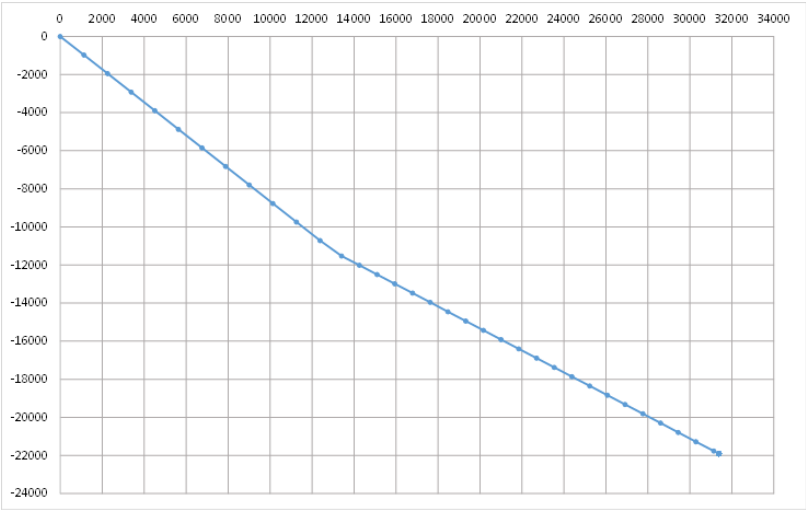
$$y = y_0 + \pi D \left(\omega_1 \cdot t_1 + \omega_2 \cdot t_2 \cdot sin\left(-\frac{2\pi}{3}\right) + \omega_3 \cdot t_3 \cdot sin\left(\frac{2\pi}{3}\right)\right) \quad (5.6)$$

В какой-то момент времени один из моторов останавливается. Следовательно, необходимо выполнить две последовательные итерации вычисления координат:

- в первой итерации вращаются все три колеса
- во второй итерации - только два колеса (т.к. у третьего колеса $\omega = 0$)

После остановки всех моторов мы получаем конечные координаты робота X и Y, которые и выводим в ответ.

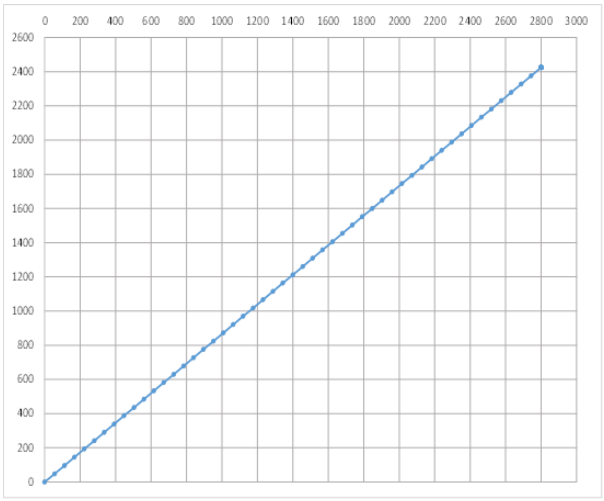
Примеры графиков движений по наборам данных представлены на рис.5.3 и 5.4.



d	w_1	w_2	w_3	t_1	t_2	t_3	⇒	x	y
73	1.542	1.542	-1.542	666	233	666		31391.8	-21909.8

Рис. 5.3: График движения всенаправленной тележки (пример 1)

Пример программы-решения



d	w_1	w_2	w_3	t_1	t_2	t_3	⟹	x	y
40	1.4	-1.4	1.4	100	100	100		2800.0	2424.9

Рис. 5.4: График движения всенаправленной тележки (пример 2)

Ниже представлено решение на языке C++

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

#define PI 3.1415926

class Vector
{
public:
        Vector(double endX, double endY)
        {
                this->x = endX;
                this->y = endY;
        }
        void setX(double endX) { this->x = endX; }
        void setY(double endY) { this->y = endY; }
        double getX() { return x; }
        double getY() { return y; }
        void add(Vector vec)
        {
                x += vec.getX();
                y += vec.getY();
        }
        void subtract(Vector vec)
        {
                x -= vec.getX();
                y -= vec.getY();
        }
        void mult(double num)
        {
                x *= num;
                y *= num;
        }
        Vector operator+(Vector vec)
        {
```

```
35                  Vector newVec(*this);
36                  newVec.add(vec);
37                  return newVec;
38          }
39          Vector operator+=(Vector vec)
40          {
41                  add(vec);
42                  return *this;
43          }
44          Vector operator-(Vector vec)
45          {
46                  Vector newVec(*this);
47                  newVec.subtract(vec);
48                  return newVec;
49          }
50          Vector operator-=(Vector vec)
51          {
52                  subtract(vec);
53                  return *this;
54          }
55          Vector operator*(double num)
56          {
57                  Vector newVec(*this);
58                  newVec.mult(num);
59                  return newVec;
60          }
61          Vector operator*=(double num)
62          {
63                  mult(num);
64                  return *this;
65          }
66          bool operator==(int zero)
67          {
68                  return zero == 0 && x == 0 && y == 0;
69          }
70
71  private:
72          double x;
73          double y;
74  };
75
76  struct Motion
77  {
78  public:
79          Motion() {}
80          Motion(double m1w, double m2w, double m3w, double time)
81          {
82                  motor1w = m1w;
83                  motor2w = m2w;
84                  motor3w = m3w;
85                  this->time = time;
86          }
87
88          double
89                  motor1w,
90                  motor2w,
91                  motor3w;
92          double time;
93          double radius;
94
```

```
95          int getType()
96          {
97                  if (time == 0) return 0;
98                  if (motor1w == motor2w && motor2w == motor3w) return 0;
99                  if (motor1w*motor2w*motor3w == 0) return 1;
100                 else return 2;
101         }
102         Vector getMotionVector()
103         {
104                 int type = getType();
105                 switch (type)
106                 {
107                 case 0: return Vector(0, 0);
108                 case 1:
109                         if (motor1w == 0)
110                         {
111                                 double speed = motor2w * radius;
112                                 Vector sVec(0, -speed * cos(PI / 6));
113                                 return sVec;
114                         }
115                         if (motor2w == 0)
116                         {
117                                 double speed = motor1w * radius;
118                                 Vector sVec(speed * pow(cos(PI / 6), 2), -speed * sin(PI / 6) * cos(P
119                                 return sVec;
120                         }
121                         if (motor3w == 0)
122                         {
123                                 double speed = motor1w * radius;
124                                 Vector sVec(speed * pow(cos(PI / 6), 2), speed * sin(PI / 6) * cos(PI
125                                 return sVec;
126                         }
127                 case 2:
128                         if (motor1w == motor2w)
129                         {
130                                 double speed3 = motor3w * radius;
131                                 return Vector(motor1w*radius, speed3*sin(PI / 3));
132                         }
133                         if (motor1w == motor3w)
134                         {
135                                 double speed2 = motor2w * radius;
136                                 return Vector(motor1w*radius, -speed2 * sin(PI / 3));
137                         }
138                         if (motor2w == motor3w)
139                         {
140                                 double speed1 = motor1w * radius;
141                                 return Vector(speed1, 0);
142                         }
143                 }
144         }
145 };
146
147 using namespace std;
148
149 int main()
150 {
151         double d, w1, t1, w2, t2, w3, t3;
152         cin >> d >> w1 >> t1 >> w2 >> t2 >> w3 >> t3;
153
154         //split motion into simple motions by timespans
```

```
155         Motion m1, m2, m3;
156         m1.radius = m2.radius = m3.radius = d / 2;
157         if (t1 >= t2 && t2 >= t3)
158         {
159                 m1.motor1w = w1; m1.motor2w = w2; m1.motor3w = w3;
160                 m1.time = t3;
161                 m2.motor1w = w1; m2.motor2w = w2; m2.motor3w = 0;
162                 m2.time = t2 - t3;
163                 m3.motor1w = w1; m3.motor2w = 0; m3.motor3w = 0;
164                 m3.time = t1 - t2;
165         }
166         else if (t1 >= t3 && t3 >= t2)
167         {
168                 m1.motor1w = w1; m1.motor2w = w2; m1.motor3w = w3;
169                 m1.time = t2;
170                 m2.motor1w = w1; m2.motor2w = 0; m2.motor3w = w3;
171                 m2.time = t3 - t2;
172                 m3.motor1w = w1; m3.motor2w = 0; m3.motor3w = 0;
173                 m3.time = t1 - t3;
174         }
175         else if (t2 >= t1 && t1 >= t3)
176         {
177                 m1.motor1w = w1; m1.motor2w = w2; m1.motor3w = w3;
178                 m1.time = t3;
179                 m2.motor1w = w1; m2.motor2w = w2; m2.motor3w = 0;
180                 m2.time = t1 - t3;
181                 m3.motor1w = 0; m3.motor2w = w2; m3.motor3w = 0;
182                 m3.time = t2 - t1;
183         }
184         else if (t2 >= t3 && t3 >= t1)
185         {
186                 m1.motor1w = w1; m1.motor2w = w2; m1.motor3w = w3;
187                 m1.time = t1;
188                 m2.motor1w = 0; m2.motor2w = w2; m2.motor3w = w3;
189                 m2.time = t3 - t1;
190                 m3.motor1w = 0; m3.motor2w = w2; m3.motor3w = 0;
191                 m3.time = t2 - t3;
192         }
193         else if (t3 >= t1 && t1 >= t2)
194         {
195                 m1.motor1w = w1; m1.motor2w = w2; m1.motor3w = w3;
196                 m1.time = t2;
197                 m2.motor1w = w1; m2.motor2w = 0; m2.motor3w = w3;
198                 m2.time = t1 - t2;
199                 m3.motor1w = 0; m3.motor2w = 0; m3.motor3w = w3;
200                 m3.time = t3 - t1;
201         }
202         else if (t3 >= t2 && t2 >= t1)
203         {
204                 m1.motor1w = w1; m1.motor2w = w2; m1.motor3w = w3;
205                 m1.time = t1;
206                 m2.motor1w = 0; m2.motor2w = w2; m2.motor3w = w3;
207                 m2.time = t2 - t1;
208                 m3.motor1w = 0; m3.motor2w = 0; m3.motor3w = w3;
209                 m3.time = t3 - t2;
210         }
211
212         //starting motion
213         Vector result(0, 0);
214         result += m1.getMotionVector() * m1.time;
```

```
215         result += m2.getMotionVector() * m2.time;
216         result += m3.getMotionVector() * m3.time;
217 
218         cout << result.getX() << ' ' << result.getY() << endl;
219 }
```

Задача 5.1.2. Управление всенаправленной тележкой (10 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`). С некоторой кинематической моделью робота можно познакомиться по ссылке: `https://bharat-robotics.github.io/blog/kinematic-analysis-of-holonomic-robot/`.

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. 5.5). Расположение моторов является постоянным и соответствует рисунку 5.5.

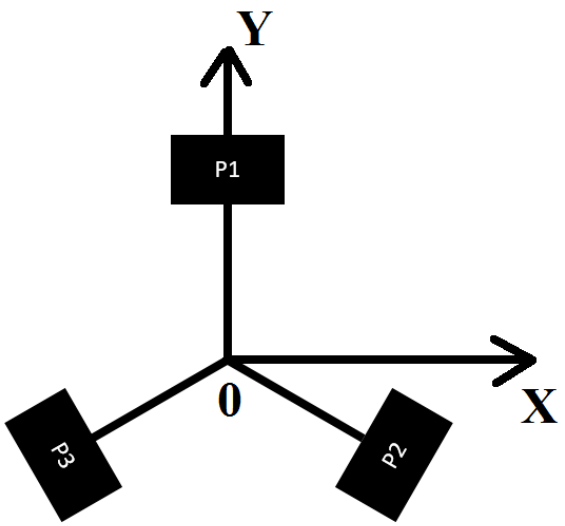


Рис. 5.5: Расположение моторов робота относительно центра глобальной системы отсчета в начальный момент времени

Считать, что мощность достигается мгновенно и может подаваться на все моторы одновременно. Колёса вращаются без проскальзывания. В случае когда на моторы ничего не подаётся, их скорость равна 0.

Формат входных данных

Первая строчка содержит четыре целых числа через пробел – d, p, t, N, где:

- d — диаметр колёс в мм ($30 \leqslant d \leqslant 100$);
- p — длина оси (осевой балки) от центра робота до колеса в мм ($50 \leqslant p \leqslant 125$);
- t — общее время работы моторов в с ($10 \leqslant t \leqslant 1000$);
- N — количество измерений ($1 \leqslant N \leqslant 1000$).

Далее идут 3 строки — для первого, второго и третьего моторов, соотвественно.

В каждой строке находится N вещественных чисел через пробел — подаваемая на мотор скорость w_i, через равные промежутки времени. (-2 рад/с $\leqslant w_i \leqslant 2$ рад/с)

Формат выходных данных

Одна строка, содержащая два целых числа через пробел — координаты центра робота в мм, где он закончил своё движение. Допускается погрешность в 1 мм по каждой из координат.

Примеры

Пример №1

Стандартный ввод

`30 50 12 4`
`0 0 2 2`
`-2 -2 0 0`
`2 2 2 2`

Стандартный вывод
`77.486 167.373`

Пример №2

Стандартный ввод

`35 55 16 8`
`-0.8 -0.8 -0.8 0.6 0.6 0.6 0 0`
`-0.8 -0.8 -0.8 0 0 0 0 0`
`0.8 0.8 0.8 0.6 0.6 0.6 0 0`

Стандартный вывод
`1.504 130.544`

Решение

В данной задаче модель всенаправленной тележки аналогична модели в предыдущей задаче (рис. 5.6) и по условиям задачи тележка движется прямолинейно, т.е. мы можем использовать уравнения кинематики из прошлой задачи.

Рассмотрим отличия от предыдущей задачи:

- Время вращения всех моторов постоянное и одинаковое, t с.
- У моторов (у всех одновременно) происходит изменение угловых скоростей, N раз за все время движения t

Время, используемое в этой формуле, вычисляется делением всего временного интервала, в течение которого двигается робот на количество измерений (формула 5.7).

$$dt = \frac{t}{N} \tag{5.7}$$

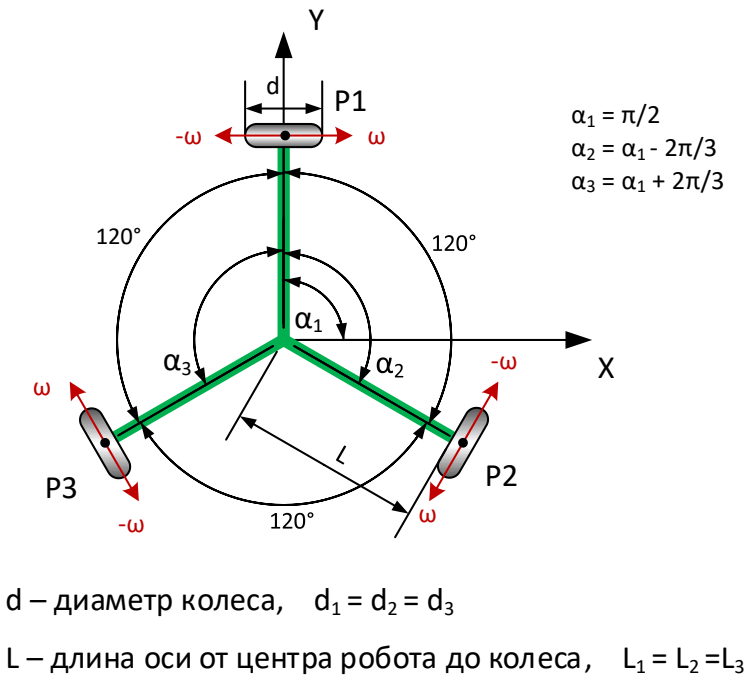


Рис. 5.6: Схема всенаправленной тележки

Таким образом, чтобы получить итоговые координаты центра тележки, нужно итеративно (через время dt) производить вычисление новых координат, подставляя в каждой итерации очередное значение мощности для каждого мотора.

Пример программы-решения

Ниже представлено решение на языке C++

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

#define PI 3.1415926

using namespace std;

class Point
{
public:
        Point(double x, double y) { _x = x; _y = y; }
        Point() { _x = _y = 0; }
        double getX() { return _x; }
        double getY() { return _y; }
        void add(Point p) { _x += p._x; _y += p._y; }
        void subtract(Point p) { _x -= p._x; _y -= p._y; }
        void multiply(double num) { _x *= num; _y *= num; }
        Point operator+(Point p)
        {
```

```
23                  Point newP(*this);
24                  newP.add(p);
25                  return newP;
26          }
27          Point operator-(Point p)
28          {
29                  Point newP(*this);
30                  newP.subtract(p);
31                  return newP;
32          }
33          Point operator*(double num)
34          {
35                  Point newP(*this);
36                  newP.multiply(num);
37                  return newP;
38          }
39          Point operator+=(Point p)
40          {
41                  add(p);
42                  return *this;
43          }
44          Point operator-=(Point p)
45          {
46                  subtract(p);
47                  return *this;
48          }
49          Point operator*=(double num)
50          {
51                  multiply(num);
52                  return *this;
53          }
54          bool operator==(Point p) { return abs(_x - p._x) < 1E-2 && abs(_y - p._y) < 1E-2; }
55  private:
56          double _x, _y;
57  };
58
59  class Vector
60  {
61  public:
62          Vector(Point start, Point end)
63          {
64                  _start = start;
65                  _end = end;
66          }
67          Point getStart() { return _start; }
68          Point getEnd() { return _end; }
69          void add(Vector vec)
70          {
71                  double
72                          offsetX = vec._end.getX() - vec._start.getX(),
73                          offsetY = vec._end.getY() - vec._start.getY();
74                  _end += Point(offsetX, offsetY);
75          }
76          void subtract(Vector vec)
77          {
78                  double
79                          offsetX = vec._end.getX() - vec._start.getX(),
80                          offsetY = vec._end.getY() - vec._start.getY();
81                  _end -= Point(offsetX, offsetY);
82          }
```

```
83          void multiply(double num)
84          {
85                  Point unit = _end - _start;
86                  unit *= num;
87                  _end = _start + unit;
88          }
89          Vector operator+(Vector vec)
90          {
91                  Vector newVec(*this);
92                  newVec.add(vec);
93                  return newVec;
94          }
95          Vector operator+=(Vector vec)
96          {
97                  add(vec);
98                  return *this;
99          }
100         Vector operator-(Vector vec)
101         {
102                 Vector newVec(*this);
103                 newVec.subtract(vec);
104                 return newVec;
105         }
106         Vector operator-=(Vector vec)
107         {
108                 subtract(vec);
109                 return *this;
110         }
111         Vector operator*(double num)
112         {
113                 Vector newVec(*this);
114                 newVec.multiply(num);
115                 return newVec;
116         }
117         Vector operator*=(double num)
118         {
119                 multiply(num);
120                 return *this;
121         }
122         bool operator==(int zero)
123         {
124                 return zero == 0 && _end - _start == Point(0, 0);
125         }
126         operator Point()
127         {
128                 return _end - _start;
129         }
130         double lenght()
131         {
132                 Point p = Point(*this);
133                 return sqrt(pow(p.getX(), 2) + pow(p.getY(), 2));
134         }
135
136 private:
137         Point _start;
138         Point _end;
139 };
140
141 struct Motion
142 {
```

```
143 public:
144         Motion() {}
145         Motion(double m1w, double m2w, double m3w, double time)
146         {
147                 motor1w = m1w;
148                 motor2w = m2w;
149                 motor3w = m3w;
150                 this->time = time;
151         }
152 
153         double
154                 motor1w,
155                 motor2w,
156                 motor3w;
157         double time;
158         double radius;
159 };
160 
161 class Robot
162 {
163 public:
164         Robot(double originDistance, double wheelRadius)
165         {
166                 _wheelRadius = wheelRadius;
167                 _originDistance = originDistance;
168                 _orientation = 0;
169                 _origin = Point(0, 0);
170                 _motor1 = Point(originDistance, 0);
171                 _motor2 = Point(originDistance * cos(PI / 6), -originDistance * sin(PI / 6));
172                 _motor3 = Point(-originDistance * cos(PI / 6), -originDistance * sin(PI / 6));
173         }
174         void startMotion(Motion motion)
175         {
176                 if (_origin == Point(0, 0))
177                 {
178                         double tOrientation = _orientation;
179                         if (motion.motor1w == -motion.motor2w)
180                         {
181                                 rotate(2. / 3 * PI - tOrientation);
182                                 Vector projX(Point(0, 0), Point(motion.motor3w*motion.radius, 0));
183                                 Vector projY(Point(0, 0), Point(0, -motion.motor1w*motion.radius*sin(
184                                 Vector res = (projX + projY)*motion.time;
185                                 _motor1 += (Point)res;
186                                 _motor2 += (Point)res;
187                                 _motor3 += (Point)res;
188                                 _origin += (Point)res;
189                                 rotate(-2. / 3 * PI + tOrientation);
190                                 return;
191                         }
192                         if (motion.motor1w == -motion.motor3w)
193                         {
194                                 rotate(-2. / 3 * PI - tOrientation);
195                                 Vector projX(Point(0, 0), Point(motion.motor2w*motion.radius, 0));
196                                 Vector projY(Point(0, 0), Point(0, motion.motor1w*motion.radius*sin(P
197                                 Vector res = (projX + projY)*motion.time;
198                                 _motor1 += (Point)res;
199                                 _motor2 += (Point)res;
200                                 _motor3 += (Point)res;
201                                 _origin += (Point)res;
202                                 rotate(2. / 3 * PI + tOrientation);
```

```
203                     return;
204             }
205             if (motion.motor2w == -motion.motor3w)
206             {
207                     Vector projX(Point(0, 0), Point(motion.motor1w*motion.radius, 0));
208                     Vector projY(Point(0, 0), Point(0, -motion.motor2w*motion.radius*sin
209                     Vector res = (projX + projY)*motion.time;
210                     _motor1 += (Point)res;
211                     _motor2 += (Point)res;
212                     _motor3 += (Point)res;
213                     _origin += (Point)res;
214                     return;
215             }
216             if (motion.motor1w == motion.motor2w)
217             {
218                     rotate(-2. / 3 * PI - tOrientation);
219                     Vector proj(Point(0, 0), Point(motion.motor2w*motion.radius - motion
220                     if ((proj - Vector(Point(0, 0), Point(-motion.motor3w*motion.radius*c
221                     {
222                             _motor1 += (Point)proj;
223                             _motor2 += (Point)proj;
224                             _motor3 += (Point)proj;
225                             _origin += (Point)proj;
226                             rotate(2. / 3 * PI + tOrientation);
227                             return;
228                     }
229                     Vector speed1(Point(0, 0), Point(-motion.motor3w*motion.radius*cos(PI
230                     double proj1 = proj.lenght();
231                     if (motion.motor2w < 0) proj1 *= -1;
232                     double proj2 = speed1.lenght();
233                     if (motion.motor3w < 0) proj2 *= -1;
234                     double d = (proj2*_originDistance - proj1 * _originDistance) / (proj1
235                     Point center = _motor3 * -(d / _originDistance);
236                     double angle = proj1 * motion.time / (_originDistance - d);
237                     _motor1 -= center;
238                     _motor2 -= center;
239                     _motor3 -= center;
240                     _origin -= center;
241                     rotate(angle);
242                     _motor1 += center;
243                     _motor2 += center;
244                     _motor3 += center;
245                     _origin += center;
246                     rotate(2. / 3 * PI + tOrientation);
247                     return;
248             }
249             if (motion.motor2w == motion.motor3w)
250             {
251                     rotate(2. / 3 * PI - tOrientation);
252                     Vector proj(Point(0, 0), Point(motion.motor3w*motion.radius - motion
253                     if ((proj - Vector(Point(0, 0), Point(-motion.motor1w*motion.radius*c
254                     {
255                             _motor1 += (Point)proj;
256                             _motor2 += (Point)proj;
257                             _motor3 += (Point)proj;
258                             _origin += (Point)proj;
259                             rotate(-2. / 3 * PI + tOrientation);
260                             return;
261                     }
262                     Vector speed1(Point(0, 0), Point(-motion.motor1w*motion.radius*cos(PI
```

```
263                             double proj1 = proj.lenght();
264                             if (motion.motor2w < 0) proj1 *= -1;
265                             double proj2 = speed1.lenght();
266                             if (motion.motor1w < 0) proj2 *= -1;
267                             double d = (proj2*_originDistance - proj1 * _originDistance) / (proj1
268                             Point center = _motor1* -(d / _originDistance);
269                             double angle = proj1 * motion.time / (_originDistance - d);
270                             _motor1 -= center;
271                             _motor2 -= center;
272                             _motor3 -= center;
273                             _origin -= center;
274                             rotate(angle);
275                             _motor1 += center;
276                             _motor2 += center;
277                             _motor3 += center;
278                             _origin += center;
279                             rotate(-2. / 3 * PI + tOrientation);
280                             return;
281                     }
282                     if (motion.motor3w == motion.motor1w)
283                     {
284                             Vector proj(Point(0, 0), Point(motion.motor1w*motion.radius - motion.
285                             if ((proj - Vector(Point(0, 0), Point(-motion.motor2w*motion.radius*c
286                             {
287                                     _motor1 += (Point)proj;
288                                     _motor2 += (Point)proj;
289                                     _motor3 += (Point)proj;
290                                     _origin += (Point)proj;
291                                     return;
292                             }
293                             Vector speed1(Point(0, 0), Point(-motion.motor2w*motion.radius*cos(PI
294                             double proj1 = proj.lenght();
295                             if (motion.motor3w < 0) proj1 *= -1;
296                             double proj2 = speed1.lenght();
297                             if (motion.motor2w < 0) proj2 *= -1;
298                             double d = (proj2*_originDistance - proj1 * _originDistance) / (proj1
299                             Point center = _motor2 * -(d / _originDistance);
300                             double angle = proj1 * motion.time / (_originDistance - d);
301                             _motor1 -= center;
302                             _motor2 -= center;
303                             _motor3 -= center;
304                             _origin -= center;
305                             rotate(angle);
306                             _motor1 += center;
307                             _motor2 += center;
308                             _motor3 += center;
309                             _origin += center;
310                             return;
311                     }
312             }
313             Point tOrigin = _origin;
314             _motor1 -= _origin;
315             _motor2 -= _origin;
316             _motor3 -= _origin;
317             _origin -= tOrigin;
318             startMotion(motion);
319             _motor1 += tOrigin;
320             _motor2 += tOrigin;
321             _motor3 += tOrigin;
322             _origin += tOrigin;
```

```
323         }
324         void rotate(double angle)
325         {
326                 double
327                         cosa = cos(angle),
328                         sina = sin(angle);
329 
330                 _motor1 = Point(_motor1.getX()*cosa + _motor1.getY()*sina, -_motor1.getX()*sina + _mo
331                 _motor2 = Point(_motor2.getX()*cosa + _motor2.getY()*sina, -_motor2.getX()*sina + _mo
332                 _motor3 = Point(_motor3.getX()*cosa + _motor3.getY()*sina, -_motor3.getX()*sina + _mo
333                 _origin = Point(_origin.getX()*cosa + _origin.getY()*sina, -_origin.getX()*sina + _or
334 
335                 _orientation += angle;
336         }
337         Point getOrigin() { return _origin; }
338 
339 private:
340         Point _origin;
341         Point
342                 _motor1,
343                 _motor2,
344                 _motor3;
345         double
346                 _orientation,
347                 _wheelRadius,
348                 _originDistance;
349 };
350 
351 
352 int main()
353 {
354         double d, r, t, n;
355         cin >> d >> r >> t >> n;
356 
357         vector<double> w1(n), w2(n), w3(n);
358         for (int i = 0; i < n; i++) cin >> w1[i];
359         for (int i = 0; i < n; i++) cin >> w2[i];
360         for (int i = 0; i < n; i++) cin >> w3[i];
361 
362         Motion m;
363         m.radius = m.radius = m.radius = d / 2;
364         m.time = t / n;
365 
366         //starting motion
367         Robot robot(r, m.radius);
368         for (int i = 0; i < n; i++)
369         {
370                 m.motor1w = w1[i];
371                 m.motor2w = w2[i];
372                 m.motor3w = w3[i];
373                 robot.startMotion(m);
374         }
375 
376         cout << fixed << setprecision(3) << robot.getOrigin().getX() << ' ' << robot.getOrigin().getY
377 }
```

Задача 5.1.3. Определение размера препятствия (15 баллов)

Робот, собранный по дифференциальной схеме, передвигается по полигону. Он оснащён приёмником и дальномером. Приёмник позволяет измерять расстояния до установленных на поле и находящихся в прямой видимости маяков. Дальномер направлен влево по ходу движения робота и позволяет получать информацию о находящихся препятствиях в данном направлении. На полигоне находятся препятствие и K маяков, координаты которых передаются через входной файл. Гарантируется, что маяки не мешают роботу перемещаться и не попадают в поле зрения дальномера. Пример полигона представлен на рис. 5.7.

Необходимо найти площадь препятствия, расположенного на полигоне. Известно что, приёмник возвращает значения расстояний до препятствия в мм, считая от оси вращения робота, находящейся в центре между колёсами. Дальномер расположен в том же месте, что и приёмник.

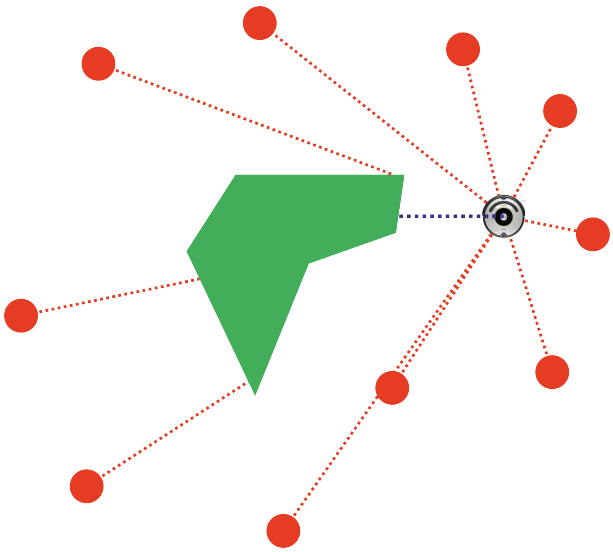


Рис. 5.7: Пример соревновательного полигона

Формат входных данных

Первая строка содержит 3 целых числа: K и N и dT:

- K — количество маяков ($10 \leqslant K \leqslant 100$);
- N — количество замеров ($10 \leqslant N \leqslant 10\ 000$);
- dT — пауза между измерениями в мс ($0 \leqslant dT \leqslant 10\ 000$).

Далее идёт K строк. Каждая строка имеет следующую структуру: i, x_i, y_i, в которой все числа вещественные и разделены пробелами, где:

- i — номер маяка по порядку ($0 \leqslant i \leqslant 10\ 000$);
- x_i — координата x i-того маяка ($-10\ 000 \leqslant x_i \leqslant 10\ 000$);
- y_i — координата y i-того маяка ($-10\ 000 \leqslant y_i \leqslant 10\ 000$).

Далее идёт N строк. Каждая строка содержит:

- d — целое число, показание дальномера в мм ($0 \leqslant d \leqslant 1\ 000$), в случае если предмет находится вне поля видимости дальномера, то его значение будет равно 1 000;

- $Value_1\ Value_2\ \ldots\ Value_k$ — вещественные числа, показания, получаемые приёмником с каждого маяка. В случае, если маяк находится вне зоны видимости, то соответствующее значение будет равно -1.

Все числа указаны через пробел.

Формат выходных данных

Одна строка, содержащая одно целове число – площадь препятствия в мм^2. Допускается погрешность в ± 1 м^2.

Примеры

Примеры входных данных и ответов к ним можно найти по `http://bit.ly/2Re5l9N`данной сслыке.

Решение

Для определения границ препятствия сначала определим координаты робота в момент замеров. Для этого мы найдем точки пересечения двух окружностей, образованных от центров маяков, которые "видят"робота в данный момент времени. Схема представлена на рис.5.8, где:

- M_1 - маяк N, установленный в координатах $(a_1; b_1)$
- M_2 - маяк K, установленный в координатах $(a_2; b_2)$
- r_1 - радиус первой окружности (расстояние от маяка N до центра робота)
- r_2 - радиус второй окружности (расстояние от маяка K до центра робота)

Сначала найдем расстояние между центрами маяков: d $(M_1;M_2)$ (можно включить проверку отсутствие решений: $d = |M_1 - M_2|$, при $d > r_1 + r_2$ (круги не пересекаются) или $d < |r_2 - r_1|$ (одна окружность находится внутри другой)). На схеме (рис.5.8) видно, что r_1 и r_2 - катеты $\triangle M_1M_2K_1$, а искомая сторона M_1M_2 - гипотенуза. Исходя из теоремы Пифагора, найдем гипотенузу по формуле (5.8).

$$d = \sqrt{(a_2 - a_1)^2 + (b_2 - b_1)^2} \tag{5.8}$$

Далее нам надо найти r_2, $M_1S = d_1$; $M_2S = d_2$.

$$\begin{cases} d_1 + d_2 = d \\ h = \sqrt{{r_1}^2 - {d_1}^2} = \sqrt{{r_2}^2 - {d_2}^2} \end{cases} \Rightarrow \begin{cases} d_1 + d_2 = d \\ {r_1}^2 - {d_1}^2 = {r_2}^2 - {d_2}^2 \end{cases} \Rightarrow$$

$$\begin{cases} d_1 + d_2 = d \\ {d_2}^2 - {d_1}^2 = {r_2}^2 - {r_1}^2 \end{cases} \Rightarrow \begin{cases} d_1 + d_2 = d \\ (d_2 - d_1)(d_2 + d_1) = {r_2}^2 - {r_1}^2 \end{cases} \Rightarrow$$

$$(d_2 - d_1) = \frac{{r_2}^2 - {r_1}^2}{d} \Rightarrow d_2 = \frac{{r_2}^2 - {r_1}^2}{d} + d_1 = \frac{{r_2}^2 - {r_1}^2}{d} + d - d_2 \Rightarrow$$

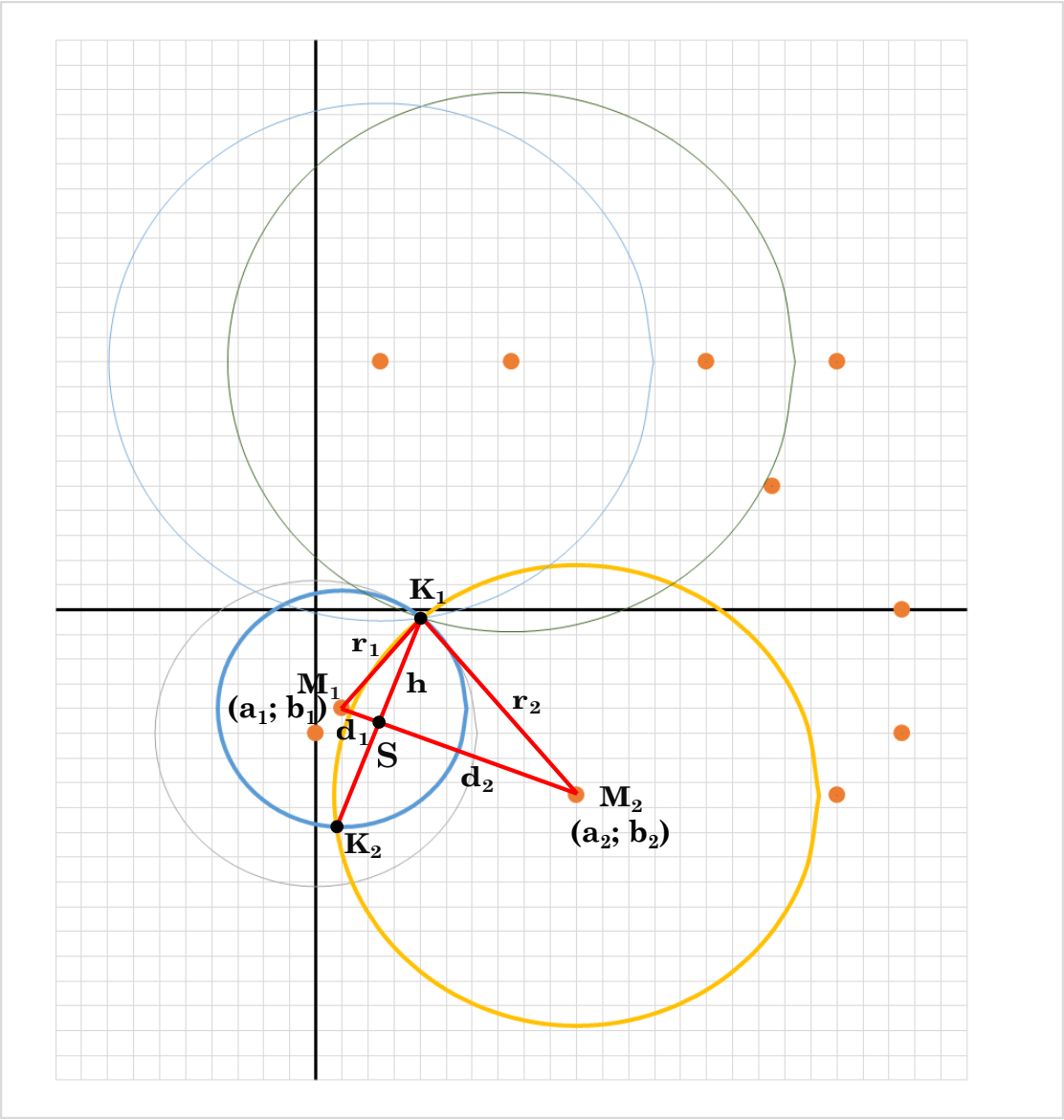


Рис. 5.8:

$$2 \cdot d_2 = \frac{{r_2}^2 - {r_1}^2}{d} + d \qquad \Rightarrow \qquad d_2 = \frac{{r_2}^2 - {r_1}^2}{2 \cdot d} + \frac{d}{2} \tag{5.9}$$

Теперь найдем катет d_2 в треугольнике K_1M_2S по формуле (5.9).

Зная расстояния d и d_2, находим d_1 по формуле (5.10):

$$d_1 = d - d_2 \tag{5.10}$$

Зная катет d_1 и гипотенузу r_1 найдем катет h в $\triangle M_1K_1S$ по формуле (5.11):

$$h = \sqrt{{r_1}^2 - {d_1}^2} \tag{5.11}$$

Далее находим координаты точки S через формулу деления отрезка в данном

отношении (через координаты M_1 и M_2 и отношения k):

$$k = \frac{d_1}{d_2}$$

$$x_s = \frac{a_1 + k \cdot a_2}{1 + k} \tag{5.12}$$

$$y_s = \frac{b_1 + k \cdot b_2}{1 + k} \tag{5.13}$$

Далее найдем координаты обоих точек пересечения окружностей по формулам (5.14, 5.15, 5.16, 5.17):

$$x_1 = \frac{h}{d} \cdot (b_1 - b_2) + x_s \tag{5.14}$$

$$y_1 = \frac{h}{d} \cdot (a_2 - a_1) + y_s \tag{5.15}$$

$$x_2 = -\frac{h}{d} \cdot (b_1 - b_2) + x_s \tag{5.16}$$

$$y_2 = -\frac{h}{d} \cdot (a_2 - a_1) + y_s \tag{5.17}$$

Для выбора координат из полученных (x_1, y_1) и (x_2, y_2) проверим значения по координатам 3 маяка (на выбор).

Для вычисления координат точки на препятствии, видимую датчиком расстояния, сначала найдем угол α робота по формуле (5.18):

$$\alpha_{robot} = atan2(x_i - x_{i-1}; y_i - y_{i-1}) \tag{5.18}$$

после чего найдем угол дальномера β (с учетом того, что дальномер направлен влево по ходу движения робота) по формуле (5.19):

$$\beta = \alpha_{robot} + \frac{\pi}{2} \tag{5.19}$$

координаты точки на препятствии, вычисляемые с показаний дальномера, находим по формулам (5.20, 5.21):

$$x_o = x_{i\,robot} + d \cdot cos(\beta) \tag{5.20}$$

$$y_o = y_{i\,robot} + d \cdot sin(\beta) \tag{5.21}$$

где d - показания с дальномера до препятствия.

Результаты вычислений показаны на рис. 5.9

По полученным координатам точек многоугольника находим площадь препятствия по формуле площади Гаусса (5.22):

$$\begin{aligned} A &= \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right| \\ &= \frac{1}{2}|x_1 y_2 + x_2 y_3 + \cdots + x_{n-1} y_n + x_n y_1 - x_2 y_1 - x_3 y_2 - \cdots - x_n y_{n-1} - x_1 y_n| \end{aligned} \tag{5.22}$$

где,

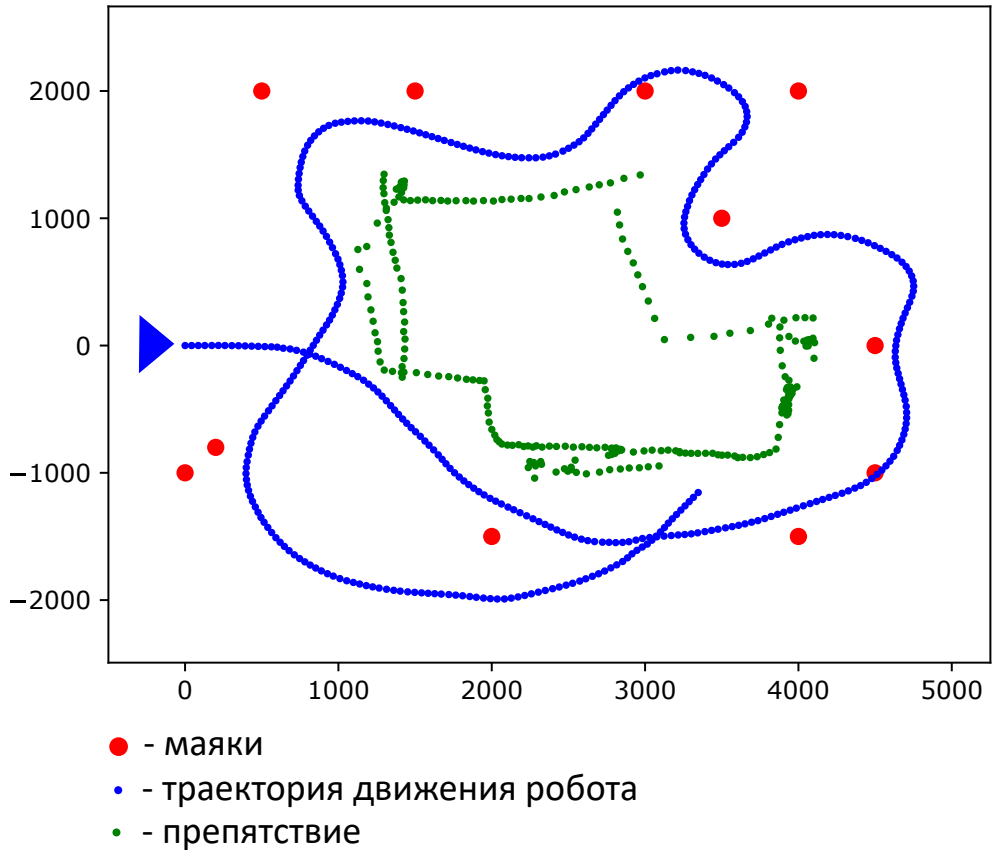


Рис. 5.9: Траектория движения робота и границы препятствия

- A - площадь многоугольника
- n - количество сторон многоугольника
- (x_i, y_i) при $i = 1, 2, \ldots n$ – координаты вершин многоугольника

В ответ выводим полученную площадь препятствия.

Пример программы-решения

Ниже представлено решение на языке Python3

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


def gauss(polygon):
    # polygon в формате [(x1,y1),...(xi,yi)] или [[x1,y1], ... [xi,yi]]
    polygon.append(polygon[0])
    S = 0
    for i in range(len(polygon) - 1):
        S += polygon[i][0] * polygon[i + 1][1]       # Xi * Yi+1
        S -= polygon[i][1] * polygon[i + 1][0]       # Yi * Xi+1
    S = abs(S) * 0.5
    return int(S)

```

```
16
17  def circles(lst):
18      a1, b1, r1, a2, b2, r2, a3, b3, r3 = lst
19      # расстояние между центрами маяков M1 (a1, b1), M2 (a2, b2)
20      d = math.sqrt((a2 - a1) ** 2 + (b2 - b1) ** 2)
21
22      if d >= r1 + r2 or d < abs(r1-r2):
23          return False
24
25      d2 = (r2 ** 2 - r1 ** 2) / (2 * d) + (d / 2)
26      d1 = d - d2
27
28      h = math.sqrt(r1 ** 2 - d1 ** 2)
29      k = d1 / d2
30
31      xs = (a1 + k * a2) / (1 + k)
32      ys = (b1 + k * b2) / (1 + k)
33
34      x1 = (h / d) * (b1 - b2) + xs
35      y1 = (h / d) * (a2 - a1) + ys
36
37      x2 = -(h / d) * (b1 - b2) + xs
38      y2 = -(h / d) * (a2 - a1) + ys
39
40      test1 = int(math.sqrt((x1 - a3) ** 2 + (y1 - b3) ** 2))
41      test2 = int(math.sqrt((x2 - a3) ** 2 + (y2 - b3) ** 2))
42
43      if abs(test1 - r3) <= abs(test2 - r3):
44          return (x1, y1)
45      else:
46          return (x2, y2)
47
48
49  data = sys.stdin.readlines()
50
51  K, N, dT = list(map(int, data[0].strip().split(' ')))
52  data.pop(0)
53  """
54  K  - кол-во маяков
55  N  - кол-во замеров
56  dT - пауза между замерами, мс
57  """
58
59  beacons, measures = [], []
60
61  for i in range(K):
62      beacons.append(list(map(float, data[i].strip().split(' '))))
63
64  for i in range(N):
65      measures.append(list(map(float, data[i + K].strip().split(' '))))
66
67  xy_obstacle, xy_robot = [], []
68  alpha = 0
69
70  for i in range(N):
71      b_idx = []
72      for b in range(len(beacons)):
73          if measures[i][b + 1] >= 0:
74              b_idx.append(b)
75          if len(b_idx) == 3:
```

```
76              break
77
78      b_info = []
79      for  idx in b_idx:
80          b_info.append(beacons[idx][1])
81          b_info.append(beacons[idx][2])
82          b_info.append(measures[i][idx + 1])
83
84      xy = circles(b_info)
85      xy_robot.append(xy)
86
87      if (measures[i][0] < 1000) and len(xy_robot) > 1:               # замеры с дальномера
88          dY = xy_robot[-1][1] - xy_robot[-2][1]
89          dX = xy_robot[-1][0] - xy_robot[-2][0]
90          alpha = (math.atan2(dY, dX))
91
92          betta = alpha + math.radians(90)
93          xy = [xy_robot[-1][0] + measures[i][0] * math.cos(betta), xy_robot[-1][1] + measures[i][0] *
94
95          xy_obstacle.append(xy)
96
97  print(gauss(xy_obstacle))
```

Задача 5.1.4. Определение цветных цилиндров (15 баллов)

Робот, собранный по дифференциальный схеме и оснащенный дальномером, направленным прямо по ходу движения, перемещается по робототехническому полигону. На данном полигоне установлены цилиндры различного цвета. Робот также является цилиндром.

В процессе своего передвижения по полигону (см рис. 5.10) робототехническое устройство измерило расстояние до возникающих прямо препятствий и их цвет в шестнадцатиричном формате вида $RRGGBB$, где RR - 16тиричное число R составляющей данного элемента матрицы, GG и BB - 16тиричные числа G и B составляющих, соответственно.

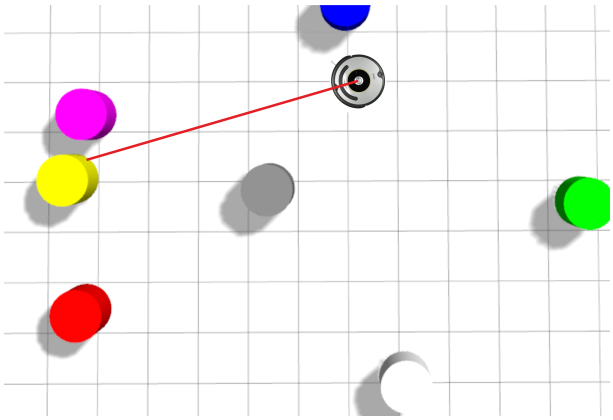


Рис. 5.10: Пример соревновательного полигона

Необходимо определить между какими цилиндрами робот не сможет проехать, если известен маршрут робота и показания датчика в процессе движения. Размер цилиндров и робота: 20 см в диаметре. Дальномер возвращает расстояние до цилиндра, считая от оси вращения робота, находящейся в центре между колёсами. Также гарантируется, что робот измерил каждый цилиндр не менее трех раз, и разница

между этими измерениями составляет минимум 5 дуговых градусов (при измерении по дуге цилиндра).

Формат входных данных

Первая строка содержит 2 целых числа: N и dT:

- N — количество замеров ($10 \leqslant N \leqslant 10000$);
- dT — пауза между измерениями в мс ($0 \leqslant dT \leqslant 10000$).

Далее идёт N строк. Каждая строка имеет следующую структуру: Vel_{linear}, $Vel_{angular}$, $Distance$, $Color$, в которой все числа разделены пробелами, где:

- Vel_{linear} — линейная скорость в см/с, с которой движется робот в данный момент ($-100 \leqslant Vel_{linear} \leqslant 100$);
- $Vel_{angular}$ — угловая скорость в рад/мс, с которой движется робот в данный момент ($-10 \leqslant Vel_{angular} \leqslant 10$);
- $Distance$ — показания датчика расстояния в см в диапазоне “5–255” см, если предмет вне зоны видимости, то будет выведено 255;
- $Color$ — цвет обнаруженного циллиндра в шестнадцатиричном формате вида $RRGGBB$. В случае если цилиндр не обнаружен, будет выведено “000000”.

Значения скоростей и показания датчика являются вещественными числами. Моторы достигают скоростей мгновенно. Робот движется без проскальзывания. Значения цвета — целые числа.

Формат выходных данных

Одна строка, в которой указана пара цветов цилиндров в шестнадцатиричной записи через пробел, в порядке возрастания чисел. В случае если таких пар несколько, то каждую пару следует выводить в отдельной строке. Несколько строк следует выводить в порядке возрастания первых чисел, а в случае их равенства, в порядке возрастания вторых.

Примеры

Примеры входных данных и ответов к ним можно найти по `http://bit.ly/2UoC22Z`данной сслыке.

Решение

Рассмотрим движение робота между моментами изменения скоростей. В этот момент линейная и угловая скорости постоянны и мы можем рассматривать движение робота, как показано на рис. 5.11.

По условиям задачи нам известны угловая (ω) и линейная (v) скорости робота, а также время между замерами (t).
Угол поворота (α) вычисляем по формуле (5.23).
В начале движения робот находится в координатах ($x_0 = 0; y_0 = 0$) и направлен вдоль оси X, т.е. угол $\alpha_0 = 0$.

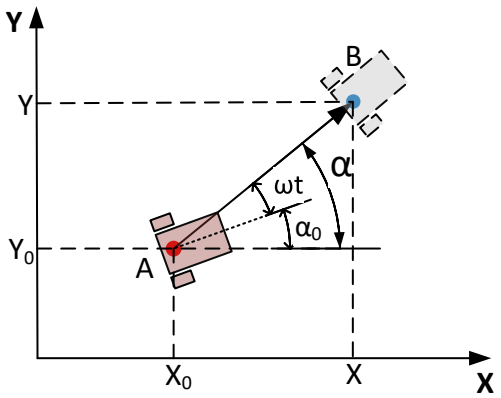


Рис. 5.11: Движение между измерениями скоростей

$$\alpha = \alpha_0 + w \cdot t \tag{5.23}$$

Длина пройденного пути (отрезок AB) находится по формуле (5.24):

$$l = v \cdot t \tag{5.24}$$

Координаты робота в точке B находятся по формулам (5.25) и (5.26):

$$x_i = x_0 + l \cdot cos(\alpha) \tag{5.25}$$
$$y_i = y_0 + l \cdot sin(\alpha) \tag{5.26}$$

После обработки всех данных получаем путь перемещения робота (рис. 5.12).

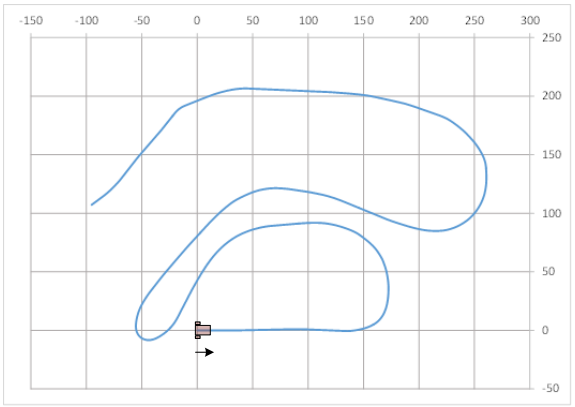


Рис. 5.12: Рассчитанный путь перемещения робота

Зная координаты робота и получаемое расстояние от дальномера (d) мы можем рассчитать координаты точек на окружностях цилиндров (с учетом того, что дальномер направлен прямо по ходу движения робота, см. рис. 5.13) по формулам (5.27) и (5.28):

$$x'_c = x_{robot} + cos(\alpha) \cdot d \tag{5.27}$$

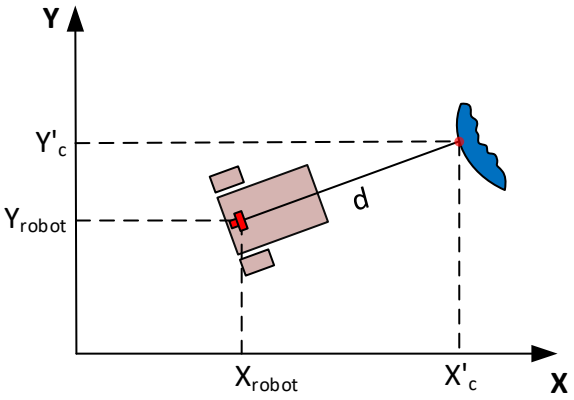


Рис. 5.13:

$$y'_c = y_{robot} + sin(\alpha) \cdot d \tag{5.28}$$

Координаты точек относятся к цилиндрам определенного цвета. Результат показан на рис. 5.14.

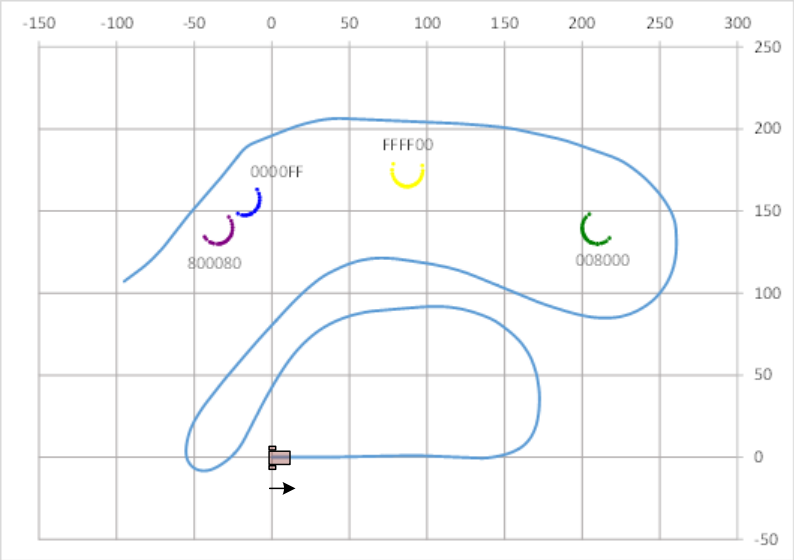


Рис. 5.14: Наборы точек, распределенные по цветам цилиндров

Сейчас нам известны координаты всех точек на окружности каждого цилиндра - найдем координаты центра каждого цилиндра. Для этого нам надо выбрать 3 точки из всего множества для каждого цилиндра. Например, две точки - самые крайние с обеих сторон, третья точка - посередине между ними (рис. 5.15). В примере на рис. 5.15 указан набор точек желтого цвета ($FFFF00$) (из рис. 5.14).

Уравнение прямой, проходящей через две точки, имеет вид:

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

В этом случае угловой коэффициент k определятся по формуле:

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

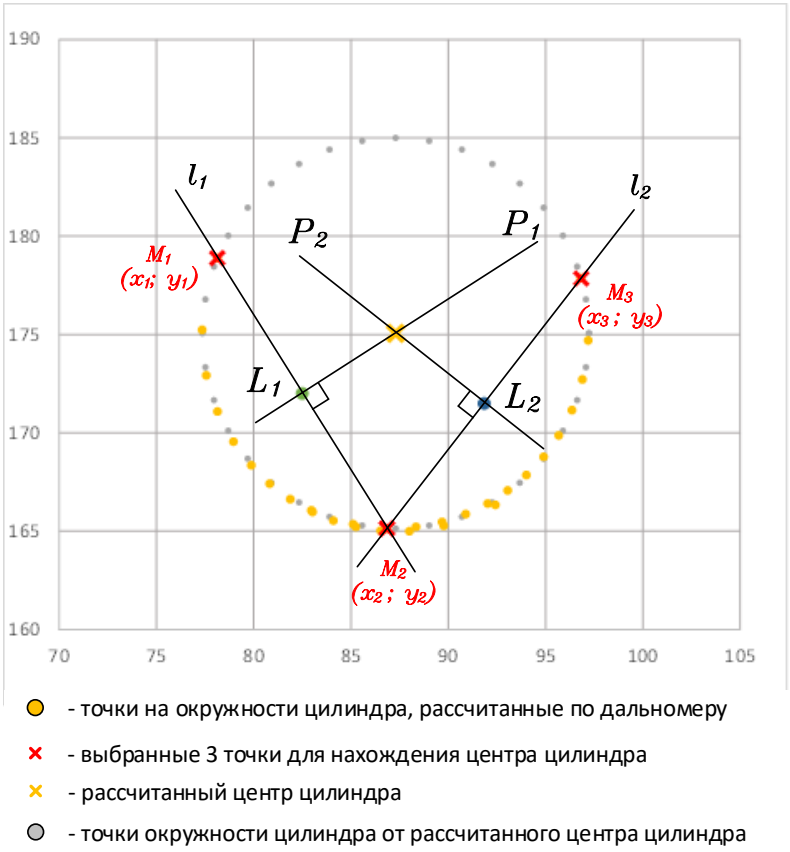


Рис. 5.15: Схема нахождения центра цилиндра по трем точкам

Найдем коэффициенты k и b для прямых l_1 и l_2 по формулам (5.29) и (5.29), (5.29) и (5.29):

$$\begin{aligned} k_{l_1} &= \frac{y_2 - y_1}{x_2 - x_1} \\ b_{l_1} &= y_2 - k_{l_1} \cdot x_2 \\ k_{l_2} &= \frac{y_3 - y_2}{x_3 - x_2} \\ b_{l_2} &= y_3 - k_{l_2} \cdot x_3 \end{aligned} \tag{5.29}$$

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

Найдем середины отрезков M_1M_2 и M_2M_3 по формулам (5.30) и (5.30).

$$\begin{aligned} L_1 &= \left(\frac{x_1 + x_2}{2}; \frac{y_1 + y_2}{2} \right) \\ L_2 &= \left(\frac{x_2 + x_3}{2}; \frac{y_2 + y_3}{2} \right) \end{aligned} \tag{5.30}$$

Прямая, перпендикулярная к линии с коэффициентом наклона k имеет коэффициент наклона: $-1/k$, значит уравнения прямых P_1 и P_2, перпендикулярных l_1 и l_2,

соответственно, запишем следующим образом: для P_1 – по формулам (5.31) и (5.31), для P_2 - по формулам (5.31) и (5.31):

$$\begin{aligned} k_{P_1} &= -\frac{1}{k_{l_1}} \quad \Leftrightarrow \quad k_{P_1} = -\frac{x_2 - x_1}{y_2 - y_1} = \frac{x_1 - x_2}{y_2 - y_1} \\ b_{P_1} &= y_{L_1} - k_{P_1} \cdot x_{L_1} \\ k_{P_2} &= -\frac{1}{k_{l_2}} \quad \Leftrightarrow \quad k_{P_2} = -\frac{x_3 - x_2}{y_3 - y_2} = \frac{x_2 - x_3}{y_3 - y_2} \\ b_{P_2} &= y_{L_2} - k_{P_2} \cdot x_{L_2} \end{aligned} \tag{5.31}$$

Сейчас мы уже можем найти координаты точку центра цилиндра (или точку пересечения прямых P_1 и P_2) по формулам (5.32) и (5.32):

$$\begin{aligned} x &= \frac{b_{P_1} - b_{P_2}}{k_{P_2} - k_{P_1}} \\ y &= k_{P_1} \cdot x + b_{P_1} \end{aligned} \tag{5.32}$$

Результат расчета центров цилиндров показан на рис. (5.16):

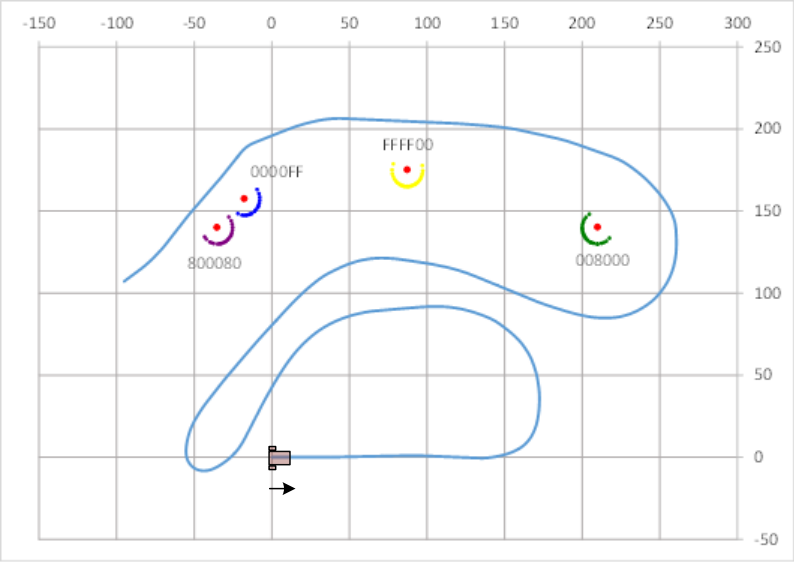


Рис. 5.16: Центры цилиндров

Далее проверяем попарно все цилиндры (каждый с каждым) на возможность проезда между ними робота. Для этого мы должны узнать расстояние между краями цилиндров по формуле (5.33). Для примера возьмем цилиндры с цветами 800080 и $0000FF$.

$$L = d - r_1 - r_2 \tag{5.33}$$

где d - расстояние между центрами цилиндров; r_1, r_2 - радиусы цилиндров. В нашем примере $d = 24.77$ см; $r_1 = 9.94$ см; $r_2 = 9.98$ см.

Если полученное расстояние L больше или равно диаметра робота ($L \geqslant D_{robot}$), значит робот сможет проехать между цилиндрами, иначе – нет.

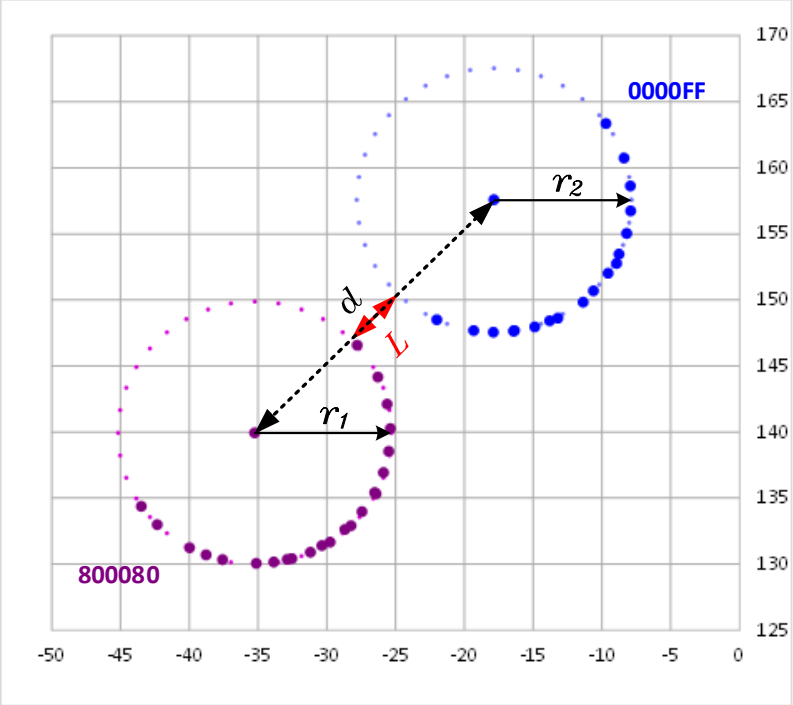


Рис. 5.17:

В примере на рис. (5.17) расстояние между краями цилиндров $L = 4.85$ см, т.е. робот между этими цилиндрами не проедет, т.к. диаметр робота 20 см.

В ответ выводим пару цветов цилиндров, между которыми робот сможет проехать. Таких пар может быть несколько.

Ответ: 0000FF 800080

Пример программы-решения

Ниже представлено решение на языке Python3

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


def line_len(x1, y1, x2, y2):
    return math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)


def extreme_points(lst):
    max_len = 0
    xy = tuple()
    for i in range(len(lst)):
        x1, y1 = lst[i]
        for j in range(len(lst)):
            if line_len(x1, y1, *lst[j]) > max_len:
                max_len = line_len(x1, y1, *lst[j])
                xy = (x1, y1, *lst[j])
    return xy
```

```
20
21
22  data = sys.stdin.readlines()
23
24  N, dT = map(int, data[0].strip().split(' '))
25  """
26  N  - кол-во замеров
27  dT - пауза между замерами, мс
28  """
29  diameter = 20                                          # robots and cylinders, in cm
30  cylinder_colors = []
31  cylinder_dist_by_colors = {}
32
33  robot_xy = [[0, 0]]
34  alpha, robot_distance = 0, 0
35
36  for i in range(N):
37      measures = data[i+1].strip().split(' ')            # v, w, dist_to_cylinder, color
38      measures[:3] = list(map(float, measures[:3]))      # v, w, dist_to_cylinder -> float()
39      measures[3] = measures[3].rjust(6, '0')
40      v, w, dist_to_cyl, color = measures
41
42      alpha += w * dT
43      robot_distance = v * dT
44
45      r_x = robot_xy[-1][0] + robot_distance * math.cos(alpha)
46      r_y = robot_xy[-1][1] + robot_distance * math.sin(alpha)
47
48      robot_xy.append((r_x, r_y))
49
50      if dist_to_cyl < 255 and color != '000000':
51          cyl_x = robot_xy[-1][0] + dist_to_cyl * math.cos(alpha)
52          cyl_y = robot_xy[-1][1] + dist_to_cyl * math.sin(alpha)
53
54          if color not in cylinder_dist_by_colors:
55              cylinder_dist_by_colors[color] = [(cyl_x, cyl_y)]
56          else:
57              cylinder_dist_by_colors[color].append((cyl_x, cyl_y))
58
59  unique_dist = {}
60  for key in cylinder_dist_by_colors:
61      xy = []
62      unique_dist[key] = []
63      for x, y in cylinder_dist_by_colors[key]:
64          str_xy = str(round(x, 4)) + ',' + str(round(y, 4))
65          if str_xy not in xy:
66              xy.append(str_xy)
67              unique_dist[key].append([x, y])
68
69  cylinder_centers = []
70  for color in unique_dist:
71      x1, y1, x2, y2 = extreme_points(cylinder_dist_by_colors[color])
72      x3, y3 = cylinder_dist_by_colors[color][8]
73
74      L1 = [(x1 + x2) / 2, (y1 + y2) / 2]
75      L2 = [(x2 + x3) / 2, (y2 + y3) / 2]
76
77      # P1
78      k1 = (x1 - x2) / (y2 - y1)
79      b1 = L1[1] - k1 * L1[0]
```

```
80
81      # P2
82      k2 = (x2 - x3) / (y3 - y2)
83      b2 = L2[1] - k2 * L2[0]
84
85      x = (b1 - b2) / (k2 - k1)
86      y = k1 * x + b1
87
88      cylinder_centers.append([x, y, color])
89
90  results = []
91  for i in range(len(cylinder_centers)):
92      for j in range(1, len(cylinder_centers)):
93                  if i != j and i < j:
94                      x1, y1 = cylinder_centers[i][0:2]
95                      clr1 = cylinder_centers[i][2]
96                      x2, y2 = cylinder_centers[j][0:2]
97                      clr2 = cylinder_centers[j][2]
98
99                      dist = math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2)
100                     if dist - diameter < diameter:
101                         results.append(sorted([clr1, clr2]))
102
103 for i in results:
104     print(*i)
```

Задача 5.1.5. Определения расстояния до препятствия (5 баллов)

Камера статично установлена на высоте h мм и отклонена на α° от горизонта. Она имеет разрешение $height \times width$ и угол обзора β°. Поясняющая картинка представлена на рисунке 5.18.

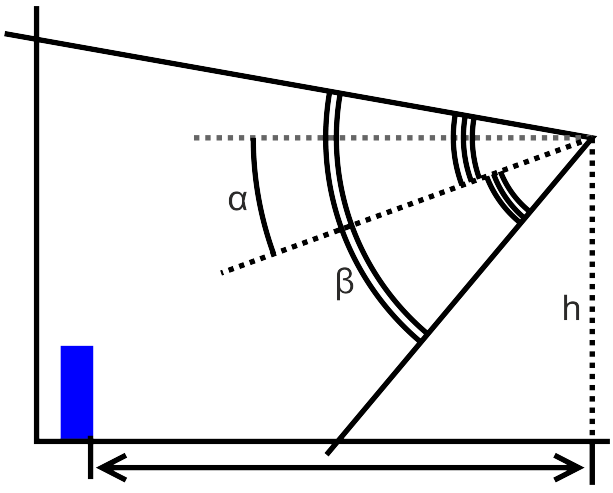


Рис. 5.18: Схема установки камеры

Необходимо найти расстояние до предмета, установленного на полигоне. Гарантируется, что предмет находится в поле зрения камеры, является однотонным. Его минимальная площадь на изображении равна $s\%$. Также известно, что помимо требуемого предмета в камеру попадает поверхность полигона и(или) борта. Поверхность полигона не однородная, а содержащая различные цвета в хаотическом порядке.

Пример изображения представлен на рис. 5.19,

Рис. 5.19: Пример изображения с камеры

Формат входных данных

Первая строка входного файла содержит 6 чисел: h, α, β, $height$, $width$, s:

- h — высота установки камеры в мм ($10 \leqslant h \leqslant 100$);
- α — угол в градусах, под которым расположена камера относительно горизонта ($0° \leqslant \alpha \leqslant 45°$);
- β — угол обзора камеры в градусах ($30° \leqslant \alpha \leqslant 180°$);
- $height$ — разрешение изображения по высоте ($10 \leqslant height \leqslant 2000$);
- $width$ — разрешение изображения по ширине ($10 \leqslant width \leqslant 2000$);
- s — минимальная площадь предмета в % от всего изображения, которое измеряется в пикселях ($1 \leqslant s \leqslant 100$).

На следующих $height$ строках расположено изображение размером $height \times width$ в виде шестнадцатиричных чисел. Данные числа имеют следующий вид: $RRGGBB$, где RR - 16тиричное число R составляющей данного элемента матрицы, GG и BB - 16тиричные числа G и B составляющих, соответственно.

Все числа являются целыми.

Формат выходных данных

Одна строка, в которой указано число — расстояние до предмета в мм. Допускается погрешность в 10 мм.

Примеры

Примеры входных данных и ответов к ним можно найти по `http://bit.ly/2FCbgAU`данной сслыке.

Решение

Составим схему по данным из условий задачи (рис. 5.20):

1. Высота установки камеры, h

2. Угол отклонения камеры от линии горизонта, α
3. Угол обзора камеры, β
4. Разрешение камеры по высоте, $height$

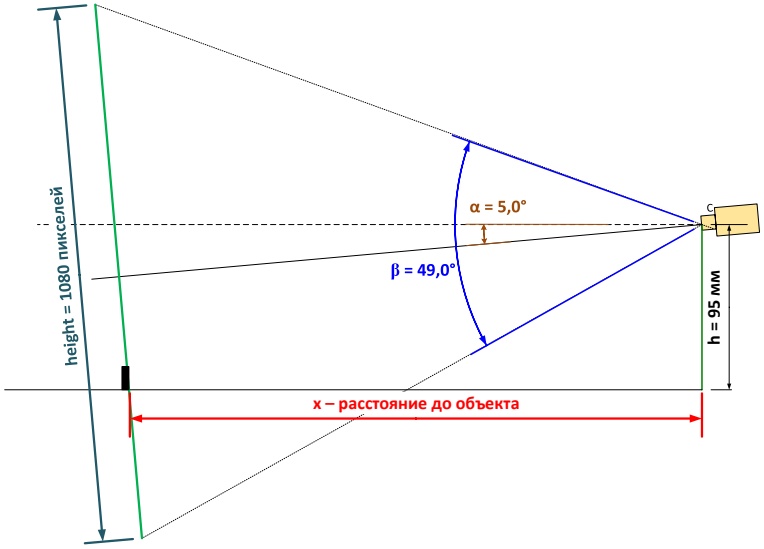


Рис. 5.20:

Теперь вполне очевидно, что искомое значение x - это катет b прямоугольного треугольника $\triangle ABC$ (рис. 5.21)

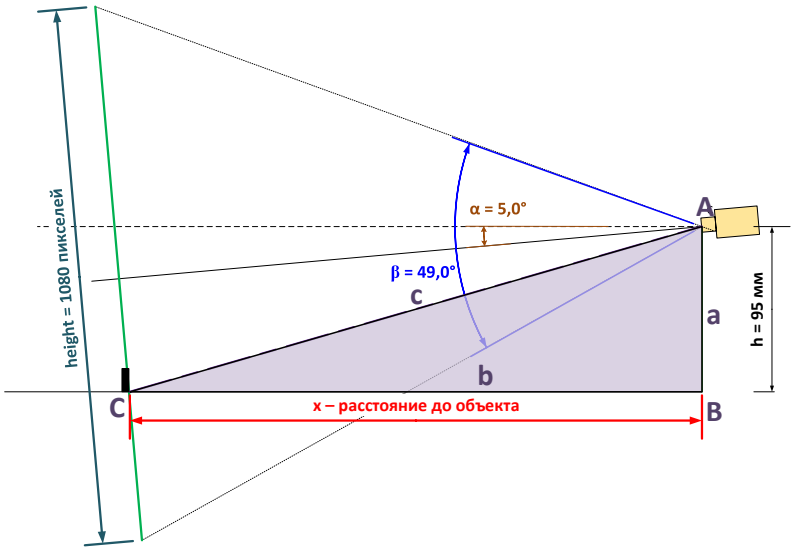


Рис. 5.21:

Как вариант, найти длину катета b можно через $tg(\angle CAB)$, тогда пусть $\angle CAB = \omega$.
Для нахождения угла ω нам надо найти углы θ и γ.
Для нахождения угла θ нам надо знать расстояние x_1. В нашем случае, расстояние x_1 - это высота в пикселях от нижней границы изображения до нижней границы предмета на изображении. см. рис.5.22.

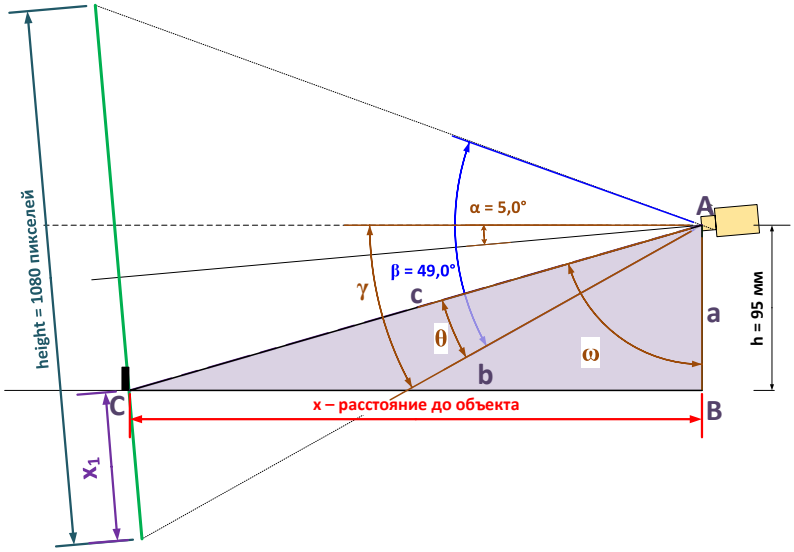


Рис. 5.22:

Нахождение расстояния x_1: По условиям задачи известно, что искомый объект на изображении - "однотонный". Алгоритм поиска однотонного изображения заключается в поиске близлежащих пикселей одинакового цвета. Для этого используется построчное сканирование изображения с применением алгоритма BFS к каждому непросмотренному пикселю. При этом надо вести учет уже просмотренных ранее пикселей через BFS. Начальный пиксель находится в левом верхнем углу.

После обработки всего изображения сравниваем площади полученных объектов и находим объект с площадью, максимально приближенной к задаваемой в условии задачи.

Далее у этого объекта берем значения пикселя с ближайшими координатами к нижней границе изображения, рис. 5.23.

Рис. 5.23:

Находим расстояние x_1:

$x_1 = height$ – максимальная координата Y объекта $= 1080 - 720 = 320$ пикселей

Угол θ найдем через пропорцию:

$$\frac{\beta}{\theta} = \frac{height}{x_1} \qquad \Rightarrow \qquad \theta = \frac{x_1 \cdot \beta}{height} \qquad \Rightarrow \qquad \theta = \frac{320 \cdot 49}{1080} = 14.52^\circ$$

Угол γ в нашем случае - это сумма угла α и половины угла β:

$$\gamma = \alpha + \frac{\beta}{2} \qquad \Rightarrow \qquad \gamma = 5 + \frac{49}{2} = 29.5^\circ$$

И, наконец, найдем угол ω:

$$\omega = 90 - \gamma + \theta \qquad \Rightarrow \qquad \omega = 90 - 29.5 + 14.52 = 75.02^\circ$$

Теперь найдем X, но предварительно переведем градусы в радианы:

$$\omega' = \frac{\omega \cdot \pi}{180} \qquad \Rightarrow \qquad \omega' = \frac{95 \cdot \pi}{180} = 1,309346005 \text{ радиан}$$

$$x = h \cdot tg(\omega') \qquad \Rightarrow \qquad x = 95 \cdot tg(1,309346005) = 355 \text{ мм}$$

Ответ: расстояние от камеры до объекта равно 355 мм

Пример программы-решения

Ниже представлено решение на языке Python3

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


def find_obj2(pic):
    row_max = len(pic)
    col_max = len(pic[0])
    visited = [[False for j in range(col_max)] for i in range(row_max)]

    colors = {}
    for row in range(row_max - 1, 0, -1):
        for col in range(col_max):
            if not visited[row][col]:
                clr = pic[row][col]
                key = clr + ',' + str(row) + ',' + str(col)
                colors[key] = [(row, col)]
                queue = [(row, col)]
                while len(queue) > 0:
                    r, c = queue.pop(0)  # row, col
                    if not visited[r][c]:
                        visited[r][c] = True
                        colors[key].append((r, c))

```

```
25                          if r + 1 < row_max:
26                              if pic[r + 1][c] == clr and not visited[r + 1][c]:
27                                  queue.append((r + 1, c))
28                          if r - 1 >= 0:
29                              if pic[r - 1][c] == clr and not visited[r - 1][c]:
30                                  queue.append((r - 1, c))
31                          if c + 1 < col_max:
32                              if pic[r][c + 1] == clr and not visited[r][c + 1]:
33                                  queue.append((r, c + 1))
34                          if c - 1 >= 0:
35                              if pic[r][c - 1] == clr and not visited[r][c - 1]:
36                                  queue.append((r, c - 1))
37
38                  if len(colors[key]) < 10:
39                      colors.pop(key)
40                  else:
41                      # добавляем площадь цвета от общего размера картинки
42                      colors[key].insert(0, len(colors[key]) / (pic_h * pic_w) * 100)
43
44      # выбираем цвет, у которого минимальная дельта к площади из условий (sq)
45      # for clr in colors:
46      #     print (clr, len(colors[clr]))
47
48      min_delta = float('inf')
49      for key in colors:
50          delta = abs(colors[key][0] - sq)
51          if  delta < min_delta:
52              key_clr = key
53              min_delta = delta
54
55      max_x = 0
56      for xy in colors[key_clr][1:]:
57          if xy[0] > max_x:
58              max_x = xy[0]
59
60      return max_x
61
62
63  def avg(arr):
64      return sum(arr) / len(arr)
65
66  virtual = True
67
68  results = []
69  data = sys.stdin.readlines()
70
71  cam_h, cam_alpha, cam_beta, pic_h, pic_w, sq = list(map(int, data[0].strip().split(' ')))
72  data.pop(0)
73
74  """
75  cam_h                   - высота установки камеры, мм
76  camAlpha          - угол камеры относительно горизонта, градусы
77  camBeta                 - угол камеры, градусы
78  picH                  - высота изображения, пиксели
79  picW                  - ширина изображения, пиксели
80  sq                  - минимальная площадь предмета, % от общей площади картинки
81  """
82
83  picRaw, picDec = [], []
84
```

```
for row in range(pic_h):
    picRaw.append([])
    lineRaw = list(data[row].strip().split(' '))
    picRaw[row] = lineRaw

object_bottom = find_obj2(picRaw)                # pixels from picture_bottom to object_bottom
angle = 90 - cam_alpha - cam_beta/2 + (pic_h - object_bottom)/pic_h * cam_beta

result = cam_h * math.tan(math.radians(angle))
print (round(result))
```

Задача 5.1.6. Определение вектора перемещения робота в заданном виде (20 баллов)

Робот, собранный по дифференциальной схеме, оборудован камерой. Камера установлена так, что смотрит вниз перпендикулярно поверхности, по которой движется робот. Известны характеристики камеры такие, как разрешение ($height \times width$), 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. Всего кадров было сделано N.

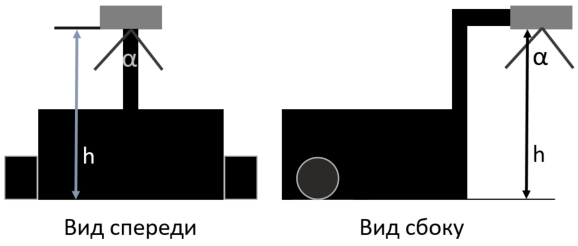


Рис. 5.24: Вид робота. Поясняющая картинка

Зная, что робот движется с постоянной линейной (v) и постоянной угловой (ω) скоростью, необходимо определить перемещение по X и по Y в мм.

Формат входных данных

Входной файл состоит из нескольких строк.

Первая строка содержит 5 целых чисел: N, h, α, $height$, $width$:

- N — количество замеров камерой ($2 \leqslant N \leqslant 16$);
- h — высота установки камеры в мм ($10 \leqslant h \leqslant 2000$);
- α — угол обзора камеры в градусах ($15° \leqslant \alpha \leqslant 175°$);
- $height$ — разрешение изображения по высоте ($5 \leqslant height \leqslant 16$);
- $width$ — разрешение изображения по ширине ($5 \leqslant width \leqslant 16$).

Далее расположено N строк, на каждой из которых расположено изображение размером $height \times width$ в виде шестнадцатиричных чисел слева направо, сверху вниз. Данные числа имеют следующий вид: $RRGGBB$, где RR - 16тиричное число

R составляющей данного элемента матрицы, GG и BB — 16тиричные числа G и B составляющих соответственно.

Все числа являются целыми.

Формат выходных данных

Вывести пару вещественных чисел с точностью до целых — перемещение по X и по Y в мм в данном порядке через пробел.

Примеры

Примеры входных данных и ответов к ним можно найти по `http://bit.ly/2EP1c5g`данной ссылке.

Решение

Для примера рассмотрим схему установки камеры (рис. 5.25), снимки с камеры (рис. 5.26) и набор данных:

- Количество замеров камерой, $N = 3$
- Высота установки камеры, $h = 100$ мм
- Угол обзора камеры, $\alpha = 30°$
- Разрешение снимка камеры по высоте, $height = 15$ пикселей
- Разрешение снимка камеры по ширине, $width = 15$ пикселей

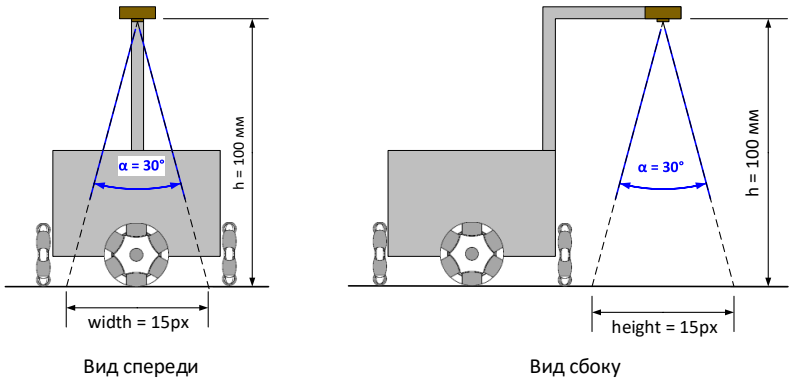


Рис. 5.25: Схема установки камеры с исходными данными

Для решения задачи сначала вычисляем перемещения робота через смещения пикселей на снимках (рис. 5.27). Сравниваем по два смежных снимка: $_{i-1i}$. После обработки всех снимков получаем конечное смещение в пикселях.

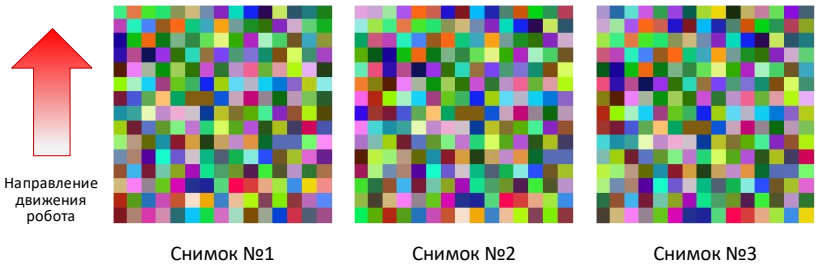


Рис. 5.26: Пример снимков (1 цв.квадрат = 1 пиксель) с камеры размером 15x15 пикселей

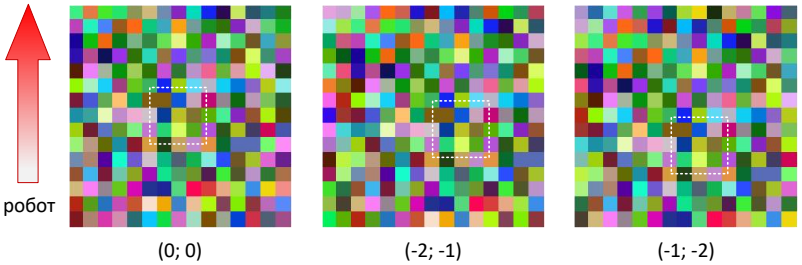


Рис. 5.27: Вычисление перемещения робота

Для пересчета смещения из пикселей в мм рассмотрим схему на рис. 5.28 и переведем полученные относительные координаты $(x; y)$ в мм.

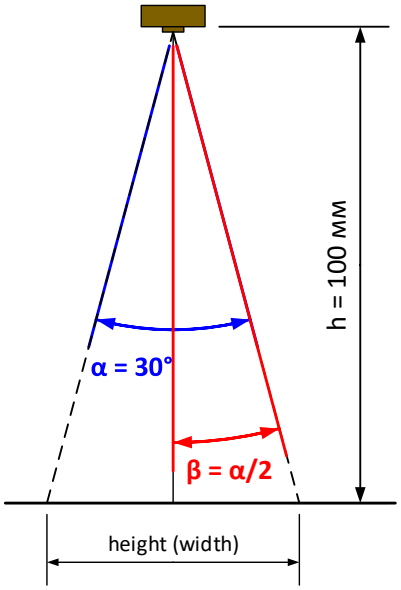


Рис. 5.28:

Вычислим размерность 1 пикселя для ширины и высоты снимка (формулы 5.34 и 5.35).

$$pixel\_width\_mm = \frac{h \cdot tg(\beta) \cdot 2}{width} \Rightarrow \frac{100 \cdot tg(0.2618) \cdot 2}{15} = 3.57 \text{ мм} \qquad (5.34)$$

$$pixel\_height\_mm = \frac{h \cdot tg(\beta) \cdot 2}{height} \Rightarrow \frac{100 \cdot tg(0.2618) \cdot 2}{15} = 3.57 \text{ мм} \tag{5.35}$$

Переводим пиксели в мм:

$$x = x \cdot 3.57 = -1 \cdot 3.57 = -3.6(\text{мм})$$
$$y = y \cdot 3.57 = -2 \cdot 3.57 = -7.1(\text{мм})$$

Ответ: -3.6 -7.1

Пример программы-решения

Ниже представлено решение на языке Python3

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

import math
import sys

def check(mat1, mat2):  #функция для нахождения смещения между двумя кадрами
    maximum = 0
    max_y = 0
    max_x = 0
    for i in range(-pic_h + 1, pic_h):
        for j in range(-pic_w + 1, pic_w):
            max_this = 0
            for y in range(pic_h):
                for x in range(pic_w):
                    if (y + i) >= 0 and (x + j) >= 0 and (y + i) < pic_h and (x + j) < pic_w:
                        if mat1[y][x] == mat2[y + i][x + j]:
                            max_this += 1
            if max_this > maximum:
                max_x = -j
                max_y = -i
                maximum = max_this
    return [maximum, max_x, max_y]

data = sys.stdin.readlines()

shots_n, cam_h, cam_alpha, pic_h, pic_w = list(map(int, data[0].strip().split(' ')))
data.pop(0)
'''
#shots_n    - кол-во замеров
#h              - высота установки камеры, мм
#cam_alpha  - угол обзора камеры, градусы
#pic_h           - высота изображения
#pic_w           - ширина изображения
'''
shots = []

for i in range(shots_n):
    shots.append([])                                                    # очередной кадр
    shot_raw = data[i].strip().split(' ')                    # кадр в виде одномерного массива пиксе
    shot_dec = list(map(lambda hx: int(hx,16), shot_raw))    # кадр в виде одномерного массива пиксел
    for r in range(0, pic_h * pic_w, pic_w):
```

```
42          shots[-1].append(shot_dec[r:r + pic_w])                    # очередная строка пикселей в кад
43
44  x, y = 0, 0
45  for i in range(1, shots_n):
46      temp = check(shots[i-1], shots[i])
47
48      x += int(temp[1])
49      y += int(temp[2])
50
51  x = math.tan(math.radians(cam_alpha / 2)) * 2 * cam_h / pic_w * x
52  y = math.tan(math.radians(cam_alpha / 2)) * 2 * cam_h / pic_h * y
53  print(str(round(x, 1)) + " " + str(round(y, 1)))
```

Задача 5.1.7. Распознавание ARTag маркера (10 баллов)

Для определения своего местоположения квадрокоптер использует камеру, снимающую поверхность, над которой перемещается робот. Изображение с камеры приходит в управляющую программу в виде набора $height \times width$ точек, где каждая точка закодирована в RGB-формате.

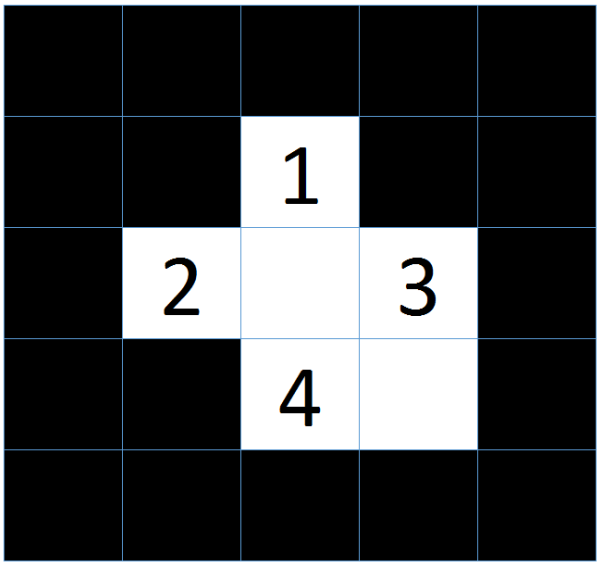


Рис. 5.29: Нумерация битов в маркера

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 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 (см. рис. 5.29).

Например, на маркере с рис. 5.30 закодировано число 0011_2, что эквивалентно 3_{10}.

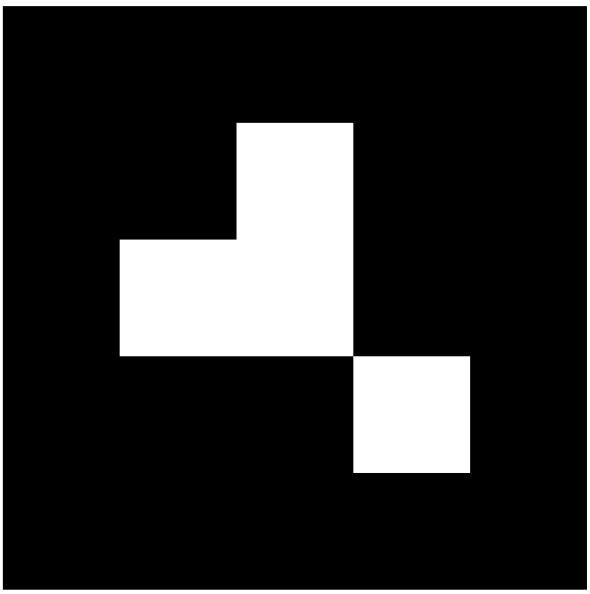


Рис. 5.30: Маркер с закодированным значением - 0011_2

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, тон и добавляют блики на изображение. Также квадрокоптеру не всегда удается полностью захватить маркер, и необходимо обрабатывать данные с нескольких снимков.

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

Входной файл состоит из нескольких строк.

Первая строка содержит 3 целых числа: N, $height$, $width$:

- N — количество замеров камерой ($1 \leqslant N \leqslant 100$);
- $height$ — разрешение изображения по высоте ($10 \leqslant height \leqslant 1000$);
- $width$ — разрешение изображения по ширине ($10 \leqslant width \leqslant 1000$).

Далее расположено N строк на каждой расположено изображение размером $height \times width$ ввиде шестнадцатиричных чисел слева направо сверху вниз. Данные числа имеют следующий вид: $RRGGBB$, где RR - 16тиричное число R составляющей данного элемента матрицы, GG и BB — эт0 16ти-ричные числа G и B составляющих соответственно.

Все числа являются целыми.

Формат выходных данных

Вывести одно целое число в десятичной системе счисления, которое было закодированное на маркере. В случае если невозможно определить число, следует вывести

−1.

Примеры

Примеры входных данных и ответов к ним можно найти по `http://bit.ly/2DHww5w`данной сслыке.

Решение

Декомпозиция задачи
Для каждого кадра:

1. Перевод цветного изображения в ч/б (состоит из "0"и "1")
2. Распознавание метки ARTag
   (a) Нахождение углов многоугольника (до 8)
   (b) если углов больше 4 - "достраиваем"срезанные углы
   (c) Нахождение центра метки, он же - бит четности (parity bit)
   (d) Нахождение ключевого бита (key bit)
   (e) "Сборка"двоичного числа
   (f) Сверка с битом четности
   (g) Перевод в десятичную систему счисления

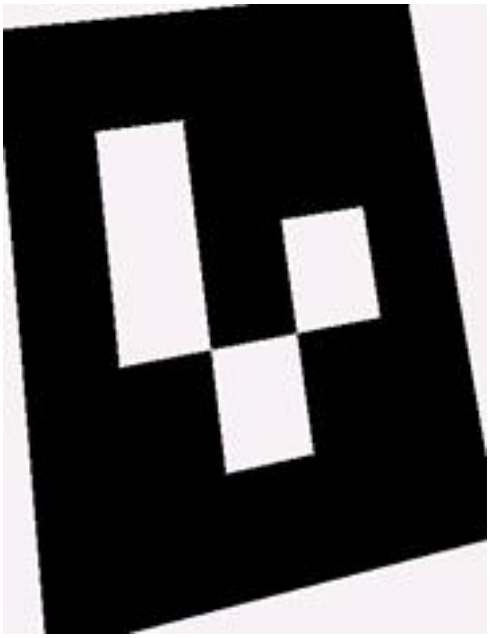


Рис. 5.31: Исходное изображение метки ARTag

С алгоритмами перевода изображения в черно-белый вариант можно ознакомиться в презентации `http://robolymp.ru/forum/index.php?PAGE_NAME=list&FID=116` Алгоритмические основы технического зрения

по адресу: `http://robolymp.ru/forum/index.php?PAGE\_NAME=list\&FID=116`.

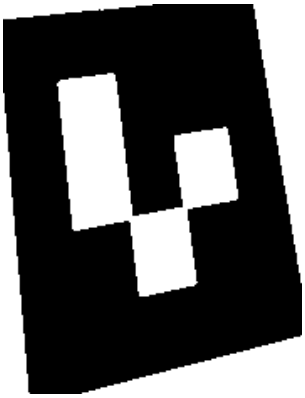


Рис. 5.32: Изображение в ч/б варианте

Находим координаты углов n-угольника. В нашем случае получается 8-угольник:

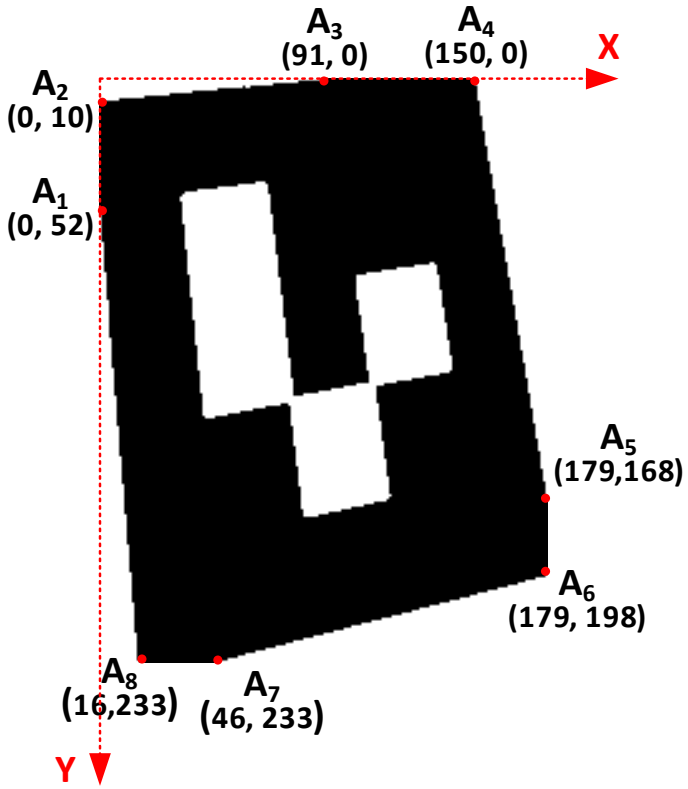


Рис. 5.33: Координаты углов метки ARTag

Получается, что все 4 угла 'срезаны' значит нам надо их 'достроить'.

Каждый угол образуется пересечением отрезков близлежащих сторон метки. На-

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
X	0	0	91	150	179	179	46	16
Y	52	10	0	0	168	198	233	233

Таблица 5.2: Координаты углов метки ARTag

пример, ∠A (верхний левый) образуется пересечением отрезков $(A_1; A_8)$ и $(A_2; A_3)$, т.е. для нахождения координат угла нам достаточно найти точку пересечения прямых.

Исходя из линейной функции $y = k \cdot x + b$ найдем коэффициенты $k_{1,2}$ и $b_{1,2}$ для каждого отрезка:

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

Для отрезка $(A_1; A_8)$:

$$k_1 = \frac{233 - 52}{16 - 0} = 11.31; \qquad b_1 = 233 - 11.31 \cdot 16 = 52.04$$

Для отрезка $(A_2; A_3)$:

$$k_2 = \frac{0 - 10}{91 - 0} = -0.11; \qquad b_2 = 0 + 0.11 \cdot 91 = 10.01$$

Теперь находим координаты $x; y$ ∠A:

$$x = \frac{b_1 - b_2}{k_2 - k_1} = \frac{52.04 - 10.01}{-0.11 - 11.31} = -3.68$$
$$y = k_1 \cdot x + b_1 = 11.31 \cdot -3.68 + 52.04 = 10.42$$

Аналогичным образом находим остальные углы и получаем координаты 4х угольника:

Далее находим центр метки, он же будет "битом четности". В нашем случае центр метки - это точка пересечения диагоналей метки: $AC \cap BD$, в результате получаем координаты центра: $O(90; 104)$.

После чего находим центры сторон 4х-угольника: AB, BC, CD, DA. Для этого мы разделим отрезок каждой стороны на 2 равные части. Разберем на примере стороны AB:

$$x_{AB} = \frac{x_1 + x_2}{2} = \frac{-4 + 149}{2} = 72.5$$
$$y_{AB} = \frac{y_1 + y_2}{2} = \frac{10 - 6}{2} = 2$$

Аналогичным способом находим координаты середин остальных сторон, в результате получаем (рис.5.35).

Теперь каждый отрезок от точки O до точки на периметре 4х-угольника делим на 2.5 части. Почему на 2.5? В нашем случае маркер представляет квадрат из 5х5 клеток и центр находится в середине клетки, поэтому расстояние от центра метки до любой точки на краю метки составляет 2.5 клетки (рис. 5.36).

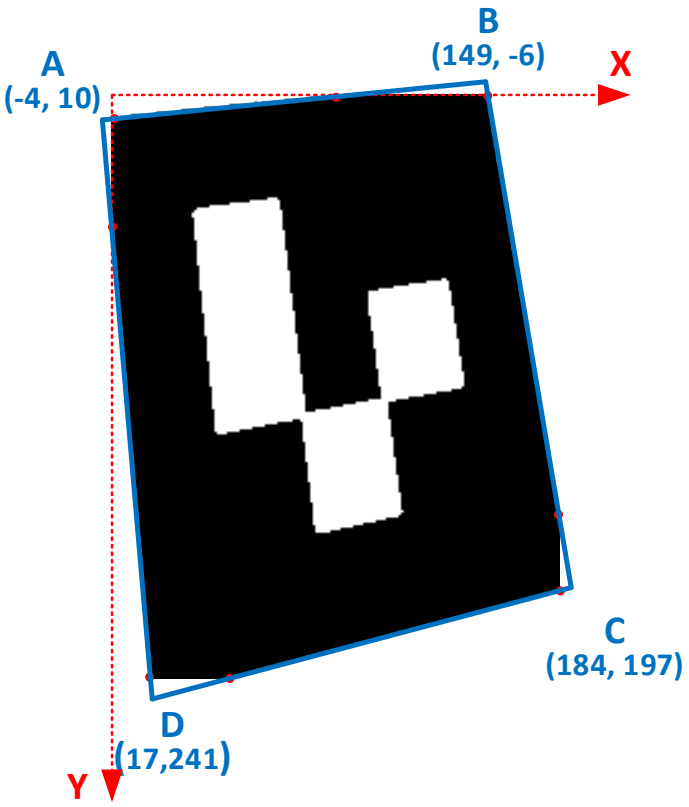


Рис. 5.34: Координаты полученного 4-х угольника

	O	AB	BC	CD	DA
X	90	73	167	101	7
Y	104	2	96	219	126

Таблица 5.3: Координаты центра метки и середин сторон периметра метки

По полученным точкам OA_1, OB_1, OC_1, OD_1 находим "ключевой бит по которому определятся начало отсчета двоичного числа на метке. В нашем случае "ключевой бит"находится в точке OA_1.

По точкам OAB_1, OBC_1, OCD_1, ODA_1 составляем двоичный код. Старший разряд двоичного числа является самым верхним от расположения "ключевого бита".

Расположение информационных битов, в зависимости от "ключевого бита показаны на рис. 5.1.7 и 5.1.7.

В нашем случае получается двоичное число: 0001
Сверяем двоичное число с "битом четности": если в двоичном числе количество "1"нечетное, значит "бит четности"должен быть равен "1 если же количество "1"в двоичном числе четное, то "бит четности"должен быть равен "0".

Если проверка на четность прошла успешно, то принимаем ответ как правильный, если проверка не прошла, значит обрабатываем следующий кадр.

В нашем случае проверка прошла успешно, значит переводим полученное число

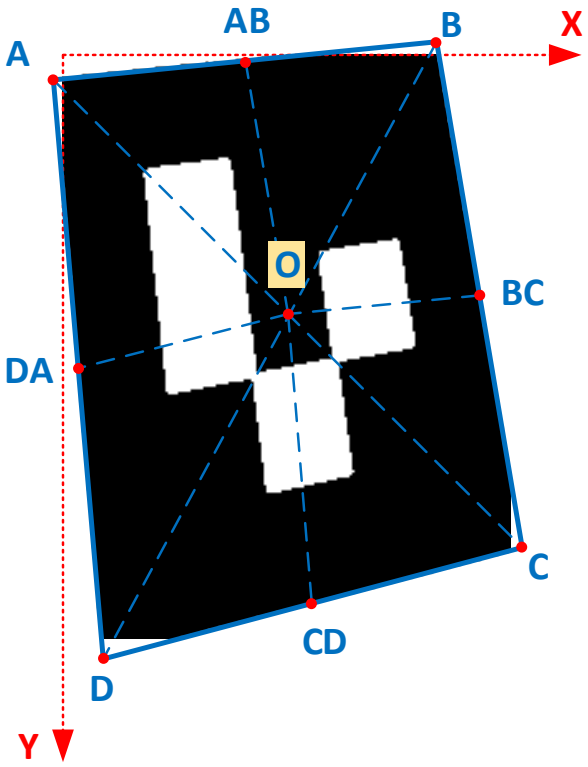


Рис. 5.35: Разделение сторон периметра метки пополам

из двоичной системы в десятичную и принимаем как правильный ответ, получается: 1.

Если кадров несколько, то лучше все их распознать и сравнить полученные результаты.

Ответ: на метке закодировано число 1.

Пример программы-решения

Ниже представлено решение на языке Python3

```
# -*- coding: utf-8 -*-
import sys
import math
import numpy as np

def cross_find(coords):
    # coords  - массив с координатами 4х угольника (ABCD)
    x1, y1 = coords[0]
    x2, y2 = coords[2]
    x3, y3 = coords[1]
    x4, y4 = coords[3]

    x = -(((x1 * y2 - x2 * y1) * (x4 - x3) - (x3 * y4 - x4 * y3) * (x2 - x1)) / (
                (y1 - y2) * (x4 - x3) - (y3 - y4) * (x2 - x1)))
```

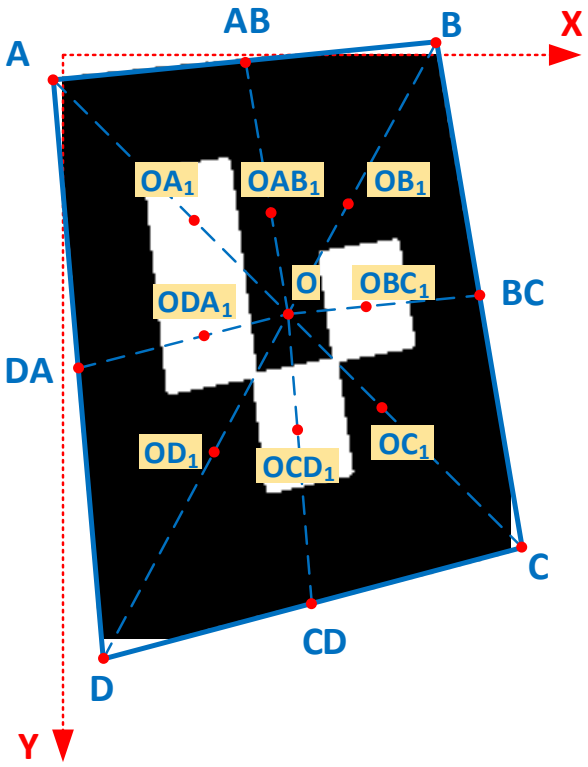


Рис. 5.36: Полученные точки в центрах клеток

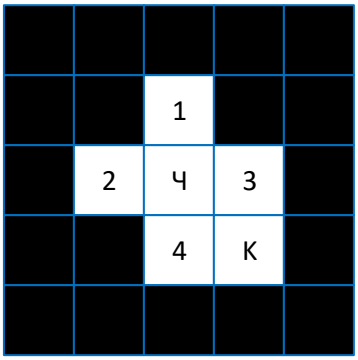


Рис. 5.37: Общая схема метки ARTag

```
15      y = ((y3 - y4) * (-x) - (x3 * y4 - x4 * y3)) / (x4 - x3)
16
17      return [int(x), int(y)]
18
19
20  def lineToSegments(begin, end, segments = 2):
21      # begin, end - массивы с координатами отрезка (начало, конец). начало, как правило - центр
22      x0 = round((end[0] - begin[0]) / segments)
23      y0 = round((end[1] - begin[1]) / segments)
24
25      points = [begin]
26      for i in range(1, segments):
27          points.append([])
28          points[i].append(points[i - 1][0] + x0)
```

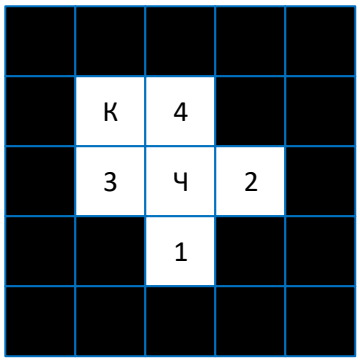


Рис. 5.38: Расположение битов для нашего примера

```
29          points[i].append(points[i - 1][1] + y0)
30      points.append([end])
31      return points
32
33
34  def rgb2clr(rgb):
35      #return (rgb[0] * 256 ** 2 + rgb[1] * 256 + rgb[2])
36      if type(rgb) == int:                # один цвет, значит переводим в серый
37          rgb = [rgb]
38          for i in range(2):
39              rgb.append(rgb[0])
40      return int(sum([rgb[i] * 256**(2-i) for i in range(len(rgb))]))
41
42
43  def clr2rgb(clr):
44      r = (clr & 0xFF0000) >> 16
45      g = (clr & 0x00FF00) >> 8
46      b = (clr & 0x0000FF)
47      return (r, g ,b)
48
49
50  def blur(pic_in): # div, offset):
51      blur5 = [
52          [0.000789, 0.006581, 0.013347, 0.006581, 0.000789],
53          [0.006581, 0.054901, 0.111345, 0.054901, 0.006581],
54          [0.013347, 0.111345, 0.225821, 0.111345, 0.013347],
55          [0.006581, 0.054901, 0.111345, 0.054901, 0.006581],
56          [0.000789, 0.006581, 0.013347, 0.006581, 0.000789]
57      ]
58      blur3 = [
59          [0.054901, 0.111345, 0.054901],
60          [0.111345, 0.225821, 0.111345],
61          [0.054901, 0.111345, 0.054901]
62      ]
63
64      pic_out, pic_blur = [], []
65      size = 3                    # "window" size
66      side = size //2
67
68      pic_h, pic_w = len(pic_in), len(pic_in[0])
69      for i in range(side, pic_h - side):
70          pic_out.append([])
71          for j in range(side, pic_w - side):
72
73              # pixels - 2D список цветов пикселей "окна" в каналах R,G,B
```

```
74              # очередное "окно" из пикселей 3*3 для применения фильтра
75              pixels = []
76              for p_y in range(i-side, i+side+1):
77                  pixels.append([])
78                  for p_x in range(j-side, j+side+1):
79                      pixels[-1].append(clr2rgb(pic_in[p_y][p_x]))
80
81              # pixels_RGB - 2D список списков цветов по-канально [R], [G], [B]
82              pixels_RGB = []
83              for ii in range(3):
84                  pixels_RGB.append([[rgb[ii] for rgb in row] for row in pixels])
85
86              pic_blur = []
87              for channel in range(3):
88                  pic_blur.append([])
89                  for ii in range(size):
90                      pic_blur[-1].append([])
91                      for jj in range(size):
92                          if size == 3:
93                              pic_blur[-1][-1].append(pixels_RGB[channel][ii][jj] * blur3[ii][jj])
94                          elif size == 5:
95                              pic_blur[-1][-1].append(pixels_RGB[channel][ii][jj] * blur5[ii][jj])
96
97              pixels_RGB_blur = [int(sum([sum(row) for row in ch_blur])) for ch_blur in pic_blur]
98              pic_out[-1].append(rgb2clr(pixels_RGB_blur))
99
100     return pic_out
101
102
103 def pic_to_RGB (pic_in):
104     pic_out = []
105     for row in range(len(pic_in)):
106         pic_out.append([])
107         for col in range(len(pic_in[0])):
108             pic_out[-1].append(clr2rgb(pic_in[row][col]))
109     return pic_out
110
111
112 def rgb2grey(pic_in, mode):
113     RGB, pic_out = [], []
114     pic_h, pic_w = len(pic_in), len(pic_in[0])
115     for i in range(pic_h):
116         pic_out.append([])
117         for j in range(pic_w):
118             RGB = clr2rgb(pic_in[i][j])
119             if mode == 1:                # "V from HSV", V = max(r, g, b)
120                 result = max(RGB)
121             elif mode == 2:             # через Y ' from Y'UV(средневзвешенное) вариант 1
122                 # result = Math.round((0.299 * RGB[0]) + (0.587 * RGB[1]) + (0.114 * RGB[2]))
123                 result = round((0.2126 * RGB[0]) + (0.7152 * RGB[1]) + (0.0722 * RGB[2]))
124             elif mode == 3:  # через Y ' from Y' UV(средневзвешенное) вариант 2
125                 clr = pic_in[i][j]
126                 # tmp = ((((clr >> 16) & 0xff) * 76) + (((clr >> 8) & 0xff) * 150) + ((clr & 0xff) *
127                 tmp = ((((clr >> 16) & 0xff) * 76) + (((clr >> 8) & 0xff) * 150) + ((clr & 0xff) * 29
128                 result = tmp << 16 | tmp << 8 | tmp  # r | g | b
129             elif mode == 4:          # Поиск ближайшей точки нейтральной оси: L = (max(R, G, B) + min
130                 result = round((max(RGB) + min(RGB)) / 2)
131             elif mode == 5:          # Среднее арифметическое компонент R, G, B
132                 result = round(sum(RGB) / 3)
133
```

```
134             #print (result,rgb2clr(result))
135             pic_out[-1].append(rgb2clr(result))
136 
137     return pic_out
138 
139 
140 def rgb2bw(pic_in, method = 'niblack'):
141     #pic_in_RGB = pic_to_RGB(pic_in)
142 
143     pic_h, pic_w = len(pic_in), len(pic_in[0])
144     pic_out = [[16777215 for _ in range(pic_w)] for _ in range(pic_h)]
145 
146     delta = 0
147     dot_side = 3
148     radius = int(dot_side/2)                    # pixels from center 'big dot' to each side
149 
150     if method == 'niblack' or method == 'sauvola':
151         '''
152         Niblack:
153         T = m + k * s
154         m - mean of local area pixels
155         k - 0.2     - value by author
156         s - standart deviation of local pixel area
157 
158         Sauvola:
159         T = m * (1 - k * 1 - S / R)
160         m - mean of pixels under window area
161         S - dynamic range of variance
162         k - may be in the range of 0-1 (?)
163         k=0.5 and R=128         - values by author
164 
165         '''
166         for row in range(delta + radius, pic_h - delta - radius):
167             for col in range(delta + radius, pic_w - delta - radius):
168                 pixels = []
169                 for p_r in range(dot_side):
170                     for p_c in range(dot_side):
171                         pixels.append(pic_in[row - radius + p_r][col - radius + p_c])
172                 #print (row,col,list(map(hex, pixels)))
173                 mean = sum(pixels)/len(pixels)
174                 if method == 'niblack':
175                     sd = standart_deviation(pixels)
176                     T = int(mean + sd * 0.2)
177                     #print (mean, T)
178                 else:
179                     S = standart_variance(pixels)
180                     T = int(mean * (1 - 0.5 * (1 - S / 128)))
181 
182                 # pix = 16777215 if pic_in[row][col] >= T else 0
183                 if pic_in[row][col] < T:
184                     pic_out[row][col] = 0
185 
186     elif method == 'threshold':
187         for row in range(pic_h):
188             for col in range(pic_w):
189                 if pic_in[row][col] < 1500000:
190                     pic_out[row][col] = 1
191                 else:
192                     pic_out[row][col] = 0
193 
```

```
194     return pic_out
195
196
197 def standart_deviation(nums):
198     mean = sum(nums)/len(nums)
199     return ((1/len(nums) * sum([(num - mean) **2 for num in nums])/(len(nums))) ** 0.5)
200
201
202 def standart_variance(nums):
203     mean = sum(nums)/len(nums)
204     return (sum([(num - mean) **2 for num in nums])/len(nums))
205
206
207 def k_b(P1, P2):
208     x1, y1 = P1
209     x2, y2 = P2
210     k = (y2 - y1) / (x2 - x1)
211     b = y2 - k * x2
212     return (k, b)
213
214 def mean_xy(corner1, corner2):
215     # вычисление средних координат x, y двух точек
216     mean_x = int(corner1[0] + corner2[0]) / 2
217     mean_y = int(corner1[1] + corner2[1]) / 2
218     return (mean_x, mean_y)
219
220
221 def bias_xy(corner, m_xy):
222     # отклонение координат x, y  от среднего значения
223     return (abs(corner[0] - m_xy[0]), abs(corner[1] - m_xy[1]))
224
225
226 def close_corners(corners):
227     BIAS = 10                                   # допустимая погрешность
228     abcd = [()] * 4
229
230     meanXY = mean_xy(corners[0],corners[1])
231     bias = bias_xy(corners[0], meanXY)
232     print (corners[6], corners[0])
233     if sum(bias) > BIAS and (corners[6] != corners[0] and corners[2] != corners[1]):
234         k1, b1 = k_b(corners[6], corners[0])
235         k2, b2 = k_b(corners[2], corners[1])
236         x = (b1 - b2) / (k2 - k1)
237         y = k1 * x + b1
238     else:
239         x, y = meanXY
240     abcd[0] = (int(x), int(y))
241
242     meanXY = mean_xy(corners[2], corners[3])
243     bias = bias_xy(corners[3], meanXY)
244     if sum(bias) > BIAS and (corners[1] != corners[2] and corners[5] != corners[3]):
245         k1, b1 = k_b(corners[1], corners[2])
246         k2, b2 = k_b(corners[5], corners[3])
247         x = (b1 - b2) / (k2 - k1)
248         y = k1 * x + b1
249     else:
250         x, y = meanXY
251     abcd[1] = (int(x), int(y))
252
253     meanXY = mean_xy(corners[4], corners[5])
```

```
254     bias = bias_xy(corners[4], meanXY)
255     if sum(bias) > BIAS and (corners[3] != corners[5] and corners[7] != corners[4]):
256         k1, b1 = k_b(corners[3], corners[5])
257         k2, b2 = k_b(corners[7], corners[4])
258         x = (b1 - b2) / (k2 - k1)
259         y = k1 * x + b1
260     else:
261         x, y = meanXY
262     abcd[2] = (int(x), int(y))
263 
264     meanXY = mean_xy(corners[7], corners[6])
265     bias = bias_xy(corners[7], meanXY)
266     if sum(bias) > BIAS and (corners[4] != corners[7] and corners[0] != corners[6]):
267         k1, b1 = k_b(corners[4], corners[7])
268         k2, b2 = k_b(corners[0], corners[6])
269         x = (b1 - b2) / (k2 - k1)
270         y = k1 * x + b1
271     else:
272         x, y = meanXY
273     abcd[3] = (int(x), int(y))
274 
275     return abcd
276 
277 
278 def get_corners(pic_in):
279     corners = [(-1, -1)] * 8       # left_up x 2, right_up x2, right_down x2, left_down x2
280     #pic_h, pic_w = len(pic_in), len(pic_in[0])
281 
282     out_l = out_r = False
283     for row in range(pic_h):                        # Up -> Down
284         for col_l in range(pic_w):                  # L -> R
285             if pic_in[row][col_l] == 1 and not out_l:
286                 corners[0] = (row, col_l)
287                 out_l = True
288 
289         for col_r in range(pic_w-1, -1, -1):        # L <- R
290             if pic_in[row][col_r] == 1 and not out_r:
291                 corners[1] = (row, col_r)
292                 out_r = True
293 
294     out_l = out_r = False
295     for row in range(pic_h-1, -1, -1):              # Up <- Down
296         for col_l in range(pic_w):                  # L -> R
297             if pic_in[row][col_l] == 1 and not out_l:
298                 corners[4] = (row, col_l)
299                 out_l = True
300 
301         for col_r in range(pic_w-1, -1, -1):        # L <- R
302             if pic_in[row][col_r] == 1 and not out_r:
303                 corners[5] = (row, col_r)
304                 out_r = True
305 
306     out_u = out_d = False
307     for col in range(pic_w):                        # Left -> Right
308         for row_u in range(pic_h):                  # U -> D
309             if pic_in[row_u][col] == 1 and not out_u:
310                 corners[6] = (row_u, col)
311                 out_u = True
312 
313         for row_d in range(pic_h-1, -1, -1):        # D <- U
```

```
314             if pic_in[row_d][col] == 1 and not out_d:
315                 corners[7] = (row_d, col)
316                 out_d = True
317 
318     out_u = out_d = False
319     for col in range(pic_w-1, -1, -1):              # Left <- Right
320         for row_u in range(pic_h):                  # U -> D
321             if pic_in[row_u][col] == 1 and not out_u:
322                 corners[2] = (row_u, col)
323                 out_u = True
324 
325         for row_d in range(pic_h-1, -1, -1):        # D <- U
326             if pic_in[row_d][col] == 1 and not out_d:
327                 corners[3] = (row_d, col)
328                 out_d = True
329 
330     print(corners)
331     if sum(map(len, corners)) == 0:                             # не найдено ни одного угла
332         return -1
333     elif no_corner(corners[0], corners[1]) or no_corner(corners[2], corners[3]) or \
334         no_corner(corners[4], corners[5]) or no_corner(corners[6], corners[7]):
335         return -1                                               # нету одного из углов
336     else:
337         return close_corners(corners)
338 
339 
340 def find_corners2(pic_in, delta = 0):
341 
342     looked = []
343 
344     def check_xy(yx):
345         if pic_h - delta > yx[0] >= 0 + delta and pic_w - delta > yx[1] >= 0 + delta:
346             return yx not in looked and pic_in[yx[0]][yx[1]] == 1
347         else:
348             return False
349 
350 
351     def find_x(y0x0, direction = 1):
352         print (y0x0)
353         y0, x0 = y0x0
354         prev_yx = (0, 0)
355         dX, dY = prev_yx
356 
357         while True:
358             y, x = y0, x0
359             if check_xy((y - direction, x)):
360                 y -= direction
361                 dY += 1
362             elif check_xy((y, x + direction)):
363                 x += direction
364                 dX += 1
365             elif check_xy((y + direction,x)):
366                 y += direction
367                 dY += 1
368 
369             if (y, x) not in perimeter:
370                 perimeter.append((y, x))
371 
372             if x != x0 and dY > 0:
373                 dX, dY = 0, 0
```

```
374
375             if x == x0 and dY > 1:
376                 looked.pop()
377                 return (prev_yx)
378
379             #print('yx0:',y0, x0, '   yx:', y, x, '   ', y0 == y, x0 == x, '  dYX:',dY, dX)
380             looked.append((y, x))
381             prev_yx = (y0, x0)
382             if y0 == y and x0 == x:
383                 x += direction
384             y0, x0 = y, x
385
386
387     def find_y(y0x0, direction = 1):
388         y0, x0 = y0x0
389         prev_yx = (0, 0)
390         dX, dY = prev_yx
391
392         while True:  # from up to down
393             y, x = y0, x0
394             if check_xy((y, x + direction)):
395                 x += direction
396                 dX += 1
397             elif check_xy((y + direction, x)):
398                 y += direction
399                 dY += 1
400             elif check_xy((y, x - direction)):
401                 x -= direction
402                 dX += 1
403
404             if (y, x) not in perimeter:
405                 perimeter.append((y, x))
406
407             if y0 != y and dX > 0:
408                 dX, dY = 0, 0
409
410             if y == y0 and dX > 1:
411                 looked.pop()
412                 return prev_yx
413
414             #print('yx0:',y0, x0, '   yx:', y, x, '   ', y0 == y, x0 == x, '  dYX:',dY, dX, '  black
415
416             looked.append((y, x))
417             prev_yx = (y0, x0)
418             if y0 == y and x0 == x:
419                 if (y,x) not in looked:
420                     y += direction
421                 else:
422                     break
423             y0, x0 = y, x
424
425     pic_h, pic_w = len(pic_in), len(pic_in[0])
426     start = (-1, -1)
427
428     for y in range(delta, pic_h - delta):
429         for x in range(delta, y + 1): # (delta):
430             if pic_in[x][y - x + delta] == 1:
431                 start = (x, y - x + delta)
432                 break
433         if sum(start) >= 0:
```

```
434             break
435
436     perimeter = [start]
437     corners0 = [start]
438     corners0.append(find_x(corners0[-1]))
439     corners0.append(find_y(corners0[-1]))
440     print (corners0)
441     corners0.append(find_x(corners0[-1], -1))
442     find_y(corners0[-1], -1)                        # для заполнения точек периметра
443
444 #    for i in perimeter:
445 #        print (*i)
446
447     corners = corners0.copy()
448
449     def corner_at_border(xy):
450         return xy[0] == pic_h-1 or xy[0] == 0 or xy[1] == pic_w - 1 or xy[1] == 0
451
452     if not corner_at_border(corners[0]) or not corner_at_border(corners[1]) or \
453         not corner_at_border(corners[2]) or not corner_at_border(corners[3]):
454         c = 1
455         for (y0, x0) in corners0:
456             dX = sum([1 for y, x in perimeter if y == y0]) - 1
457             dY = sum([1 for y, x in perimeter if x == x0]) - 1
458             yx = (y0, x0)
459
460             if dY > dX:
461                 if (y0 + dY, x0) in perimeter:
462                     yx = (y0 + dY, x0)
463                 else:
464                     yx = (y0 - dY, x0)
465             elif dX > dY:
466                 if (y0, x0 + dX) in perimeter:
467                     yx = (y0, x0 + dX)
468                 else:
469                     yx = (y0, x0 - dX)
470             corners.insert(c, yx)
471             c += 2
472
473         #for c in corners:
474         #    print (*c)
475
476     return corners
477
478
479 def no_corner(c1, c2):
480     if sum(c1) < 0 or sum(c2) < 0:
481         return True
482
483
484 def decode_artag(abcd):
485     center = cross_find(abcd)                # center of ARTag
486     parity = pic_bw[center[0]][center[1]]
487
488     '''
489        K (keys) cells
490        -------------
491        | 3 |   | 0 |
492        -------------
493        |   |   |   |
```

```
494         -------------
495         | 2 |   | 1 |
496         -------------
497     '''
498     # отрезки от центр к углам
499     # из полученных точек берем только [1], т.к. [0] - это центр метки
500     K = []
501     for i in range(4):
502         K.append(tuple(map(int, lineToSegments(center, abcd[i], 3)[1])))
503     #print('K',K)
504 
505     KEY = -1
506     for i in range(4):
507         y, x = K[i]
508         if pic_bw[y][x] == 0:
509             KEY = i
510             break
511     print ('key',KEY)
512     if KEY == -1:                                   # не найдена "ключевая" клетка
513         return -1
514 
515     '''
516         bin's cells
517         -------------
518         |   | 3 |   |
519         -------------
520         | 2 |   | 0 |
521         -------------
522         |   | 1 |   |
523         -------------
524     '''
525 
526     bins = []
527     for i in range(4):
528         x, y = map(int, lineToSegments(K[i], K[(i+1) % 4])[1])
529         bins.append(pic_bw[x][y])
530     print ('bins', bins)
531     states = [(2, 3, 1, 0), (3, 2, 0, 1), (0, 3, 1, 2), (1, 0, 2, 3)]
532 
533     bin_num = ''
534     for i in range(4):
535         bin_num += str(bins[states[KEY][i]])
536 
537     # четное "1" - бит Ч в "0",    нечетое "1" - бит Ч в "1"
538     if bin_num.count('1') % 2 == 0 and parity == 0 or \
539             bin_num.count('1') % 2 != 0 and parity == 1:                # проверка на бит четности
540         return int(bin_num, 2)
541     else:
542         return -1                                                       # не прошел контроль четност
543 
544 
545 def bfs_edges(pic_in):
546     def check_range(y, x):
547         if pic_h > y >= 0 and pic_w > x >= 0:
548             return True
549         else:
550             return False
551 
552     def check_white(y, x):
553         for i in range(3):
```

```
554             for j in range(3):
555                 if i == j == 1:
556                     break
557                 if check_range(y - 1 + i, x - 1 + j):
558                     if pic_in[y - 1 + i][x - 1 + j] == 0:           # white pixel
559                         return True
560         return False
561 
562     def check_black(y, x):
563         bl = 0
564         for i in range(3):
565             for j in range(3):
566                 if check_range(y - 1 + i, x - 1 + j):
567                     if pic_in[y - 1 + i][x - 1 + j] == 1:           # black pixel
568                         bl += 1
569         if bl < 7:
570             return True
571         else:
572             return False
573 
574     pic_h, pic_w = len(pic_in), len(pic_in[0])
575     start = (-1, -1)
576     for i in range(pic_h):
577         for j in range(i):
578             if pic_in[j][i-j] == 1:
579                 start = (j, i-j)
580                 break
581         if sum(start) >= 0:
582             break
583 
584     visited = [[False for _ in range(pic_w)] for _ in range(pic_h)]
585     queue = [start]
586     edges = []
587     while len(queue) > 0:
588         y, x = queue.pop(0)
589         if not visited[y][x]:
590             visited[y][x] = True
591             if check_white(y, x):
592                 edges.append((y, x))
593             elif (x == 0 or y == 0) and check_black(y, x):
594                 edges.append((y, x))
595             elif (x == pic_w-1 or y == pic_h-1) and check_black(y, x):
596                 edges.append((y, x))
597 
598             for i in range(3):
599                 for j in range(3):
600                     if not i == j == 1:
601                         yy = y - 1 + i
602                         xx = x - 1 + j
603                         if check_range(yy, xx) and not visited[yy][xx] and pic_in[yy][xx] == 1:
604                             queue.append((yy, xx))
605 
606         if (y, x) == start and len(edges) > 1:
607             break
608 
609     # оставляем только периметр из точек
610     perimeter = []
611     looked = []
612     queue = [edges[0]]
613     while len(queue) > 0:
```

```
614         y0, x0 = queue.pop(0)
615         if not (y0, x0) in looked:
616             looked.append((y0, x0))
617             perimeter.append((y0, x0))
618             for y, x in edges:
619                 dY = abs(max(y0, y) - min(y0, y))
620                 dX = abs(max(x0, x) - min(x0, x))
621                 if (dY <= 1 and dX <= 1) and (y, x) not in looked and (y, x) not in queue:
622                     queue.append((y, x))
623 
624     return perimeter
625 
626 
627 path = "C:\\_docs\\!RobonesT\\!!Проекты\\Разбор НТИ ИРС\\zad7\\"
628 fileIn = path + "datasets\\0"
629 fileOut = path + "zad7_out.txt"
630 
631 with open(fileIn) as f:
632     data = f.readlines()
633 
634 # data = sys.stdin.readlines()
635 shots_n, pic_h, pic_w = list(map(int, data[0].strip().split(' ')))
636 data.pop(0)
637 """
638 #shots_h     - кол-во замеров камерой
639 #pic_h             - ширина изображения, пиксели
640 #pic_w             - высота изображения, пиксели
641 """
642 
643 pics_dec = []
644 for picN in range(shots_n):
645     pics_dec.append([])
646     for row in range(pic_h):
647         line = list(map(lambda h: int(h, 16), data[row + picN * pic_h].strip().split(' ')))
648         pics_dec[picN].append(line)
649 
650 print ('get file done')
651 
652 pic_grey = rgb2grey(pics_dec[0], 2)
653 hist = {}
654 for row in pic_grey:
655     for clr in row:
656         clr_rgb0 = clr2rgb(clr)[0]
657         if str(clr_rgb0) not in hist:
658             clr_txt = str(clr_rgb0).rjust(3,'0')
659             hist[clr_txt] = 1
660         else:
661             hist[clr_txt] += 1
662 
663 for k, v in hist.items():
664     print(k, v)
665 
666 pic_blur = blur(pic_grey)
667 
668 pic_bw = rgb2bw(pic_blur, 'threshold')
669 
670 with open(fileOut, mode='w') as f:
671     for line in pic_bw:
672         f.write(','.join(list(map(str,line))) + '\n')
673 
```

```
674 print(find_corners2(pic_bw, 5))
675
676 #z = np.array(pic_bw, dtype = int)
677
678 #for i in z:
679 #print (sum(z[:,41]))
680 #ed = bfs_edges(pic_bw)
681 #print (ed)
682 #corners = [(-1,-1)] * 8
683 #print (sorted(ed))
684 #corners[0] = ed[0]
685 #for y in ed:
686 #    print (*y)
687
688
689 """
690
691 results = []
692 for i in range(shots_n):
693     pic_bw = rgb2bw(pics_dec[i],'threshold')
694     cs = get_corners(pic_bw)
695     if cs == -1:
696         res = -1
697     else:
698         res = decode_artag(cs)
699     results.append(res)
700     print ('result', results)
701
702 # vertical output
703 '''
704 spc = '16711680,' * pic_w                                          # красный цвет
705 spc = spc[:-1] + '\n'
706 with open(fileOut, mode='w') as f:
707 for i in pics_dec[0]:
708     #for j in range(len(pics_dec[0])):
709        #f.write(','.join(list(map(str, pics_dec[i][j]))) + '\n')
710    f.write(str(i).replace("[","").replace("]","") + '\n')
711 f.write(spc)
712 f.write(spc)
713 '''
714
715 # horizontal output
716 '''
717 spc = '16711680, 16711680, 16711680'
718 pics_out = []
719 for i in range(pic_h):
720     line = ''
721     for shot in range(shots_n):
722         line += ','.join(list(map(str, pics_dec[shot][i])))
723         line += ' ,' + spc
724     pics_out.append(line)
725 '''
726
727 """
728
729 #with open(fileOut, mode='w') as f:
730  #   for line in pic_bw:
731  #       f.write(','.join(list(map(str,line))) + '\n')
```

Задача 5.1.8. Определение лабиринта (10 баллов)

Роботы в количестве N, собранные по дифференциальной схеме, оснащёны тремя дальномерами, направленными налево, прямо и направо относительно направления движения робота. Роботы движутся одновременно из заранее известных секторов по неизвестному лабиринту, размером $K \times M$ секторов.

Между секторами могут быть стены, нахождение препятствия, внутри сектора недопустимо. Отсчёт координат секторов начинается с $(0; 0)$ в левом верхнем углу, X возрастает вправо, Y –вниз.

В процессе своих перемещений они получают значения с дальномеров, позволяющих узнать структуру лабиринта. В результате полученных данных необходимо понять в какие сектора полигона может попасть первый робот, без учёта нахождения на поле других роботов. В случае если данное значение невозможно определить, следует вывести количество секторов, посещённых в процессе движения данным роботом.

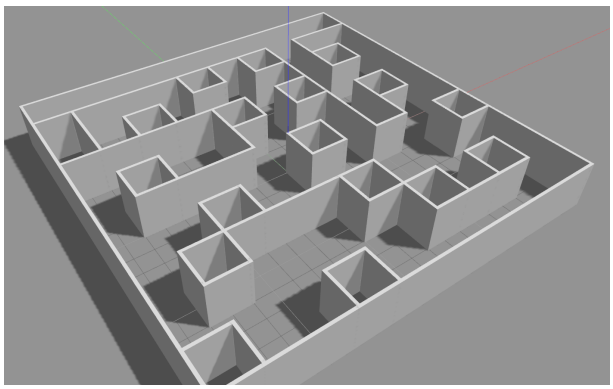


Рис. 5.39: Пример соревновательного полигона

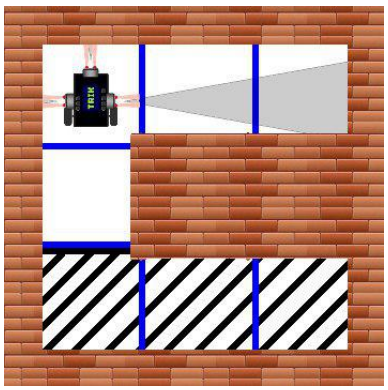


Рис. 5.40: Расположение датчиков

Формат входных данных

Первая строка содержит 4 целых числа: N, K, M и i:

- N — количество роботов на поле ($2 \leqslant N \leqslant 5$);
- K — ширина (по оси X) лабиринта в секторах ($4 \leqslant K \leqslant 20$);
- M — длина (по оси Y) лабиринта в секторах ($4 \leqslant M \leqslant 20$);

- i — количество показаний каждого робота.

Далее идет N строк, содержащие координаты старта каждого робота, а также направление робота при старте, т.е. одна строка имеет следующую структуру: x_s, y_s, dir, где:

- x_s — координаты старта данного робота по оси X;
- y_s — координаты старта данного робота по оси Y;
- dir — направление робота при старте:
  - U — робот направлен вверх;
  - L — робот направлен влево;
  - D — робот направлен вниз;
  - R — робот направлен вправо;

Далее идет N блоков по i строк. На каждой строке находится действие робота и показания всех датчиков после этого действия через пробел: $Movement$, D_l, D_f, D_r:

- $Movement$ — выполненное действие робота:
  - F — проезд робота в следующий по ходу движения сектор;
  - L — поворот робота в данном секторе налево;
  - R — поворот робота в данном секторе направо;
- D_l — показания датчика расстояния, направленного влево:
  - 1 — присутствует препятствие;
  - 0 — отсутствует препятствие;
- D_f — показания датчика расстояния, направленного вперёд:
  - 1 — присутствует препятствие;
  - 0 — отсутствует препятствие;
- D_r — показания датчика расстояния, направленного вправо:
  - 1 — присутствует препятствие;
  - 0 — отсутствует препятствие.

Формат выходных данных

Одно число - количество секторов, которое возможно посетить. Если невозможно определить, то вывести число секторов посещённых первым роботом.

Примеры

Примеры входных данных и ответов к ним можно найти по `http://bit.ly/2LK5FrH`данной сслыке.

Решение

По условиям задачи нам известно количество роботов на поле, размер лабиринта по горизонтали и вертикали (в секторах), стартовые сектора каждого робота и направление каждого робота (рис. 5.41).

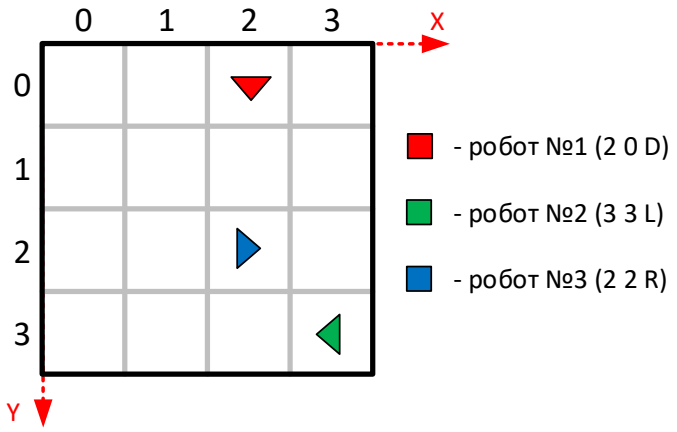


Рис. 5.41: Расположение роботов на старте

Представим лабиринт в виде неориентированного графа, где вершины графа - сектора лабиринта. В этом случае решение задачи заключается в составлении матрицы смежности графа, т.е. возможности проезда из любого сектора в смежные сектора. На старте матрица смежности "пустая". Нумерация вершин - абсолютная, начиная с "0"(рис. 5.42).

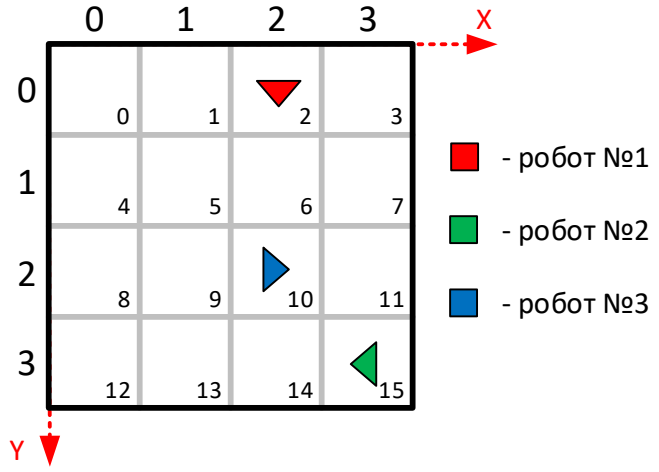


Рис. 5.42: Нумерация вершин (номер клетки = номер вершины графа)

Формула получения номера вершины из текущих координат робота (x, y):

$$cell\_number = y \cdot N_{cols} + x, \text{ где } N_{cols} \text{ - количество столбцов поля}$$

Например, чтобы получить номер стартовой клетки робота №1:

$$cell\_number = 0 \cdot 4 + 2 = 2$$

Для дискретной одометрии (информация о координатах робота и его направлении в текущий момент времени) для каждого робота нужно постоянно отслеживать его координаты секторов (x, y) и азимут (направление движения).

Далее по командам перемещения и показаниям датчиков для каждого робота заполняем матрицу смежности. По условиям задачи роботы перемещаются одновременно, т.е. за одну итерацию каждый робот совершает 1 действие (проезд прямо или

поворот).

Разберем на примере 3 первые команды для каждого робота (таблица 5.4).

	Робот #1	Робот #2	Робот #3
Строка команд #1	F 1 1 0	F 1 0 1	F 0 1 1
Строка команд #2	R 1 0 0	F 1 1 0	L 0 0 1
Строка команд #3	F 0 0 1	R 1 0 0	F 1 0 1

Таблица 5.4: Примеры команд для перемещений роботов

Команда "F 1 1 0" обозначает, что робот проезжает прямо 1 сектор ("F"). Показания датчиков после остановки: слева - стена ("1"), впереди - стена("1"), справа - свободно ("0") и т.п.

Состояние лабиринта и положение роботов после выполнения 1 строки команд показаны на рис. 5.43.

Рис. 5.43: Состояние лабиринта и положения роботов после 1й итерации

Робот #1, пример заполнения матрицы смежности.
Начальный сектор движения - 2, после проезда прямо (робот направлен вниз), робот оказывается в секторе 6. По показаниям датчиков корректируем матрицу смежности:

- Из сектора 2 можно проехать в сектор 6 $\Rightarrow$ вершины 2 и 6 смежные
- Показания датчика слева = '1', т.е. между секторами 6 и 7 стена $\Rightarrow$ вершины 6 и 7 не смежные.
- Показания датчика спереди = '1', т.е. между секторами 6 и 10 стена $\Rightarrow$ вершины 6 и 10 не смежные.
- Показания датчика справа = '0', т.е. из сектора 6 можно проехать в сектор 5 $\Rightarrow$ вершины 6 и 5 смежные.

Подобным образом корректируем матрицу смежности после проезда каждого робота. Также необходимо учитывать посещенные сектора и/или проверенные дальномером.

Лабиринт и роботы после выполнения 2 строки команд:

Рис. 5.44: Состояние лабиринта и положения роботов после 2й итерации

Лабиринт и роботы после выполнения 3 строки команд:

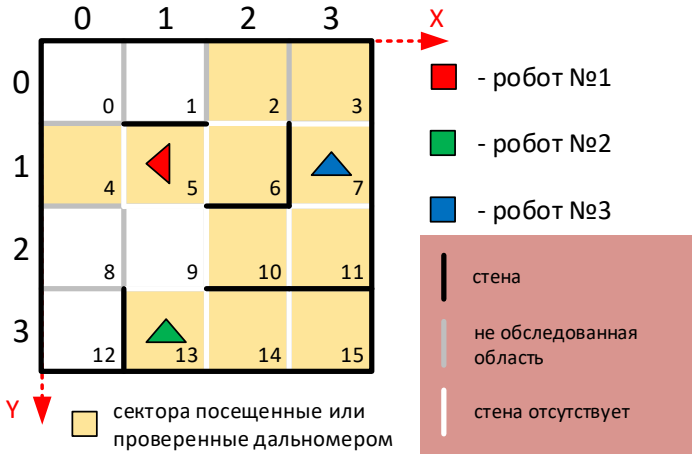


Рис. 5.45: Состояние лабиринта и положения роботов после 3й итерации

Лабиринт и роботы после выполнения всех команд (рис. 5.46).

После выполнения всех команд по перемещению роботов проверяем карту лабиринта на предмет наличия не посещенных и не проверенных дальномером секторов без явно обнаруженных стен, т.к. в этом случае мы не знаем точно - можно ли из этого сектора перемещаться в смежные сектора или нельзя, поэтому:

- Если такие сектора есть, то ответом будет количество секторов, которые проехал Робот #1
- Если таких секторов нет, то через алгоритм BFS проверяем количество секторов, которые сможет посетить Робот #1 начиная со стартовой позиции. В ответ выводим полученное количество секторов. В нашем примере это 10.

Ответ: 10 секторов

Пример программы-решения

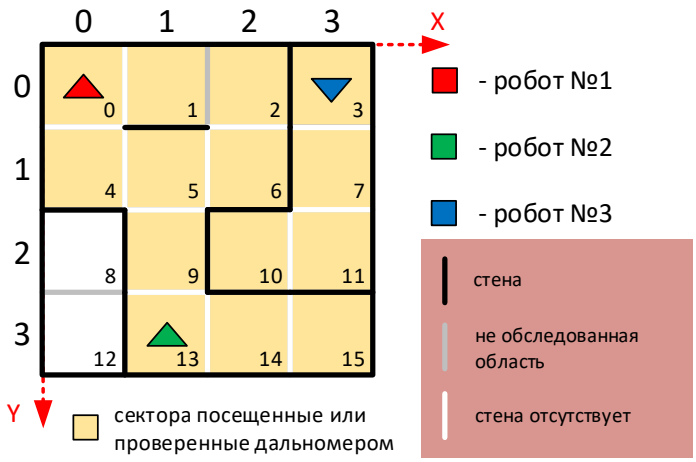


Рис. 5.46: Состояние лабиринта и положения роботов после выполнения всех итераций

Ниже представлено решение на языке Python3

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

def write_matrix(cell1, cell2, value):
    global matrix
    matrix[cell1][cell2] = int(not value)
    matrix[cell2][cell1] = int(not value)

def edit_matrix(m):
    global LMooved

    for rr in range(N):
        x, y, az = robots_moves[rr]
        cells = [0, 0, 0]                       # sensors: [left, forward, right]
        if az == 0:
            cells[0] = coord2cell(x - 1, y)
            cells[1] = coord2cell(x, y - 1)
            cells[2] = coord2cell(x + 1, y)
        elif az == 1:
            cells[0] = coord2cell(x, y - 1)
            cells[1] = coord2cell(x + 1, y)
            cells[2] = coord2cell(x, y + 1)
        elif az == 2:
            cells[0] = coord2cell(x + 1, y)
            cells[1] = coord2cell(x, y + 1)
            cells[2] = coord2cell(x - 1, y)
        elif az == 3:
            cells[0] = coord2cell(x, y + 1)
            cells[1] = coord2cell(x - 1, y)
            cells[2] = coord2cell(x, y - 1)

        robot_obstacle = [False] * 3
        for b in range(N):                                  # проверяем каждого робота
            for ss in range(3):                             # на пересечение с каждым дальномером
                if coord2cell(robots_moves[b][0], robots_moves[b][1]) == cells[ss]:
                    robot_obstacle[ss] = True

        cell = coord2cell(x, y)
```

```
39          for s in range(3):                                      # 3 sensors
40              if cells[s] >= K * M or cells[s] < 0:
41                  continue
42
43              if not robot_obstacle[s]:
44                  write_matrix(cell, cells[s], measures[rr][m][s + 1])
45                  write_matrix(cells[s], cell, measures[rr][m][s + 1])
46
47              if not robot_obstacle[s] and not measures[rr][m][s + 1]:
48                  LMooved[cells[s]] = True
49
50  def cell2coord(cell):
51      y = int(cell / K)
52      x = cell - y * K
53      return (x, y)
54
55  def coord2cell(x, y):
56      # K - ширина лабиринта, секторов
57      return y * K + x
58
59  def bfs():
60      start = coord2cell(robots[0][0], robots[0][1])
61      visited = [False for i in range(K * M)]
62
63      path = []
64      queue = [start]
65      while len(queue) > 0:
66          p = queue.pop(0)
67          if not visited[p]:
68              visited[p] = True
69              path.append(p)
70
71              for i in range(K * M):
72                  if not visited[i] and matrix[p][i] > 0:
73                      queue.append(i)
74
75              x, y = cell2coord(p)
76              if x + 1 < K:
77                  if matrix[p][p + 1] < 0 and not LMooved[p + 1]:
78                      return []
79
80              if x - 1 >= 0:
81                  if matrix[p][p - 1] < 0 and not LMooved[p - 1]:
82                      return []
83
84              if y + 1 < M:
85                  if matrix[p][p + K] < 0 and not LMooved[p + K]:
86                      return []
87
88              if y - 1 >= 0:
89                  if matrix[p][p - K] < 0 and not LMooved[p - K]:
90                      return []
91
92      return path
93
94  data = sys.stdin.readlines()
95  N, K, M, cnt_sens = list(map(int, data[0].strip().split(' ')))
96  data.pop(0)
97
98  #N         - кол-во роботов на поле
```

```
#K         - ширина лабиринта, секторов
#M         - длина лабиринта, секторов
#cnt_sens - кол-во показаний датчиков

measures, robots = [], []
az = ['U','R','D','L']
for i in range(N):
    line = data[0].strip().split(' ')
    robots.append([int(line[0]), int(line[1]), az.index(line[2])])
    data.pop(0)

for move in range(N):
    measures.append([])
    for m in range(cnt_sens):
        line = data[move * cnt_sens + m].strip().split(' ')
        measures[-1].append([line[0]])
        for i in range(3):
            measures[-1][-1].append(int(line[i + 1]))

matrix = [[-1 for j in range(K*M)] for i in range(K*M)]      # пустая матрица смежности
move_cells = [1 for i in range(N)]                           # подсчет посещенных клеток роботами
LMooved = [False for i in range(K * M)]                                  # просмотренные или посещ

robots_moves = []
for i in range(N):
    robots_moves.append(robots[i])                           # стартовые позиции роботов

for m in range(cnt_sens):
    for r in range(N):
        x, y, azimut = robots_moves[r]
        LMooved[coord2cell(x, y)] = True

        action = measures[r][m][0]
        if action == 'L':   azimut -= 1
        elif action == 'R': azimut += 1
        azimut %= 4

        if action == 'F':
            x0, y0 = x, y
            if   azimut == 0: y -= 1
            elif azimut == 1: x += 1
            elif azimut == 2: y += 1
            elif azimut == 3: x -= 1
            move_cells[r] += 1

            write_matrix(coord2cell(x,y),coord2cell(x0,y0), 0)

        robots_moves[r] = [x, y, azimut]

    edit_matrix(m)

result = bfs()
if len(result) == 0 :
    print (move_cells[0])
else:
    print (len(result))
```

Задача 5.1.9. Планирование движения в лабиринте (15 баллов)

Роботы в количестве N, собранные по дифференциальной схеме, движутся по заранее известному лабиринту и представленному на рис. 5.47.

Необходимо доехать из начального в конечный сектор каждым роботом, если они движутся одновременно и параллельно и имеют следующие команды:

- F — проезд робота в следующий по ходу движения сектор;
- L — поворот робота в данном секторе налево и проезд в следующий по ходу движения сектор;
- R — поворот робота в данном секторе направо и проезд в следующий по ходу движения сектор.

Считать что роботы поворачиваются мгновенно, а после одновременно начинают движение вперёд с одинаковой скоростью.Робот является цилиндром и занимает 2/3 площади клетки. Каждое перемещение он начинает и заканчивает в центре клетки.

Также гарантируется, что данных команд будет достаточно для выполнения задачи. При движении столкновение не допускается и перемещение необходимо осуществлять так, чтобы роботы получили наименьшее число команд. Иначе говоря, необходимо, чтобы сумма длин строк, содержащих команды передаваемые на робота в хронологическом порядке, без пробелов и запятых, была минимально возможной.

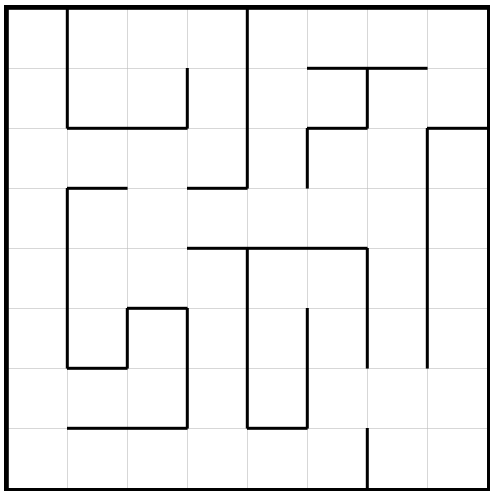


Рис. 5.47: Структура лабиринта для решения задачи

Формат входных данных

Первая строка содержит 1 целое число: N — количество роботов на поле ($1 \leqslant N \leqslant 3$).

Далее идет N строк, содержащие координаты старта и финиша каждого робота, а также направление робота при старте, т.е. одна строка имеет следующую структуру: $x_s,\ y_s, dir,\ x_f,\ y_f$, где:

- x_s — координаты старта данного робота по оси X;
- y_s — координаты старта данного робота по оси Y;
- dir — направление робота при старте:
  - U – робот направлен вверх;
  - L – робот направлен влево;
  - D – робот направлен вниз;
  - R – робот направлен вправо;
- x_f — координаты финиша данного робота по оси X;
- y_f — координаты финиша данного робота по оси Y;

Все данные указаны через пробел, числа являются целыми.

Формат выходных данных

N строк, каждая строка содержит команды, передаваемые на данного робота в хронологическом порядке (самая первая команда расположена в начале строки), без пробелов и запятых.

Комментарии

Дополнительные наборы входных данных доступны по `http://bit.ly/2KtN4z9`ссылке.

Примеры

Пример №1

Стандартный ввод

`2`
`6 7 U 0 2`
`7 7 U 1 2`

Стандартный вывод
`FLLRFFFFRFFFF` `FLRFFFLLRFFRL`

Пример №2

Стандартный ввод

`2`
`0 0 D 2 5`
`0 1 R 2 1`

Стандартный вывод
`FFFFFFLFL` `RFFFFFLFFLFFLRFRLFLL`

Решение

Одним из вариантов решения данной задачи является составление графа, где вершины – состояние позиций роботов внутри лабиринта, а ребра - действия всех роботов, приведшие к переходу в это состояние.

Например, если по лабиринту перемещаются три робота, то ребро "FRF"определяет, что следующее состояние получается посредством перемещения первого робота прямо, второго - поворотом направо, третьего - также проездом прямо. Необходимо помнить, что согласно условиям задачи, поворот направо/налево - это поворот в текущем секторе направо/налево и проезд прямо.

При построении новых ребер необходимо учитывать, что роботы не должны сталкиваться. Также возможны такие ребра, где один из роботов стоит - т.е. в одном из предыдущих состояний он приехал в точку финиша.

На рис. 5.48 изображен пример графа для очень простого лабиринта и двух роботов. Очевидно, в таком графе можно применить поиск в ширину для нахождения вершины, когда все роботы находятся в требуемом секторе. Последовательность ребер из базового состояния в финишное состояние - будет кодировать путь каждого робота.

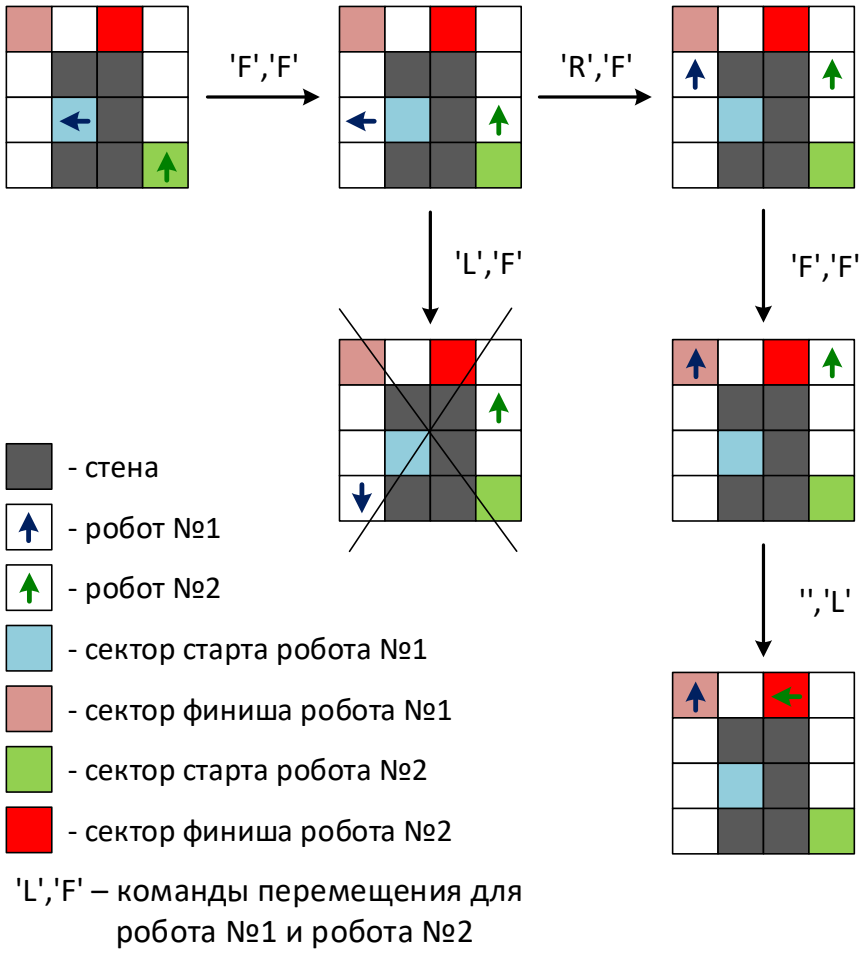


Рис. 5.48: Граф перемещения роботов

Пример программы-решения

Ниже представлено решение на языке C++

```cpp
// Состояние робота - это пара чисел cell и dir робота
// Количество различных состояний робота для лабиринта размером AxB:
// K_robot = 4*A*B
// Состояние лабиринта - это упорядоченный набор состояний роботов
// Количество различных состояний робота для N роботов:
// K_maze = K_robot^N
// С ограничениями по времени и памяти программа справится с N <= 3

#include <iostream>
#include <vector>
#include <string>
#include <queue>
#include <algorithm>
#include <ctime>

using namespace std;

// Выводить ли отладочную информацию?
const bool debug = false;

// Размеры лабиринта
const int height = 8;
const int width = 8;

// Лабиринт
// adj[i] содержит массив вида {U, R, D, L}, где:
// U - номер клетки сверху от i-ой клетки
// R - номер клетки справа от i-ой клетки
// D - номер клетки снизу от i-ой клетки
// L - номер клетки слева от i-ой клетки
// Если с одной из сторон стена, то номер клетки = -1
const int adj[height * width][4] = {{-1, -1, 8, -1}, {-1, 2, 9, -1}, {-1, 3, 10, 1}, {-1, -1, 11, 2},
// Количество роботов на поле
int robot_count;

// Номер клетки и направление старта роботов
vector <int> cell_start, dir_start;

// Номер клетки и направления финиша роботов
vector <int> cell_finish;

// Количество возможных состояний лабиринта
int state_count = 0;

// used[i] - просмотрено i-ое состояние лабиринта или нет
vector <bool> used;
// used[i] - из какого состояния получили i-ое состояние лабиринта
vector <int> from;
// actions[i] - какое действия совершили роботы, чтобы перейти в i-ое состояние лабиринта
// вместо
vector <char*> actions;

// path[i] - путь для i-ого робота
vector <string> path;

// Состояние лабиринта в которое перешли роботы, когда оказались на точках финиша
int state_finish = -1;

```

```
// Переводит символьное представление направления в числовое
// 'U' -> 0
// 'R' -> 1
// 'D' -> 2
// 'L' -> 3
int dir_to_int(char c) {
    if (c == 'U')
        return 0;
    else if (c == 'R')
        return 1;
    else if (c == 'D')
        return 2;
    else if (c == 'L')
        return 3;
    else {
        cerr << "Invalid direction!: " << c << "\n";
        throw(1);
    }
}

// Переводит вектор состоний робота в числовое представление
int state_to_int(vector <pair <int, int>> state) {
    int res = 0;
    for (auto el : state) {
        res *= (4 * height * width);
        res += el.first * 4 + el.second;
    }
    return res;
}

// Переводит числовое представление состояния лабиринта в вектор состояний робота
vector <pair <int, int>> state_to_vector(int state) {
    vector <pair <int, int>> res;
    while (res.size() != robot_count) {
        res.emplace_back(state % (4 * height * width) / 4, state % (4 * height * width) % 4);
        state /= (4 * height * width);
    }
    reverse(res.begin(), res.end());
    return res;
}

// Возвращает следующую комбинация для строки (например FFF -> FFL; FFR -> FLF; RRR -> END)
// (S - бездействовать)
string next_combination(string s) {
    int i = s.size() - 1;
    bool cont = true;
    while (cont && i >= 0) {
        cont = false;
        if (s[i] == 'S')
            s[i] = 'F';
        else if (s[i] == 'F')
            s[i] = 'L';
        else if (s[i] == 'L')
            s[i] = 'R';
        else if (s[i] == 'R') {
            s[i] = 'S';
            cont = true;
            --i;
        }
    }
```

```
119     if (i < 0)
120         return "END";
121     else
122         return s;
123 }
124 
125 // Считывание данных
126 void get_data() {
127     cin >> robot_count;
128     cell_start.resize(robot_count);
129     dir_start.resize(robot_count);
130     cell_finish.resize(robot_count);
131     state_count = 1;
132     for (int robot = 0; robot < robot_count; ++robot) {
133         int xs, ys, xf, yf;
134         char ds;
135         cin >> xs >> ys >> ds >> xf >> yf;
136         cell_start[robot] = xs + ys * width;
137         cell_finish[robot] = xf + yf * width;
138         dir_start[robot] = dir_to_int(ds);
139         state_count *= 4 * height * width;
140     }
141 }
142 
143 // Вывод входных данных
144 void print_data() {
145     cout << "--------------------\n";
146     for (int robot = 0; robot < robot_count; ++robot) {
147         cout << "Robot " << robot + 1 << endl;
148         cout << "start: " << cell_start[robot] << " " << dir_start[robot] << endl;
149         cout << "finish: " << cell_finish[robot] << endl;
150         cout << "--------------------\n";
151     }
152 };
153 
154 // Поиск в ширину по состояниям лабиринта
155 void bfs() {
156     if (debug)
157         cout << "bfs started\n";
158     used.resize(state_count, false);
159     from.resize(state_count, -1);
160     actions.resize(state_count);
161     vector <pair <int, int>> cur;
162     for (int robot = 0; robot < robot_count; ++robot)
163         cur.emplace_back(cell_start[robot], dir_start[robot]);
164     used[state_to_int(cur)] = true;
165     queue <int> q;
166     q.push(state_to_int(cur));
167     while (!q.empty()) {
168         int cur_st = q.front();
169         cur = state_to_vector(cur_st);
170         q.pop();
171         // Проверить, если все роботы на точках финиша
172         bool finish = true;
173         for (int robot = 0; robot < robot_count; ++robot)
174             if (cur[robot].first != cell_finish[robot]) {
175                 finish = false;
176                 break;
177             }
178         if (finish) {
```

```
            state_finish = state_to_int(cur);
            break;
        }
        // перебираем все строки действий вида FFF, FFL, FFR, ...
        string s = string(robot_count, 'S');
        while (s != "END") {
            auto next = cur;
            bool ok = true;
            for (int robot = 0; robot < robot_count; ++robot) {
                if (s[robot] == 'S') {
                    if (cur[robot].first == cell_finish[robot]) {
                        int next_dir = next[robot].second;
                        int next_cell = next[robot].first;
                    } else {
                        ok = false;
                        break;
                    }
                } else if (from[cur_st] == -1 || actions[cur_st][robot] != 'S') {
                    if (s[robot] == 'F'){
                        int next_dir = next[robot].second;
                        int next_cell = adj[next[robot].first][next_dir];
                        if (next_cell == -1) {
                            ok = false;
                            break;
                        }
                        next[robot].first = next_cell;
                        next[robot].second = next_dir;
                    } else if (s[robot] == 'L') {
                        int next_dir = (next[robot].second + 3) % 4;
                        int next_cell = adj[next[robot].first][next_dir];
                        if (next_cell == -1) {
                            ok = false;
                            break;
                        }
                        next[robot].first = next_cell;
                        next[robot].second = next_dir;
                    } else if (s[robot] == 'R') {
                        int next_dir = (next[robot].second + 1) % 4;
                        int next_cell = adj[next[robot].first][next_dir];
                        if (next_cell == -1) {
                            ok = false;
                            break;
                        }
                        next[robot].first = next_cell;
                        next[robot].second = next_dir;
                    }
                } else {
                    ok = false;
                    break;
                }
            }
            for (int i = 0; i < robot_count; ++i)
                for (int j = 0; j < robot_count; ++j) {
                    if (i != j && next[i].first == cur[j].first && next[i].second != next[j].second)
                        ok = false;
                        break;
                    }
                    if (i != j && next[i].first == next[j].first) {
                        ok = false;
                        break;
```

```
239                     }
240                 }
241             if (ok && !used[state_to_int(next)]) {
242                 used[state_to_int(next)] = true;
243                 int state_int = state_to_int(next);
244                 // convert string to C-string
245                 actions[state_int] = new char[robot_count + 1];
246                 for (int i = 0; i < robot_count; ++i)
247                     actions[state_int][i] = s[i];
248                 actions[state_int][robot_count] = '\n';
249                 from[state_to_int(next)] = state_to_int(cur);
250                 q.push(state_to_int(next));
251             }
252             s = next_combination(s);
253         }
254     }
255     if (debug)
256         cout << "bfs finished\n";
257 }
258 
259 void print_path() {
260     if (debug)
261         cout << state_finish << endl;
262     if (state_finish == -1) {
263         cout << "No solution!\n";
264     } else {
265         path.resize(robot_count, "");
266         int cur_state = state_finish;
267         while (from[cur_state] != -1) {
268             for (int robot = 0; robot < robot_count; ++robot)
269                 path[robot].push_back(actions[cur_state][robot]);
270             cur_state = from[cur_state];
271         }
272         for (int robot = 0; robot < robot_count; ++robot) {
273             reverse(path[robot].begin(), path[robot].end());
274             path[robot] += 'S';
275             path[robot] = path[robot].substr(0, path[robot].find('S'));
276             cout << path[robot] << endl;
277         }
278     }
279 }
280 
281 int main() {
282     get_data();
283     if (debug)
284         print_data();
285     unsigned int start_time = clock();
286     bfs();
287     print_path();
288     if (debug) {
289         unsigned int end_time = clock();
290         cout << "Execution time: " << (end_time - start_time) / 1000.0 << " s" << endl;
291     }
292     return 0;
293 }
```

Задача 5.1.10. Определение пересекающихся траекторий (10 баллов)

Имеется набор байт — трафик, собранный на концентраторе, во время общения в локальной сети нескольких устройств, включая робототехнические устройства.

Известно, что робототехнические устройства общались по протоколу, построенному поверх UDP (`https://ru.wikipedia.org/wiki/UDP`) и реализованному следующим образом: три целых числа – t_i, X_i, Y_i – каждое из которых занимает 4 байта, где t_i — время, в которое были сняты данные показатели координат, а X_i и Y_i — координаты местоположения робототехнической тележки в данный момент времени. При записи чисел использовалась нотация BigEndian (`https://en.wikipedia.org/wiki/Endianness`).

Необходимо определить IP-адреса (`https://ru.wikipedia.org/wiki/IPv4`) устройств, чьи траектории пересекались.

Считать, что между изменениями робототехнические тележки перемещались прямо. Гарантируется, что через данный концентратор проходит лишь необходимый трафик, т.е. отсутствуют пакеты, не относящиеся к данной задаче.

Формат входных данных

Первая строка содержит 1 целое число: N — количество переданых пакетов через концентратор ($4 \leqslant N \leqslant 10^3$).

Далее идут N строк, каждая из которых содержит один пакет, переданный через концентратор в побитовом формате, который содержит t_i, X_i, Y_i, где:

- t_i — время в мс, в которое были сняты данные показатели координат – 4 байта ($0 \leqslant t_i < 2^{32}$);
- X_i — координаты по оси X местоположения робототехнической тележки в данный момент времени — 4 байта ($0 \leqslant X_i < 2^{32}$);
- Y_i — координаты по оси Y местоположения робототехнической тележки в данный момент времени — 4 байта ($0 \leqslant Y_i < 2^{32}$).

Формат выходных данных

Необходимо вывести два IP-адресса в десятичном формате через пробел в порядке их возрастания – адреса устройств, чьи траектории пересекались.

В случае если пересекались несколько пар роботов, эти пары следуют в порядке первых пересечений траекторий робототехнических устройств.

В случае если таких пересечений нет, следует вывести -1.

Примеры

Примеры входных данных и ответов к ним можно найти по `http://bit.ly/2QkcNzv`данной сслыке.

Решение

Решение задачи начнем с изучения структуры пакета UDP.
По условиям задачи длина пакета составляет 32 байта в шестнадцатеричном формате. Каждый пакет содержит следующую информацию:

- IP-адрес отправителя - 4 байта
- IP-адрес получателя - 4 байта
- Время снятия показаний координат устройства - 4 байта
- Координата по оси X в данный момент времени - 4 байта
- Координата по оси Y в данный момент времени - 4 байта

Из описания структура протокола UDP (для IPv4) нам известно, что IP-адреса отправителя и получателя находятся в начале пакета и занимают по 4 байта каждый (таблица 5.49).

байт	0		1		2		3		4		5		6		7		8		9		10	
данные	**7**	**4**	**1**	**b**	**c**	**7**	**8**	**a**	**4**	**9**	**2**	**7**	**2**	**a**	**2**	**b**	**0**	**0**	1	1	0	0
	IP отправителя								IP получателя													

Таблица 5.5: IP-адреса отправителя и получателя в пакете UDP

Нас интересует только IP-адрес отправителя. В нашем примере это $[74.1b.c7.8a]_{16} \Rightarrow [116.27.199.138]_{10}$

В середине пакета UDP находятся данные, которые нам не нужны в рамках нашей задачи.

Время снятия показаний t, координаты X и Y находятся в конце пакета UDP (таблица 5.6).

байт	20		21		22		23		24		25		26		27		28		29		30		31	
данные	**0**	**0**	**0**	**0**	**0**	**4**	**e**	**b**	**0**	**0**	**0**	**0**	**0**	**3**	**e**	**7**	**0**	**0**	**0**	**0**	**0**	**5**	**2**	**9**
	время								координата X								координата Y							

Таблица 5.6: Время, координаты X и Y робота в пакете UDP

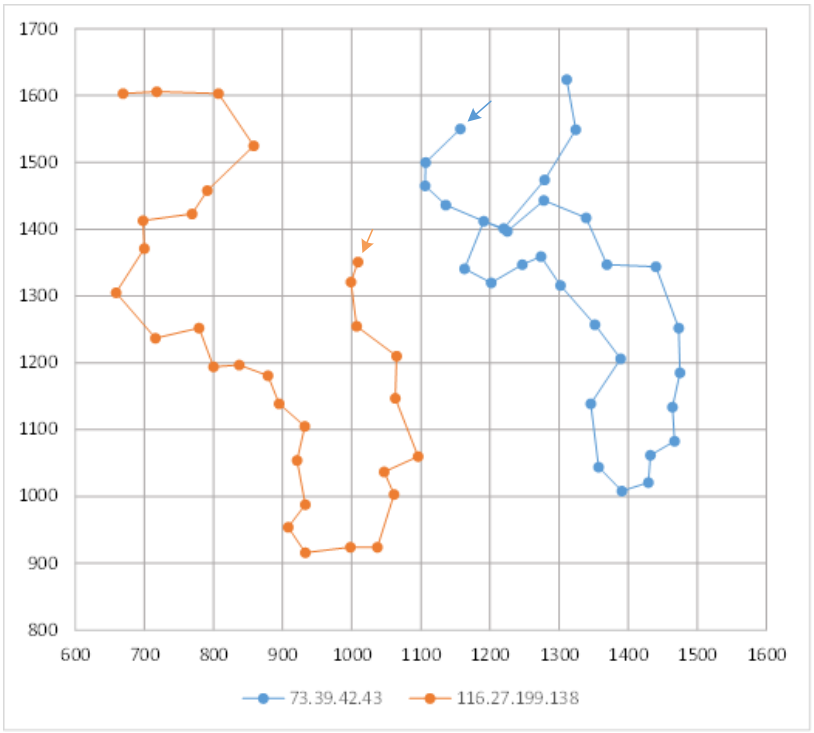
Обрабатываем все пакеты данных и отбираем в новый массив (список) IP-адрес отправителя, время, координату X, координату Y.

Далее по полученным данным проверяем пересечения с другими роботами. Для этого мы проверяем пересечение каждого отрезка пути текущего робота: $(x_i; y_i)$ и $(x_{i-1}y_{i-1})$ с каждым отрезком пути остальных роботов. Если отрезки пересекаются, значит и пути пересеклись.

Если пересечений не обнаружено, то в качестве ответа выводим -1.

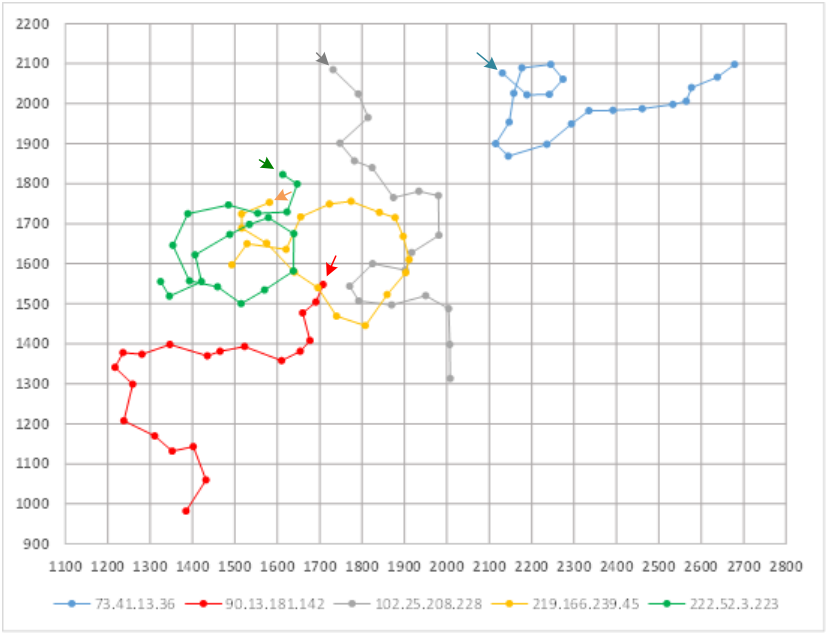
Если же пути пересекались, то выводим пару IP-адресов роботов в десятичном формате в порядке возрастания. В случае, если пересекались несколько пар роботов, то эти пары выводим в порядке первых пересечений траекторий роботов.

В случае с траекториями движения роботов, как на рис. 5.49 ответ: -1



Стрелками указаны начальные точки движения роботов

Рис. 5.49: Пример не пересекающихся траекторий роботов



Стрелками указаны начальные точки движения роботов

Рис. 5.50: Пример пересекающихся траекторий роботов

В случае с траекториями движения роботов, как на рис. 5.50 ответ:

```
219.166.239.45 222.52.3.223
90.13.181.142 219.166.239.45
102.25.208.228 219.166.239.45
```

Пример программы-решения

Ниже представлено решение на языке Python3

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

def hex2ip(hex_num):
    #формат входного IP-адреса:     FFFFFFFF
    #формат выходного IP-адреса:    255.255.255.255
    ip = []
    for i in range(0, len(hex_num), 2):
        ip.append(int(hex_num[i: i + 2], 16))
    return '.'.join(map(str, ip))

def cross(coords):
    x1, y1, x2, y2, x3, y3, x4, y4 = coords
    v2 = (x2 - x1) * (y3 - y1) - (y2 - y1) * (x3 - x1)
    v3 = (x2 - x1) * (y4 - y1) - (y2 - y1) * (x4 - x1)
    v0 = (x4 - x3) * (y1 - y3) - (y4 - y3) * (x1 - x3)
    v1 = (x4 - x3) * (y2 - y3) - (y4 - y3) * (x2 - x3)
    return ((v2 < 0 and v3 > 0) or (v2 > 0 and v3 < 0)) and \
           ((v0 < 0 and v1 > 0) or (v0 > 0 and v1 < 0))

data = sys.stdin.readlines()

packets = int(data[0].strip())      # кол - во переданных пакетов
times = []                          # время в мс, в которое были сняты координаты
coord_x = []                         # координаты оси X положения робота в момент times[]
coord_y = []                         # координаты оси Y положения робота в момент times[]
ip_sender = []                       # IP отправителя

for i in range(packets):
    ip_sender.append(data[i+1].strip()[0:8])
    times.append( int(data[i+1].strip()[40: 48], 16))
    coord_x.append(int(data[i+1].strip()[48: 56], 16))
    coord_y.append(int(data[i+1].strip()[56: 64], 16))

data0 = []
data_by_ip = {}
for i in range(packets):
    ip_out, t, x, y = ip_sender[i], times[i], coord_x[i], coord_y[i]
    data0.append((t, ip_out, x, y))

data = sorted(data0)

for i in range(0, len(data)):
    if data[i][1] not in data_by_ip:
        data_by_ip[data[i][1]] = [data[i]]
    else:
        data_by_ip[data[i][1]].append(data[i])

crosses = []
for key1 in data_by_ip:
    for key2 in data_by_ip:
        if key1 == key2:
            break
        for k1 in range(1, len(data_by_ip[key1])):
            x1y1 = data_by_ip[key1][k1 - 1][2:]
            x2y2 = data_by_ip[key1][k1][2:]
```

```
57
58              for k2 in range(1, len(data_by_ip[key2])):
59                  x3y3 = data_by_ip[key2][k2 - 1][2:]
60                  x4y4 = data_by_ip[key2][k2][2:]
61
62                  if cross(x1y1 + x2y2 + x3y3 + x4y4):
63                      keys = sorted((key1, key2))
64                      if keys not in crosses:
65                          crosses.insert(0, keys)
66
67  if len(crosses) == 0:
68      print (-1)
69  else:
70      for i in range(len(crosses)):
71          print (*([hex2ip(crosses[i][0]), hex2ip(crosses[i][1])]))
```

Задача 5.1.11. Определение собственных координат (10 баллов)

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

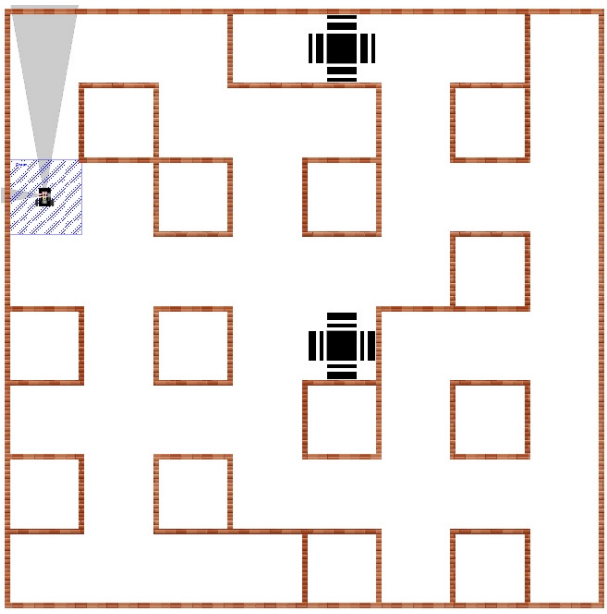


Рис. 5.51: Пример начального расположения робота

Для решения задачи робот запускается на поле, состоящим из 8x8 квадратных секторов. На поле между некоторыми секторами установлены препятствия для ограничения перемещения робота.

Роботу необходимо проехать некоторое количество секторов, заданное через входной файл *input.txt*, по правилу "левой руки", после остановиться и вывести на экран относительные координаты, т.е. свои координаты относительно точки старта, в формате (X, Y), где начало отсчета - точка старта, направление оси Y совпадает с первоначальным направлением движения, ось X направлена вправо перпендикулярно оси Y.

Конфигурация робота

Подключение моторов:

- Левый мотор - порт M3;
- Правый мотор - порт M4.

Подключение датчиков:

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

Формат входных данных

Входной файл содержит только одну строчку. В строке - целое число N ($5 \leqslant N \leqslant 40$), определяющее количество секторов, которое необходимо проехать. Сектор старта не учитывается в подсчёте.

Ограничения

Робот не должен выполнять задание дольше 3 минут.

Примеры

Для лабиринта, представленного на рис. 5.51, и при числе 7 во входных данных, робот должен остановиться в соответствии с рис. 5.52 и вывести на экран $(4, 1)$.

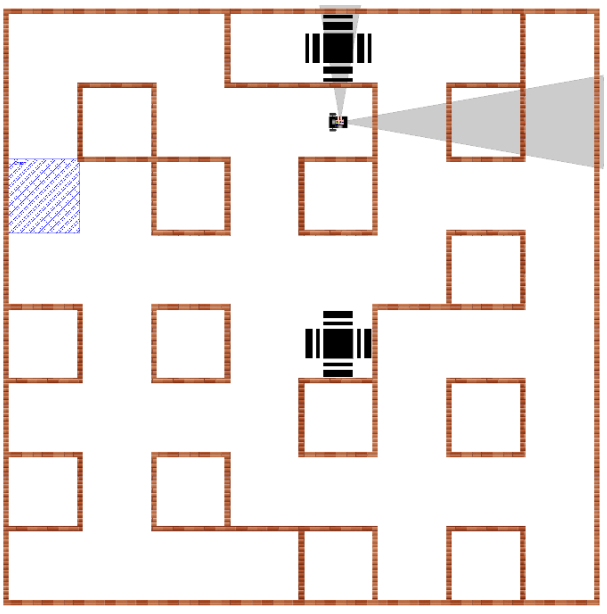


Рис. 5.52: Расположение робота, стартовавшего из позиции на рис. 5.51 и проехавшего 7 секторов

Решение

Для решения задачи в симуляторе TRIK-Studio определим следующую конфигурацию моторов и датчиков:

Оборудование	**Порт**
Правый мотор	M4
Левый мотор	M3
Датчик расстояния, направленный вперед	A1
Датчик расстояния, направленный влево	A2

Для перемещения в лабиринте нам необходимо подготовить функции:

- Проезд робота вперед на заданное расстояние
- Разворот робота на заданный угол (по энкодерам или гироскопу)
- Динамическое вычисление координат текущего сектора лабиринта (X; Y) и направления робота (азимут)
- Движение робота в лабиринте по правилу "левой руки"

Алгоритмы "прямолинейного движения на заданное расстояние" и "разворот робота на месте" являются базовыми для начальной робототехники и подробно рассматривать мы их не будем.

Алгоритм "левой руки": Для движения по лабиринту по правилу "левой руки" на роботе установлено 2 датчика расстояния: первый - спереди, второй – слева по ходу движения.

Псевдокод правила "левой руки":

1. *Если слева пусто (датчик возвращает расстояние больше, чем длина одного сегмента лабиринта), то:*
   (a) Поворот налево на 90 градусов
   (b) Проезд прямо на 1 сегмент
2. *иначе:*
   - если спереди пусто (датчик возвращает расстояние больше, чем длина одного сегмента), то:

   проезд прямо на 1 сегмент иначе: поворот направо на 90 градусов

Дискретная одометрия: Для вычисления азимута будем использовать следующее кодирование сторон света (см. рис.5.53)

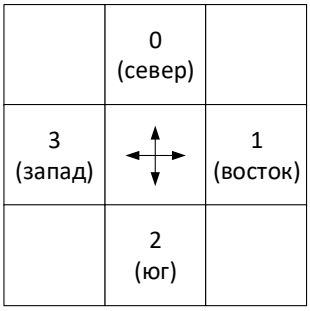


Рис. 5.53: Определение числовых вариантов сторон света

Вполне очевидно, что азимут робота меняется только при поворотах, а координаты сектора меняются только при перемещении между ними, но изменение координат зависит от азимута.

Азимут			
0	1	2	3
X-1	Y+1	X+1	Y-1

Таблица 5.7: Пересчет координат робота (при движении ”вперед”), в зависимости от текущего азимута

По условиям задачи робот в начальный момент времени установлен в направлении ”восток”. Полные координаты робота в момент старта $[x, y, azimut]$ показаны на рис. 5.54 :

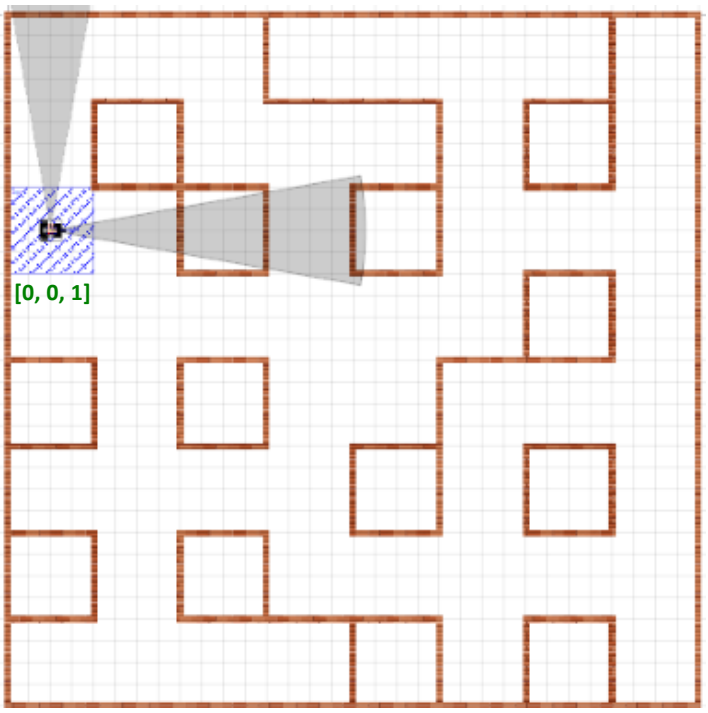


Рис. 5.54: Расположение робота на старте и начальные координаты робота [x, y, азимут]

Далее, согласно алгоритму ”левой руки”, робот проверяет возможность перемещения влево. В нашем случае - ”пусто”, т.е. робот поворачивает налево и проезжает вперед 1 сектор. При повороте налево меняется азимут с 1 на 0. При проезде прямо меняются координаты $(X; Y)$ в зависимости от азимута. Не забываем, что по условиям задачи робот на старте направлен вдоль оси Y, а ось X направлена вправо от робота (см. табл.5.7)

Подсчет посещенных секций делаем только при перемещении между ними (при поворотах секции не меняются).

Когда количество посещенных секций будет равно заданному значению - останавливаем робота и выводим на экран координаты робота. Обращайте внимание, в какое место экрана и в каком виде должен выводиться ответ.

Пример выполнения задания с количеством секций $= 17$ показан на рис.5.55.

Для нашего примера робот остановится в координатах $(-3; 2)$ с азимутом 3 (запад).

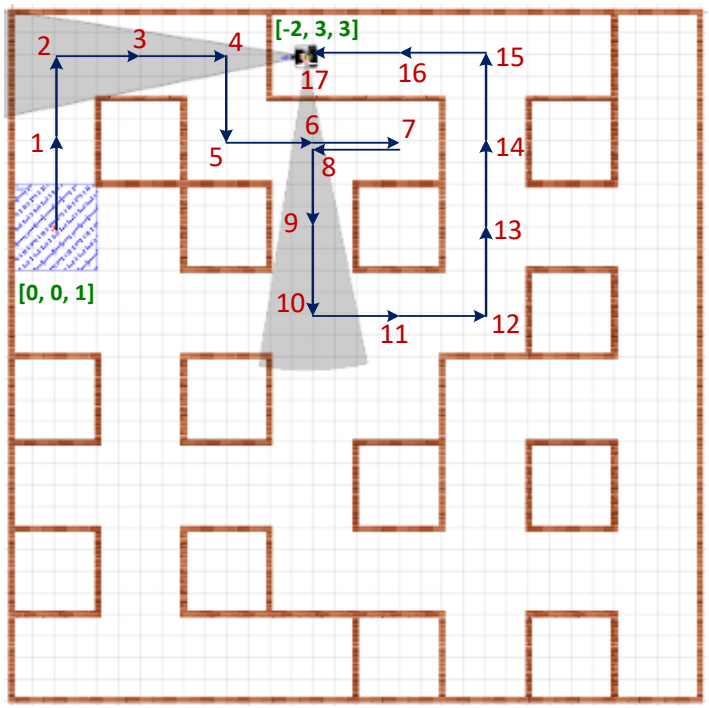


Рис. 5.55: Перемещения робота по правилу ”левой руки” с номерами посещенных секторов

Ответ выводим на экран TRIK’а: **(-2, 3)**

Пример программы-решения

Ниже представлено решение на языке JavaScriot

```
function sign(num) { return num >= 0 ? 1 : -1}
function motors(mL, mR){ mR = mR || mL; brick.motor('M4').setPower(mL); brick.motor('M3').setPower(mR)}

function move(cm, turn){
        var L = (cm / (Math.PI * robot.D)) * robot.cpr
        L += encL()
        var sgn = sign(cm)
        motors(100 * sgn)
        if (sgn == 1){
                while (encL() <= L) script.wait(10)
        } else {
                while (encL() >= L) script.wait(10)
        }

        if (turn == undefined){
                step += 1
                switch (robot.a){
                        case 0: robot.x--; break;
                        case 1: robot.y++; break;
                        case 2:         robot.x++; break;
                        case 3:         robot.y--; break;
                }
        }
        motors(0)
}
```

```
26
27  robot = {
28          D: 5.6,
29          track: 17.5,
30          x: 0,
31          y: 0,
32          a: 1,
33          cpr: 360
34  }
35
36  cellLength = 17.5 * 4
37
38  sensF = brick.sensor('A1').read
39  sensL = brick.sensor('A2').read
40  encL  = brick.encoder('E4').read
41  encR  = brick.encoder('E3').read
42
43  wait = script.wait
44
45  steps = script.readAll('input.txt')
46  step = 0
47
48  function turn (angle){
49          var sgn = sign(angle)
50          move(17.5 * 0.5, true)
51          var lAngle = ((robot.track * angle) / (robot.D * 360)) * robot.cpr
52          lAngle += encL()
53
54          motors(50 * sgn, -50 * sgn)
55
56          if (sgn == 1){
57                  while (encL() <= lAngle) wait(10)
58                  robot.a++
59          } else {
60                  while (encL() >= lAngle) wait(10)
61                  robot.a--
62          }
63
64          if (robot.a > 3) robot.a = 0
65          if (robot.a < 0) robot.a = 3
66
67          motors(0)
68          move(17.5 * -0.5, true)
69  }
70
71
72  while (true){
73          if (sensL() > cellLength){
74                  turn(-90)
75                  move(cellLength)
76          } else {
77                  if (sensF() > cellLength)
78                          move(cellLength)
79                  else
80                          turn (90)
81          }
82          script.wait(100)
83          if (step == steps)
84                  break
85  }
```

```
86  motors(0)
87
88
89  // test
90
91  a = 1
92
93
94  out = '(' + robot.x + ',' + robot.y + ')'
95  brick.display().addLabel(out, 1, 1)
96  brick.display().redraw()
```

3. ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП

Предметный тур

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

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

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

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

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

6.1. Математика. 9 класс

Задача 6.1.1. (20 баллов)

У правильного десятиугольника отметили все вершины и еще 2019 точек внутри. Некоторые из 2029 отмеченных точек соединили отрезками так, что исходный десятиугольник оказался разбит на треугольники и каждая точка является вершиной хотя бы одного треугольника, причем любые два треугольника либо не имеют общих точек, либо имеют общую вершину, либо имеют общую сторону. (Такое разбиение фигур называется **триангуляцией**.)

а) Сколько треугольников получилось?

б) Докажите, что количество треугольников не зависит от способа триангуляции.

Решение

а) Можно последовательно ставить точки внутри многоугольника, соединять со всеми вершинами этого многоугольника и считать сколько треугольников

добавилось. Изначально треугольников не было. Поставим точку и соединим со всеми вершинами. Появятся 10 треугольников. Потом ставим следующую точку внутрь какого-то треугольника. Один треугольник вычитается и добавляется три, т.е. плюс два треугольника. Всего получается $10 + 2018 \cdot 2 = 4046$.

б) Пусть при некоторой триангуляции получились n треугольников. Сосчитаем сумму всех углов всех треугольников. При каждой точке внутри десятиугольника сумма углов равна $360°$, а сумма углов в вершинах равна сумме внутренних углов десятиугольника. Получаем: $360° \cdot 2019 + 180° \cdot (10 - 2) = 180° \cdot n$. Откуда $n = 2 \cdot 2019 + 8 = 4046$.

Задача 6.1.2. (30 баллов)

Пусть переменные x, y, z принимают только натуральные значения. Докажите, что уравнение

$$x^2 + y^2 = z^3$$

а) имеет решение;

б) имеет бесконечное количество решений;

в) имеет бесконечное количество решений, в которых z — нечетное число.

Решение

а) $x = y = z = 2$ является решением данного уравнения, так как $2^2 + 2^2 = 2^3$.

б) Тройки чисел $(2n^3; 2n^3; 2n^2)$ для любого натурального n являются решениями:

$$(2n^3)^2 + (2n^3)^2 = 2 \cdot 2^2 n^{3 \cdot 2} = (2n^2)^3.$$

в) Найдем одно такое решение, например, $(5; 10; 5)$ или $(11; 2; 5)$ и воспользуемся приемом из пункта б). Тройка

$$(5n^3; 10n^3; 5n^2)$$

при нечетных n будет решением исходного уравнения, в котором $z = 5n^2$ нечетное число.

Критерии оценки

Только за пункт а) — 10 баллов, за пункт б) — 20 баллов (включает пункт а)), за пункт в) — 30 баллов (включает первые два пункта).

Задача 6.1.3. (50 баллов)

На одном из полей 8×8 находится робот-разведчик, который каждую секунду перемещается в соседнее поле. Охранная система каждую секунду может проверить любые n полей, есть ли там робот. Может ли охранная система за одну минуту наверняка обнаружить шпиона, если

а) $n = 15$;

б) $n = 9$;

в) $n = 8$?

(Считается, что если у робота есть шанс не оказаться на проверяемом поле, то он не обнаружен.)

Решение

а) Приведем алгоритм проверки, для которого в некоторый момент робот окажется в проверямом поле. Пронумеруем поля как в таблице:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Сперва проверим все поля с 1 по 15. На следующей секунде проверим поля с 8 по 22, затем с 15 по 29. Каждый раз первые 7 проверяемых убираем и добавляем следующие 7. Таким образом добираемся до конца таблицы за 8 секунд. У шпиона нет возможности "проскочить" не оказавшись на проверяемом поле. Следовательно, охранная система в какой-то момент обнаружит шпиона не зависимо где он находился в начале и как передвигался.

б) Алгоритм примерно такой же, только вместо 7 полей передвигаемся на 1 поле: на первой секунде проверяем с 1 по 9, на второй со 2 по 10, затем с 3 по 11 и т.д. Шпион будет обнаружен за не более чем 56 секунд.

в) Раскрасим поля в шахматном порядке. Предположим, что шпион в первой секунде проверки оказался на черном поле. Тогда в нечетные секунды он будет в черных, а в четных секундах в белых полях. Начиная с угла пронумеруем диагонали в одном направлении с 1 по 15. Пусть угловая клетка (первая диагональ) будет черного цвета. Проверим 1 и 3 ряд клеток (черные) и еще какие-то. На второй секунде проверим 2-й и 4-й ряд (белые). Потом 5-й ряд, 6-й ряд, и т.д. по одному. Ряды 12 и 14 можем проверить вместе, так же как и ряд 13 с 15-м. Шпион не сможет пройти с непроверенных полей в проверенные, не оказавшись в проверяемом. Единственный момент, когда это можно сделать, только если предположение вначале неверно, т.е. на первой секунде шпион находился в белой клетке. Если охранная система проверит еще раз по тому же алгоритму с 16 по 30 секунду, то сможет поймать шпиона не более чем за 30 секунд.

Критерии оценки

Только за пункт а) — 10 баллов, за пункт б) — 25 баллов (включает пункт а)), за пункт в) — 50 баллов (включает первые два пункта).

Ответ: а) да; б) да; в) да.

6.2. Математика. 10-11 класс

Задача 6.2.1. (20 баллов)

Функция $f(x)$ задана следующим образом:

$$f(x) = \frac{x^2 \cdot (-1)^{-[x]} + x}{|x|},$$

где $[x]$ означает наибольшее целое число, не превосходящее числа x.

Найдите значение выражения:

$$\underbrace{f(f(\ldots f}_{2019}(20.19)\ldots)).$$

Решение

За исключением некоторых целых точек, $f(f(x)) = -x$. Поэтому

$$\underbrace{f(f(\ldots f}_{2018}(20.19)\ldots)) = -20.19.$$

Остается вычислить $f(-20.19) = -21.19$.

Ответ: -21.19

Задача 6.2.2. (40 баллов)

Пусть переменные x, y, z принимают только натуральные значения. Докажите, что уравнение

$$x^{2018} + y^{2018} = z^{2019}$$

а) имеет решение;

б) имеет бесконечное количество решений;

в) имеет бесконечное количество решений, в которых z — нечетное число.

Решение

а) $x = y = z = 2$ является решением данного уравнения, так как

$$2^{2018} + 2^{2018} = 2 \cdot 2^{2018} = 2^{2019}.$$

б) Тройки чисел $(2n^{2019}; 2n^{2019}; 2n^{2018})$ для любого натурального n являются решениями:

$$(2n^{2019})^{2018} + (2n^{2019})^{2018} = 2 \cdot 2^{2018} n^{2019 \cdot 2018} = (2n^{2018})^{2019}.$$

в) Воспользуемся тождеством:

$$a^n(a^n+b^n)^n + b^n(a^n+b^n)^n = (a^n+b^n)\cdot(a^n+b^n)^n = (a^n+b^n)^{n+1}.$$

Возьмем a любое нечетное число, b – четное и $n = 2018$. Тогда решениями уравнения являются все тройки чисел $x = a(a^{2018}+b^{2018})$, $y = b(a^{2018}+b^{2018})$, $z = a^{2018}+b^{2018}$. Нечетность z очевидна.

Критерии оценки

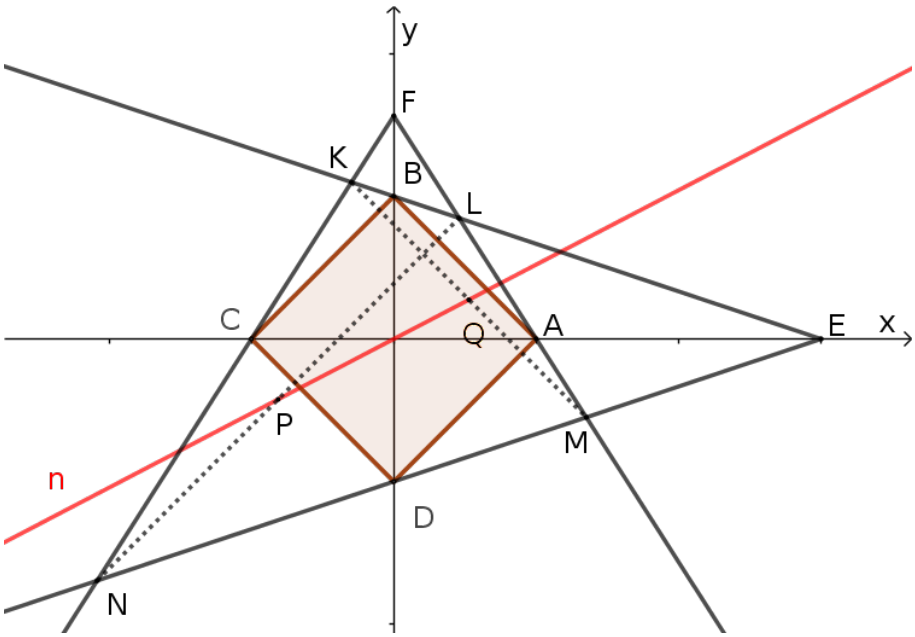
Только за пункт а) — 10 баллов, за пункт б) — 20 баллов (включает пункт а)), за пункт в) — 40 баллов (включает первые два пункта)

Задача 6.2.3. (40 баллов)

На продолжениях диагоналей AC и BD квадрата $ABCD$ отметили точки E и F соответственно. Точки пересечения лучей EB и ED с лучами FA и FC образовали выпуклый четерехугольник $KLMN$. Докажите, что прямая, проходящая через середины диагоналей получившегося четырехугольника $KLMN$, проходит через центр квадрата $ABCD$.

Решение

Введем систему координат так, как показано на рисунке (точка $A(1;0)$, точка $B(0;1)$):



Пусть точки E и F имееют координаты $(e;0)$ и $(0;f)$ соответственно. Запишем уравнения прямых

$$BE:\ \frac{x}{e}+y=1;$$

$$DE:\ \frac{x}{e}-y=1;$$

$$AF:\ x+\frac{y}{f}=1;$$

$$CF:\ -x+\frac{y}{f}=1.$$

Решая соответствующие уравнения найдем координаты вершин четырехугольника $KLMN$:

$$K\left(\frac{e-ef}{ef+1};\frac{ef+f}{ef+1}\right),$$

$$L\left(\frac{ef-e}{ef-1};\frac{ef-f}{ef-1}\right),\ M\left(\frac{ef+e}{ef+1};\frac{f-ef}{ef+1}\right),\ N\left(\frac{-e-ef}{ef-1};\frac{-ef-f}{ef-1}\right).$$

Тогда точки P и Q, середины диагоналей LN и KM, имеют координаты:

$$P\left(\frac{-e}{ef-1};\frac{-f}{ef-1}\right),\ Q\left(\frac{e}{ef+1};\frac{f}{ef+1}\right).$$

Для центра квадрата $O(0;0)$ вектора $\overrightarrow{OP}\left(\frac{-e}{ef-1};\frac{-f}{ef-1}\right)$ и $\overrightarrow{OQ}\left(\frac{e}{ef+1};\frac{f}{ef+1}\right)$ коллинеарны, поскольку $\overrightarrow{OQ}=\frac{1-ef}{1+ef}\cdot\overrightarrow{OP}$. Следовательно, точки P, O, Q лежат на одной прямой.

6.3. Информатика

Задача 6.3.1. Игра (100 баллов)

Как-то Ильнар и Азат придумали игру для двух человек. На доске пишется число n, после чего игроки делают свои ходы.

Ходят по очереди. Первым ходит Ильнар. За один ход игрок должен заменить написанное на доске число (обозначим его m) на число $m-x$, где x является степенью числа 2, то есть $x=2^k$ для некоторого целого неотрицательного числа k, и выполняется условие $1\leqslant x\leqslant m$. Игрок, который не может сделать ход, проигрывает.

Определите, кто выиграет при оптимальной игре.

Формат входных данных

В первой строке вводится одно целое число n ($0\leqslant n\leqslant 10^9$) – начальное число.

Формат выходных данных

Выведите *Ilnar won!*, если выиграет Ильнар, или *Azat won!*, если выиграет Азат.

Примеры

Пример №1

Стандартный ввод
2

Стандартный вывод
Ilnar won!

Пример №2

Стандартный ввод
0

Стандартный вывод
Azat won!

Способ оценки работы

За решение задачи начислялось:

- **40 баллов**, если пройдена только первая группа тестов;
- **70 баллов**, если пройдена первая и вторая группы тестов;
- **100 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке C++

```
#include "testlib.h"
#include <string>
#include <vector>
#include <sstream>

using namespace std;

bool compareWords(string a, string b)
{
    vector<string> va, vb;
    stringstream sa;

    sa << a;
    string cur;
    while (sa >> cur)
        va.push_back(cur);

    stringstream sb;
    sb << b;
    while (sb >> cur)
        vb.push_back(cur);

    return (va == vb);
}

int main(int argc, char * argv[])
{
    setName("compare files as sequence of tokens in lines");
    registerTestlibCmd(argc, argv);

    std::string strAnswer;

    int n = 0;
    while (!ans.eof())
    {
        std::string j = ans.readString();

        if (j == "" && ans.eof())
          break;
```

```
        std::string p = ouf.readString();
        strAnswer = p;

        n++;

        if (!compareWords(j, p))
            quitf(_wa, "%d%s lines differ - expected: '%s', found: '%s'", n, englishEnding(n).c_str()
    }

    if (n == 1)
        quitf(_ok, "single line: '%s'", compress(strAnswer).c_str());

    quitf(_ok, "%d lines", n);
}
```

Решение

Можно заметить, что все выигрышные для Азата числа делятся на 3, а все не делящиеся на 3 числа – выигрышные для Ильнара.

Доказательство:

1. 0 делится на 3 и это выигрышное для Азата число.

2. Если на доске написано положительное число, которое делится на 3, то в результате любого хода получается число, которое не делится на 3.

3. Если на доске написано число, которое не делится на 3, то мы можем просто отнять остаток от деления на 3 (то есть 1 или 2) и на доске снова будет написано число, которое делится на 3.

Асимптотика: $O(1)$

Пример программы-решения

Ниже представлено решение на языке Python3

```
1 n = int(input())
2 print("Azat won!" if n % 3 == 0 else "Ilnar won!")
```

Задача 6.3.2. Бактерии (100 баллов)

На планете Нирокку существует n различных видов бактерий. Виды бактерий пронумерованы от 1 до n. В лаборатории "Кадзи" хранятся экземпляры некоторых из этих видов.

В лаборатории существует n колоний бактерий, в i-ой из которых находятся все имеющиеся в лаборатории бактерии i-го вида. Некоторые колонии могут пустовать. Также стоит отметить, что в "Кадзи" поддерживают условия для существования, но не размножения бактерий.

Вечером в лабораторию приезжает министр экспериментов, который хочет увидеть коллекцию бактерий. Учёные лаборатории решили угодить ему: они хотят, чтобы суммарное количество бактерий стало равно любимому числу министра. Для этого учёные воспользуются раствором ускорения роста численности бактерий в колонии.

Раствор можно применять только к тем колониям, количество бактерий в которых не менее двух и ранее к ним не применяли данный раствор. В результате действия раствора количество бактерий в колонии увеличится в k раз. Число k может быть любым целым числом, большим 1, и выбирается заново перед каждым применением раствора, то есть оно может меняться.

До вечера остаётся мало времени, и потому учёные успеют применить раствор не более чем к одной колонии. Также учёные потеряли и не могут найти информацию о том, к каким колониям уже применялся раствор. В случае повторного применения раствора к некоторой колонии происходит взрыв, последствия которого до приезда министра устранить не является возможным.

Помогите учёным определить, можно ли угодить министру, не рискуя при этом взорвать лабораторию.

Формат входных данных

В первой строке два целых числа:

$1 \leqslant n \leqslant 10^5$ – количество различных видов бактерий на планете Нирокку;

$0 \leqslant m \leqslant 10^{18}$ – любимое число министра экспериментов.

Во второй строке n целых чисел:

$0 \leqslant a_i \leqslant 10^6$ – количество бактерий i-го $(1 \leqslant i \leqslant n)$ вида в лаборатории "Кадзи".

Формат выходных данных

В случае если угодить министру невозможно, выведите строку "No" (без кавычек).

В случае отсутствия необходимости применять раствор, выведите строку "Yes" (без кавычек).

В остальных случаях выведите строку "Yes i k" (без кавычек), где i – номер колонии, к которой учёным можно и нужно применить раствор, а k – количество раз, в которое необходимо увеличить численность i-ой колонии.

Если правильных ответов несколько, выведите любой из них.

Примеры

Пример №1

Стандартный ввод
5 5 0 0 1 2 0

Стандартный вывод
Yes 4 2

Пример №2

Стандартный ввод
2 7 3 3
Стандартный вывод
No

Пример №3

Стандартный ввод
4 8 0 3 2 3
Стандартный вывод
Yes

Способ оценки работы

За решение задачи начислялось:

- **30 баллов**, если пройдена только первая группа тестов;
- **60 баллов**, если пройдена первая и вторая группы тестов;
- **100 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке C++

```
#include "testlib.h"

std::string upper(std::string sa) {
    for (size_t i = 0; i < sa.length(); i++)
        if ('a' <= sa[i] && sa[i] <= 'z')
            sa[i] = sa[i] - 'a' + 'A';
    return sa;
}

bool is_prime(long long n) {
    for (long long k = 2LL; k * k <= n; k++)
        if (n % k == 0LL)
            return 0;
    return 1;
}

int main(int argc, char * argv[]) {
    registerTestlibCmd(argc, argv);

        int n = inf.readInt();
    long long m = inf.readLong();
        long long a[n];
        long long sum = 0;
    for (int i = 0; i < n; i++){
        a[i] = inf.readLong();
                sum += a[i];
        }

    std::string ja = upper(ans.readWord());
    std::string pa = upper(ouf.readWord());
```

```
    if (ja != "YES" && ja != "NO")
        quitf(_fail, "YES or NO expected in answer, but %s found", ja.c_str());

    if (pa != "YES" && pa != "NO")
        quitf(_pe, "YES or NO expected, but %s found", pa.c_str());

    if (ja != pa)
        quitf(_wa, "expected %s, found %s", ja.c_str(), pa.c_str());

    if (ja == "NO")
        quitf(_ok, ":)");

        long long sumja = sum;
        if (!ans.seekEof()) {
                int i = ans.readInt();
                if (i < 1 || i > n)
                        quitf(_fail, "Invalid i (i = %d)", i);
                i--;
                if (a[i] < 2LL || !is_prime(a[i]))
                        quitf(_fail, "Invalid a[%d] (a[%d] = %d)", i, i, a[i]);
                if (ans.seekEof())
                        quitf(_fail, "Answer does not contain k");
                long long k = ans.readLong();
                if (k < 2LL)
                        quitf(_fail, "Invalid k (k = %d)", k);
                if (!ans.seekEof())
                        quitf(_fail, "Answer contains extra elements");
                sumja += (k - 1LL) * a[i];
        }

        if (sumja != m)
                quitf(_fail, "Jury sum not equal to m");

        long long sumpa = sum;
        if (!ouf.seekEof()) {
                int i = ouf.readInt();
                if (i < 1 || i > n)
                        quitf(_pe, "Invalid i (i = %d)", i);
                i--;
                if (a[i] < 2LL || !is_prime(a[i]))
                        quitf(_wa, "Invalid a[%d] (a[%d] = %d)", i, i, a[i]);
                if (ouf.seekEof())
                        quitf(_pe, "Output does not contain k");
                long long k = ouf.readLong();
                if (k < 2LL)
                        quitf(_wa, "Invalid k (k = %d)", k);
                if (!ouf.seekEof())
                        quitf(_pe, "Output contains extra elements");
                sumpa += (k - 1LL) * a[i];
        }

        if (sumpa != m)
                quitf(_wa, "Participant sum not equal to m");

    quitf(_ok, ":)");
}
```

Решение

Запишем в переменную sum текущее суммарное количество бактерий.

Посмотрим на значение $m - sum$.

Если оно отрицательно, то ответ *No*, так как мы не можем уменьшить суммарное количество бактерий.

Если равно нулю, то очевидно, что ответ – *Yes*.

Подробнее рассмотрим случай, когда значение $m - sum$ положительно.

Из условия понятно, что если число a_i удовлетворяет некоторому условию, то мы можем умножить его на любое целое число, которое больше 1.

Что это за условие?

Не должно существовать целого числа, которое больше 1 и его можно умножить на целое число k $(k > 1)$ и таким образом получить число a_i.

Не трудно заметить, что число a_i должно быть простым (не иметь делителей кроме 1 и самого себя), иначе возможно, что к i-ой колонии уже применяли раствор.

Таким образом нужно определить, существует ли такое простое a_i, что $sum - a_i + a_i \cdot k = m$, где число k – целое и больше 1.

Проверку на простоту можно сделать с помощью, например, решета Эратосфена, поскольку $a_i \leqslant 10^6$ для всех i.

Асимптотика: $O(n + max(a_i) \cdot \log(\log(max(a_i))))$

Пример программы-решения

Ниже представлено решение на языке Python3

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

sum = sum(a)

if sum == m:
    print("Yes")
elif sum > m:
    print("No")
else:
    sz = max(a)
    prime = [i > 1 for i in range(sz + 1)]
    for i in range(sz + 1):
        if prime[i]:
            j = i * i
            while j <= sz:
                prime[j] = False
                j += i
    for i in range(n):
        if prime[a[i]] and (m - sum) % a[i] == 0:
            print("Yes", i + 1, (m - sum) // a[i] + 1)
            break
    else:
        print("No")
```

Задача 6.3.3. Царство (100 баллов)

В тридевятом царстве n городов и $n-1$ дорог, по которым можно из любого города добраться в любой другой (возможно, проезжая через другие города). Города пронумерованы от 1 до n, а дороги – от 1 до $n-1$. Все дороги с двусторонним движением.

Автомобилисты царства последнее время сильно озабочены ценами на бензин. Они называют пару городов (i, j) странной, если из города i можно добраться до города j и стоимость бензина в городе i больше, чем в городе j.

А царь озабочен народными волнениями и хочет перекрыть все дороги. На время. А также хочет всегда знать степень недовольства своих подданных. Одним из факторов, влияющих на степень недовольства, является количество пар странных городов.

Ваша задача – определить количество пар странных городов до перекрытия дорог и после каждого очередного перекрытия дороги. Не справитесь – не сносить вам головы!

Формат входных данных

В первой строке вводится целое число n ($2 \leqslant n \leqslant 3 \cdot 10^5$).

Во второй строке вводится n целых чисел c_i ($1 \leqslant c_i \leqslant 10^9$) – стоимость бензина в i-ом городе.

В следующих $n-1$ строках содержатся описания дорог: пары чисел a_i и b_i ($1 \leqslant a_i, b_i \leqslant n$), говорящие о том, что i-ая дорога проложена между городами a_i и b_i.

В последней строке задается порядок перекрытия дорог, то есть $n-1$ целых чисел q_i ($1 \leqslant q_i \leqslant n-1$) – номер дороги, перекрытой i-ой по счёту.

Гарантируется, что до перекрытия дорог в царстве можно было из любого города добраться в любой другой, а в конечном итоге все дороги перекрыли.

Формат выходных данных

Выведите в одной строке n целых чисел – количество пар странных городов до перекрытия дорог и после каждого очередного перекрытия дороги.

Заметьте, что ответ может не помещаться в 32-битный тип данных.

Примеры

Пример №1

Стандартный ввод
3 1 2 3 1 2 2 3 1 2

Стандартный вывод
3 1 0

Пример №2

Стандартный ввод
5 1 2 3 1 2 1 2 2 3 3 4 4 5 2 3 1 4
Стандартный вывод
8 4 2 1 0

Способ оценки работы

За решение задачи начислялось:

- **40 баллов**, если пройдена только первая группа тестов;
- **70 баллов**, если пройдена первая и вторая группы тестов;
- **100 баллов**, если пройдены все тесты.

Для проверки результата использовался следующий код на языке C++

```
#include "testlib.h"
#include <sstream>

using namespace std;

int main(int argc, char * argv[])
{
    setName("compare ordered sequences of signed int%d numbers", 8 * int(sizeof(long long)));

    registerTestlibCmd(argc, argv);

    int n = 0;
    string firstElems;

    while (!ans.seekEof() && !ouf.seekEof())
    {
        n++;
        long long j = ans.readLong();
        long long p = ouf.readLong();
        if (j != p)
            quitf(_wa, "%d%s numbers differ - expected: '%s', found: '%s'", n, englishEnding(n).c_str
        else
            if (n <= 5)
            {
                if (firstElems.length() > 0)
                    firstElems += " ";
                firstElems += vtos(j);
            }
    }

    int extraInAnsCount = 0;

    while (!ans.seekEof())
```

```
    {
        ans.readLong();
        extraInAnsCount++;
    }

    int extraInOufCount = 0;

    while (!ouf.seekEof())
    {
        ouf.readLong();
        extraInOufCount++;
    }

    if (extraInAnsCount > 0)
        quitf(_wa, "Answer contains longer sequence [length = %d], but output contains %d elements",

    if (extraInOufCount > 0)
        quitf(_wa, "Output contains longer sequence [length = %d], but answer contains %d elements",

    if (n <= 5)
        quitf(_ok, "%d number(s): \"%s\"", n, compress(firstElems).c_str());
    else
        quitf(_ok, "%d numbers", n);
}
```

Решение

Переведём задачу в терминологию графов:

- Город = вершина;
- Дорога = ребро;
- Странная пара городов (i, j) = вершины i и j, которые находятся в одной компоненте связности, причём $c_i > c_j$.

Теперь "развернём" задачу: посчитаем ответ с конца. Будем добавлять рёбра в граф, а не удалять из него. Причём делать мы это будем в обратном порядке.

Не сложно заметить, что после каждого добавления ребра объединяются две компоненты и ответ нужно пересчитать только относительно вершин, которые были в этих двух компонентах.

Давайте поддерживать словари (map'ы) для каждой компоненты связности. Ключи словарей – значения вершин (для i-ой вершины это число c_i) в соответствующих компонентах. Значения, хранящиеся по этим ключам – количество вершин с таким же значением c_i в компоненте.

Для быстрого определения того, к каким двум компонентам относятся концы ребра, которое мы сейчас добавляем, воспользуемся СНМ (Системой Непересекающихся Множеств).

Таким образом, при добавлении нового ребра для пересчёта ответа достаточно объединить 2 map'ы.

Если всегда переносить информацию о вершина из меньшей по количеству ключей map'ы в большую, то можно доказать, что информация о каждой вершине перенесётся не более $\log_2(n)$ раз.

Асимптотика: $O(n \cdot \log^2(n))$

Пример программы-решения

Ниже представлено решение на языке Python3

```python
def dsu_init(n):
    for i in range(n):
        p[i] = i
    return


def dsu_get(v):
    if v != p[v]:
        p[v] = dsu_get(p[v])
    return p[v]


def dsu_union(a, b):
    a = dsu_get(a)
    b = dsu_get(b)
    if a != b:
        map_union(a, b)
    return


def map_union(a, b):
    if len(mp[b]) > len(mp[a]):
        map_union(b, a)
        return

    global cur
    cur += cnt[a] * cnt[b]
    p[b] = a
    cnt[a] += cnt[b]
    for i in mp[b]:
        mp[a].setdefault(i, 0)
        cur -= mp[b][i] * mp[a][i]
        mp[a][i] += mp[b][i]
    return


n = int(input())
c = [int(i) for i in input().split()]

p = [0] * n
cnt = [0] * n
mp = [{} for i in range(n)]
cur = 0

for i in range(n):
    mp[i][c[i]] = 1
    cnt[i] += 1

a = [0] * (n - 1)
b = [0] * (n - 1)
for i in range(n - 1):
    a[i], b[i] = [int(i) - 1 for i in input().split()]

q = [int(i) - 1 for i in input().split()]
q.reverse()

```

```
dsu_init(n)
ans = [0]
for i in q:
    dsu_union(a[i], b[i])
    ans.append(cur)

ans.reverse()
print(' '.join(map(str, ans)))
```

Командный тур

В командной части заключительного этапа участники должны спроектировать автономную систему управления групой роботов-погрузчиков для решения задач навигации на модели логистического центра с использованием алгоритмов компьютерного зрения.

Командная часть заключительного этапа проходит в течении 3.5 дней (всего 26 астрономических часов), которые включают работу по оснащению роботов необходимыми датчиками, программированию, пробные заезды на макете логистического центра, зачетные попытки.

7.1. Легенда

Недалёкоое будущее, в автоматических логистических центрах практически нет людей, всю работу выполняют роботы-погрузчики, которыми управляет интеллектуальное ПО по распределению задач.

Несмотря на отсутствие человеческого фактора, не следует думать, что в таких центрах никогда не будет проблем. Форс-мажор может произойти в любой момент и система из роботов и ПО должна уметь справляться с такими ситуациями.

Поэтому для финальной задачи профиля «Интеллектуальные робототехнические системы» предлагается рассмотреть следующий эпизод: в логистическом центре произошла перезагрузка всех систем, что привело к сбросу информации о местоположениях работающих в данный момент роботов-погрузчиков.

В начале выполнения задания считается, что робототехнические устройства активируются в каких-то секторах (каждый в своём) логистического центра. При этом 2 из них имеют информацию о своём местоположении, а у 3-го была повреждена память и по этому эта информация была потеряна. Структура логистического центра известна заранее всем устройствам. Роботы-погрузчики должны переместиться в сектор своей приписки. Координаты секторов сервисного обслуживания известны заранее. Информацию о местоположении необходимого сектора приписки можно узнать в секторе сервисного обслуживания. В нём располагается ARTag маркер, в котором закодирован номер робота и координаты сектора его приписки. При перемещении роботы не должны сталкиваться, а также повреждать логистический центр.

Задача участников Олимпиады — разработать программу управления несколькими робототехническими устройствами для выполнения задания описанного выше.

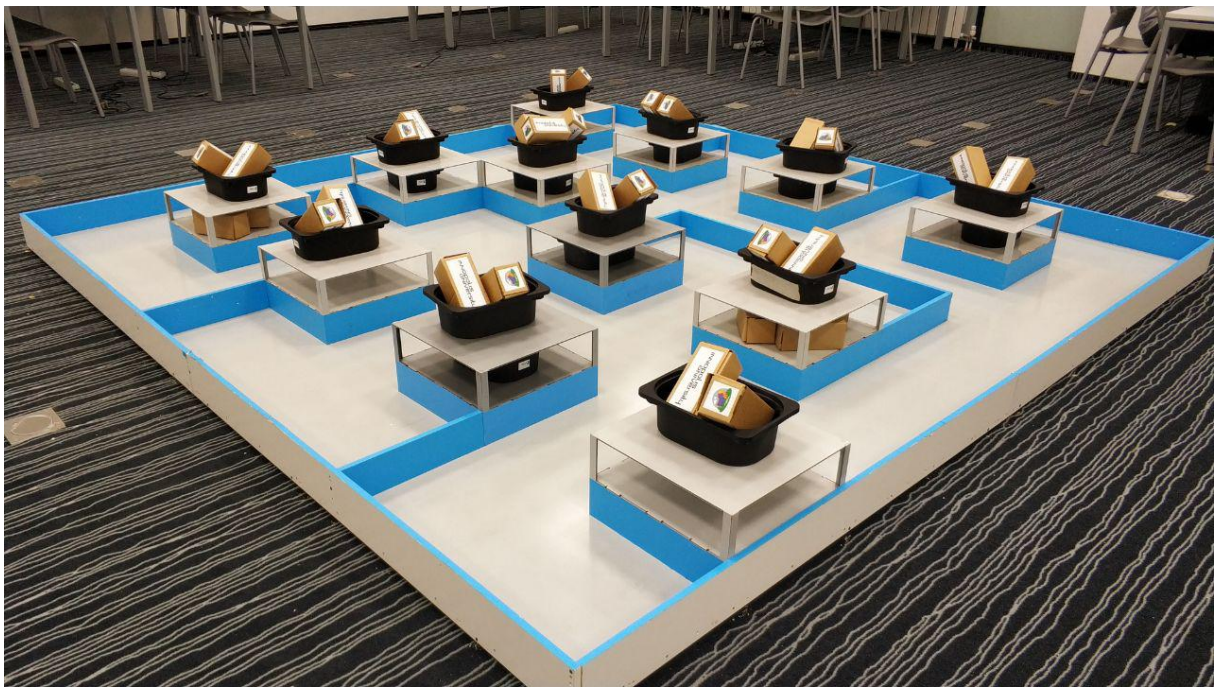


Рис. 7.1: Полигон для запуска робототехнических устройств на финале Олимпиады НТИ

7.2. Набор заданий

Решение командной задачи разбито на 5 этапов. Первые четыре этапа итеративно подводят участников к решению полной финальной задачи, осуществляемому во время последнего пятого этапа. На каждом этапе в проверку решения заданий данного этапа входят:

- способность проверить гипотезу о работоспособности алгоритма через демонстрацию решения в симуляторе;
- полнота решения задания конкретного этапа;
- воспроизводимость результатов — робототехническое устройство участников должно было неоднократно выполнить требуемые действия.

Первый этап

Задача: три робототехнических устройства располагаются в модели логистического центра. Им необходимо обменяться данными о наличии или отсутствии стен слева от робота, перед собой и справа от робота, где 1 — присутствует стена, 0 — отсутствует. Выводить показания следует на экран всех роботов непрерывно. Данные, принятые с каждого робота, необходимо вывести на разных строках при этом показания с одного робота следует выводить через пробел в порядке указаном выше.

Включая содержательные задачи:

- Нахождение порогового значения для датчиков расстояния;
- Реализация коммуникации между робототехническими устройствами.

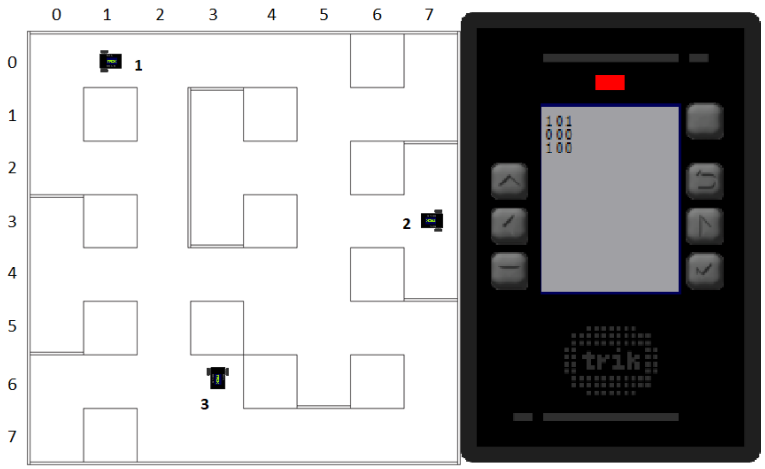


Рис. 7.2: Пример расположения роботов и показания выведенные на экран

Второй этап

Задача: три робототехнических устройства должны определить своё местоположение в модели логистического центра из случайных точек старта, одна из которых заранее не известна.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов локализации для группы робототехнических устройств;
- Реализация алгоритмов распределения группы роботов с целью достижения общей задачи;
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

Третий этап

Задача: три робототехнических устройства должны проехать по модели логистического центра из точек старта в сектора финишей, не столкнувшись.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

Четвёртый этап

Задача: робототехническое устройство должно проехать по модели логистического центра из сектора старта в сектор финиша через сектор сервисного обслуживания. Координаты сектора финиша робот должен определить самостоятельно по ARTag метке, расположенной на стеллаже, прилегающем к сектору сервисного обслуживания.

Включая содержательные задачи:

- Калибровка камеры робототехнического устройства;

- Реализация алгоритмов компьютерного зрения: считывание ARTag меток без использования дополнительных библиотек;
- Декодирование бинарного кода с использованием кода Хэмминга.

Пятый этап

Задача: три робототехнических устройства должны проехать по модели логистического центра из случайных точек старта, одна из которых заранее не известна, в соответсвующие сектора финиша. Координаты секторов финиша для каждого робота указаны в ArTag маркерах, расположенных в секторах сервисного обслуживания.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов локализации для группы робототехнических устройств;
- Реализация алгоритмов распределения группы роботов с целью достижения общей задачи;
- Реализация алгоритмов компьютерного зрения: считывание ARTag меток без использования дополнительных библиотек;
- Декодирование бинарного кода, с использованием кода Хэмминга;
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

7.3. Описание модели логистического центра

Полигон - квадратное поле 3200 × 3200 мм., разделенное на квадратные сектора 400 × 400 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. 7.3) размером 200 × 300 × 100 мм (груз - 2-3 коробки (рис. 7.4) размером 180 × 80 × 85 мм).

Рис. 7.3: Внешний вид контейнера

Рис. 7.4: Внешний вид коробки

Полигон окружен бортом высотой 100 мм. Конфигурация полигона определяется в первый день финального этапа и объявляется участникам. Данная конфигурация

будет использоваться во все дни финального этапа.

На нижней полке стеллажа, прилегающего одной из четырех сторон к сектору сервисного обслуживания, на высоте от 100-150 мм от уровня поверхности поля закреплен ARTag маркер (`https://goo.gl/WaTFMB`), определяющий номер робота и координаты сектора приписки (сектор финиша для конкретного робота). Размер маркера - 30 × 30 мм. Маркер обращен лицевой стороной внутрь сектора сервисного обслуживания, координаты которого известны заранее. 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 маркера на стелаже представлен на рисунке 7.5

Рис. 7.5: Стеллаж с установленным ARTag маркером

Маркер состоит из 6 × 6 элементов одинакового размера. Элементы маркера, расположенные по его границе — всегда черные. Четыре элемента, находящиеся в углах внутреннего 4 × 4 квадрата определяют ориентацию маркера таким образом, что только один из них — белый. Оставшиеся 12 элементов 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. Нумерация элементов относительно ориентационных элементов обозначена на рисунке 7.6.

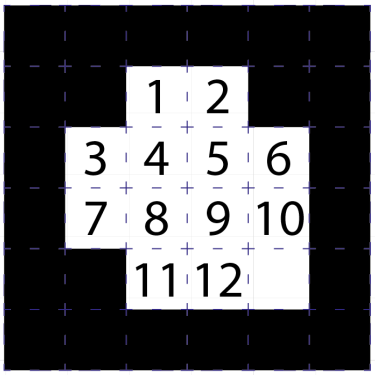


Рис. 7.6: Нумерация элементов маркера относительно ориентационных элементов

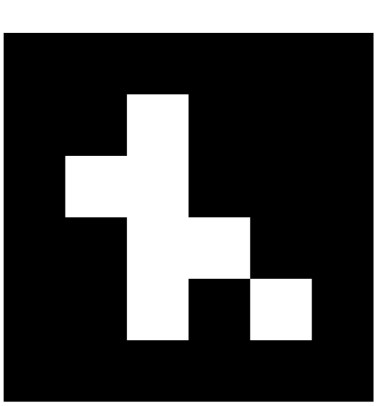


Рис. 7.7: Маркер с закодированным значением - 010011100101_2

Закодированное на маркере двоичное двенадцатибитное число закодировано с использованием кода Хэмминга (https://habr.com/ru/post/140611/), в котором 8 информационных битов и 4 контрольных бита:

1 Первый контрольный бит;

2 Второй контрольный бит;

3 Старший бит номера робота $N(1 \leqslant N \leqslant 3)$;

4 Третий контрольный бит;

5 Младший бит номера робота $N(1 \leqslant N \leqslant 3)$;

6 Первый (старший) бит координаты X робота N $(0 \leqslant X_N \leqslant 7)$;

7 Второй бит координаты X робота N $(0 \leqslant X_N \leqslant 7)$;

8 Четвёртый (последний) контрольный бит;

9 Третий (младший) бит координаты X робота N $(0 \leqslant X_N \leqslant 7)$;

10 Первый (старший) бит координаты Y робота N $(0 \leqslant Y_N \leqslant 7)$;

11 Второй бит координаты Y робота N $(0 \leqslant Y_N \leqslant 7)$;

12 Третий (младший) бит координаты Y робота N $(0 \leqslant Y_N \leqslant 7)$;

Маркер на рисунке 7.7 кодирует число 010011100101_2 — $N = 1$, $X = 6$, $Y = 5$. Таким образом, сектор приписки для робота 1 находится в позиции с координатами $(6, 5)$ — см. рисунок 7.8.

Гарантируется, что сектора сервисного обслуживания будут располагаться в таких местах полигона, где к сторонам сектора прилегает как минимум один стеллаж.

Сектора активаций роботов (стартов), сектора сервисного обслуживания и сектора приписки никак не обозначаются на поле и определяются непосредственно перед каждым заездом робота.

7.4. Описание конструктора

В первый день финального тура каждой команде выдаются три комплекта:

- 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 построе-

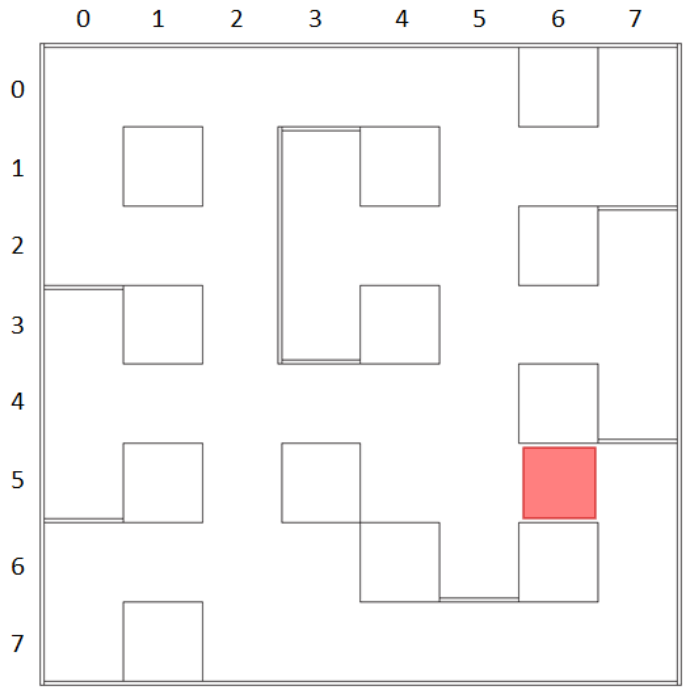


Рис. 7.8: Расположение сектора финиша для маркера из рисунка 7.7

на 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: 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 могут пользоваться своими ноутбуками.

7.5. Условия проведения

1. Из полученного набора датчиков команды могут выбирать те, с помощью которых, по мнению участников, можно решить задачу наиболее эффективным способом.
2. Команды могут вносить любые изменения в мобильные наземные платформы.
3. Участники во время командного этапа финального тура могут использовать интернет и заранее подготовленные библиотеки для решения задачи.
4. Участники не могут использовать помощь тренера, сопровождающего лица или привлекать третьих лиц для решения задачи.
5. Финальная задача формулируется участникам в первый день финального тура, но участники выполняют решение задачи поэтапно. Критерии прохождения каждого этапа формулируются для каждого дня финального тура. За подзадачи, решенные в конкретном этапе начисляются баллы. Баллы за под-

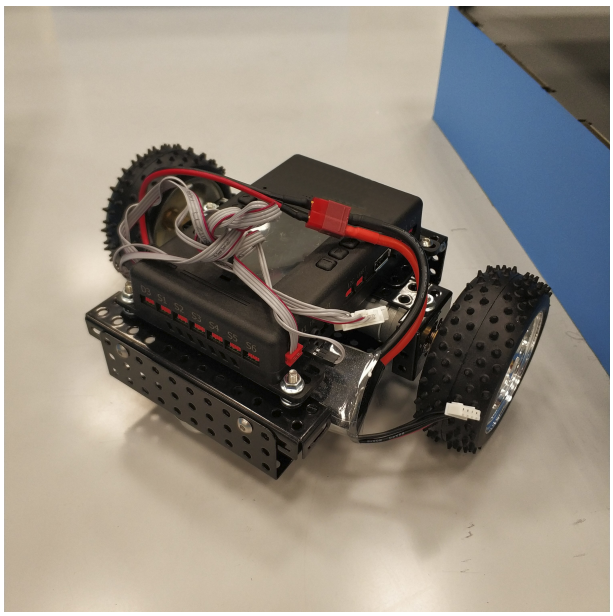


Рис. 7.9: Мобильная платформа TRIK в сборе

задачи можно получить только в день, закрепленный за конкретным этапом.

6. Некоторые подзадачи строго требуют выполнение каких-то предыдущих подзадач. Выполнение данных подзадач без выполнения предыдущих допускается, однако данная попытка будет оценена в 0 баллов.
7. 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.
8. 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.
9. Испытания на полигоне должны осуществляться так, чтобы не мешать другим командам, проводящим в это время свои испытания на полигоне. Для этого всем командам может быть назначено ограничение по времени, которое они могут тратить на одно испытание. После истечения этого времени, команда должна дать возможность проводить испытания следующей команде.
10. 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.
11. У команды есть не более двух попыток для сдачи решения подзадач в симуляторе:
    (a) Решения принимаются на проверку до истечения первых 6 часов работы в соответствующий соревновательный день. Может меняться в зависимости от дня.

(b) До истечения 6 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.

(c) До истечения указанного времени команды могут изменять файл с решением сколько угодно раз. Проверяться будет всегда только последняя доступная версия.

(d) Если решение отправлено на проверку в течение первых 4 часов работы в соответствующий соревновательный день, то команда имеет право на вторую попытку, если результаты проверки решения ее не устраивают.

(e) Если команда хочет воспользоваться правом проверки решения до истечения 4 часов, то она должна загрузить в каталог на *Google Drive* программу со своим решением и сообщить об этом судьям.

12. Часть подзадач для реальных роботов может быть запрещена к приемке без успешного прохождения 60% всех тестов, предназначенных для проверки решения соответствующей подзадачи в симуляторе.

13. 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.

14. Перед сдачей робота в карантин команды должны загрузить на роботов управляющие программы, подготовленные 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.

15. После момента, когда все роботы сданы в карантин, судьи по одной вызывают команды для приемки решения подзадач, закрепленных за этапом конкретного дня финального тура.

16. Может быть предусмотрено до двух попыток сдачи решения одной и той же подзадачи на реальных роботах. Конкретное количество попыток определяется в конкретных подзадачах.

17. 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.

18. Роботы должны выполнять задание полностью автономно. Удаленное управление не допускается. Касание какого-либо робота участником команды после его старта во время приемочных запусков не допускается. Алгоритм, реализующий систему управления группой роботов, должен планировать свое выполнение, полагаясь только на информацию с датчиков.

19. Введение данных в программу до старта устройства (например, координат робота в начале работы) разрешается только для тех задач, где это явно прописано. Во всех других случаях введение данных в программу роботов перед запуском запрещено.

20. Для всех роботов программа должна быть одинаковой, допускается отличие лишь в разрешенных входных данных.
21. Если какая-то подзадача подразумевает считывание информации с элементов, расположенных на полигоне, запрещается при запуске роботов вводить информацию о положении этих элементов или значениях, которые данные элементы определяют.
22. Если во время приемочных запусков у судьи возникли сомнения о том, что задачи подэтапа решены корректно (роботы не выполняют задачу полностью автономно, участник вводит значения в каких-либо роботов перед запуском), то он вправе провести инспекцию кода. По результатам инспекции, судья вправе снять с команды баллы, набранные за данный этап.
23. 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.
24. Команда вправе обсуждать с судьей результаты приемочных запусков до вызова следующей команды, но финальное решение остается о начислении баллов остается за судьей.

7.6. Процедура проведения приемочных запусков и критерии оценки

Первый этап

1. Командам будет предоставлено две попытки на демонстрацию решения задачи на реальном роботе.
2. Обе попытки будут осуществляться 7 марта.
3. Необходимо сдать данную задачу в любой момент периода отладки.
4. Правила именования файлов с программой управления:
   (a) для первой попытки: `part1_1.js`.
   (b) для второй попытки: `part1_2.js`.
5. Максимальное время выполнения одной попытки - 30 секунд.
6. Баллы за решение задач этапа:
   (a) **Реальный робот:**
      i. Один из роботов вывел показания всех 9ти датчиков расстояния — 2 балла;
      ii. Все роботы вывели показания всех 9ти датчиков расстояния один раз — 2 балла;
      iii. Все роботы выводят показания всех 9ти датчиков расстояния непрерывно — 2 баллов;
7. Баллы за две попытки суммируются.
8. Выполнение всех критериев в каждой из двух попыток всех двух подзадач дает дополнительные 2 балла.
9. Максимальное количество баллов за этап — 14.

Второй этап

1. Командам необходимо подготовить две задачи для симулятора:
   (a) В качестве первой задачи участникам необходимо выполнить следующее:
      i. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Необоходимо в процессе движения опредлить своё местоположение, остановиться и вывести на экран координаты робота в формате "(X, Y)" без пробелов и без кавычек.

      Ожидаемый результат: После запуска программы робот определил своё местоположение, остановился и вывел на экран координаты своего местоложожения в формате "(X, Y)" без пробелов и без кавычек.

   (b) В качестве второй задачи:
      i. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Известно, что два сектора данного логистического центра заблокированы другими роботами. Координаты этих секторов передаются через входной файл. Необоходимо в процессе движения опредлить своё местоположение, остановиться и вывести на экран координаты робота в формате "(X, Y)" без пробелов и без кавычек.

      Входные данные: через файл `input.txt` управляющей программе передаются:
         A. В первой строке через пробел — координаты одного заблокированного сектора X_1, Y_1, $0 \leqslant X_1, Y_1 \leqslant 7$;
         B. Во второй строке через пробел — координаты другого заблокированного сектора X_2, Y_2, $0 \leqslant X_2, Y_2 \leqslant 7$;

      Ожидаемый результат: После запуска программы робот определил своё местоположение, остановился и вывел на экран координаты своего местоложожения в формате "(X,Y)" без пробелов и без кавычек.

   (c) Правила именования файлов с управляющей программой для проверки решений в симуляторе:
      i. Для первой задачи: `sim_part2_1.qrs`;
      ii. Для второй задачи: `sim_part2_2.qrs`.
2. Команде необходимо будет подготовить решения для двух разных подзадач для реальных роботов. На демонстрацию каждого решения предоставляется 2 попытки.
3. Все попытки осуществляются 7 марта.
4. После сдачи в карантин для 1ой подзадачи судья определяет сектор старта и направление робота в секторе старта.
5. За 5 мин до сдачи в карантин для 2ой подзадачи судья определяет сектора старта и направления в секторах старта двух роботов. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
6. После сдачи в карантин для 2ой подзадачи судья определяет сектор старта

и направление в секторе старта для оставшегося робота.

7. Секторы старта могут быть различными в разных попытках. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
8. Правила именования файлов с программой управления:
   (a) для первой попытки первой подзадачи: `part2_1_1.js`;
   (b) для второй попытки первой подзадачи: `part2_1_2.js`;
   (c) для первой попытки второй подзадачи: `part2_2_1.js`;
   (d) для второй попытки второй подзадачи: `part2_2_2.js`.
9. Максимальное время выполнения одной попытки - 5 минут.
10. Баллы за решение задач этапа:
    (a) **Первая задача в симуляторе**: робот смог определить своё местоположение на всех проверочных полигонах — 12 баллов.
    (b) **Вторая задача в симуляторе**: робот смог определить своё местоположение на всех проверочных полигонах — 14 баллов.
    (c) **Первая подзадача на реальном роботе:** Робот располагается с случайном секторе робототехнического полигона. Робот смог определить своё местоположение на поле, остановился, издал звуковой сигнал, вывел на экран свои координаты в формате "(X, Y)" — 14 баллов.
    (d) **Вторая подзадача на реальном роботе:** Роботы располагается с случайных секторах робототехнического полигона. Роботы смогли определить своё местоположение на поле, остановились, издали звуковой сигнал, вывел на экран свои координаты в формате "(X, Y)", а также вывели `finish` — 18 баллов.
11. Баллы за первую подзадачу не начисляются, если не было частично засчитано решение первой задачи в симуляторе
12. Баллы за вторую подзадачу не начисляются, если не были частично засчитаны решения за все задачи в симуляторе.
13. Баллы за все попытки в каждой подзадаче суммируются.
14. Выполнение всех критериев в каждой из двух попыток всех двух подзадач дает дополнительные 4 балла.
15. Максимальное количество баллов за этап — 94.

Третий этап

1. Командам необходимо подготовить две задачи для симулятора:
   (a) В качестве первой задачи участникам необходимо выполнить следующее:
      i. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Задача робота проехать из точки старта в точку финиша.

      Входные данные: через файл `input.txt` управляющей программе передаются:

A. 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), $0 \leqslant X_s, Y_s \leqslant 7$;

B. В второй строке через пробел — координаты сектора финиша X_f, Y_f, $0 \leqslant X_f, Y_f \leqslant 7$.

Ожидаемый результат: После запуска программы робот перемещается в сектор финиша по оптимальному пути. Оптимальным является путь, длина строки маршрута, которого наименьшая. Для маршрута используется следующая нотация действий без пробелов:

A. F — проезд в следующий сектор по ходу движения;

B. L — поворот налево в данном секторе;

C. R — поворот направо в данном секторе.

После остановки на экран робота выведен путь в описанной нотации выше.

(b) В качестве второй задачи:

i. Роботы устанавливаются в модели логистического центра в заранее неизвестном секторе. Задача роботов проехать из точки старта в точки финиша.

Входные данные: через файл `input.txt` управляющей программе передаются:

A. В первой строке через пробел — координаты сектора старта первого робота X_{1s}, Y_{1s}, направление старта первого робота D_{1s} (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке) и координаты сектора финиша первого робота X_{1f}, Y_{1f}, $0 \leqslant X_{1s}, Y_{1s}, X_{1f}, Y_{1f} \leqslant 7$;

B. Во второй строке через пробел — координаты сектора старта второго робота X_{2s}, Y_{2s}, направление старта второго робота D_{2s} (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке) и координаты сектора финиша вторго робота X_{2f}, Y_{2f}, $0 \leqslant X_{2s}, Y_{2s}, X_{2f}, Y_{2f} \leqslant 7$;

C. В третьей строке через пробел — координаты сектора старта третьего робота X_{3s}, Y_{3s}, направление старта третьего робота D_{3s} (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке) и координаты сектора финиша третьего робота X_{3f}, Y_{3f}, $0 \leqslant X_{3s}, Y_{3s}, X_{3f}, Y_{3f} \leqslant 7$;

Ожидаемый результат: После запуска программы происходит вычисление планируемых маршрутов для каждого робота. После вычисления в файл “output.txt” записана одна строка следующего вида: $1P_1 2P_2 3P_3$, без пробелов, где $1P_1$, P_2 и P_3 – маршруты соответсвующих роботов. Для вывода маршрута используется следующая нотация без пробелов:

- F — проезд в следующий сектор по ходу движения;
- L — поворот налево в данном секторе;
- R — поворот направо в данном секторе;
- S — остановка в данном секторе.

Роботы движутся без столкновений и так, что команды выполняются паралельно и любое действие занимает одинаковое количество времени.

(c) Правила именования файлов с управляющей программой для проверки решений в симуляторе:

i. Для первой подзадачи: `sim_part3_1.js`;

ii. Для второй подзадачи: `sim_part3_2.js`.

2. Команде необходимо будет подготовить решения для двух разных подзадач для реальных роботов. На демонстрацию каждого решения предоставляется 2 попытки.
3. Все попытки осуществляются 8 марта.
4. За 15 минут до времени сдачи роботов в карантин для данных подзадач судья определяет сектора старта и финиша, а также направление робота в секторе старта. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
5. Секторы старта и финиша могут быть различными в разных попытках.
6. Правила именования файлов с программой управления:
   (a) для первой попытки первой подзадачи: `part3_1_1.js`;
   (b) для второй попытки первой подзадачи: `part3_1_2.js`;
   (c) для первой попытки второй подзадачи: `part3_2_1.js`;
   (d) для второй попытки второй подзадачи: `part3_2_2.js`.
7. Максимальное время выполнения одной попытки - 5 минут.
8. Баллы за решение задач этапа:
   (a) **Первая задача в симуляторе**: робот проехал из точки старта в точку финиша и вывел на экран маршрут робота на всех проверочных полигонах — 6 баллов.
   (b) **Вторая задача в симуляторе**: В консоль выведены верные маршруты перемещения из точки старта в точку финиша для всех роботов на всех проверочных полигонах.
      i. Маршруты позволяют роботам доехать из начальных в конечные координаты – 10 баллов.
      ii. В маршруте кажного робота было использовано не более 3х команд S – 4 балла.
   (c) **Первая подзадача на реальном роботе:** Робот доехал до сектора финиша, остановился и вывел `finish` — 8 баллов.
   (d) **Вторая подзадача на реальном роботе:** Роботы располагается с случайных секторах робототехнического полигона.
      i. Первый робот доехал до сектора финиша не задев остальных роботов, остановился и вывел `finish` — 10 баллов.

ii. Второй робот доехал до сектора финиша не задев остальных роботов, остановился и вывел `finish` — 12 баллов.

iii. Третий робот доехал до сектора финиша не задев остальных роботов, остановился и вывел `finish` — 12 баллов.

9. За первую подзадачу начисляется только половина возможных баллов, если не было засчитано решение первой задачи в симуляторе
10. За вторую подзадачу начисляется только половина возможных баллов, если не были засчитаны решения за все задачи в симуляторе.
11. Баллы за все попытки в каждой подзадаче суммируются.
12. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 4 балла.
13. Максимальное количество баллов за этап — 108.

Четвёртый этап

1. В качестве задачи для симулятора участникам необходимо выполнить следующее:
   (a) Робот устанавливается в модели логистического центра в случайных секторе. При этом структура логистического центра известна заранее. Задача робота проехать из точки старта в точку финиша, чьи координаты заданы изображением ARTag маркера, заранее считанным с камеры реального устройства. На финише необходимо вывести на экран слово `finish`.

   Входные данные: через файл `input.txt` управляющей программе передаются:

   - В первой строке 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 $P_{1,i}$ $(0 \leqslant P_{1,i} \leqslant 2^{24})$ — изображение ARTag маркера;

   Каждое число в маркере — точка, закодированная в формате RGB, т.е. строка с изображением маркера эквивалентна снимку разрешением 160×120 точек. Один маркер кодирует координаты для первого робота, в данной ситуации являющегося единственным.

   Ожидаемый результат: После запуска программы робот перемещается в сектор финиша. После остановки на экран робота выведено `finish`.

   (b) Имя файла с управляющей программой для проверки решения в симуляторе: `sim_part4.js`.
2. Команде необходимо будет подготовить решения для двух разных подзадач для реального робота. На демонстрацию каждого решения предоставляется 2 попытки.
3. Все попытки осуществляются 9 марта.
4. За 15 минут до времени сдачи роботов в карантин для 2ой подзадачи судья

определяет сектор старта и направление робота в секторе старта. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.

5. За 15 минут до времени сдачи роботов в каранин судья определяет сектор сервисного обслуживания (сектор, из которого необходимо сканировать ARTag метку) и направление расположения ARTag метки (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.
6. Сектор старта и сектор сервисного обслуживания может быть разным для каждой попытки.
7. Правила именования файлов с программой управления:
   (a) для первой подзадачи: `part4_1.js`;
   (b) для первой попытки второй подзадачи: `part4_2_1.js`;
   (c) для второй попытки второй подзадачи: `part4_2_2.js`.
8. Максимальное время выполнения одной попытки - 3 минуты.
9. Баллы за решение задач этапа:
   (a) **Симулятор**: робот проехал из точки старта в точку финиша на всех проверочных полигонах — 6 баллов.
   (b) **Первая подзадача на реальном роботе:** Робот верно распознал ARTag метку, которая установлена прямо перед камерой, и вывел на экран верное значение, закодированное на метке — 2 балла.
   (c) **Вторая подзадача на реальном роботе:** Робот располагается за два сектора до сектора сервисного обслуживания:
      i. Робот доехал до сектора сервисного обслуживания, распределения задач, остановился, издал звуковой сигнал, вывел на экран верное ARTag метки — 4 балла.
      ii. Робот доехал до указаных в метке координат, остановился, издал сигнал и вывел на экран `finish`— 8 баллов.
10. За вторую подзадачу начисляется только половина возможных баллов, если не было засчитано решение в симуляторе, а также если не сдана первая подзадача.
11. Выводить значение метки и `finish` следует не менее 10 секунд.
12. Баллы за все попытки в каждой подзадаче суммируются.
13. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 4 балла.
14. Максимальное количество баллов за этап — 38.

Пятый этап

1. В качестве задачи для симулятора участникам необходимо выполнить следующее:
   (a) Робот устанавливается в модели логистического центра в заранее неизвестном секторе. При этом структура логистического центра из-

вестна заранее. Задача робота локализоваться и доехать в точку финиша, чьи координаты заданы изображением ARTag маркера, заранее считанным с камеры реального устройства. На финише необходимо вывести на экран слово `finish`.

Входные данные: через файл `input.txt` управляющей программе передаются:

- В первой строке 19200, разделенных пробелом, целых чисел $P_{1,i}$ ($0 \leqslant P_{1,i} \leqslant 2^{24}$) — изображение ARTag маркера;

Каждое число в маркере — точка, закодированная в формате RGB, т.е. строка с изображением маркера эквивалентна снимку разрешением 160×120 точек. Один маркер кодирует координаты для первого робота, в данной ситуации являющегося единственным.

Ожидаемый результат: После запуска программы робот перемещается в сектор финиша. После остановки на экран робота выведено `finish`.

(b) Имя файла с управляющей программой для проверки решения в симуляторе: `sim_part5.qrs`.

2. Командам будет предоставлено две попытки на демонстрацию решения задачи на реальном роботе.
3. Все попытки осуществляются 10 марта.
4. За 15 мин до сдачи в карантин судья определяет сектора старта и направления в секторах старта двух роботов. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
5. После сдачи в карантин судья определяет сектор старта и направление в секторе старта для оставшегося робота.
6. В начале соревновалельного дня определяются сектора сервисного обслуживания (сектора, из которых 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 самостоятельно.
7. Сектора сервисного обслуживания являются одинаковыми для всех попыток.
8. Сектора сервисного обслуживания: первый — “$(2, 1)$ 3”, второй — “$(2, 5)$ 3”, третий — “$(5, 2)$ 1”.
9. Сектора старта могут быть различными в разных попытках.
10. Правила именования файлов с программой управления:
    (a) для первой попытки: `part5_1.js`.
    (b) для второй попытки: `part5_2.js`.
11. Максимальное время выполнения одной попытки - 5 минут.
12. 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.
    (b) **Реальный робот:** Роботы располагаются в секторах старта, задача определить своё местоположение, распознать arTag метки располо-

женные с секторах сервисного обслуживания, и доехать до финиша.

i. Все роботы определили своё местоположение, остановившись, издали звуковой сигнал и вывели на экран свои координаты в формате $(X;Y)$ и через 10 секунд продолжили движение — 14 баллов.

ii. Один из роботов смог распознать один из arTag маркеров, издал звуковой сигнал, находясь в секторе сервисного обслуживания, вывел на экран номер робота для которого предназначется данный сектор финиша и его координаты в формате $N\ (X, Y)$, где N — номер робота, и через 10 секунд продолжил движение — 10 баллов.

iii. Все arTag маркеры были распознаны согласно предыдущему пункту — 14 баллов.

iv. Один из роботов доехал до сектора финиша и вывел `finish` — 10 баллов.

v. Все роботы достигли соотвествующх секторов финиша и вывели `finish` — 12 баллов.

13. За подзадачу начисляется только половина возможных баллов, если не было засчитано решение подзадачи в симуляторе.
14. Баллы за все попытки в каждой подзадаче суммируются.
15. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 4 балла.
16. Максимальное количество баллов за этап — 132.

7.7. 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;
```

```
25
26  var readGyro = brick.gyroscope().read
27
28  function readYaw() {
29    return -readGyro()[6];
30  }
31
32  var direction = 0; // absolute angle of direction movement
33  var directionOld = 0;
34  var azimut = 0; // we should go on azimut or turn to it
35  print("-----------------------------------------");
36
37  eLeft.reset();
38  eRight.reset();
39
40  var el = eLeft.readRawData();
41  var er = eRight.readRawData();
42  mLeft(0);
43  mRight(0);
44
45  //инициализация и калибровка гироскопа
46  brick.gyroscope().calibrate(12000);
47  script.wait(13000);
48  print("gyro inited");
49  brick.display().addLabel(30, 10, "Start!");
50  brick.display().redraw();
51  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);
112 }
113
114 // проезд в перед на количество ячеек _kceel со скоростью _v
115 // выравниваясь по гироскопу на угол azimut
116 function forward(_v, _kcell) {
117   print("azimut = " + azimut);
118   _alpha = azimut;
119   var _vel = _v == undefined ? 40 : _v; // скорость по умолчанию
120   var u = 0;
121   eLeft.reset();
122   eRight.reset();
123   var el = Math.abs(eLeft.readRawData());
124   while (Math.abs(eLeft.readRawData()) < (el + (_kcell * cellLength))) {
125     u = 1.5 * (_alpha - direction / 1000);
126     mLeft(_vel - u);
127     mRight(_vel + u);
128     script.wait(5);
129   }
130   brick.motor(M3).powerOff();
131   brick.motor(M4).powerOff();
132   script.wait(300);
133 }
134
135 min = function(a, b) {
136   return a < b ? a : b;
137 }
138 max = function(a, b) {
139   return a > b ? a : b;
140 }
141
142 //============================================================================
143 // массив для изображения
144 var pic = [];
```

```
145
146 var marker_size = 5;
147 var total_width = 320 / 2;
148 var total_height = 240 / 2;
149 var prob = 25 * 5 * 5;
150
151 function getColor(pic, x, y) {
152   return pic[y * total_width + x];
153 }
154
155 function squareAverage(pic, x, y, diam) {
156   var sum = 0;
157   var start_row = y * total_width;
158   var end_row = start_row + diam * total_width;
159
160   for (var index = start_row + x; index < end_row; index += total_width)
161     for (var j = 0; j < diam; j += 1)
162       sum += pic[index + j];
163
164   return sum;
165 }
166
167 // lu - left-up corner. Coordinates: (x, y)
168 // ld - left-down corner
169 // ru - right-up corner
170 // rd - right-down corner
171 function getCenterColor(pic, lu, ld, ru, rd, diam) {
172   var x = (lu[0] + ld[0] + ru[0] + rd[0]) >> 2;
173   var y = (lu[1] + ld[1] + ru[1] + rd[1]) >> 2;
174   var color = squareAverage(pic, x - diam, y - diam, diam << 1);
175   return color;
176 }
177
178 function findGridCorners(corners, marker_size) {
179   var grid_corners = [];
180
181   var vertical_lines = [];
182   var upper_line_x1 = corners[0][0];
183   var upper_line_y1 = corners[0][1];
184   var upper_line_x2 = corners[2][0];
185   var upper_line_y2 = corners[2][1];
186   var down_line_x1 = corners[1][0];
187   var down_line_y1 = corners[1][1];
188   var down_line_x2 = corners[3][0];
189   var down_line_y2 = corners[3][1];
190
191   var mks = 1.0 / marker_size;
192   var k_ux = (upper_line_x2 - upper_line_x1) * mks;
193   var k_uy = (upper_line_y2 - upper_line_y1) * mks;
194   var k_dx = (down_line_x2 - down_line_x1) * mks;
195   var k_dy = (down_line_y2 - down_line_y1) * mks;
196
197   for (var i = 0; i < marker_size + 1; i += 1) {
198
199     var up_x = upper_line_x1 + k_ux * i;
200     var up_y = upper_line_y1 + k_uy * i;
201
202     var down_x = down_line_x1 + k_dx * i;
203     var down_y = down_line_y1 + k_dy * i;
204
```

```
    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;
    }
```

```
385       mapColorToLetter[i] = grayscale[curBin];
386     }
387   }
388
389   // Возвращает значение ARTag
390   function getARTagValue() {
391     source_pic = getPhoto();
392
393     // init pic, grayscale mode
394     for (var i = 0; i < total_height; i += 1) {
395       for (var j = 0; j < total_width; j += 1) {
396         var x = (j + i * scale * total_width) * scale;
397         var p = source_pic[x];
398         p = (((p & 0xff0000) >> 18) + ((p & 0xff00) >> 10) + ((p & 0xff) >> 2));
399         pic[j + i * total_width] = p;
400       }
401     }
402     calculateHistogram();
403     getRange();
404     initMapColorToLetter();
405
406     var thresh = threshold2(rangeBins[1], pic, total_height, total_width);
407     var corners = findCorners(thresh, 9);
408     var grid_corners = findGridCorners(corners, marker_size)
409     var values = detectCode(thresh, grid_corners, 3);
410
411     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],
```

```
  [5, 14, 21, -1],
  [-1, 15, -1, 13],
  [7, -1, -1, 14],
  [8, 17, -1, -1],
  [-1, 18, -1, 16],
  [10, -1, 26, 17],
  [11, 20, 27, -1],
  [-1, 21, -1, 19],
  [13, -1, 29, 20],
  [-1, -1, -1, -1],
  [-1, -1, 31, -1],
  [-1, -1, 32, -1],
  [-1, -1, -1, -1],
  [18, -1, 34, -1],
  [19, -1, -1, -1],
  [-1, -1, -1, -1],
  [21, 30, 37, -1],
  [-1, 31, -1, 29],
  [23, -1, 39, 30],
  [24, 33, 40, -1],
  [-1, 34, -1, 32],
  [26, 35, 42, 33],
  [-1, 36, -1, 34],
  [-1, 37, 44, 35],
  [29, -1, 45, 36],
  [-1, -1, -1, -1],
  [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;
```

```
505 var destroyedSectors = 0;
506
507 // получаем маршрут перемещения до необходимой точки из текущей
508 // в нотации F, L, R
509 function getPath(finishSector) {
510   from = [];
511   for (var i = 0; i < 64; i++) {
512     from[i] = [];
513     for (var j = 0; j < 4; j++)
514       from[i][j] = "N";
515
516   }
517
518   var queue = [];
519   queue.push([positionOfRobot, directionOfRobot]);
520   var finishDir = 0;
521   while (queue.length > 0) {
522     temp = queue.shift();
523     currentSector = temp[0], currentDir = temp[1];
524     if (currentSector == finishSector) {
525       finishDir = currentDir;
526       break;
527     }
528     // Вперед
529     if (lab[currentSector][currentDir] > -1) {
530       adjSector = lab[currentSector][currentDir];
531       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
712   // Количество обрушенных секций
713   destroyedSectors = 2;
714   positionOfRobot = xStart + yStart * 8;
715
716   travelThroughoutMaze();
717   follow_path(xARTag + yARTag * 8);
718   turnTo((directionOfARTag + 3) % 4);
719
720   //detecting ARTag markers until it isn't successful
721   value1 = 0;
722   value2 = 0;
723   dist = 0.3;
724   while (value1 < 8 && value2 < 8 || value1 > 7 && value2 > 7) {
725     readARTag(dist);
726     dist += 0.05;
727   }
728
729   // Координаты финиша
730   xFinish = 0;
731   yFinish = 0;
732   if (value1 < 8) {
733     xFinish = value1;
734     yFinish = value2 - 8;
735   } else {
736     xFinish = value2;
737     yFinish = value1 - 8;
738   }
739
740   print(xFinish + " " + yFinish);
741   finishSector = xFinish + yFinish * 8;
742   follow_path(finishSector);
743   print(countUnavailable);
744   return;
```

```
}
```

7.8. Критерии определения команды-победителя командного тура

1. Максимальный балл за командный этап — 400 баллов.
2. Сумма баллов, набранных за решения подзадач всех этапов командной части финального этапа, определяет итоговую результативность команды (измеряемую в баллах).
3. Команды ранжируются по результативности.
4. Команда победитель определяется, как команда с максимальной результативностью.

4. КРИТЕРИИ

Критерии определения победителей и призеров заключительного этапа

8.1. Первый отборочный этап

В первом отборочном этапе участники решали задачи по двум предметам - математика и информатика, в каждом предмете максимально можно было набрать 100 баллов. Для того, чтобы пройти во второй этап участники должны были набрать в сумме по обоим предметам:

- 9 классы и младше — не менее 80 баллов.
- 10-11 класс — не менее 100 баллов.

8.2. Второй отборочный этап

Количество баллов, набранных при решении всех задач, суммируется. Призерам второго отборочного этапа было необходимо набрать не менее 40 баллов.

Победители второго отборочного этапа являются финалистами олимпиады и должны были набрать 46 баллов и выше независимо от уровня.

8.3. Заключительный этап

В заключительном этапе олимпиады баллы участника складываются из двух частей:

1 - участник получает баллы за индивидуальное решение задач по предметам (математика и информатика)

2 - участник получает баллы за командное решение практической задачи.

Итоговый личный балл участника рассчитывается по формуле:

$$S = Si + Sm + St, \text{ где:}$$

Si — сумма баллов по информатике, набранная в рамках индивидуальной части заключительного этапа (максимум 300 баллов), нормированная на 100;

Sm — сумма баллов по математике, набранная в рамках индивидуальной части заключительного этапа (максимум 100 баллов);

St — количество баллов, набранное в рамках командной части заключительного этапа (максимум 400 баллов).

Критерий определения победителей и призеров:

Категория	Количество баллов
Победители	160 баллов и выше
Призеры	От 140 до 159 баллов