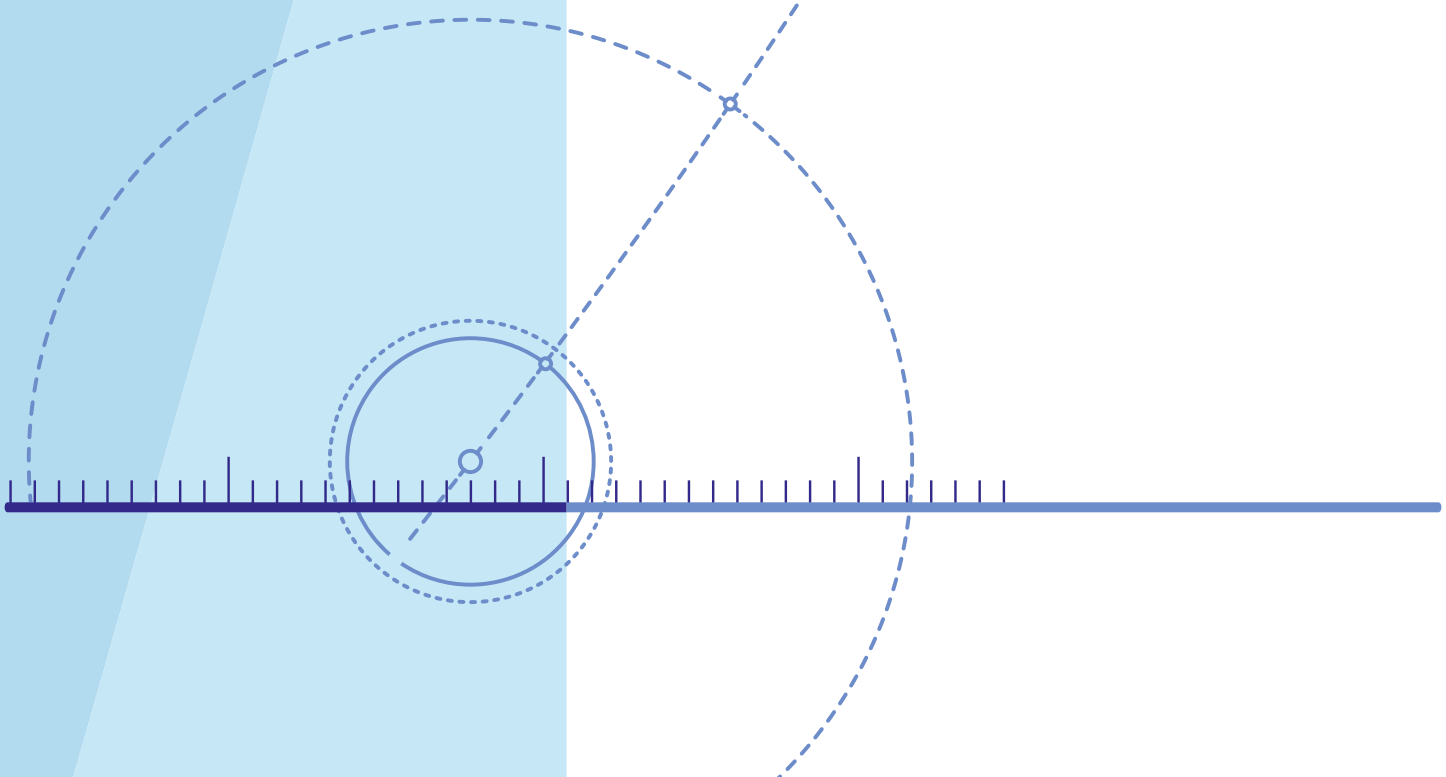
Олимпиада НТИ

Сборник задач
Олимпиады НТИ – 2017
по профилю:
«Интеллектуальные
робототехнические системы»



ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Олимпиада Национальной технологической инициативы[1] (далее – Олимпиада НТИ) – это командная инженерная олимпиада школьников, завершающаяся разработкой действующего устройства, системы устройств или компьютерной программы. Олимпиада является проектом Агентства стратегических инициатив, элементом дорожной карты НТИ «Кружковое движение» и ключевым механизмом вовлечения инженерно-ориентированных школьников в образовательные программы высшего образования, ориентированные на рынки НТИ. 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», «Интеллектуальные робототехнические системы», «Беспилотные авиационные системы», «Ядерные технологии», «Нанотехнологии (Современные структуры и материалы)», «Технологии беспроводной связи» и «Электронная инженерия: Умный дом».

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.

Олимпиада НТИ была впервые проведена в 2015/16 учебном году. В отборочных этапах олимпиады приняли участие несколько тысяч школьников, около ста из них были приглашены к участию в заключительном этапе.

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

Заключительный этап Олимпиады и торжественные мероприятия проводились на площадке ОЦ «Сириус» в лабораториях и помещениях Парка Наук и Искусств. Вечером проходили лекции и встречи от представителей технологических компаний для участников.

[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.

Организаторы и партнеры Олимпиады НТИ – 2017

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

Вузы-организаторы олимпиады:

- ФГБОУ ВО «Московский политехнический университет»;
- ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»;
- ФГАОУ ВО «Национальный исследовательский Томский политехнический университет»;
- ФГАОУ ВО «Дальневосточный федеральный университет»;
- АНО ВО «Университет Иннополис»;
- ФГАОУ ВПО «Национальный исследовательский ядерный университет «МИФИ»;
- ФГАОУ ВО «Московский физико-технический институт (государственный университет)»;
- ФГАОУ ВО «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики» (Университет ИТМО);
- ФГБОУ ВО «Московский авиационный институт (национальный исследовательский университет)»;
- ФГБОУ ВО «Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»;
- ФГАОУ ВО «Новосибирский национальный исследовательский государственный университет».

К работе методической комиссии был привлечен профессорско-преподавательский состав вузов-организаторов и представители реального сектора экономики. Объективную оценку работы осуществляет жюри, представленное основателями технологических компаний, а также представителями вузов-организаторов. Вузы-организаторы, входящие в Оргкомитет Олимпиады НТИ, ведут непрерывную работу с талантливыми школьниками.

Московский политехнический университет:

Московский политехнический университет при активном участии Инженерной школы (факультета) регулярно ведет мероприятия для школьников. 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. Стратегическим партнером в этой области является образовательный фонд «Талант и успех». Летом 2016 года Московский Политех выступил соорганизатором трех направлений проектной деятельности в ОЦ «Сириус» и осуществил экспертную поддержку деятельности ОЦ «Сириус» по направлениям «Транспорт» и «Космос». В ноябре 2016 г. участники всероссийского форума в г. Ярославле «Будущие интеллектуальные лидеры России» имели возможность решать кейсы под руководством сотрудников Московского Политеха, в том числе в рамках подготовки к Олимпиаде. В планах факультета - создание инженерных кружков Московского Политеха на базе

Сириуса (радиотехника, аэрокосмическая инженерия и беспилотные транспортные системы). Под курированием понимаются регулярные встречи и помощь как детям, так и педагогам.

Инженерная школа (факультет) является соорганизатором инженерно-конструкторских школ «Лифт в будущее» - программа БФ «Система» по поддержке талантливой молодежи. В течение месяца вместе с преподавателями Московского Политеха в ВДЦ «Орленок» дети разрабатывают настоящие технологические проекты и пробуют себя в роли стратегов, генеральных конструкторов и инженеров.

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

- Кружок прототипирования;
- Кружок радиоэлектроники и программирования;
- Автомобильный кружок;
- Кружок транспортной робототехники.

Вместе с тем Московский Политех принимает участие в организации других олимпиад, входящих в перечень олимпиад школьников Минобрнауки России 2016/2017 учебного года:

- Многопрофильная инженерная олимпиада «Звезда». Основная цель олимпиады — развитие и стимулирование интереса у обучающихся к научно-исследовательской и инженерной деятельности, формирование целостного представления о приоритетных направлениях финансово-экономического развития страны и мотивации к поступлению на инженерные специальности
- Объединенная межвузовская математическая олимпиада (ОММО). Проводится для одиннадцатиклассников по инициативе группы московских вузов с 2009 года.
- 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.

Также Московский Политех является организатором ряда олимпиад, не входящих в перечень Минобрнауки России:

- Олимпиада для школьников «Технология художественной обработки материалов» проводится в РФ и странах СНГ с 2013 г.
- Олимпиада для учащихся общеобразовательных школ по профильной дисциплине «Инженерная графика» имени О.В. Гордона проводится в РФ и странах СНГ с 2015 года
- Олимпиада «Рисунок автомобиля» проводится в РФ и странах СНГ с 2011 г.
- Олимпиада «Промышленный дизайн» проводится в РФ и странах СНГ с 2016 г.

Московский Политех также участвует (имеет статус организатора или соорганизатора) в следующих мероприятиях: инженерно-конструкторское направление предпрофессиональной олимпиады Московской олимпиады школьников 2016/17 гг., предпрофессиональный экзамен для инженерных классов в Москве 2017 г., инженерное направление Московского конкурса научно-

исследовательских и проектных работ учащихся, проектные смены ОЦ «Сириус», турнир двух столиц по робототехнике и т. д.

Санкт-Петербургский политехнический университет Петра Великого:

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

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

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

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

Томский политехнический университет:

Томский политехнический университет является одним из 9 опорных вузов в программе инновационного развития ОАО «Газпром». В 2009 году ТПУ и ОАО «Газпром» подписали соглашение о сотрудничестве. В Институте природных ресурсов создан и действует полный учебно-научный цикл. Он включает обучение (рабочим профессиям, прохождение производственных преддипломных практик, переподготовку и повышение квалификации специалистов компании, создание кадрового задела) и исследования (Инновационный научно-образовательный центр трубопроводного транспорта нефти и газа, в составе которого действует Международная научно-образовательная лаборатория «Нефтегазовая гидродинамика и теплообмен», лаборатория неразрушающего контроля).

Участие со своим треком «Электронная инженерия: Умный дом» в рамках Олимпиады НТИ стало логичным продолжением работы Томского политехнического университета по развитию инженерных навыков у школьников и студентов. На траектории элитного технического образования ТПУ студенты воплощают в жизнь свои самые смелые инженерные идеи. Но многолетний опыт показывает, что деятельность по развитию интереса к инженерии и закладыванию базовых знаний в предметной области нужно прививать гораздо раньше, поэтому Томский политехнический университет ведет постоянную работу по привлечению в инженерные профессии талантливых школьников. Одним из таких мероприятий стал проект «Юный инженер», который проводится

совместно с Департаментом образования Томской области. Суть проекта в том, чтобы студенты ТПУ, уже имеющие опыт в реализации инженерных проектов, готовили и проводили учебные занятия со школьниками г. Томска и Томской области. Цель проекта сделать доступным современное дополнительное образование для детей всего региона.

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

«Агентам» предстоит выполнять специальные задания не только в виртуальном, но и в живом режиме, поучаствовать в живой миссии, которую необходимо выполнить на базе ТПУ. Это может быть реальное событие — проходящая в вузе конференция, за ходом которой школьники будут следить в Сети, и так далее.

Многопользовательский интерфейс позволяет игрокам общаться друг с другом с помощью чатов внутри игры, а также выполнять совместные задачи.

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

Со школьниками проводится много других интересных и полезных мероприятий: «Университетские субботы» - проект, позволяющий школьнику вникнуть в естественные и точные науки (физика, химия, математика, информатика), узнать о потенциале ТПУ, его лабораториях и передовых разработках.

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

Количество учащихся в 2016 г.:

- курсы ТПУ – 430 чел.;
- заочные курсы – 125 чел.;
- комплексные учебные курсы – 320 чел.

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

Томский политехнический университет принимает участие в организации олимпиад, входящих в перечень олимпиад школьников Минобрнауки России 2016/2017 учебного года:

- Интернет-олимпиада школьников по физике, целью олимпиады является повышение качества знаний по физике, повышение мотивации к поступлению в технические вузы. С 30.03.2017 г. олимпиада получила статус международной.

- Многопрофильная инженерная олимпиада «Звезда». Участие в данной олимпиаде дает возможность школьнику существенно расширить свой кругозор, получить полное представление о тенденциях развития экономики и техники.
- Межрегиональная олимпиада школьников по математике САММАТ.
- Олимпиада школьников Сибирского Федерального округа «Будущее Сибири» - мотивирует школьников для поступления в технические вузы и расширяет знания в предметной области.
- Межрегиональная олимпиада школьников «Высшая проба» дает возможность учащимся попробовать себя в разных областях знаний, как в технических, так и в гуманитарных, что способствует гармоничному развитию личности.

Также ТПУ является соорганизатором таких олимпиад как:

- Политехническая олимпиада по математике, физике, химии, информатике, русскому языку, обществознанию, проводимая на территории стран СНГ. Данная олимпиада нацелена на выявление талантливых абитуриентов в странах СНГ, развитие и проверку глубоких знаний по предложенным дисциплинам.
- Олимпиада ПАО «Газпром», направленная на развитие креативного подхода к решению инженерных задач, проверки знаний и мотивации к изучению нефтегазового дела.

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 довузовской подготовки Университет проводит большое количество образовательных мероприятий и интеллектуальных конкурсов с целью выявления и развития талантливых школьников:

- Школы олимпиадной подготовки по информатике, математике, робототехнике.
- Открытая олимпиада Университета Иннополис по математике и информатике (с 2016 года входят в перечень РСОШ)
- Всероссийская робототехническая олимпиада, с 2014 года Университет является 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 5 — 11 классов
- Олимпиада по финансовым информационным технологиям
- Университет активно работает со школьными учителями и педагогами дополнительного образования, организуются курсы подготовки для педагогов по олимпиадной робототехнике, программированию, педагогические конференции.

Московский физико-технический институт:

В рамках непрерывной работы со школьниками Московский физико-технический институт (МФТИ) регулярно ведет мероприятия для школьников, такие как летние и

зимние школы (Международная физико-математическая школа «Phystech.International» – 200 участников, Летняя школа «Прикладные математика и физика» - 400 участников, Аэрокосмическая школа МФТИ – 80 участников, Электронно-молекулярная школа МФТИ – 80 участников), дни открытых дверей (8 января 2017 года – 1 300 участников, 9 апреля – 2 000 участников), конференции (Международная научно-техническая конференция школьников «Старт в науку» - 1100 участников), а также принимает участие в организации других олимпиад, входящих в перечень олимпиад школьников Минобрнауки России 2016/2017 учебного года:

- Олимпиада «Физтех» (профиль олимпиады – физика и математика)
- Открытая химическая олимпиада (профиль олимпиады – химия)
- Открытая олимпиада школьников по программированию (профиль олимпиады – информатика)
- Олимпиада Курчатов (математика и физика)
- Олимпиада школьников по программированию "ТехноКубок" (информатика)

Также МФТИ является организатором ряда олимпиад (2016/2017), не входящих в перечень Минобрнауки России:

- Выездная физико-математическая олимпиада школьников, является одним из отборочных этапов на олимпиады «Физтех», в ней приняло участие 28 000 участников;
- Онлайн-этап олимпиады Физтех – 47 000 участников;
- Столичная физико-математическая олимпиада МФТИ – 4 000 участников;
- Международная физико-математическая олимпиада Phystech.International 2016 – 5 900 участников.

НИЯУ МИФИ:

Национальный исследовательский ядерный университет «МИФИ» является организатором большого числа олимпиад и конкурсов, в том числе входящих в Перечень олимпиад РСОШ в 2016/2017 учебном году. Среди них:

- Отраслевая физико-математическая олимпиада «Росатом» (№ 71 Перечня, физика – 1 уровень, математика – 2 уровень). Предметная олимпиада, очные туры которой проходят более чем в 35 городах РФ и трех странах ближнего зарубежья. В 2016/2017 году в олимпиаде приняло участие более 19 000 школьников 7-11 классов
- Инженерная олимпиад школьников (№ 17 Перечня, физика - 1 уровень). Проводится НИЯУ МИФИ совместно с СПБГЭТУ «ЛЭТИ», НГТУ им. Р.Е. Алексеева, МГУПС (МИИТ) и Самарским университетом. В олимпиаде принимает участие более 7 000 школьников 9-11 классов. Олимпиада проходит в более, чем 20 городах РФ (в том числе в городах-спутниках АЭС) и трех странах ближнего зарубежья. Победители и призеры Олимпиады обладают преимущественным правом при получении целевых направлений на обучения в вузах РФ от АО «Концерн «Росэнергоатом».
- Конкурс научных работ школьников «Юниор» (№ 10 Перечня, естественные науки – 3 уровень, инженерные науки – 3 уровень). Участники Конкурса представляют результаты своих научных работ по 6 секциям: физика, математика, робототехника, биология и экология, информатика, химия. Ежегодно в Конкурсе принимают участие более 500 школьников из большинства регионов РФ.

НИЯУ МИФИ вместе с рядом вузов также принимает участие в организации олимпиад:

- Объединенная межвузовская математическая олимпиада (№43 Перечня, 2 уровень)
- Многопрофильная инженерная олимпиада «Звезда» (Техника и технологии, №39 Перечня, 3 уровень). Секция: «Ядерные технологии»
- Московская предпрофессиональная олимпиада (технологическое направление). Совместно с НИТУ «МИСиС» и МФТИ.

Новосибирский государственный университет:

НГУ – классический университет, главная задача которого заключается в подготовке интеллектуальной элиты для науки, образования, наукоёмкого производства, бизнеса на основе фундаментального образования, позволяющего выпускникам быстро адаптироваться к меняющимся потребностям общества.

Основная цель НГУ в рамках довузовской подготовки - выявление детей, проявивших способности к науке и создание условий для развития творческих способностей школьников, их самостоятельности, интереса к научной деятельности.

Для этих целей был создан СУНЦ НГУ. Специализированный учебно-научный центр физико-математического и химико-биологического профиля является структурным подразделением НГУ. В СУНЦ НГУ 500 учащихся проживают в условиях интерната. Их обучают более 260 высококвалифицированных преподавателей, в их числе: 2 академика РАН, 1 академик РАО, 24 доктора наук, 98 кандидатов наук, 19 профессоров, 83 доцента. Более половины преподавателей – ученые Сибирского отделения РАН, преподаватели Новосибирского государственного университета.

В составе СУНЦ НГУ уже 50 лет работает Заочная школа, в которой ежегодно обучаются более 2000 учащихся 5-11 классов из 40 регионов Сибири, Урала, Дальнего Востока и стран СНГ на 8 отделениях (математика, физика, химия, биология, русский, английский, немецкий, французский язык). В процессе дистанционного обучения используются электронные образовательные ресурсы, разработанные в СУНЦ НГУ.

СУНЦ НГУ ежегодно входит в ТОП-25 в рейтинге лучших школ России по данным МЦНМО.

Среди выпускников ФМШ около 4 тысяч кандидатов наук, более 500 докторов наук, 7 членов-корреспондентов РАН, 4 академика РАН и академик РАО, члены других академий. Многие выпускники оказывают существенное влияние на развитие отечественной науки, они занимают лидирующие позиции в научно-исследовательских институтах Российской академии наук, являются руководителями ведущих научных школ. Среди выпускников ФМШ-СУНЦ НГУ - организаторы крупных производств, компаний и банков, высококвалифицированные специалисты в сфере финансов и инновационного бизнеса.

С каждым годом, на базе НГУ проводятся все больше олимпиад и интеллектуальных турниров, в которых принимают участие школьники и студенты не только со всей России, но и со всего мира:

- Всероссийская предметная олимпиада школьников – на базе НГУ проводится региональный этап по естественным наукам и информатике.
- Всесибирская открытая олимпиада школьников - проводится на базе НГУ и СУНЦ НГУ, входит в список РСР. Основной задачей Всесибирской олимпиады является привлечение школьников к изучению естественнонаучных предметов. Особенностью олимпиады является также то, что призеры олимпиады приглашаются в Летнюю физико-математическую школу, проводимую в Академгородке (г. Новосибирск), по результатам

обучения в которой старшеклассники принимаются в физико-математическую школу, Специализированный учебно-научный центр Новосибирского государственного университета. Олимпиада проводится по следующим профильным предметам: математике, физике, химии, биологии и информатике (им. И.В. Поттосина).

- Турнир городов – международная олимпиада по математике для школьников. Особенность Турнира городов в том, что он ориентирует участников не на спортивный успех, а на углублённую работу над задачей, т. е. развивает качества, необходимые в исследовательской работе. Проводится на базе НГУ, входит в список РСР.
- NSUCRYPTO – международная олимпиада по криптографии. Проводится на базе НГУ совместно с Институтом математики им. С.Л.Соболева. Цель олимпиады – привлечь молодых исследователей к решению математических задач современной криптографии, в том числе - нерешенных.

Университет ИТМО:

Соорганизатор Олимпиады Санкт-Петербургский Национальный Исследовательский Университет Информационных технологий, механики и оптики (ИТМО) с целью популяризации естественных наук и привлечения талантливых абитуриентов ежегодно проводит ряд мероприятий:

- Конгресс молодых ученых;
- олимпиады для школьников, в том числе, входящие в перечень Министерства образования и науки РФ (Открытая олимпиада школьников «Информационные технологии», Открытая олимпиада школьников по математике, Интернет-олимпиада школьников по физике, Объединенная межвузовская математическая олимпиада школьников, Интернет-олимпиада по информатике);
- организовано взаимодействие с региональными профильными образовательными учебными заведениями среднего образования;
- в различных школах и кружках, организованных на базе Университета ИТМО ежегодно принимает участие более 1000 школьников из Санкт-Петербурга и Ленинградской области;
- для привлечения абитуриентов и информирования их о направлениях подготовки в Университете ИТМО, создан массовый онлайн-конкурс под названием «Физика: игра света», размещенный в настоящее время на портале Newtonew;
- в январе 2017 г. Университет ИТМО принял участие в организации и проведении Балтийского научно-инженерного конкурса – одного из самых крупных научных соревнований для школьников в России, сочетающего в себе строгое судейство научных проектов учеными и преподавателями вузов и современные традиции в организации научных молодежных праздников.
- на сайте Школьной лиги Роснано в рамках проекта «Школа на ладони» проведен конкурс «Как с помощью лазерной указки заглянуть в наномир» в котором приняли участие более 100 школьников из различных регионов России.
- в июле 2017 года стартует новая проектная смена образовательного центра «Сириус». В этом году Университет ИТМО совместно с компанией НТ-СПб подготовил в рамках смены сразу четыре задания по профилю «Микромир и зондовая микроскопия».

- В Университете ИТМО ведется активная работа со школьниками с самого младшего возраста. Для этого созданы различные школы, кружки, коворкинг, проводится День открытых дверей для школьников 5-10 классов ITMO.START. В частности, на базе физико-технического факультета Университета ИТМО в период летних каникул 2017 г. планируется проведение летних научных практикумов для школьников из Санкт-Петербурга.
- Молодежный коворкинг-центр Университета ИТМО – это регулярная бесплатная образовательная программа по развитию личностных навыков у школьников 8-10 классов, которая включает в себя обучающие курсы и мастер-классы по таким направлениям, как: основы робототехники; составление интеллект-карт; основы программирования; фотоискусство и др.
- На базе кафедры графических технологий работает Детско-юношеский компьютерный центр. Ученики центра получают начальные знания в области компьютерного дизайна и мультипликации, программирования, 3D-моделирования, web-проектирования, видеомонтажа и других аналогичных сфер. Это позволяет школьникам, заинтересованным в графических технологиях, определиться с выбором профессии и подготовиться к поступлению на соответствующую кафедру Университета ИТМО.
- Школа лазерных технологий Университета ИТМО с 2010 года знакомит учеников старших классов с лазерами и оптической техникой, рассказывает об их применении в современной науке, а также дает возможность школьникам почувствовать себя студентами путем привлечения учащихся к научной деятельности. Учебная программа ШЛТ включает два курса: Сезонную и Летнюю школы лазерных технологий.

Московский авиационный институт:

Московский авиационный институт (национальный исследовательский университет) регулярно проводит мероприятия для школьников, такие как программа работы с талантливыми школьниками «Инженерный класс в московской школе», «Университетские субботы», профориентационные мероприятия, конкурсы школьных проектов: Международная молодежная научная конференция «Гагаринские чтения – секция «Юные учёные», Московский открытый Городской Конкурс проектных, исследовательских и реферативных работ школьников по астрономии и ракетной технике и космонавтике с участием регионов РФ «Через тернии к звездам». Научная программа конференции «Гагаринские чтения – секция «Юные учёные» предусматривает публикацию тезисов работ школьников 8-11 классов, занимающихся научно-техническим творчеством. Целью проведения научной секции является выявление наиболее способных и подготовленных учащихся и их привлечение к учебе в МАИ. Целью Московского открытого Городского Конкурса проектных, исследовательских и реферативных работ школьников по астрономии, ракетной технике и космонавтике с участием регионов РФ «Через тернии к звездам» является привлечение учащихся к изучению истории и достижений отечественной космонавтики и астрономии, раскрытие творческого потенциала участников конкурса, развитие навыков проведения самостоятельных исследований, использования литературных источников и современных информационных технологий, умений представить свою работу в виде публичного доклада, а также профессиональная ориентация учащихся.

Также университет принимает участие в организации олимпиад, входящих в перечень олимпиад школьников Минобрнауки России 2016/2017 учебного года:

- Многопрофильная инженерная олимпиада «Звезда». Основная цель Многопрофильной инженерной олимпиады «Звезда» — развитие и

стимулирование интереса у обучающихся к научно-исследовательской и инженерной деятельности, формирование целостного представления о приоритетных направлениях финансово-экономического развития страны и мотивации к поступлению на инженерные специальности.

- Объединенная межвузовская математическая олимпиада школьников. Ключевой задачей данной олимпиады является выявление талантливых школьников и обеспечение их сертификатами, которые повысят шанс поступления на желаемое место обучения в высшем учебном заведении. В настоящее время олимпиада проводится для школьников 9-11 классов, что позволяет не только создать условия для интеллектуального развития и поддержки одаренных детей, но и оказывает содействие в обеспечении профессиональной ориентации и продолжении образования.
- Интернет-олимпиада школьников по физике. Проводится среди учащихся 7-11 классов, которым интересна физика, математика и компьютерные технологии. Данная олимпиада помогает найти учащихся со способностями в области экспериментальной деятельности, умеющих применять на практике свои знания.

На ряду с этим МАИ является организатором ряда олимпиад, не входящих в перечень Минобрнауки России:

- Российская аэрокосмическая олимпиада школьников. Олимпиада проводится среди школьников с целью развития творческой инициативы, повышения познавательного интереса обучающихся общеобразовательных учреждений к углубленному изучению предметов.
- Олимпиада по авиации для школьников (ПАО Компания «Сухой»). ПАО Компания «Сухой» совместно с факультетами МАИ проводят олимпиаду среди школьников 10-11 классов, по результатам которой формируются группы целевого обучения для подготовки по индивидуальной программе в МАИ и в филиале ПАО Компании «Сухой» «ОКБ Сухого».
- Олимпиада имени Андрея Николаевича Туполева. ПАО «Туполев» совместно с Московским авиационным институтом (национальным исследовательским университетом) проводит для учащихся 10-11 классов олимпиаду имени А.Н. Туполева. Из участников-победителей формируются группы целевого набора для получения профессионального образования на целевых бюджетных местах в МАИ.

С целью популяризации инженерных наук и привлечения талантливых абитуриентов Московский авиационный институт (национальный исследовательский университет) проводит ряд масштабных мероприятий:

- Московский молодёжный фестиваль «МАЙский взлёт», участниками которого ежегодно становятся около 7000 старшеклассников и студентов. Цель мероприятия — привлечение внимания молодёжи к инженерным профессиям и научно-техническому творчеству в области аэрокосмических и наукоёмких отраслей промышленности; презентация инновационных идей и проектов в области авиа- и ракетостроения; популяризация научно-технических знаний среди школьников.
- День науки в МАИ. День науки позволяет абитуриентам, посетить лаборатории и ресурсные центры МАИ, тем самым узнать ближе современную

аэрокосмическую отрасль. В мероприятии ежегодно принимает участие более 1000 школьников.

- Дни открытых дверей. В день открытых дверей учащиеся имеют возможность посетить факультеты МАИ, а также узнать об особенностях обучения, студенческих проектах и перспективах обучения в МАИ. В каждом мероприятии принимает участие более 1000 учащихся.

Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева (ранее СибГАУ):

В 2016-2017 учебном году ФГБОУ ВО «СибГАУ» провел олимпиады, включенные в Перечень РСОШ:

- Герценовская олимпиада школьников (иностранные языки, биология, география). Организатор – ФГБОУ ВО «Российский государственный педагогический университет имени А.И. Герцена».
- Интернет-олимпиада школьников по физике (физика). Организатор - ФГБОУ ВО «Санкт-Петербургский государственный университет».
- Межрегиональная олимпиада МПГУ для школьников (география). Организатор – ФГБОУ ВПО «Московский педагогический государственный университет».
- Межрегиональная олимпиада школьников «Будущие исследователи – будущее науки» (биология, химия, русский язык). Организатор - ФГБОУ ВО «Нижегородский государственный университет им. Н.И. Лобачевского».
- Межрегиональная предметная олимпиада Казанского федерального университета (физика, химия). Организатор – ФГАОУ ВО «Казанский (Приволжский) федеральный университет».
- Олимпиада по химии «Юные таланты» (химия). Организатор - ФГБОУ ВО «Пермский государственный национальный исследовательский университет».
- Многопрофильная инженерная олимпиада «Звезда» (русский язык, физика-математика, история). Организатор – ФГБОУ ВО «Южно-Уральский государственный университет».
- Московская олимпиада школьников (химия). Организатор – Департамент образования города Москвы.
- Олимпиада школьников «Ломоносов» (химия). Организатор - ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова».
- Олимпиада школьников Санкт-Петербургского государственного университета (химия, история, социология, биология, обществознание, физика, математика, экономика, география, медицина, право). Организатор - ФГБОУ ВО «Санкт-Петербургский государственный университет».
- Открытая межвузовская олимпиада школьников Сибирского Федерального округа «Будущее Сибири» (физика, химия). Организатор – ФГБОУ ВО «Новосибирский государственный технический университет».
- Региональный конкурс школьников Челябинского университетского образовательного округа (иностранные языки). Организатор – ФГБОУ ВО «Челябинский государственный университет».
- Впервые на базе ФГБОУ ВО «СибГАУ» проводилась олимпиада Донского государственного университета «Я – бакалавр», междисциплинарная многопрофильная олимпиада Ассоциации инновационных регионов России (АИРР) «Технологическое предпринимательство».

С целью популяризации технических дисциплин, привлечения внимания школьников к университету, привлечения талантливых и одаренных абитуриентов в вузе проводятся мероприятия:

- Неделя техники и технологии «Магия инженерной науки». В рамках мероприятия проходят:
  - Олимпиада для школьников «Мы – зажигаем звезды» по физике, астрономии, химии;
  - Краевой технический турнир;
  - Лектории в образовательных учреждениях города;
  - Экскурсии по университету (планетарий, музей, инженерные центры);
  - Профтестирование школьников с целью помощи в выборе профессии;
  - Встреча школьников с знаменитым выпускником университета в рамках проведения круглого стола «Сто вопросов взрослому».

  К проведению привлекаются ведущие предприятия Красноярского края, Министерство образования Красноярского края, Союз машиностроителей и т.д.

  Грамоты победителей и призеров мероприятий, позволяют получить дополнительные баллы при поступлении в университет.
- Неделя химии и биологии в Красноярском крае. В рамках мероприятия проводятся:
  - Краевой химический турнир для школьников Красноярского края;
  - Региональная олимпиада по химии «Интеллектуальный химический бум»;
  - Региональная олимпиада по биологии «Биология – царица наук»;
  - Лектории и лабораторные практикумы по химии и биологии в образовательных учреждениях города;
  - Экскурсии для школьников на кафедры химического и биологического профиля с демонстрацией возможностей научных лабораторий; экскурсии в Институт леса им. В.Н. Сукачева СО РАН, в ФГБУ Государственный природный заповедник «Столбы», в Красноярский парк флоры и фауны «Роев ручей».
- Краевая зимняя политехническая школа-симпозиум «Мы - будущее России». Работа школы направлена на выявление одарённых детей, поддержку и развитие интеллектуального творчества школьников, организацию сотрудничества молодых исследователей и учёных, содействие профессиональному самоопределению молодежи. В течение недели ребята из Красноярска и других городов и поселков Красноярского края работают по различным направлениям: Космос; Биология и Экология; Химия; Физика; Информатика; Математика; Техническая; Технология деревообработки; Английский язык; Химическая технология и биотехнология; Менеджмент командной деятельности; Экономика; Бизнес–школа. Участникам школы предоставляется возможность проведения исследований в специализированных аудиториях, лабораториях, компьютерных классах университета.
- Неделя физики и информатики в Красноярском крае. Проводится с целью популяризации технических дисциплин, привлечения внимания школьников к университету, популяризация научно–технического творчества, привлечение талантливых и одаренных абитуриентов. В рамках реализации мероприятия проводятся:
  - Олимпиада по физике «Потомки Ньютона»;
  - Международный интернет-конкурс технографики и медиаресурсов;

- Олимпиада по графическим дисциплинам «Карандаш и ластик»;
- Турнир по информатике;
- Лектории и мастер-классы по физике, технике, технологии и информатике в образовательных учреждениях города;
- Экскурсии для школьников на профильные кафедры с демонстрацией возможностей научных лабораторий.

- Краевой гуманитарный турнир «Траектория успеха». Проводится опорным университетом в рамках реализации потребности вуза в профессионально ориентированных абитуриентах с высоким уровнем фундаментальной подготовки и создания оптимальных условий для выявления способных и одаренных школьников, их дальнейшего увлечения и популяризации гуманитарных наук.

  Турнир - командное соревнование, состоящее в решении нестандартных заданий по гуманитарным направлениям и защиты своих решений. В каждой команде может быть до 5 участников (с 7-го по 11-й класс). От каждого учебного заведения может быть представлено не более двух команд.
- Летняя лабораторная школа «Мир открывающихся возможностей» Работа школы направлена на выявление одарённых детей, поддержку и развитие интеллектуального творчества школьников, организацию сотрудничества молодых исследователей и учёных, содействие профессиональному самоопределению молодежи. Проводится в течение недели, базируясь на всех кафедрах университета с отрывом от учебы

Дальневосточный Федеральный Университет:

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

1. Программы дополнительного образования:

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

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

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

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

Важными мероприятиями для Университета являются краткосрочные интенсивные программы, такие как «Тихоокеанская математическая школа». В марте

2017 года такой интенсив прошли 59 школьников 10-11 классов со всего Приморского края.

2. Олимпиады и конкурсы:

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

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

- «Полуфинал Всероссийской командной школьной олимпиады по программированию»;
- Всероссийская олимпиада школьников (муниципальный и региональный этапы);
- Межрегиональная Всероссийская олимпиада школьников «Ломоносов» по биологии и математике (2016 год - 48 участников, из них – 1 победитель, 6 призеров из территорий Приморского края, Хабаровского край, Республики Саха (Якутия));
- Всероссийская «Объединенная межвузовская математическая олимпиада школьников» по математике (2016 год – 24 участника из территорий Приморского края, Хабаровского край, Амурской области);
- Многопрофильная инженерная олимпиада школьников «Звезда» по технике и технологии кораблестроения и морской техники, машиностроению, технике и технологии наземного транспорта, авиационной и ракетно-космической технике, технологии материалов, ядерной энергетике, электронике, радиотехнике и системе связи (2016 год - 250 участников (7 - 11 классы), призеры – 30, победители – 10);
- Всероссийская «Северо-Восточная олимпиада школьников» по математике, информатике и химии (2016 год - 36 участников, из них – 1 победитель, 9 призеров из территорий Приморского края, Хабаровского края, Амурской области).

Дальневосточный федеральный университет также выступил как *организатор следующих олимпиад*:

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

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

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

Олимпиада НТИ проводится при поддержке технологических партнеров, количество которых увеличилось, по сравнению с прошлым годом, среди них: АО «Российская венчурная компания» (в роли генерального партнера), НП «Лифт в Будущее», АО «Р-Фарм», АО «Роснано», ОАО ОРКК, сеть детских технопарков

«Кванториум», ООО «Спутникс», ООО Полюс-НТ, проекты Servers.ru, BiTronicsLab, STEM-игры, АНО «Агентство стратегических инициатив по продвижению новых проектов» и др. Полный список организаторов и партнеров олимпиады размещен в соответствующем разделе на официальном сайте: http://nti-contest.ru/about/.

«Глобальная цель Олимпиады НТИ – дать возможность школьникам использовать свои инженерные и научные хобби для более осознанного выбора дальнейшей профессии, – говорит Евгений Кузнецов – директор дочерних фондов АО «РВК» – это эффективный инструмент привлечения школьников в университетские образовательные программы, ориентированные на подготовку кадров для развития рынков НТИ».

Структура отбора участников Олимпиады НТИ

Соревнование проходит в три этапа. Первый и второй отборочные этапы проходили с октября 2016 по февраль 2017 в заочной форме на интернет-платформе «Stepik» (http://stepik.org) и в инженерных онлайн-симуляторах.

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

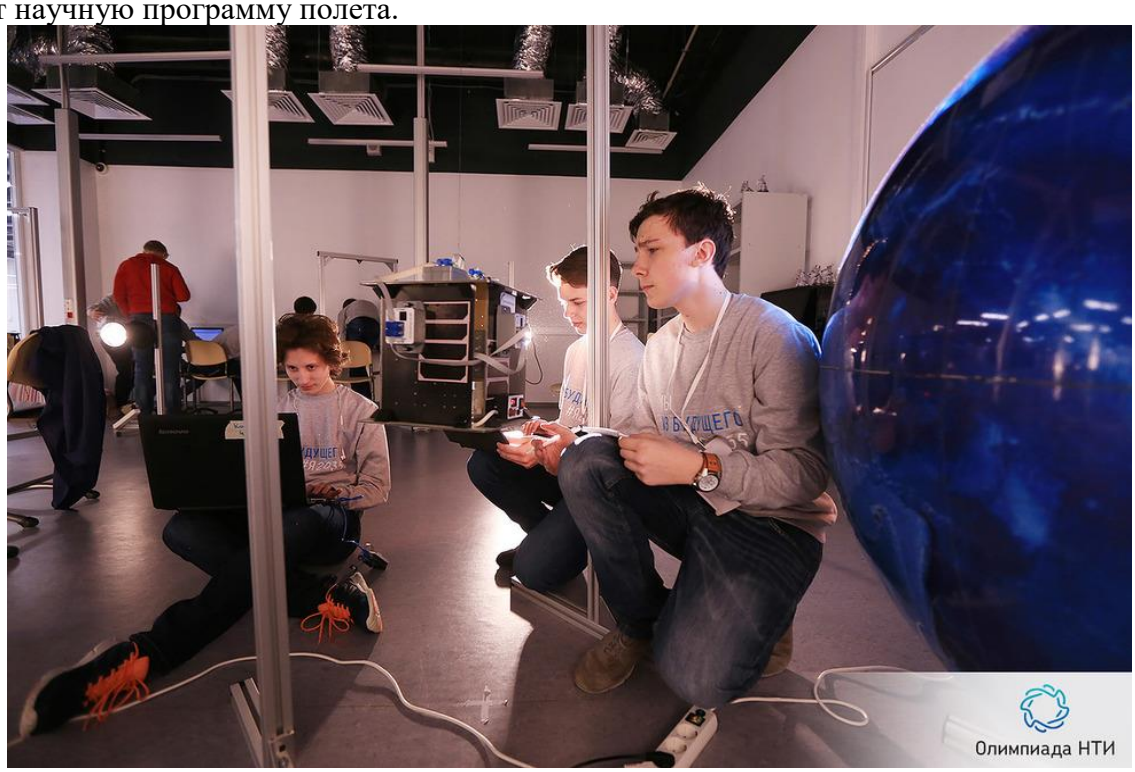
На фото: Торжественная церемония открытия заключительного этапа Олимпиады НТИ

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.

Участникам профиля «Интеллектуальные энергетические системы» необходимо создать непрерывно работающую энергосеть для модели поселка, состоящую из разных источников и накопителей энергии.

На фото: участники профиля «Интеллектуальные энергетические системы» 2016/17 года

Финалисты профиля «Системы связи и дистанционного зондирования Земли» собирают модель орбитального спутника, настраивают бортовой компьютер и системы, задают научную программу полета.

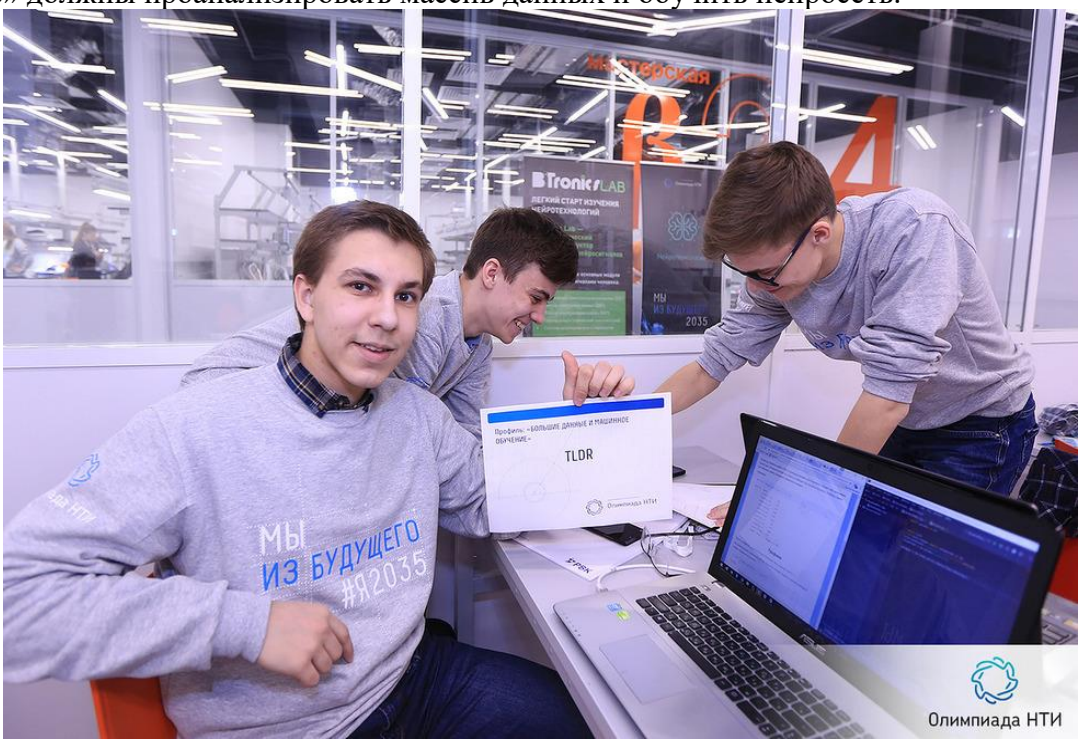


На фото: участники профиля «Системы связи и дистанционного зондирования Земли» 2016/17 года

Задание по профилю «Автономные транспортные системы» предполагает разработку действующей модели беспилотного автомобиля, роботизированного катамарана, летательного аппарата и спутника связи.

На фото: полигон для испытаний устройств профиля «Автономные транспортные системы» 2016/17 года

Участники заключительного этапа по профилю «Большие данные и машинное обучение» должны проанализировать массив данных и обучить нейросеть.

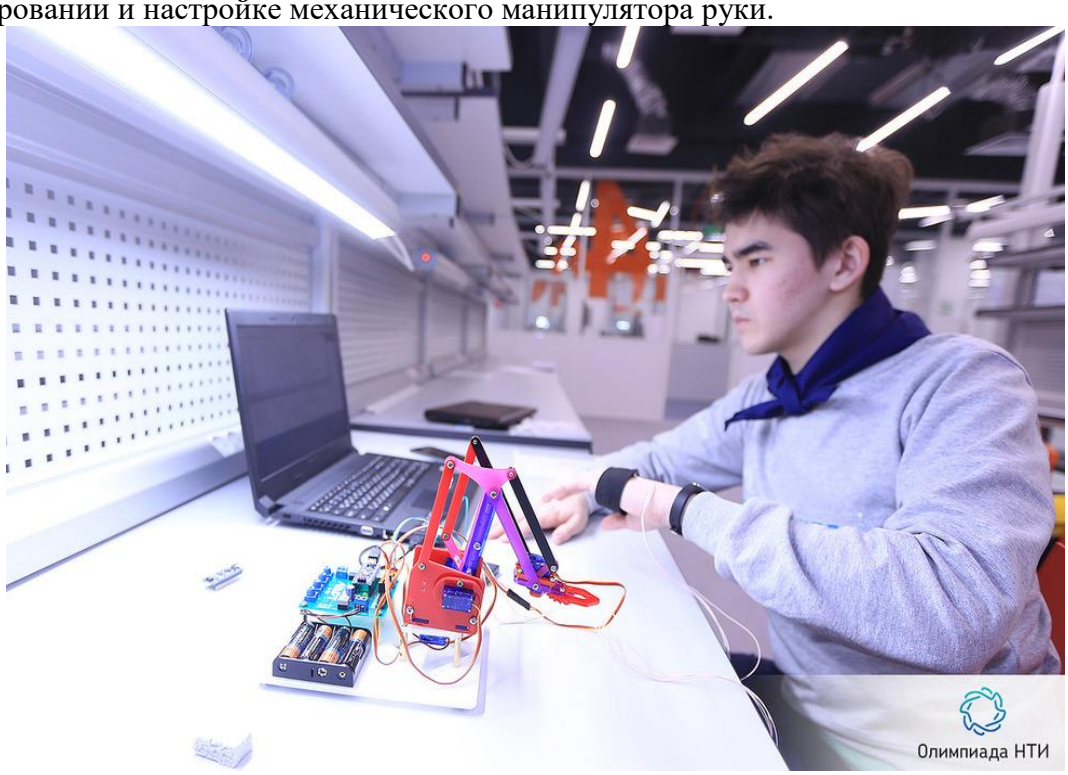


На фото: участники профиля «Большие данные и машинное обучение» 2016/17 года

Участникам финала по направлению «Беспилотные летательные аппараты» необходимо было собрать и настроить коптеры и решить задачи по работе с маршрутами.

На фото: участники профиля «Беспилотные авиационные системы» 2016/17 года во время испытания устройства

Для финалистов направления «Нейротехнологии» задача заключалась в конструировании и настройке механического манипулятора руки.



На фото: участник профиля «Нейротехнологии» 2016/17 года и собранный его командой манипулятор

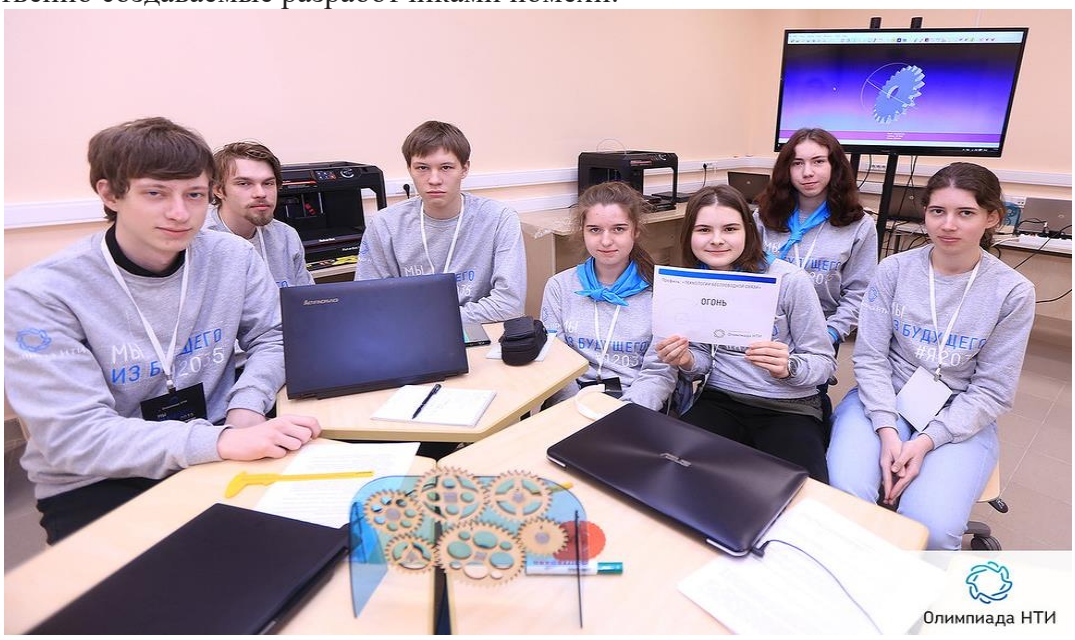
Задачи направления «Инженерные биологические системы» заключались в настройке экосистемы аквариума для уровня сложности «9 класс» и работе по получению целевого белка путем проведения клонирования и трансформации клеток при помощи методов молекулярной биологии и генной инженерии для уровня сложности «10-11 класс».

На фото: участники профиля «Инженерные биологические системы» 2016/17 года во время работы в лаборатории

Участникам профиля «Электронная инженерия: Умный дом» необходимо оборудовать умный дом системами охранно-пожарной сигнализации на основе датчиков присутствия, дыма, протечек, а также обеспечить его контролем доступа, видеонаблюдением, интеллектуальным освещением, рассчитать тепло- и энергопотребление.

На фото: стенд профиля «Электронная инженерия: Умный дом»

Участникам заключительного этапа профиля «Технологии беспроводной связи» необходимо решить задачу беспроводной безопасной передачи сложных данных в условиях зашумленного и ограниченного канала связи, а именно собрать данные о модели рабочего механизма, передать их с помощью беспроводных технологий через искусственно создаваемые разработчиками помехи.



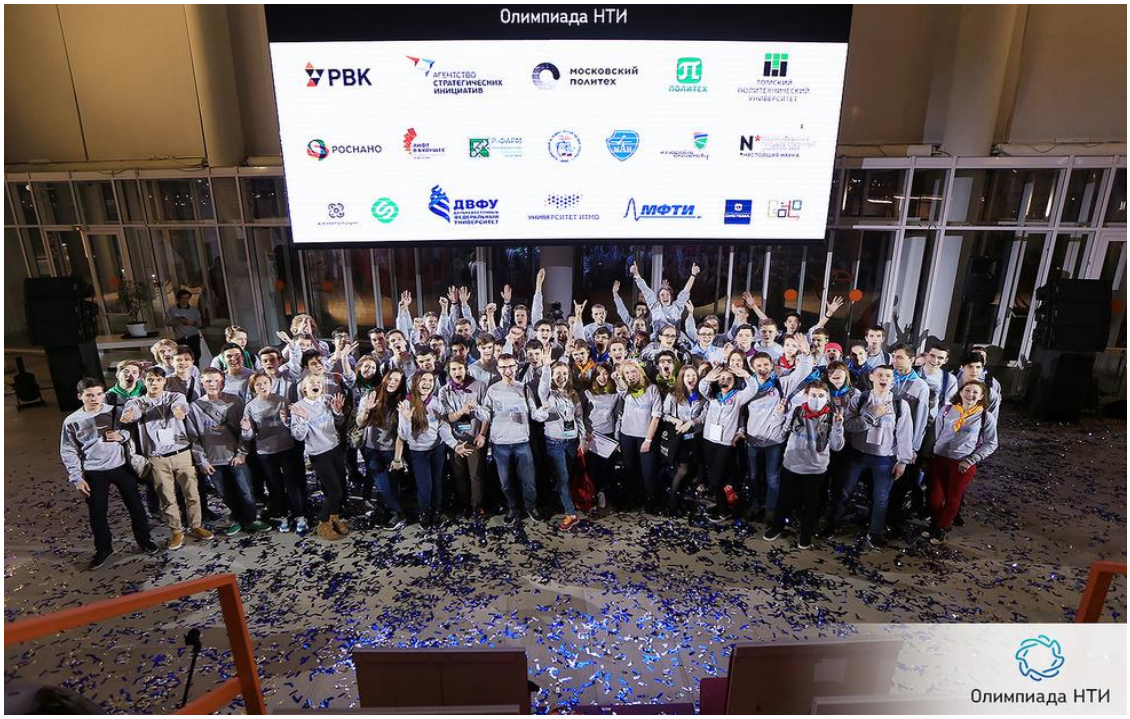
На фото: участники профиля «Технологии беспроводной связи»

Участники профиля «Интеллектуальные робототехнические системы» решают 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» 2016/17 года вокруг полигона.

Задание заключительного этапа профиля «Нанотехнологии» (Современные структуры и материалы) связано с получением фотонных пленок.

На фото: команда-участница профиля «Нанотехнологии» 2016/17 года



На фото: участники заключительного этапа Олимпиады НТИ 2016/17 во время торжественной церемонии закрытия.

Работа с участниками

Организаторы Олимпиады заинтересованы в дальнейшем сопровождении ее участников. 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.

В 2016 году четыре профиля Олимпиады НТИ («Автономные транспортные системы», «Большие данные и машинное обучение», «Системы связи и дистанционного зондирования Земли», «Интеллектуальные энергетические системы») вошли в Перечень олимпиад школьников МОН, таким образом победители и призеры этого года смогут воспользоваться льготами при поступлении в вузы России (зависит от правил приема конкретного вуза). Победители и призеры новых профилей также могут воспользоваться бонусами при поступлении в вузы-организаторы.

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

Партнерство с инженерными соревнованиями

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

Подготовка участников

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

1. «Автономные транспортные системы» - http://nti-contest.ru/profiles/transport/
2. «Системы связи и дистанционного зондирования Земли» - http://nti-contest.ru/profiles/space/
3. «Интеллектуальные энергетические системы» - http://nti-contest.ru/profiles/energy/
4. «Большие данные и машинное обучение» - http://nti-contest.ru/profiles/data/
5. «Системы беспроводной связи» - http://nti-contest.ru/profiles/telecom/
6. «Нанотехнологии (Современные структуры и материалы)» - http://nti-contest.ru/profiles/nanomaterial/
7. «Электронная инженерия: Умный дом» - http://nti-contest.ru/profiles/smarthouse/

8. «Нейротехнологии» - http://nti-contest.ru/profiles/neurotech/
9. «Интеллектуальные робототехнические системы» - http://nti-contest.ru/profiles/irs/
10. «Инженерные биологические системы» - http://nti-contest.ru/profiles/biotech/
11. «Беспилотные авиационные системы» - http://nti-contest.ru/profiles/aero/
12. «Ядерные технологии» - http://nti-contest.ru/profiles/nuclear/

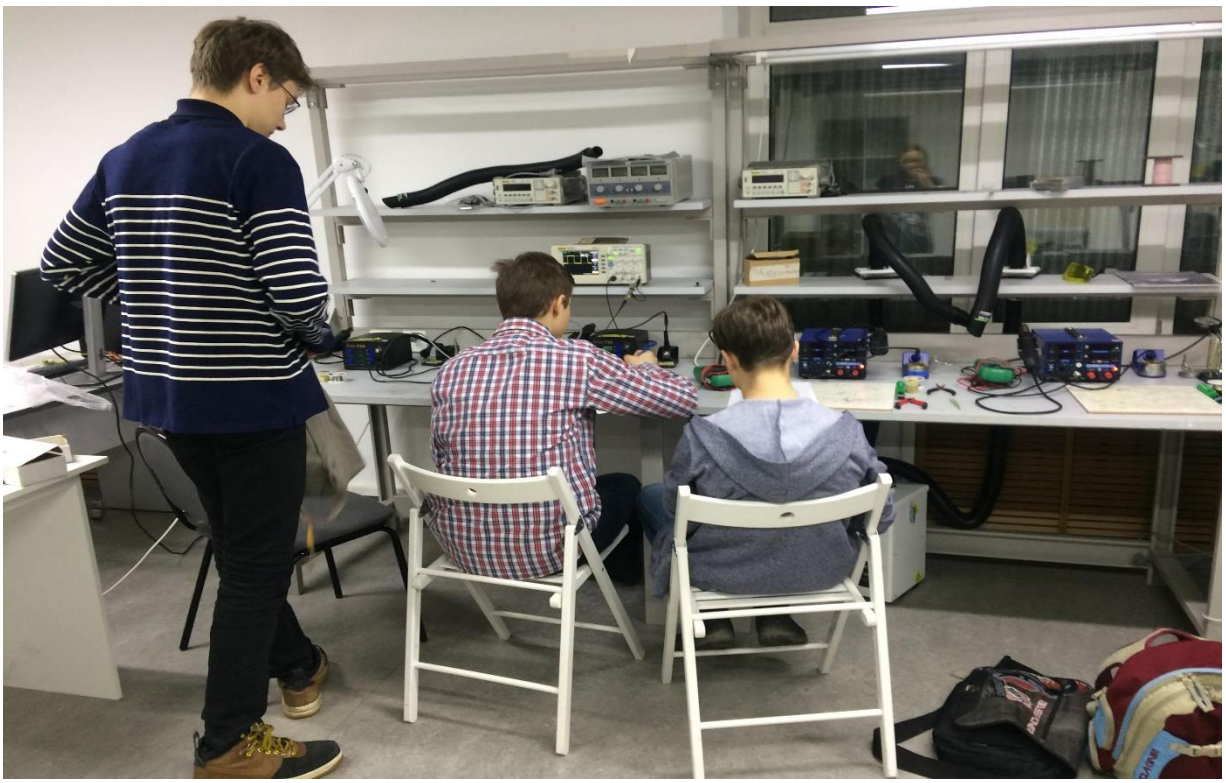
Прошедшая олимпиада является промежуточным итогом работы по реализации дорожной карты НТИ «Кружковое движение»: подготовка к ней велась в проектной школе программы НП «Лифт в будущее», фаблабах, ЦМИТах, детских технопарках, на базе активных школ и лицеев, центров дополнительного образования по всей России. 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.

Для более глубокого погружения в область практической работы участники имели возможность посетить политехническую смену инженерно-конструкторской школы «Лифт в будущее». Школа проходила во Всероссийском детском центре «Орленок» в Краснодарском крае.

В рамках краткосрочной углубленной подготовки по направлениям «Автономные транспортные системы» и «Системы связи и дистанционного зондирования Земли» был проведен интенсив для учеников 171 школы города Москвы на выездной зимней школе. Для участников Свердловской области в рамках фестиваля «Город Технотворчества» были проведены подготовительные мероприятия по направлениям «Автономные транспортные системы» и «Электронная инженерия: Умный дом».

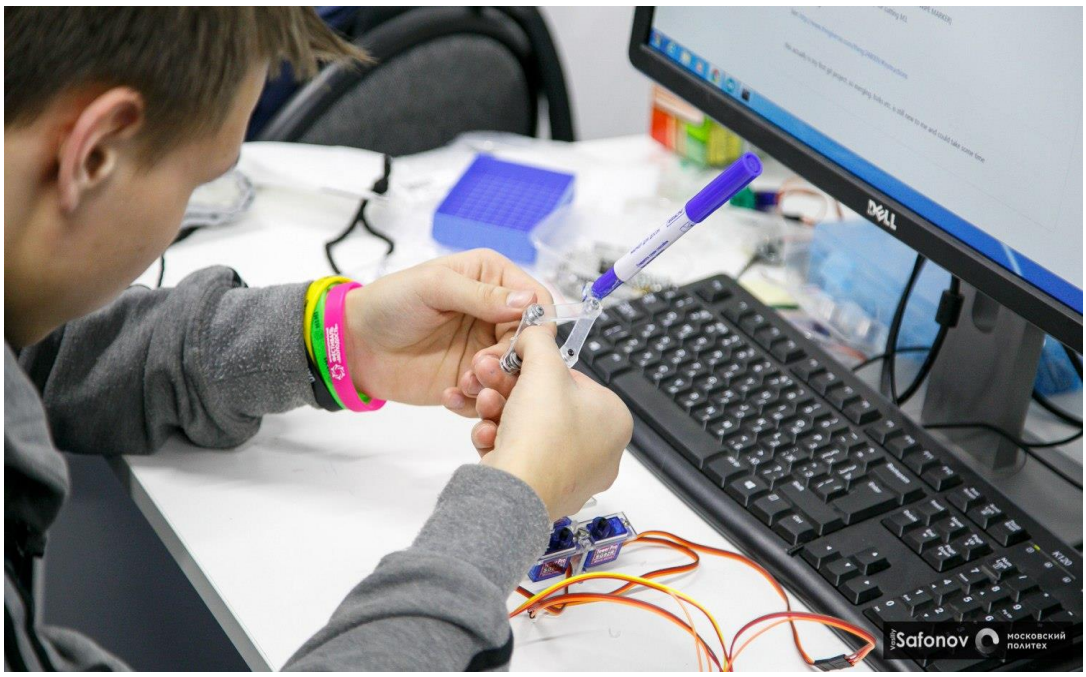
Для подготовки участников Олимпиады по направлению «Большие данные и машинное обучение» в Москве 9-11 декабря был проведен хакатон образовательного проекта GoTo. Командам помогали эксперты по анализу данных и социологи.

С декабря по март 2017 года на базе Московского Политеха проводились одно- или двухдневные подготовительные мероприятия для участников из Москвы, Московской области и других близлежащих областей, например, Тульской, Калужской и проч. по направлениям «Автономные транспортные системы», «Системы связи и дистанционного зондирования Земли», «Большие данные и машинное обучение», «Системы беспроводной связи», «Нанотехнологии» (Современные структуры и материалы).



На фото: Хакатон профиля «Интеллектуальные энергетические системы»

На фото: Хакатон профиля «Системы связи и дистанционного зондирования Земли»



На фото: Хакатон профиля «Автономные транспортные системы»

На базе Университета Иннополис был проведен двухдневный интенсив по направлению «Интеллектуальные робототехнические системы».

На фото: участники интенсива на базе Университета Иннополис.

На базе Московского авиационного университета был проведен однодневный интенсив по направлению «Беспилотные авиационные системы». Участники хакатона работали с различными датчиками, которые установлены на летательном аппарате, микроконтроллерами и электрическими двигателями; а также разрабатывали программное обеспечение в среде Arduino, позволяющее запускать двигатель и управлять его оборотами, получать информацию с датчиков в реальном масштабе

времени, отрабатывали взлёт-посадку с помощью кода, который они написали для автоматизированного полета.

Также в рамках работы по практической подготовке участников и развитию детско-взрослого инженерного сообщества была сформирована региональная сеть партнерских площадок по подготовке на базе школ, ЦМИТов, центров дополнительного образования и сети детских технопарков «Кванториум». Информация о партнерских площадках размещена в специальном разделе официального сайта олимпиады: http://nti-contest.ru/places_to_prepare/.

В 2016/2017 учебном году Олимпиада НТИ прошла с широким освещением мероприятий в различных средствах массовой информации (телевидение, печатные издания, электронные издания), среди которых как узкопрофильные издания, аудитория которых включает школьников старших классов, учителей, преподавателей в определенной научной среде, так и крупные федеральные издания общего характера. Во время проведения отборочных этапов Олимпиада НТИ освещалась в публикациях специализированных научных и общеобразовательных порталов (Занимательная робототехника, Дневник.ру, НИА Наука, Научная Россия, 5 углов, Нейроновости), официальных региональных порталов (Калининград, Тюмень, Курск, Курган, Тамбов, Мурманск, Новогород, Вологда и т.д.), в печатных изданиях («Качество образования»). Заключительный этап Олимпады НТИ (25-29 марта 2017 года) проходил при участии журналистов таких печатных изданий, как Российская газета, Комсомольская правда, Огонек, Сноб, Известия, Учительская газета; федеральных телевизионных каналов (Первый канал, Россия 24), научно-популярных блогов Мел, Постоянная Планка, Химия-Просто, Первый научный. Широкое освещение мероприятий заключительного этапа имеет своей целью распространение информации среди потенциальных участников Олимпиады НТИ будущего года - учеников 6-10 классов и направлено на привлечение талантливых школьников со всей России.

ИНТЕЛЛЕКТУАЛЬНЫЕ РОБОТОТЕХНИЧЕСКИЕ СИСТЕМЫ

Профиль “Интеллектуальные робототехнические системы” посвящен решению классических задач робототехники: автономной локализации мобильного наземного робототехнического устройства, планирования и построения маршрута мобильного робототехнического устройства или его частей, навигации, распознавания графической информации.

Профиль включает в себя задачи по двум школьным предметам: **математика** и **информатика**.

§1 Первый отборочный этап

Первый отборочный тур проводится индивидуально в сети Интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Для каждого из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике - общие для всех участников. Решение задач по информатике предполагало написание программ, допускалось использовать один из нескольких языков программирования: C, C++, Java, Python или JavaScript. На решение задач каждого предмета первого отборочного этапа участникам давалось 2 дня. У участников было два временных слота по 2 дня каждый, когда они могли решать задачи по предмету. Решение каждой задачи дает определенное количество баллов. Баллы зачисляются в полном объеме за правильное решение задачи. Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и информатика) — суммарно от 0 до 20 баллов.

1.1 Первая попытка
Задачи по математике (9 класс)

Задача 1.1.1 (1 балл)

Условие:

Известно, что $a^{16} = 2$. Вычислите сумму

$$\frac{1}{1-a} + \frac{1}{1+a} + \frac{2}{1+a^2} + \frac{4}{1+a^4} + \frac{8}{1+a^8}$$

Решение:

Преобразуем сумму

$$S = \left(\frac{1}{1-a} + \frac{1}{1+a}\right) + \frac{2}{1+a^2} + \frac{4}{1+a^4} + \frac{8}{1+a^8} =$$

$$= \left(\frac{2}{1-a^2} + \frac{2}{1+a^2}\right) + \frac{4}{1+a^4} + \frac{8}{1+a^8} = \left(\frac{4}{1-a^4} + \frac{4}{1+a^4}\right) + \frac{8}{1+a^8} =$$

$$= \left(\frac{8}{1-a^8} + \frac{8}{1+a^8}\right) = \frac{16}{1-a^{16}}$$

Теперь легко найти чему равна сумма при известном a:

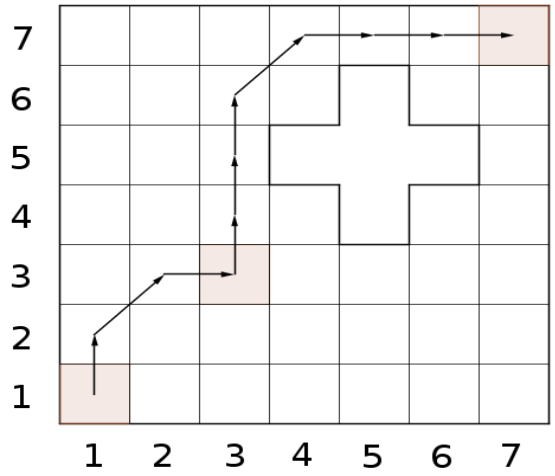
$$S = \frac{16}{1-a^{16}} = \frac{16}{1-2} = -16$$

Ответ: -16.

Задача 1.1.2 (2 балла)

Условие:

Приведенное справа поле состоит из 44 единичных клеток. Каждым своим ходом шахматный король может пойти либо на одну клетку вверх, либо на одну клетку вправо, либо на одну клетку вверх-вправо по диагонали (не выходя при этом за границы поля). Сколько существует путей короля, ведущих из клетки (1, 1) в клетку (7, 7), проходящих через клетку (3, 3) (на рисунке изображен один из возможных путей)?



Решение:

В каждую клетку нашего поля напишем число путей, ведущих в нее из клетки (1, 1), удовлетворяющих условию задачи. Очередное число в клетке вычисляется как сумма чисел в клетках непосредственно слева, снизу и слева-снизу. В итоге мы получаем, что число путей, ведущих в клетку (7, 7) равно 156 (см. рисунок).

		13	52	78	78	156
		13	26		0	78
		13				78
		13	39		26	52
1	5	13	13	13	13	13
1	3	5				
1	1	1				

Ответ: 156

Задание 1.1.3 (2 балла)

Условие:

Найдите значение выражения $\alpha^6 + \beta^6$, где α и β -- корни квадратного трехчлена $x^2 + 5x + 3$.

Решение:

Заметим, что

$$\alpha^6+\beta^6=(\alpha^3+\beta^3)^2-2(\alpha\cdot\beta)^3=\left((\alpha+\beta)^3-3(\alpha+\beta)\cdot(\alpha\cdot\beta)\right)^2-2(\alpha\cdot\beta)^3$$

Осталось вспомнить теорему Виета для квадратного трехчлена

x^2+5x+3

$$\begin{cases}\alpha+\beta=-\dfrac{5}{1}=-5\\ \alpha\cdot\beta=\dfrac{3}{1}=3\end{cases}$$

Подставим полученные значения в наше выражение

$$\alpha^6+\beta^6=\left((-5)^3-3\cdot(-5)\cdot3\right)^2-2\cdot3^3=6346$$

Ответ: 6346

Задача 1.1.4 (2 балла)

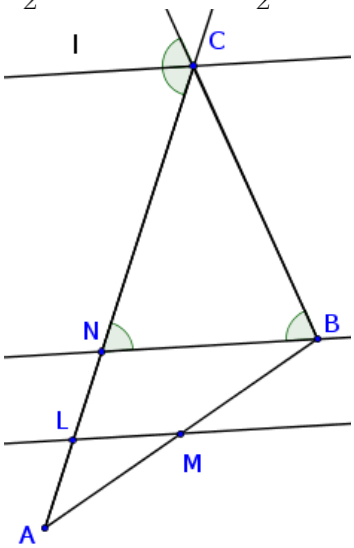
Условие:

Пусть l - биссектриса внешнего угла C треугольника ABC. Прямая, параллельная l и проходящая через середину M стороны AB, пересекает AC в точке L. Найти длину отрезка A, если AC = 5,3 и CB = 3,2.

Решение:

Проведем через точку B прямую параллельную l, точку пересечения с отрезком AC обозначим за N (см. рисунок). Тогда треугольник NCB равнобедренный (CN = CB). Так как ML || BN и M - середина отрезка AB, то ML -- средняя линия треугольника BAN, следовательно NL = LA. В итоге получаем

$$AL=\frac{1}{2}NA=\frac{1}{2}(CA-CN)=\frac{1}{2}(CA-CB)=1,05$$



Ответ: 1,05

Задача 1.1.5 (3 балла)

Условие:

Точка A лежит на окружности $x^2+y^2-8x-12y+36=0$, а точка B -- на прямой $-8x+6y-79=0$. Найдите наименьшее возможное расстояние между точками A и B.

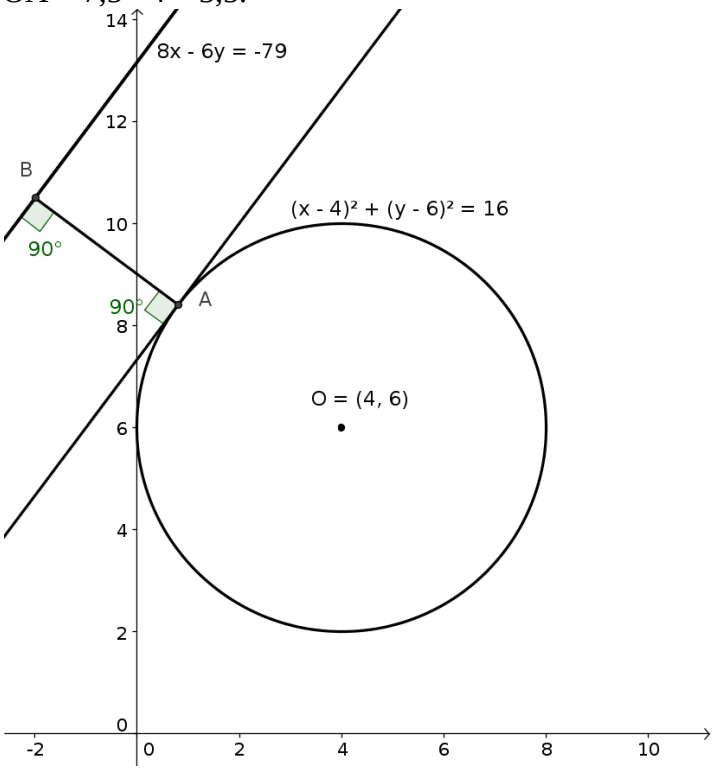
Решение:

Уравнение окружности преобразуем к виду $(x-4)^2+(y-6)^2=4^2$. Получаем окружность с центром в точке O(4, 6) радиуса 4. Длина отрезка AB будет минимальна, когда она совпадает с расстоянием между параллельными прямыми, одна из которых

исходная, а вторая касается окружности (см. рисунок). Это расстояние можно найти как разность расстояний от точки O до наших параллельных прямых AB = OB - OA (точки O, A и B лежат на одной прямой). Расстояние OA = r = 4, а расстояние OB можно вычислить как расстояние от точки до прямой

$$OB = \frac{|8 \cdot 4 + (-6) \cdot 6 + 79|}{\sqrt{8^2 + (-6)^2}} = 7,5$$

В итоге AB = OB - OA = 7,5 - 4 = 3,5.



Ответ: 3,5

1.2 Первая попытка
Задачи по математике (10-11 класс)

Задача 1.2.1 (1 балл)

Условие:

Известно, что $a = \sqrt[2048]{2}$. Вычислите сумму

$$\frac{1}{1-a} + \frac{1}{1+a} + \frac{2}{1+a^2} + \cdots + \frac{512}{1+a^{512}} + \frac{1024}{1+a^{1024}}$$

Решение:

Преобразуем сумму

$$S(a) = \left(\frac{1}{1-a} + \frac{1}{1+a}\right) + \frac{2}{1+a^2} + \cdots + \frac{1024}{1+a^{1024}} =$$
$$= \left(\frac{2}{1-a^2} + \frac{2}{1+a^2}\right) + \frac{4}{1+a^4} + \cdots + \frac{1024}{1+a^{1024}} = \cdots =$$
$$= \left(\frac{1024}{1-a^{1024}} + \frac{1024}{1+a^{1024}}\right) = \frac{2048}{1-a^{2048}}$$

Теперь легко найти, чему равна сумма при $a = \sqrt[2048]{2}$:

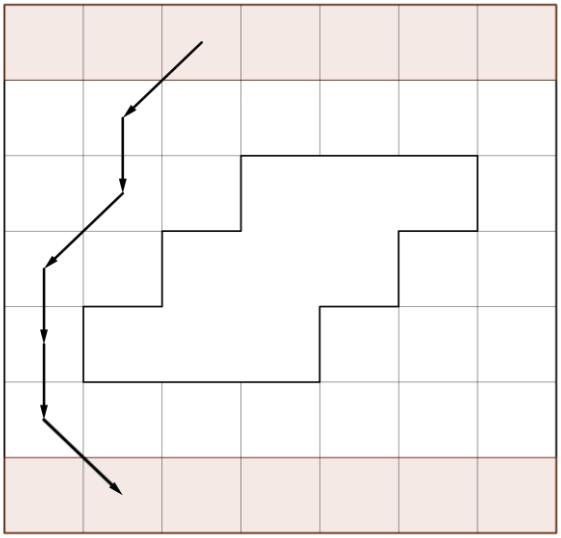
$$S(\sqrt[2048]{2}) = \frac{2048}{1 - (\sqrt[2048]{2})^{2048}} = -2048$$

Ответ: -2048

Задача 1.2.2 (2 балла)

Условие:

Приведенное справа поле состоит из 40 единичных клеток. Кузнечик из каждой клетки может прыгнуть либо на одну клетку вниз, либо на одну клетку влево-вниз по диагонали, либо на одну клетку вправо-вниз по диагонали (не выходя при этом за границы поля). Сколько существует путей кузнечика, ведущих из верхнего ряда квадратов в нижний? (На рисунке изображен один из возможных путей, верхний и нижний ряды квадратов выделены сиреневым цветом.)



Решение:

В каждую клетку нашего поля напишем число путей кузнечика, ведущих в нее из верхнего ряда клеток. Очередное число в клетке вычисляется как сумма чисел в клетках непосредственно слева-сверху, сверху и справа-сверху. В итоге мы получаем, что число путей кузнечика, ведущих в клетки нижнего ряда равно сумме 70 + 70 + 40 + 20 + 45 + 60 + 45 = 350 (см. рисунок).

1	1	1	1	1	1	1
2	3	3	3	3	3	2
5	8	9				5
13	22				5	5
35				5	10	10
35	35	0	5	15	25	20
70	70	40	20	45	60	45

Ответ: 350

Задача 1.2.3 (2 балла)

Условие:

Найдите значение выражения $\alpha^3+\beta^3+\gamma^3$, где α, β и γ - корни кубического многочлена x^3+x^2-7x+4.

Решение:

Заметим, что

$$\alpha^3+\beta^3+\gamma^3=(\alpha+\beta+\gamma)^3-3(\alpha+\beta+\gamma)(\alpha\beta+\beta\gamma+\gamma\alpha)+3\alpha\beta\gamma$$

Осталось вспомнить теорему Виета для многочлена x^3+x^2-7x+4

$$\begin{cases}\alpha+\beta+\gamma=-\dfrac{1}{1}=-1\\ \alpha\beta+\beta\gamma+\gamma\alpha=\dfrac{-7}{1}=-7\\ \alpha\beta\gamma=-\dfrac{4}{1}=-4\end{cases}$$

Подставим полученные значения в наше выражение

$$\alpha^3+\beta^3+\gamma^3=(-1)^3-3(-1)(-7)+3(-4)=-34$$

Ответ: -34

Задача 1.2.4 (2 балла)

Условие:

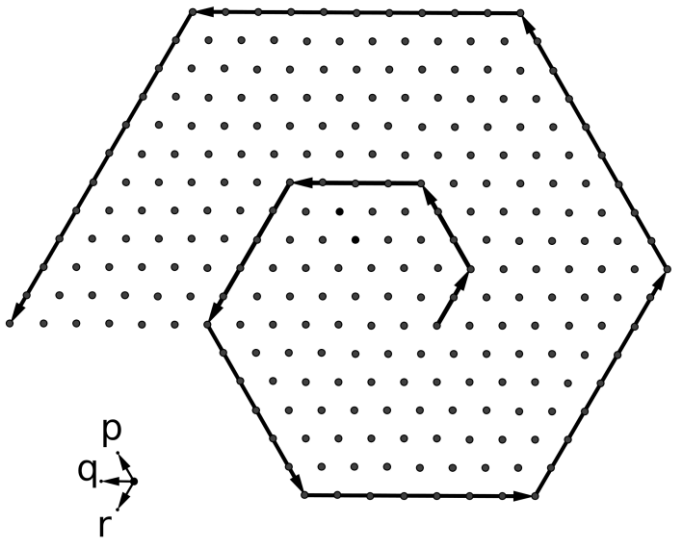
Саша собрал робота, который едет со скоростью 10 м/мин и запустил его на ровной открытой площадке. Робот проехал 2 минуты в одном направлении и повернул налево на 60 градусов, затем 3 минуты проехал прямо и повернул налево на 60 градусов, потом проехал 4 минуты и снова повернул налево на 60 градусов и т.д. после каждого поворота до следующего поворота двигался на минуту дольше. Через 65 минут закончился заряд батареи и робот остановился. На каком расстоянии робот оказался от начальной точки?

Решение:

Так как через 65 минут закончится заряд батареи, то робот проедет в итоге 20 м, 30 м, ..., 110 м при этом каждый раз поворачивая налево на 60 градусов (см. рисунок, расстояние между соседними точками 10 м). Всю траекторию робота можно представить в виде линейной комбинации трех векторов $\vec{p},\vec{q}$ и $\vec{r}$ (длины равны 10 м). В итоге робот удалится от начальной точки на вектор

$$\vec{l}=-2\vec{r}+3\vec{p}+4\vec{q}+5\vec{r}-6\vec{p}-7\vec{q}-8\vec{r}+9\vec{p}+10\vec{q}+11\vec{r}$$

Сокращая, получаем $\vec{l}=6\vec{p}+7\vec{q}+6\vec{r}$. А так как $\vec{p}+\vec{q}=\vec{r}$, то $\vec{l}=6(\vec{p}+\vec{q})+7\vec{r}=13\vec{r}$. В итоге получаем, что длина вектора $|\vec{l}|=13|\vec{r}|=130$ м.



Ответ: 130 м.

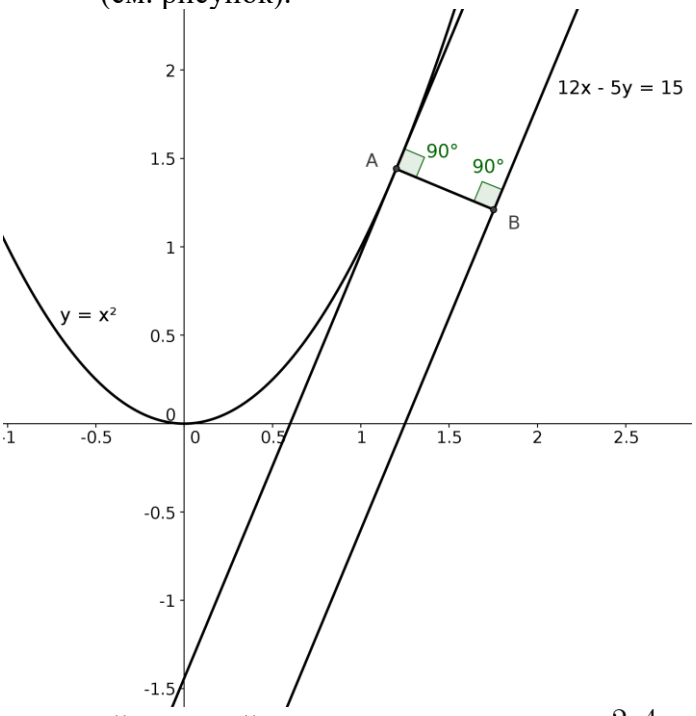
Задача 1.2.5 (3 балла)

Условие:

Точка A лежит на параболе $y = x^2$, а точка B - на прямой $12x - 5y - 15 = 0$. Найдите наименьшее возможное расстояние между A и B.

Решение:

Для того, чтобы найти минимальное расстояние от точки A до точки B нужно провести прямую параллельную $12x - 5y - 15 = 0$, которая будет касаться параболы. A - точка касания этой прямой и параболы, B - перпендикуляр, опущенный из точки A на прямую $12x - 5y - 15 = 0$ (см. рисунок).



Уравнение исходной прямой можно записать $y = 2,4x - 3$, следовательно коэффициент k = 2,4. Мы получаем, что уравнение касающейся прямой имеет вид

$y = 2,4x - a$. Так как точка пересечения с параболой $y = x^2$ единственная, то квадратное уравнение $x^2 = 2,4x - a$ имеет дискриминант равный 0, следовательно a = 1,44. Длину отрезка AB можно найти как расстояние между параллельными прямыми

$$AB = \frac{|c_1 - c_2|}{\sqrt{k^2 + 1}} = \frac{|-3 - (-1,44)|}{\sqrt{2,4^2 + 1}} = 0,6$$

Ответ: 0,6

1.3 Первая попытка
Задачи по информатике

Задача 1.3.1 (1 балл)

Условие:

Складской робот-погрузчик находится внутри одного из помещений в большом логическом центре. Проекция помещения — прямоугольник, координаты противоположных углов которого находятся в точках (0, 0) и (n, m). Вдоль стенок расположены грузы, которые погрузчик должен загрузить в себя. Какое минимальное расстояние пройдет погрузчик, чтобы добраться до одной из стенок складского помещения?

Формат входных данных:

Первая строка содержит n, m $(1 \le n, m \le 10^9)$, ширину и длину складского помещения. Вторая строка содержит два числа x, y $(0 \le x \le n, 0 \le y \le m)$, координаты погрузчика.

Формат выходных данных:

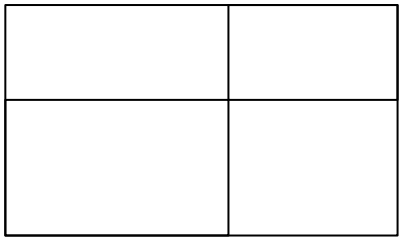
Выведите одно целое число - минимальное расстояние до одной из стенок.

Пример:

stdin:
3 4
1 2
***stdout*:**
1

Решение:

Требуется найти минимум среди четырех сторон.



Пример программы, реализующей данный алгоритм на языке Python:

```
import sys
```

```
def solve(dataset):
    n, m, x, y = [int(t) for t in dataset.split()]
    return str(min(x, y, n - x, m - y))

print(solve(sys.stdin.read()))
```

Задача 1.3.2 (2 балла)

Условие:

В логистическом центре 5 роботов-погрузчиков занимались погрузкой ящиков с грузами. Первый робот перетащил ровно A% от всех ящиков, второй робот — ровно B% от всех ящиков, третий робот — ровно C%, четвертый робот — ровно D%, пятый робот перетащил ровно E%.

Какое минимальное количество ящиков было погружено роботами?

Формат входных данных:

Единственная строка входных данных содержит пять чисел A, B, C, D, E, $(A + B + C + D + E = 100, 0 \leq A, B, C, D, E \leq 100)$

Формат выходных данных:

Выведите минимальное количество погруженных ящиков.

Пример:

stdin:
15 55 0 5 25
stdout:
20

Решение:

Очевидно, что минимальное количество ящиков не превысит 100, это просто A+B+C+D+E. То есть достаточно перебрать возможные ответы и проверить их на правильность, среди них выбрать минимальный.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    a = [int(x) for x in dataset.split()]
    for i in range(1, 101):
        failed = False
        for x in a:
            if i * x % 100 != 0:
                failed = True
        if not failed:
            return str(i)
    return str(100);

print(solve(sys.stdin.read()))
```

Задача 1.3.3 (2 балла)

Условие:

Каждая задача для робота на погрузку на логистическом заводе определяется координационной системой. Система для определения номера робота использует 3 числа, присвоенные каждому грузу: идентификатор получателя, идентификатор грузовой компании и идентификатор отправителя. Номер робота определяется путем перевода перечисленных идентификаторов в двоичную систему счисления и побитовым применением булевой функции от трех аргументов:

X	***Y***	***Z***	***f(x, y, z)***
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

где X — это n-ый бит из идентификатора получателя, Y — n-ый бит из идентификатора грузовой компании, а Z — n-ый бит из идентификатора отправителя.

Найдите номер робота путем применения этой функции к идентификаторам груза.

Формат входных данных:

Единственная строка входных данных содержит три числа x, y, z $(0 \leq x, y, z \leq 2^{64} - 1)$ — идентификатор получателя, идентификатор грузовой компании и идентификатор отправителя, соответственно.

Формат выходных данных:

Выведите номер требуемого робота.

Пример №1:

stdin:
76 58 109
stdout:
40

Пример №2:

stdin:
7 8 5
stdout:
2

Пояснение:

Побитовое применение функции означает, что функция применяется к каждому разряду числа в двоичном представлении.

Пример:

$x = 76_{10} = 1001100_2$
$y = 58_{10} = 111010_2$
$z = 109_{10} = 1101101_2$

Применяем функцию начиная с младших разрядов, при нехватке дополняем лидирующими нулями.

$ans = 0101000_2 = 40_{10}$

Решение:

Задача на реализацию, требуется просто провести битовые манипуляции с числами.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    a, b, c = ["{:064b}".format(int(t))
               for t in dataset.split()]
    ans = [0] * 64;
    for i in range(0, 64):
        if a[i] == '1' and b[i] == '1':
            ans[i] = 1;
        if b[i] == '1' and c[i] == '1':
            ans[i] = 1;
        if a[i] == '1' and b[i] == '0' and c[i] == '0':
            ans[i] = 1;
    s = ''.join(str(x) for x in ans)
    return str(int(s, 2))

print(solve(sys.stdin.read()))
```

Задача 1.3.4 (2 балла)

Условие:

На логистическом центре каждый ящик маркируется парой символов латинского алфавита, при этом используются только строчные буквы (пример маркировки, ab или bc). Для отправки грузов из логистического центра, грузы должны быть упакованы по 3 ящика. Для того, чтобы грузы были правильно упакованы по три действует следующее правило: три ящика можно упаковать если первая буква маркировки среднего ящика совпадает с последней буквой первого ящика, а последняя буква среднего ящика совпадает с первой буквой третьего (например, ab|bc|cd — упакованы верно).

Необходимо узнать, сколько различных упаковок можно составить из заданного набора ящиков с грузами, если каждый ящик можно использовать в упаковке только один раз. Две упаковки s и t считаются различными, если существует такой индекс i, что маркировка s_i не равна маркировке t_i.

Формат входных данных:

Первая строка входных данных содержит целое число n — количество ящиков с грузами $(3 \leq n \leq 100)$. Во второй строке записаны n маркировок грузов, разделенных пробелом.

Формат выходных данных:

Выведите количество различных упаковок.

Пример №1:

stdin:
3
ab bc cd
stdout:
1

Пример №2:

stdin:
3
ab bb ba
stdout:
3

Пример №3:

stdin:
5
ab bc da ca bd
stdout:
8

Решение:

Давайте перебирать тройки грузов. Остается только убрать повторяющиеся тройки, для этого можно использовать структуры данных set/hashset, либо добавив все тройки в массив и отсортировав их. После чего можно за линейное время найти количество уникальных. Сложность такого алгоритма $O\left(n^3 \cdot \ln\left(n^3\right)\right)$.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    a = dataset.split()
    n = int(a[0])
    a = a[1:]
    s = set()
    for i in range(n):
        for j in range(n):
            for k in range(n):
                if i != j and j != k and i != k and
                   a[i][1] == a[j][0] and
                   a[j][1] == a[k][0]:
                        s.add(a[i]+a[j][1]+a[k][1])
    return  str(len(s))

print(solve(sys.stdin.read()))
```

Задача 1.3.5 (3 балла)

Условие:

В логистическом центре у одного из роботов-погрузчиков обнаружилась проблема в программном обеспечении. Чтобы обнаружить сбойного робота персонал центра

собрался в течение ночи произвести диагностику всех роботов. Для этого они арендуют диагностические стенды у компании-разработчика роботов.

Один стенд может делать диагностику для любого количества роботов, при этом один и тот же робот может подключаться на диагностику к нескольким стендам. Каждый стенд выдает результат диагностики только утром, после проверки всех роботов, которые были подключены к нему, но есть известная проблема стенда – он может сказать, что результат диагностики роботов отрицательный, но у какого конкретно робота – неизвестно.

Стоимость аренды диагностических стендов очень высокая, поэтому задача руководства центра арендовать как можно меньше таких стендов. Помоги логистическому центру определить, какое минимальное количество диагностических стендов нужно для тестирования роботов, чтобы утром они уже знали проблемного робота.

Формат входных данных:

Единственная строка входных данных содержит число x $(1 \leq x \leq 10^9)$ — количество роботов.

Формат выходных данных:

Выведите минимальное количество стендов.

Пример №1:

stdin:
7
stdout:
3

Пример №2:

stdin:
2
stdout:
1

Решение:

Оптимальная стратегия: выпишем номера роботов в двоичной системе счисления.

$$1_{10} = 1_2, 2_{10} = 10_2, 3_{10} = 11_2, 4_{10} = 100_2 \ldots$$

Подключим к первому стенду тех роботов, у которых первый бит равен одному, ко второму стенду - роботов, у которых второй бит равен одному, к третьему - третий и так далее. На следующий день смотрим, на каких стендах результат диагностики роботов отрицательный. Если робот - сбойный, значит все стенды, к каким он был подключен, сигнализируют об отрицательной диагностике. По номерам стендов можно восстановить номер робота (перевод из двоичной системы в десятичную). При этом последнего робота можно не проверять, так как если он - сбойный, то ни один стенд не сигнализирует об этом. То есть для восьми роботов требуется три стенда, а не четыре.

Ответ на задачу: $\log_2(n)$ с округлением вверх.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
```

```
    n = int(dataset.split()[0])
    if n == 1:
        return "0"
    n -= 1
    return str(len("{:b}".format(n)))

print(solve(sys.stdin.read()))
```

1.4 Вторая попытка
Задачи по математике (9 класс)

Задача 1.4.1 (1 балл)

Условие:

Найдите остаток при делении 2016 · 2017 · 2018 · 2019 · 2020 на 11.

Решение:

Так как

$$2016 \equiv 3 \pmod{11},\ 2017 \equiv 4 \pmod{11}, \ldots,\ 2020 \equiv 7 \pmod{11},$$

то

$$2016 \cdot 2017 \cdot 2018 \cdot 2019 \cdot 2020 \equiv 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \equiv 2520 \equiv 1 \pmod{11}$$

Ответ: 1

Задача 1.4.2 (2 балла)

Условие:

В классе из 16 учащихся только 2 сделали домашнее задание. Равиль Ниязович наугад выбирает школьника для ответа. Если задание не сделано, то учащийся получает оценку “два” и садится. Затем процесс продолжается до тех пор, пока не ответит школьник, сделавший домашнее задание. Найдите вероятность того, что Равиль Ниязович поставит ровно три двойки. (Если ответом является бесконечная дробь, то результат округлите до сотых.)

Решение:

Наше вероятностное пространство состоит из 16! исходов (число всех перестановок на 16-ти учениках). Посчитаем в скольких из них Равиль Ниязович поставит ровно три двойки: первого опрашиваемого школьника можно выбрать 14 способами, второго - 13, третьего - 12, четвертого - 2, а остальных 12 школьников можно переставлять как угодно. Всего подходящих исходов будет 14 · 13 · 12 · 2 · 12!. Следовательно вероятность вычисляется

$$P = \frac{14 \cdot 13 \cdot 12 \cdot 2 \cdot 12!}{16!} = \frac{1}{10} = 0,1$$

Ответ: 0,1

Задача 1.4.3 (2 балла)

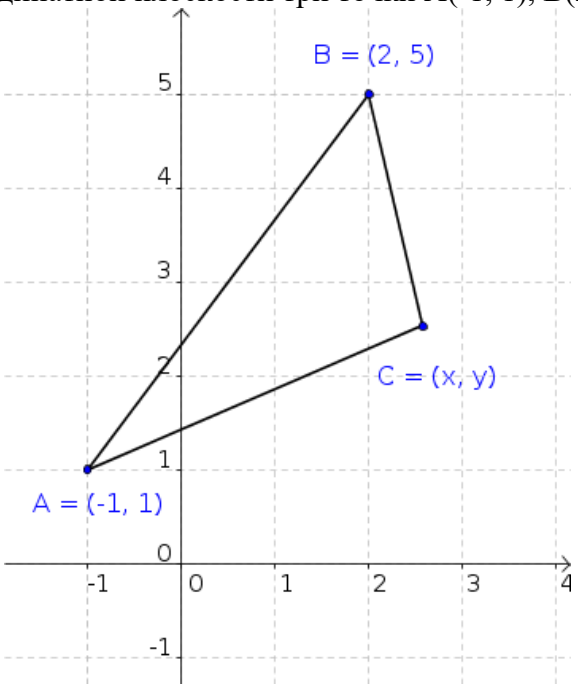
Условие:

Найдите минимальное значение выражения

$$\sqrt{(x+1)^2 + (y-1)^2} + \sqrt{(x-2)^2 + (y-5)^2}$$

Решение:

Отметим на координатной плоскости три точки A(-1, 1), B(2, 5) и C(x, y),



тогда

$$AC = \sqrt{(x+1)^2 + (y-1)^2}$$

и

$$BC = \sqrt{(x-2)^2 + (y-5)^2}$$

Следовательно, наше выражение равно сумме расстояний от точки C до точек A и B. По неравенству треугольника

$$AC + BC \geq AB = \sqrt{(2-(-1))^2 + (5-1)^2} = 5$$

Причем равенство достигается, когда точка C лежит на отрезке AB.

Ответ: 5

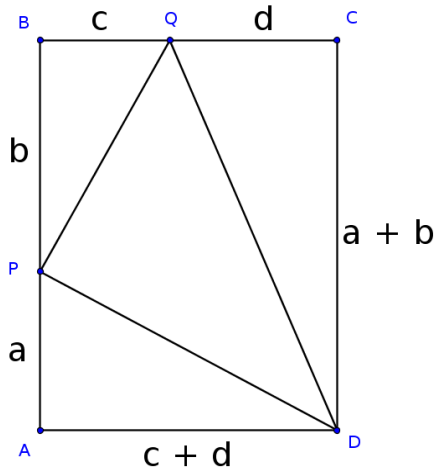
Задача 1.4.4 (2 балла)

Условие:

На сторонах AB и BC прямоугольника ABCD отмечены точки P и Q соответственно. Оказалось, что $S_{ABCD} = 2017$ и $S_{DPQ} = 505$. Чему может быть равно произведение AP · CQ?

Решение:

Обозначим длины отрезков следующим образом AP = a, PB = b, BQ = c и QC = d.



Тогда

$$S_{ABCD} = (a+b)\cdot(c+d) = a\cdot c + a\cdot d + b\cdot c + b\cdot d$$

и

$$S_{DPQ} = S_{ABCD} - S_{PDA} - S_{PQB} - S_{QCD} =$$
$$= (a+b)\cdot(c+d) - \frac{a\cdot(c+d)}{2} - \frac{b\cdot c}{2} - \frac{d\cdot(a+b)}{2} =$$
$$= \frac{a\cdot c + b\cdot c + b\cdot d}{2}$$

А так как AP · CQ = a · d, то получаем

$$AP\cdot CQ = S_{ABCD} - 2S_{PDQ} = 2017 - 2\cdot 505 = 1007$$

Ответ: 1007

Задача 1.4.5 (3 балла)

Условия:

Даны две последовательности чисел (x_n) и (y_n) следующими условиями: $x_0 = 3-\sqrt{2}$, $y_0 = 3+\sqrt{2}$ и $\forall n \in \cup\{0\}$

$$x_{n+1} = \frac{x_n + y_n}{\sqrt{2}},\ y_{n+1} = \frac{y_n - x_n}{\sqrt{2}}.$$

Найдите $x_{2016} + y_{2016}$.

Решение:

Заметим, что

$$x_{n+1} + y_{n+1} = \sqrt{2}y_n \text{ и } y_{n+1} - x_{n+1} = -\sqrt{2}x_n$$

Поэтому

$$x_{n+2} = \frac{x_{n+1} + y_{n+1}}{\sqrt{2}} = y_n \quad y_{n+2} = \frac{y_{n+1} - x_{n+1}}{\sqrt{2}} = -x_n$$

и

В итоге

$$(x_{n+8}, y_{n+8}) = (y_{n+6}, -x_{n+6}) = (-x_{n+4}, -y_{n+4}) = (-y_{n+2}, x_{n+2}) = (x_n, y_n).$$

Обе наши последовательности имеют период равный 8, а так как 2016 = 252 · 8, то (x_{2016}, y_{2016}) = (x_0, y_0). Следовательно,

$$x_{2016} + y_{2016} = x_0 + y_0 = (3-\sqrt{2}) + (3+\sqrt{2}) = 6$$

Ответ: 6

1.5 Вторая попытка
Задачи по математике (10-11 класс)

Задача 1.5.1 (1 балл)

Условие:

Найдите остаток при делении 9^{2017} на 11.

Решение:

Так как $9 \equiv -2 \pmod{11}$, то

$$9^{2017} \equiv (-2)^{2017} \equiv -2^{2017} \pmod{11}$$

Осталось заметить, что $2^{2017} = 4 \cdot (2^5)^{403}\ 2^5 \equiv -1 \pmod{11}$. Поэтому

$$9^{2017} \equiv -2^{2017} \equiv -4 \cdot (-1)^{403} \equiv 4 \pmod{11}$$

Ответ: 4

Задача 1.5.2 (2 балла)

Условие:

Найдите значение выражения

$$3 \cdot \cos\left(\arcsin\frac{1}{2}\right) \cdot \cos\left(\arcsin\frac{1}{3}\right) \cdot \ldots \cdot \cos\left(\arcsin\frac{1}{288}\right)$$

Решение:

Так как

$$\cos\left(\arcsin\frac{1}{n}\right) = \sqrt{1-\frac{1}{n^2}} = \frac{\sqrt{(n-1)(n+1)}}{n}$$

получаем

$$S = 3 \cdot \cos\left(\arcsin\frac{1}{2}\right) \cdot \cos\left(\arcsin\frac{1}{3}\right) \cdot \ldots \cdot \cos\left(\arcsin\frac{1}{288}\right) =$$
$$= 3 \cdot \frac{\sqrt{1\cdot 3}}{2} \cdot \frac{\sqrt{2\cdot 4}}{3} \cdot \ldots \cdot \frac{\sqrt{287\cdot 289}}{288}$$

Каждое натуральное число от 3 до 287 встретится ровно 2 раза под корнем в числителе, поэтому эти числа сокращаются с числами, стоящими в знаменателе.

$$S = 3 \cdot \frac{\sqrt{1\cdot 2\cdot 288\cdot 289}}{2\cdot 288} = 3 \cdot \sqrt{\frac{289}{576}} = \frac{17}{8} = 2,125$$

Ответ: 2,125

Задача 1.5.3 (2 балла)

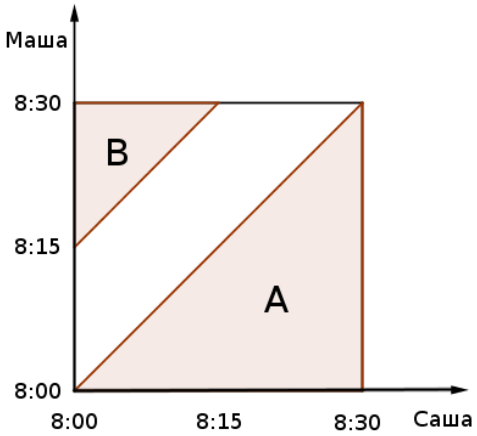
Условие:

Прием у терапевта начинается в 8 утра и для каждого пациента длится 15 минут. Саша и Маша записались на прием с 8.00 до 8.30 и приходят в случайное время в данном промежутке независимо друг от друга. Других пациентов в это время нет. Какова вероятность того, что Маше не придется ждать своей очереди?

Решение:

Рассмотрим конфигурационное пространство времен приходов Саши и Маши. Для того, чтобы Маше не пришлось ждать своей очереди, ей нужно либо прийти раньше Саши, либо на 15 или более минут позже. Первому случаю соответствуют точки, лежащие в области A конфигурационного пространства (см. рисунок), а второму случаю -- точки, лежащие в области B. Тогда ответ на эту задачу

$$P(A) + P(B) = \frac{1}{2} + \frac{1}{8} = 0,625$$



Ответ: 0,625

Задача 1.5.4 (2 балла)

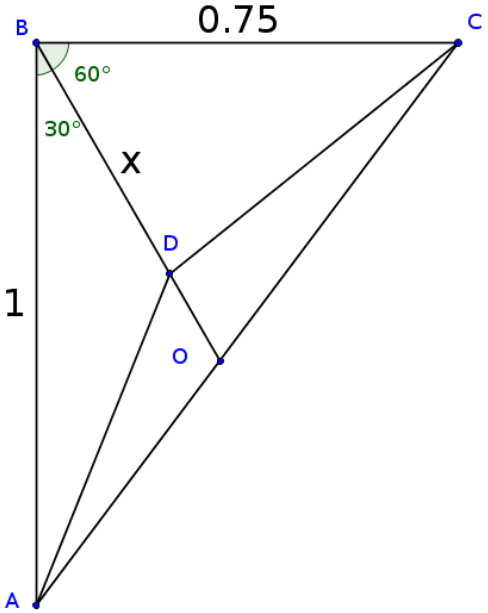
Условие:

Найдите минимум функции

$$f(x) = \sqrt{\frac{9}{16} - \frac{3}{4}x + x^2} + \sqrt{1 - \sqrt{3}x + x^2}$$

Решение:

При решении этой задачи будем использовать теорему косинусов. Рассмотрим прямоугольный треугольник ABC и отложим точку D как показано на рисунке.



Тогда по теореме косинусов для треугольника ABD:

$$AD^2 = AB^2 + BD^2 - 2AB \cdot BD \cdot \cos 30^\circ = 1 - \sqrt{3}x + x^2$$
и по теореме косинусов для треугольника CBD:
$$CD^2 = CB^2 + BD^2 - 2CB \cdot BD \cdot \cos 60^\circ = \frac{9}{16} - \frac{3}{4}x + x^2$$
Получаем, что значение нашей функции
$$f(x) = AD + CD$$
По неравенству треугольника
$$AD + CD \geq AC = \sqrt{1^2 + 0,75^2} = 1,25$$
причем равенство достигается, когда точка D лежит на гипотенузе AC, то есть совпадает с точкой O (см. рисунок).

Ответ: 1,25

Задача 1.5.5 (3 балла)

Условие:

Даны две последовательности чисел (x_n) и (y_n) следующими условиями: $x_0 = 1 - \sqrt{2}$, $y_0 = 1 + \sqrt{2}$ и $\forall n \in \cup\{0\}$
$$x_{n+1} = x_n \cdot \cos 5^\circ + y_n \cdot \sin 5^\circ, y_{n+1} = y_n \cdot \cos 5^\circ - x_n \cdot \sin 5^\circ.$$
Найдите $x_{2016} + y_{2016}$.

Решение:

Рассмотрим вектора $\vec{v}_n = (x_n, y_n)$. Заметим, что
$$|\vec{v}_{n+1}|^2 = x_{n+1}^2 + y_{n+1}^2 = (x_n \cdot \cos 5^\circ + y_n \cdot \sin 5^\circ)^2 + (y_n \cdot \cos 5^\circ - x_n \cdot \sin 5^\circ)^2 =$$
$$= x_n^2 \cdot (\cos^2 5^\circ + \sin^2 5^\circ) + y_n^2 \cdot (\cos^2 5^\circ + \sin^2 5^\circ) = x_n^2 + y_n^2 = |\vec{v}_n|^2$$
Следовательно, длины векторов не изменяются
$$|\vec{v}_{n+1}| = |\vec{v}_n| = \ldots = |\vec{v}_0| = \sqrt{(1 - \sqrt{2})^2 + (1 + \sqrt{2})^2} = \sqrt{6}$$
Пусть α_n ориентированный угол между лучом Ox и вектором $\vec{v}_n$. Тогда
$$x_n = \sqrt{6} \cdot \cos \alpha_n, y_n = \sqrt{6} \cdot \sin \alpha_n$$
следовательно,
$$\cos \alpha_{n+1} = \cos \alpha_n \cdot \cos 5^\circ + \sin \alpha_n \cdot \sin 5^\circ = \cos(\alpha_n - 5^\circ)$$
$$\sin \alpha_{n+1} = \sin \alpha_n \cdot \cos 5^\circ - \cos \alpha_n \cdot \sin 5^\circ = \sin(\alpha_n - 5^\circ)$$
В итоге мы получили, что $\alpha_{n+1} = \alpha_n - 5^\circ$, поэтому $\alpha_{2016} = \alpha_0 - 2016 \cdot 5^\circ = \alpha_0$ ($2016 \cdot 5^\circ = 28 \cdot 360^\circ$). Следовательно, наш вектор совершит 28 полных оборотов по часовой стрелке и вернется в исходное положение
$$x_{2016} + y_{2016} = (1 - \sqrt{2}) + (1 + \sqrt{2}) = 2$$

Ответ: 2

1.6 Вторая попытка
Задачи по информатике

Задача 1.6.1 (1 балл)

Условие:

Вы попали на завод роботов! Все роботы на заводе пронумерованы натуральными числами, начиная с 1. Так как производство не останавливается ни на секунду, то можно

считать, что количество роботов бесконечно. Назовем пару роботов комбинируемой, если сумма номеров роботов не превышает k.

Вам дано число k, требуется посчитать количество неупорядоченных комбинируемых пар роботов. Пара называется неупорядоченной, если для ее обозначения не важен порядок (т.е пары (1, 2) и (2, 1) не различаются).

Формат входных данных:

Задано единственное число k $(1 \le k \le 30\,000)$.

Формат выходных данных:

Выведите единственное число - количество неупорядоченных комбинируемых пар роботов.

Пример:

stdin:
5
stdout:
4

Пояснение:

В тесте из примера возможны следующие пары: (1, 2), (1, 3), (1, 4), (2, 3).

Решение:

Посмотрим сначала количество пар роботов, чтобы их сумма была ровно k, таких чисел $\frac{k-1}{2}$, таким образом нам нужно посчитать $\sum_{1}^{k} \frac{k-1}{2}$, что является арифметической прогрессией.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    k = int(dataset)
    return str((k - 1) // 2 * (k // 2))

print(solve(sys.stdin.read()))
```

Задача 1.6.2 (2 балла)

Условие:

Для управления роботами центральный компьютер посылает им команды, состоящие из символом латинского алфавита. Количество операций растет каждый день и они уже не вмещаются в память роботов. Поэтому вам поручили реализовать метод сжатия строк, основанный на повторяющихся символах.

Сжатие строки происходит по следующему алгоритму: изначальная строка делится на отрезки максимальной длины, состоящие из одинаковых символов. После чего каждый отрезок заменяется на сам повторяющийся символ и количество его повторов, но при этом, если символ повторяется только один раз, то количество его повторов не выписывается.

Формат входных данных:

Единственная строка входных данных содержит саму строку, состоящую только из строчных символов латинского алфавита, длина строки не превосходит 50000.

Формат выходных данных:

Выведите сжатую строку.

Пример №1:

stdin:
aabccc
stdout:
a2bc3

Пример №2:

stdin:
abcd
stdout:
abcd

Решение:

Запомним указатель на текущий элемент, после чего будем сдвигать его, пока символы одинаковые. Осталось посчитать длину отрезка и вывести в нужном формате. Продолжаем, пока не дойдем до конца строки. Не забываем, что в некоторых языках строки являются неизменяемыми, поэтому при добавление нового символа строка будет пересоздаваться заново.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    s = dataset.split()[0]
    res = []
    i = 0
    while i < len(s):
        j = i
        while j < len(s) and s[i] == s[j]:
            j += 1
        res.append(s[i])
        if not j - i == 1:
            res.append(j - i)
        i = j
    return ''.join(str(x) for x in res)

print(solve(sys.stdin.read()))
```

Задача 1.6.3 (2 балла)

Условие:

В логистическом центре, где всю работу выполняют роботы – очень длинные стеллажи. На стеллаже умещается n коробок с грузами. Робот-погрузчик может манипулировать одновременно только с 1 или 2 коробками. За какое минимальное количество манипуляций робот сможет разгрузить весь стеллаж от грузов, при условии,

что данное количество манипуляций должно быть кратно числу m из-за особенностей системы управления роботами-погрузчиками.

Формат входных данных:

Единственная строка входных данных содержит числа n и m $(1 \le n \le 50\,000, 1 \le m \le 100)$.

Формат выходных данных:

Выведите одно число — минимальное количество манипуляций, кратное m. Если такого числа не существует, тогда выведите -1.

Пример:

stdin:
10 2
stdout:
6

Решение:

Если мы знаем количество операций с двумя коробками, то $n - 2 \cdot i$ операций будут с одной коробкой. Поэтому переберем количество двойных манипуляций, и вычислив количество одинарных запомним минимальный ответ.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    mx = 0xfffffffff
    n, m = [int(x) for x in dataset.split()]
    ans = mx
    for i in range(0, n // 2 + 1):
        if (i + (n - 2 * i)) % m == 0:
            ans = min(ans, i + (n - 2 * i))
    return "-1" if ans == mx else str(ans)

print(solve(sys.stdin.read()))
```

Задача 1.6.4 (2 балла)

Условие:

В новом почтовом отделении вместо доставки писем по адресу, решили доставлять их в абонентские ящики, пронумерованные от 1 до n. У почтового робота есть операция доставки рассылки в абонентские ящики с номерами от L до R, включительно. Он добавляет ровно один буклет в каждый ящик. Известно, что операцию доставки использовали k раз. Помогите руководству узнать максимальное количество буклетов в ящиках.

Формат входных данных:

Первая строка входных данных содержит два числа n и k - количество абонентских ящиков и рассылок буклетов $(1 \le n \le 1\,000\,000, 1 \le k \le 25\,000)$. Следующие k строк содержат по два числа L и R, границы операции для доставки рассылок $(1 \le L \le R \le n)$.

Формат выходных данных:

Выведите единственное число - максимальное количество буклетов в ящике.

Пример:

stdin:
7 4
5 5
2 4
4 6
3 5
stdout:
3

Пояснение:

Абонентские ящики из примера: [0, 1, 2, 3, 3, 1, 0].

Решение:

Решение “в лоб” за O(n · k) операций не подходит по времени, поэтому такое решение не рассматриваем. Заметим, что одну операцию мы можем разбить на две другие: добавить 1 ко всем элементам, начиная с L, и вычесть 1 у всех элементов начиная с R + 1. Таким образом мы можем посчитать разность между всеми соседними элементами, более того, зная такие разности мы можем восстановить наш массив. После чего остается найти максимум в таком массиве.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    arr = [int(x) for x in dataset.split()]
    n, k, arr = arr[0], arr[1], arr[2:]
    a = [0] * (n + 2);
    for i in range(0, k):
        l, r = arr[2 * i] - 1, arr[2 * i + 1] - 1
        a[l] += 1
        a[r + 1] -= 1
    val = 0
    ans = 0
    for i in range(0, n):
        val += a[i]
        a[i] = val;
        ans = max(ans, val)
    return str(ans)

print(solve(sys.stdin.read()))
```

Задача 1.6.5 (3 балла)

Условие:

Логистический центр представляет из себя сложную систему переходов между различными помещениями, по которым перемещаются роботы-погрузчики. Все помещения пронумерованы от 1 до n. Один из роботов получил задачу перенести груз из первого помещения в последнее. Время перемещения от входа в одно помещение до входа в другое занимает 1 минуту, внутри помещения робот перемещается мгновенно.

Для перемещения по маршруту роботу не обязательно посещать все помещения, какие есть в центре, но он должен обязательно посетить помещение с номером k, где происходит контроль целостности груза, проверка также происходит мгновенно.

Ваша задача найти минимальное время, которое робот потратит на прохождение от первого помещения до последнего, заглянув на проверку целостности груза. Если пути не существует, выведите −1.

Формат входных данных:

Первая строка входных данных содержит три числа n, m, k - количество помещений, количество переходов, соединяющих помещения, и номер помещения проверки целостности груза, соответственно

$$(3 \le n \le 10^5, 2 \le m \le \min(n \cdot \frac{n-1}{2}, 10^5), 2 \le k < n)$$.

В следующих m строках заданы пары чисел v и u - номера помещений, которые соединены двусторонним переходом $(1 \le v \le u \le n)$. Никакие два помещения не соединены двумя переходами, и переход не соединяет помещение с самим с собой.

Формат выходных данных:

Выведите единственное число - минимальное время, необходимое для прохождения маршрута, или -1, если это невозможно.

Пример:

stdin:
5 5 3
1 2
1 3
2 3
3 4
4 5
stdout:
3

Решение:

Найдем сначала кратчайшее расстояние от первого помещения до помещения с номером k, сделать это можно алгоритмом поиска в ширину. Таким же образом найдем кратчайшее расстояние от помещения k до помещения n.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

INF = 10 ** 6

def bfs(n, start, finish, edges):
    d = [INF for i in range(n)]
    d[start] = 0
    q = deque()
    q.append(start)
    while len(q) > 0:
        u = q.popleft()
        for v in edges[u]:
            if d[u] + 1 < d[v]:
                d[v] = d[u] + 1
                q.append(v)
```

```
    return d[finish]

def solve(dataset):
    s = dataset.splitlines()
    n, m, k = map(int, s[0].split())
    edges = [[] for i in range(n)]

    for i in range(1, m + 1):
        u, v = map(int, s[i].split())
        edges[u - 1].append(v - 1)
        edges[v - 1].append(u - 1)

    res1 = bfs(n, 0, k - 1, edges)
    res2 = bfs(n, k - 1, n - 1, edges)
    if res1 == INF or res2 == INF:
        return "-1\n"
    return str(res1 + res2) + "\n"

print(solve(sys.stdin.read()))
```

1.7 Критерии отбора победителей и призеров

Количество баллов, набранных при решении задач одной попытки, суммируется. Если участник решал задачи, как в первой, так и во второй попытке, то выбирается попытка с большей суммой баллов. Призерам первого отборочного этапа необходимо было набрать 9 баллов (для 9 класса) и 10 баллов (для 10-11 класса). Победители первого отборочного этапа должны были набрать 18 баллов.

§2 Второй отборочный этап

Второй отборочный этап проводится в командном формате в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Продолжительность второго этапа составляет 2 месяца. Задачи условно разделены на задачи по математике и информатике, но носят междисциплинарный характер и в более простой форме воссоздают инженерную задачу заключительного этапа. Для каждого из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике - общие для всех участников. Участники не были ограничены в выборе языка программирования для решения задач.

Объем и сложность задач этого этапа подобраны таким образом, чтобы решение всех задач одним человеком было маловероятно. Это призвано обеспечить включение командной работы и распределения обязанностей. Решение каждой задачи дает определенное количество баллов. Если команда состоит из участников 9 и 10-11 классов, то команде засчитываются только те баллы по математике, которые набраны за решения задач 10-11 класса. Баллы зачисляются в полном объеме за правильное решение задачи. В данном этапе можно получить суммарно от 0 до 95 баллов.

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

2.1 Первая попытка
Задачи по математике (9 класс)

Задача 2.1.1 (3 балла)

Условие:

Преобразование плоскости называется *движением*, если оно сохраняет расстояние между точками. Композиция преобразований f и g обозначается g ∘ f: сначала применяем преобразование f, потом g.

Центральная симметрия Z_O относительно точки O - такое преобразование, что

$$X' = Z_O(X) \Leftrightarrow \overrightarrow{OX'} = -\overrightarrow{OX}.$$

Поворот R_A^α переводит точку X в такую точку X', что $\angle XOX' = \alpha$, измеренное против часовой стрелки.

Параллельный перенос $T_{\vec{a}}$ точку X переводит в такую точку X', что $\overrightarrow{XX'} = \vec{a}$.

Осевая симметрия S_l относительно прямой l переводит точку X в такую точку X', что l является серединным перпендикуляром к отрезку XX', при этом точки прямой l остаются на месте.

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

1. Движение является параллельным переносом тогда и только тогда, когда каждый вектор оно переводит в равный ему вектор (т.е. каждый направленный отрезок переводит в эквивалентный ему);
2. Композиция двух центральных симметрий $Z_B \circ Z_A$, есть параллельный перенос на $2 \cdot \overrightarrow{AB}$;
3. Композиция двух осевых симметрий, есть параллельный перенос или поворот;

4. Композиция двух поворотов $R_A^\alpha \circ R_B^\beta$, есть поворот вокруг некоторой точки на угол $360^\circ - \alpha - \beta$;
5. Любое движение является либо параллельным переносом, либо поворотом, либо осевой симметрией, либо центральной симметрией;
6. Существует композиция из трех осевых симметрий, оставляющая все точки на своих местах;
7. $S_l \circ T_{\vec{a}} = T_{\vec{a}} \circ S_l$;
8. Для любой точки A, любого вектора $\vec{a}$ и любого угла α существуют такие прямые n и m, что $Z_A \circ T_{\vec{a}} \circ R_A^\alpha = S_n \circ S_m$.

Ответ: 11100001

Задача 2.1.2 (3 балла)

Условие:

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

1. Расстояние d от точки (x_0; y_0) до прямой $a \cdot x + b \cdot y + c = 0$ равно
$$d = \frac{a \cdot x_0 + b \cdot y_0 + c}{\sqrt{a^2 + b^2}};$$
2. Окружность проходит через точку (2; 1) и касается осей координат. Тогда она задается уравнением $(x - 1)^2 + (y - 1)^2 = 1$;
3. В треугольнике с вершинами в точках (x_1; y_1), (x_2; y_2) и (x_3; y_3) точка пересечения медиан имеет координаты
$$\left(\frac{2x_1 + 2x_2 + 2x_3}{3};\ \frac{2y_1 + 2y_2 + 2y_3}{3}\right);$$
4. Прямая с угловым коэффициентом k проходит через точку (x_0; y_0) тогда и только тогда, когда ее уравнение принимает вид
$$y - y_0 = k \cdot (x - x_0);$$
5. Две прямые, заданные уравнениями $y = k_1 \cdot x + l_1$ и $y = k_2 \cdot x + l_2$, перпендикулярны тогда и только тогда, когда $k_1 \cdot k_2 = -1$;
6. При повороте на угол α (по часовой стрелке) с центром в начале координат точка с координатами (x; y) переходит в точку
$$(x \cdot \cos\alpha - y \cdot \sin\alpha;\ x \cdot \sin\alpha + y \cdot \cos\alpha);$$
7. Множество точек, удовлетворяющих уравнению
$$x^2 + y^2 = x + y + \frac{1}{2},$$
является окружностью;
8. Площадь треугольника с вершинами в точках (0; 0), (x_1; y_1) и (x_2; y_2) равна
$$\frac{|x_1 \cdot y_2 - x_2 \cdot y_1|}{2}.$$

Ответ: 00011011

Задача 2.1.3 (3 балла)

Условие:

Везде далее I обозначает единичную матрицу $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Вектора удобно рассматривать матрицами, имеющими один столбец: $\vec{a}\{x; y\} = \begin{pmatrix} x \\ y \end{pmatrix}$.

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

1. Дана матрица $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, тогда $detA = a \cdot d - b \cdot c$;
2. Для любой невырожденной матрицы A размера 2×2 выполняется равенство $$(A + A^{-1})^2 = A^2 + (A^{-1})^2 + 2 \cdot I;$$
3. Для любых матриц A и B размера 2×2 выполняется равенство $$(A \cdot B^T)^T = A^T \cdot B;$$
4. Если матрицы $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ и $B = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$, то $A^T \cdot B^T = \begin{pmatrix} 8 & 20 \\ 5 & 13 \end{pmatrix}$;
5. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда направленный угол $\angle(\overrightarrow{OM};\ \overrightarrow{OM'})$ равен α.
6. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда точки M и M' симметричны относительно прямой $y = \alpha \cdot x$.
7. Для любых матриц A и B размера 2×2 выполняется равенство $$(A + B)^2 = B^2 + 2 \cdot A \cdot B + A^2;$$
8. Если матрицы $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ и $B = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$, то $3 \cdot A - 2 \cdot B = 5 \cdot \begin{pmatrix} -1 & 0 \\ 0 & 2 \end{pmatrix}$.

Ответ: 11000100

Задача 2.1.4 (2 балла)

Условие:

Найдите сумму коэффициентов матрицы $\frac{1}{5^{100}} \cdot A^{100}$, если матрица $$A = \begin{pmatrix} 5 & 1 & 0 \\ 0 & 5 & 1 \\ 0 & 0 & 5 \end{pmatrix}.$$

Решение:

Пусть матрица $$B = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

тогда матрицу A можно представить в виде

$$A = 5 \cdot I + B,$$

тогда

$$A^n = (5 \cdot I + B)^n.$$

Раскроем по биному Ньютона и приведем подобные члены (проблем с приведением подобных членов не будет, так как матрицы I и B перестановочны)

$$A^n = \sum_{k=0}^{n} \frac{n!}{k! \cdot (n-k)!} \cdot 5^k \cdot I^k \cdot B^{n-k}$$

Почти все слагаемые этой суммы равны нулевой матрице, так как $B^{n-k} = 0$ при $n - k \geq 3$. Поэтому

$$A^n = 5^n \cdot I^n + n \cdot 5^{n-1} \cdot I^{n-1} \cdot B + \frac{n(n-1)}{2} \cdot 5^{n-2} \cdot I^{n-2} \cdot B^2 =$$

$$= 5^n \cdot I + n \cdot 5^{n-1} \cdot B + \frac{n(n-1)}{2} \cdot 5^{n-2} \cdot B^2$$

То есть матрица A^n имеет следующий вид

$$A^n = \begin{pmatrix} 5^n & n \cdot 5^{n-1} & \frac{n(n-1)}{2} \cdot 5^{n-2} \\ 0 & 5^n & n \cdot 5^{n-1} \\ 0 & 0 & 5^n \end{pmatrix} = 5^n \cdot \begin{pmatrix} 1 & \frac{n}{5} & \frac{n(n-1)}{50} \\ 0 & 1 & \frac{n}{5} \\ 0 & 0 & 1 \end{pmatrix}$$

Следовательно,

$$\frac{1}{5^{100}} \cdot A^{100} = \begin{pmatrix} 1 & 20 & 198 \\ 0 & 1 & 20 \\ 0 & 0 & 1 \end{pmatrix}$$

Значит ответом в этой задаче будет число 1 + 1 + 1 + 20 + 20 + 198 = 241.

Ответ: 241

Задача 2.1.5 (2 балла)

Условие:

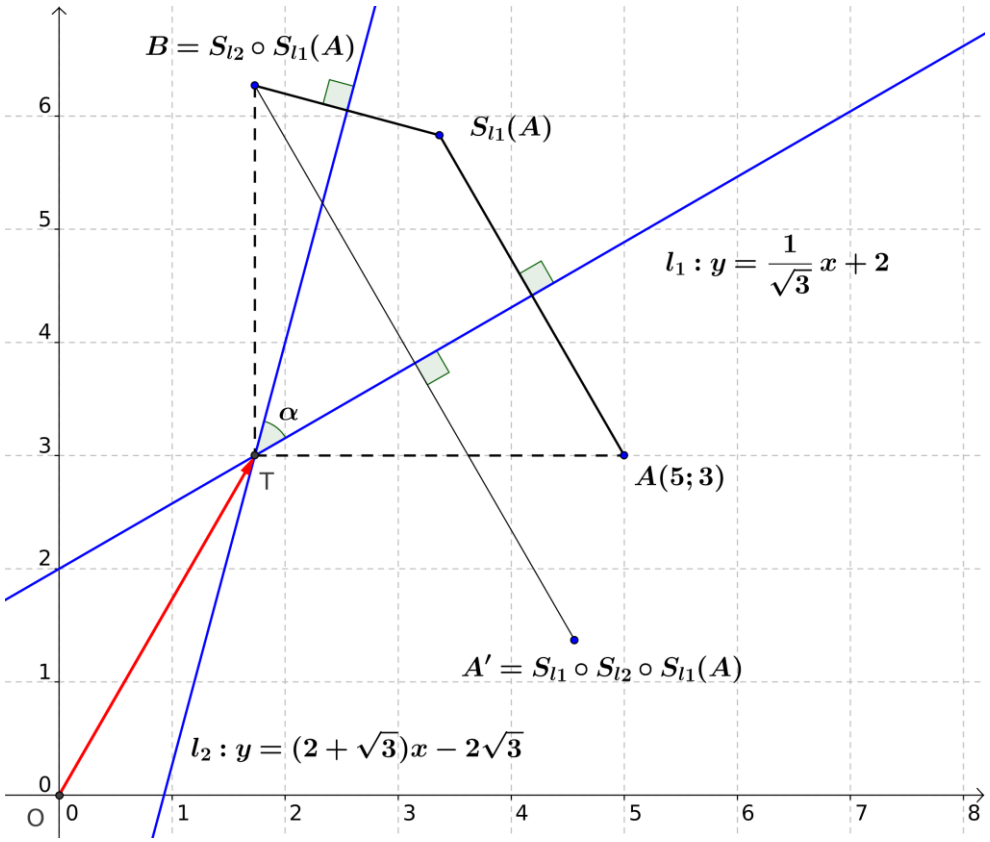
Даны две прямые $l_1: \ \sqrt{3} \cdot y = x + 2\sqrt{3}$, $l_2: \ y = (2 + \sqrt{3}) \cdot x - 2\sqrt{3}$, и точка A(5;3). Обозначим через $S_l(A)$ образ точки A при осевой симметрии относительно прямой l. Найдите координаты точки A', где

$$A' = S_{l_1} \circ S_{l_2} \circ S_{l_1}(A).$$

Значения координат округлите до тысячных. Ответ запишите в формате (-1.014; 5.000).

Решение:

Вариант 1



Пусть T - точка пересечения прямых l_1 и l_2, тогда найдем ее координаты:

$$\frac{1}{\sqrt{3}} \cdot x + 2 = (2+\sqrt{3}) \cdot x - 2\sqrt{3}$$

$$x = \frac{1+\sqrt{3}}{1+\frac{1}{\sqrt{3}}} = \frac{\sqrt{3}+3}{\sqrt{3}+1} = \sqrt{3} \qquad y = (2+\sqrt{3})\sqrt{3} - 2\sqrt{3} = 3$$

Получаем, что точка T имеет координаты $(\sqrt{3};\ 3)$. Найдем координаты точки $B = S_{l_2} \circ S_{l_1}(A)$. Так как композиция двух осевых симметрий $S_{l_2} \circ S_{l_1}$ есть не что иное, как поворот $R_T^{2\alpha}$, где α - ориентированный угол $\angle(l_1;\ l_2)$, то $B = R_T^{2\alpha}(A)$. Ввиду того, что

$$\angle(l_1;\ l_2) = \angle(Ox;\ l_2) - \angle(Ox;\ l_1),$$

то

$$\angle(l_1;\ l_2) = \arctan\frac{1}{2+\sqrt{3}} - \arctan\sqrt{3} = 75^\circ - 30^\circ = 45^\circ$$

Следовательно, $B = R_T^{90^\circ}(A)$. В данном случае координаты точки B находятся очень просто (так как точки A и T имеют одинаковую координату y). В итоге получаем $B_x = T_x = \sqrt{3}$ $B_y = T_y + |TA| = 3 + (5 - \sqrt{3}) = 8 - \sqrt{3}$.

Теперь найдем координаты точки A':

$$A' = S_{l_1} \circ S_{l_2} \circ S_{l_1}(A) = S_{l_1}(B) = T_{\overrightarrow{OT}} \circ S_{l_1} \circ T_{-\overrightarrow{OT}}(B).$$

Параллельный перенос мы делаем для того, чтобы осевая симметрия совершалась относительно прямой проходящей через начало координат. Такая осевая симметрия эквивалентна домножению слева на матрицу

$\begin{pmatrix}\cos 2\beta & \sin 2\beta \\ \sin 2\beta & -\cos 2\beta\end{pmatrix}$, где β - направленный угол между осью абсцисс и осью симметрии. Угол ..., поэтому

$$A' = T_{\overrightarrow{OT}} \circ S_{l_1} \circ T_{-\overrightarrow{OT}} \left(\begin{pmatrix} \sqrt{3} \\ 8-\sqrt{3} \end{pmatrix} \right) = T_{\overrightarrow{OT}} \circ S_{l_1} \left(\begin{pmatrix} 0 \\ 5-\sqrt{3} \end{pmatrix} \right) =$$

$$= T_{\overrightarrow{OT}} \left(\begin{pmatrix} \cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 5-\sqrt{3} \end{pmatrix} \right) = T_{\overrightarrow{OT}} \left(\begin{pmatrix} \frac{5\sqrt{3}-3}{2} \\ \frac{\sqrt{3}-5}{2} \end{pmatrix} \right) = \begin{pmatrix} \frac{7\sqrt{3}-3}{2} \\ \frac{\sqrt{3}+1}{2} \end{pmatrix}$$

Вариант 2

Как и в предыдущем решении найдем координаты точки T пересечения прямых l_1 и l_2. Точку A' можно найти следующим образом

$$A' = T_{\overrightarrow{OT}} \circ S_{l_1} \circ S_{l_2} \circ S_{l_1} \circ T_{-\overrightarrow{OT}}(A).$$

Мы знаем, что осевая симметрия относительно прямой, проходящей через начало координат эквивалентна домножению слева на матрицу $\begin{pmatrix}\cos 2\beta & \sin 2\beta \\ \sin 2\beta & -\cos 2\beta\end{pmatrix}$, где β -- направленный угол между осью абсцисс и осью симметрии.

Матрица соответствующая преобразованию S_{l_1} равна $\begin{pmatrix}\cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ\end{pmatrix}$, а преобразованию S_{l_2}: $\begin{pmatrix}\cos 150^\circ & \sin 150^\circ \\ \sin 150^\circ & -\cos 150^\circ\end{pmatrix}$. Тогда матрица преобразования $S_{l_1} \circ S_{l_2} \circ S_{l_1}$ равна

$$\begin{pmatrix} \cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ \end{pmatrix} \cdot \begin{pmatrix} \cos 150^\circ & \sin 150^\circ \\ \sin 150^\circ & -\cos 150^\circ \end{pmatrix} \cdot \begin{pmatrix} \cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ \end{pmatrix} =$$

$$= \begin{pmatrix} \cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ \end{pmatrix} \cdot \begin{pmatrix} \cos 90^\circ & -\sin 90^\circ \\ \sin 90^\circ & \cos 90^\circ \end{pmatrix} = \begin{pmatrix} \cos 30^\circ & -\sin 30^\circ \\ -\sin 30^\circ & -\cos 30^\circ \end{pmatrix}$$

В итоге получаем

$$A' = T_{\overrightarrow{OT}} \left(\begin{pmatrix} \cos 30^\circ & -\sin 30^\circ \\ -\sin 30^\circ & -\cos 30^\circ \end{pmatrix} \cdot T_{-\overrightarrow{OT}} \left(\begin{pmatrix} 5 \\ 3 \end{pmatrix} \right) \right) =$$

$$= T_{\overrightarrow{OT}} \left(\begin{pmatrix} \cos 30^\circ & -\sin 30^\circ \\ -\sin 30^\circ & -\cos 30^\circ \end{pmatrix} \cdot \begin{pmatrix} 5-\sqrt{3} \\ 0 \end{pmatrix} \right) = T_{\overrightarrow{OT}} \left(\begin{pmatrix} \frac{5\sqrt{3}-3}{2} \\ \frac{\sqrt{3}-5}{2} \end{pmatrix} \right) = \begin{pmatrix} \frac{7\sqrt{3}-3}{2} \\ \frac{\sqrt{3}+1}{2} \end{pmatrix}$$

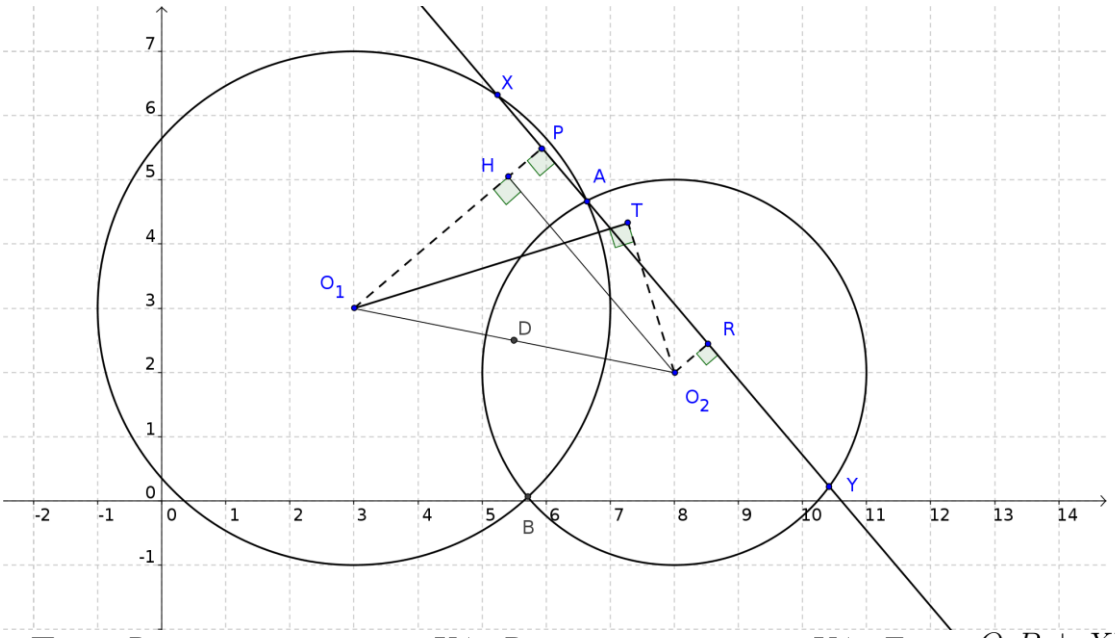
Ответ: (4,562; 1,366)

Задача 2.1.6 (2 балла)

Условие:

Даны две окружности $S_1: \ (x-3)^2 + (y-3)^2 = 16$ и $S_2: \ (x-8)^2 + (y-2)^2 = 9$, пересекающиеся в точках A и B (координаты точки A больше координат точки B). Через точку A провели две прямые l_1 и l_2. Прямая l_1 пересекает S_1 в точке X и S_2 в точке Y, а прямая l_2 пересекает S_1 в точке M и S_2 в точке N. Оказалось, что XY = MN = 8. Найдите произведение угловых коэффициентов прямых l_1 и l_2.

Решение:



Пусть P - середина хорды XA, R - середина хорды YA. Тогда $O_1P \perp XY$ и $O_2R \perp XY$. Точка H на прямой O_1P такая, что $O_2H \perp O_1P$. Тогда

$$O_2H = RP = RA + AP = \frac{XA + AY}{2} = \frac{XY}{2} = 4.$$

Аналогично находим точку T для прямой l_2 и получим $O_1T = 4$. Угловые коэффициенты прямых XY и MN равны соответственно угловым коэффициентам O_2H и O_1T. Пусть острый угол между Ox и O_1O_2 равен α и $\angle O_1O_2H = \beta$. Тогда угловой коэффициент прямой OH равен $k_1 = \tan(180 - \alpha - \beta) = -\tan(\alpha + \beta)$. А так как $\triangle O_2O_1H = \triangle O_1O_2T$, то $\angle O_2O_1T = \angle O_1O_2H = \beta$. Следовательно, угловой коэффициент прямой O_1T равен $k_2 = \tan(\beta - \alpha)$. Найдем $\tan\alpha$ и $\tan\beta$. Первый коэффициент находится из координат точек O_1 и O_2, а второй - из теоремы Пифагора и отношения катетов треугольника O_2O_1H. Получаем

$$\tan\alpha = \frac{1}{5} \text{ и } \tan\beta = \frac{\sqrt{1^2 + 5^2 - 4^2}}{4} = \frac{\sqrt{10}}{4}$$

Теперь мы можем найти ответ на задачу

$$k_1k_2 = -\tan(\alpha+\beta)\tan(\beta-\alpha) = -\frac{\tan\alpha + \tan\beta}{1 - \tan\alpha\tan\beta} \cdot \frac{\tan\beta - \tan\alpha}{1 + \tan\alpha\tan\beta} =$$
$$= \frac{\tan^2\alpha - \tan^2\beta}{1 - \tan^2\alpha\tan^2\beta} = \frac{\frac{1}{25} - \frac{10}{16}}{1 - \frac{1}{25}\cdot\frac{10}{16}} = -0.6$$

Ответ: -0,6

2.2 Первая попытка
Задачи по математике (10-11 класс)

Задача 2.2.1 (3 балла)

Условие:

Везде далее $I = (a_{i,j})$ обозначает квадратную единичную матрицу

$$a_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{else} \end{cases}.$$

Вектора удобно рассматривать матрицами, имеющими один столбец:

$$\vec{a}\{x; y\} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

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

1. Площадь параллелограмма образованного векторами $\begin{pmatrix} a \\ b \end{pmatrix}$ и $\begin{pmatrix} c \\ d \end{pmatrix}$ равна модулю определителя матрицы $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$;
2. Для любой невырожденной матрицы A выполняется равенство
$$(A + A^{-1})^2 = A^2 + (A^{-1})^2 + 2 \cdot I;$$
3. Для любых матриц A и B выполняется равенство
$$(A \cdot B^T)^T = A^T \cdot B;$$
4. Если матрицы A и B симметрические, то и матрица A · B симметрическая;
5. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда направленный угол $\angle(\overrightarrow{OM};\ \overrightarrow{OM'})$ равен α.
6. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда точки M и M' симметричны относительно прямой $y = \tan\alpha \cdot x$.
7. Для любых матриц A и B выполняется равенство
$$(A + B)^2 = B^2 + 2 \cdot A \cdot B + A^2;$$
8. Если матрицы A и B коммутативны, то для любых натуральных m и n матрицы A^n и B^m коммутативны.

Ответ: 11000101

Задача 2.2.2 (3 балла)

Условие:

Далее точки плоскости мы будем обозначать заглавными буквами, а соответствующие им комплексные числа маленькими. Если $a = x + iy$, то $\bar{a} = x - iy$ комплексное сопряжение a.

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

1. Длина отрезка AB равна $(a - b) \cdot (\bar{a} - \bar{b})$;
2. Скалярное произведение векторов равно
$$\overrightarrow{AB} \cdot \overrightarrow{CD} = \frac{1}{2} \cdot (a - b) \cdot (c - d) + \frac{1}{2} \cdot (\bar{a} - \bar{b}) \cdot (\bar{c} - \bar{d});$$
3. Поворот по часовой стрелке на 90° с центром в начале координат задается уравнением $f(z) = -i \cdot z$;
4. Вектор $\overrightarrow{AB}$ параллелен вектору $\overrightarrow{CD}$ тогда и только тогда, когда $\dfrac{b-a}{d-c}$ - действительное число;
5. Вектор $\overrightarrow{AB}$ перпендикулярен вектору $\overrightarrow{CD}$ тогда и только тогда, когда
$$(\bar{b} - \bar{a}) \cdot (d - c) + (b - a) \cdot (\bar{d} - \bar{c}) = 0;$$

6. Прямая проходящая через точки A и B задается уравнением
$$(\bar{a} - \bar{b}) \cdot z + (b - a) \cdot \bar{z} + a \cdot \bar{b} + b \cdot \bar{a} = 0;$$
7. Точки A и B лежат на единичной окружности с центром в начале координат. Тогда комплексная координата основания перпендикуляра, опущенного из точки M на прямую, проходящую через точки A и B, находится по формуле
$$\frac{1}{2} \cdot (a + b + m + a \cdot b \cdot \bar{m})$$
8. Точки A, B, C и D лежат на единичной окружности с центром в начале координат, причем $AB \perp CD$. Тогда комплексная координата точки пересечения отрезков AB и CD находится по формуле
$$\frac{1}{2} \cdot (a + b + c + d).$$

Ответ: 10111001

Задача 2.2.3 (3 балла)

Условие:

A - матрица смежности неориентированного графа G = (V; E) (без петель и кратных ребер), где множество вершин V = {v_1, v_2, ..., v_n} и длины ребер равны 1. Число, стоящее на пересечении i строки и j столбца матрицы A, будем называть (i; j)-элементом.

Через tr(A) будем обозначать сумму диагональных элементов матрицы A.

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

1. (i; j) - элемент матрицы A^3 равен количеству путей длины 4 из v_i в v_j;
2. (i; i) - элемент матрицы A^2 равен 0;
3. (i; i) - элемент матрицы A^3 равен удвоенному числу треугольников содержащих v_i;
4. Если граф G связный, то расстояние между различными вершинами v_i и v_j равно наименьшему из натуральных чисел m, для которых (i; j) - элемент матрицы A^m отличен от 0;
5. Граф G двудольный тогда и только тогда, когда матрица смежности имеет вид
$$\begin{pmatrix} 0 & \cdots & 0 & a_{1,1} & \cdots & a_{1,l} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{t,1} & \cdots & a_{t,l} \\ a_{1,1} & \cdots & a_{t,1} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{1,l} & \cdots & a_{t,l} & 0 & \cdots & 0 \end{pmatrix} \qquad a_{i,j} \in \{0; 1\}$$
для некоторых натуральных t и l;
6. Граф G двудольный тогда и только тогда, когда для любого нечетного числа n все диагональные элементы матрицы A^n равны 0;
7. Число простых циклов длины 3 графа G равно $\frac{1}{3} \cdot \mathrm{tr}(A^3)$;
8. Сумма элементов i-ой строки равна степени вершины v_i.

Ответ: 00110101

Задача 2.2.4 (2 балла)

Условие:

Найдите сумму коэффициентов матрицы $\frac{1}{5^{100}} \cdot A^{100}$, если матрица $A = \begin{pmatrix} 5 & 3 & 1 \\ 0 & 5 & 3 \\ 0 & 0 & 5 \end{pmatrix}$.

Решение:

Пусть матрица $B = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$, тогда $B^2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$. Представим нашу матрицу A в виде

$$A = 5 \cdot I + 3 \cdot B + 1 \cdot B^2$$

тогда

$$A^n = (5 \cdot I + 3 \cdot B + 1 \cdot B^2)^n$$

Раскроем по обобщенному биному Ньютона и приведем подобные члены (проблем с приведением подобных членов не будет, так как матрицы I, B и B^2 перестановочны)

$$A^n = \sum_{\substack{k_j \geq 0 \\ k_1+k_2+k_3=n}} \frac{n!}{k_1! \cdot k_2! \cdot k_3!} \cdot (5I)^{k_1} \cdot (3B)^{k_2} \cdot (B^2)^{k_3} =$$

$$\sum_{\substack{k_j \geq 0 \\ k_1+k_2+k_3=n}} \frac{n!}{k_1! \cdot k_2! \cdot k_3!} \cdot 5^{k_1} \cdot 3^{k_2} \cdot I^{k_1} \cdot B^{k_2+2k_3}$$

Заметим, что $B^k = 0$ (нулевой матрице) при $k \geq 3$, поэтому почти все слагаемые бинома Ньютона равны 0

$$A^n = 5^n \cdot I^n + n \cdot 5^{n-1} \cdot 3 \cdot I^{n-1} \cdot B + \frac{n(n-1)}{2} \cdot 5^{n-2} \cdot 3^2 \cdot I^{n-2} \cdot B^2 +$$

$$+n \cdot 5^{n-1} \cdot I^{n-1} \cdot B^2 = 5^n \cdot I + 3 \cdot n \cdot 5^{n-1} \cdot B + \frac{1}{2} \cdot 5^{n-2} \cdot n \cdot (9n+1) \cdot B^2$$

То есть матрица A^n имеет следующий вид

$$A^n = \begin{pmatrix} 5^n & 3n \cdot 5^{n-1} & \frac{n(9n+1)}{2} \cdot 5^{n-2} \\ 0 & 5^n & 3n \cdot 5^{n-1} \\ 0 & 0 & 5^n \end{pmatrix} = 5^n \cdot \begin{pmatrix} 1 & \frac{3n}{5} & \frac{n(9n+1)}{50} \\ 0 & 1 & \frac{3n}{5} \\ 0 & 0 & 1 \end{pmatrix}$$

Следовательно,

$$\frac{1}{5^{100}} \cdot A^{100} = \begin{pmatrix} 1 & 60 & 1802 \\ 0 & 1 & 60 \\ 0 & 0 & 1 \end{pmatrix}$$

Значит ответом в этой задаче будет число 1 + 1 + 1 + 60 + 60 + 1802 = 1925.

Ответ: 1925

Задача 2.2.5 (2 балла)

Условие:

На комплексной плоскости даны две точки $A(2 + 4 \cdot i)$ и $B(4 + 0 \cdot i)$. Осевая симметрия $f(z) = n \cdot \overline{z} + m$ переводит точки A и B друг в друга. Найдите значение $|n|^2 + |m|^2$.

Решение:

Обозначим комплексные числа соответствующее точкам A и B за a и b соответственно. Возьмем в комплексной плоскости произвольную точку Z(z) и ее образ Z'(z') при осевой симметрии. На основании AZ = BZ' и BZ = AZ' имеем

$$(z-a)\cdot(\overline{z}-\overline{a})=(z'-b)\cdot(\overline{z'}-\overline{b})$$ и $$(z-b)\cdot(\overline{z}-\overline{b})=(z'-a)\cdot(\overline{z'}-\overline{a})$$

Вычтем из первого уравнения второе и получим

$$(a\overline{a}-b\overline{b})+z(\overline{b}-\overline{a})+\overline{z}(b-a)=(b\overline{b}-a\overline{a})+z'(\overline{a}-\overline{b})+\overline{z'}(a-b)$$

или

$$2(a\overline{a}-b\overline{b})+(z+z')(\overline{b}-\overline{a})+(\overline{z}+\overline{z'})(b-a)=0$$

Поделим обе части на $(\overline{a}-\overline{b})$

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-(z+z')-(\overline{z}+\overline{z'})\cdot\frac{a-b}{\overline{a}-\overline{b}}=0 \qquad (*)$$

Так как AB||ZZ', то

$$\frac{z-z'}{\overline{z}-\overline{z'}}=\frac{a-b}{\overline{a}-\overline{b}} \qquad (**)$$

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-(z-z')-(\overline{z}+\overline{z'})\cdot\frac{z-z'}{\overline{z}-\overline{z'}}=2z'$$

или

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-\frac{(z-z')(\overline{z}-\overline{z'})+(\overline{z}+\overline{z'})(z-z')}{\overline{z}-\overline{z'}}=2z'$$

Раскрываем скобки и приводим подобные члены

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-\frac{2\overline{z}(z-z')}{\overline{z}-\overline{z'}}=2z'$$

Делим обе части на 2 и снова применяем (**)

$$z'=\frac{(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-\overline{z}\cdot\frac{a-b}{\overline{a}-\overline{b}}$$

Теперь найдем n и m:

$$m=\frac{(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}=\frac{20-16}{-2-4i}=\frac{-2+4i}{5}$$

$$n=-\frac{a-b}{\overline{a}-\overline{b}}=-\frac{-2+4i}{-2-4i}=\frac{3+4i}{5}$$

В итоге получаем

$$|n|^2+|m|^2=\frac{1}{25}\cdot(20+25)=\frac{9}{5}=1.8$$

Ответ: 1,8

Задача 2.2.6 (2 балла)

Условие:

Дана матрица $A=(a_{i,j})$ размера 100×100, причем

$$a_{i,j}=\begin{cases}1, & \text{if } i-j=\pm1 \text{ or } \pm99;\\ 0, & \text{else.}\end{cases}$$

Найдите $\mathrm{tr}(A^{20})$

Решение:

Заметим, что A является матрицей смежности неориентированного графа-цикла на 100 вершинах. Так как (i;j)-элемент матрицы A^{20} = $(b_{i,j})$ равен количеству путей длины 20 из вершины v_i в вершину v_j, то

$$\mathrm{tr}(A^{20}) = \sum_{i=1}^{100} b_{i,i} = 100 \cdot b_{1,1} = 100 \cdot \{\ 20\ v_1 v_1\}$$

Предпоследнее равенство следует из того, что $b_{i,i} = b_{j,j}$ для любых i и j (граф симметричен относительно любой вершины). Осталось найти чему равно $b_{1,1}$.

Так как 20 < 100, то не существует циклических путей совершающих полный оборот по нашему графу-циклу. Следовательно, в любом циклическом пути должно быть ровно 10 переходов по ребрам по часовой стрелке и 10 переходов по ребрам против часовой стрелки, причем любая такая последовательность переходов соответствует ровно одному пути ведущему из v_1 в v_1. Всего таких последовательностей C_{20}^{10} (из 20 переходов нужно выбрать 10 и сказать, что они будут по часовой стрелке). В итоге получаем

$$\mathrm{tr}(A^{20}) = 100 \cdot C_{20}^{10} = 100 \cdot \frac{20!}{10! \cdot 10!} = 18475600$$

Ответ: 18475600

2.3 Первая попытка
Задачи по информатике

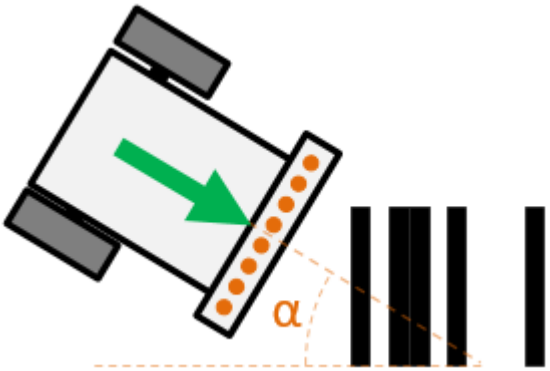
Задача 2.3.1 (10 баллов)

Условие:

Робототехническое устройство движется прямо с равномерной скоростью по плоской поверхности белого цвета. Оно оборудовано сенсором черной линии, состоящим из восьми датчиков отраженного света, расположенным на одной прямой, перпендикулярной направлению движения, на равном расстоянии друг от друга. Все датчики освещенности откалиброваны одинаково, т.е. показывают в одном и том же месте поля одинаковое значение: максимальное значение, возвращаемое датчиком на абсолютно светлой поверхности, - 100, минимальное значение, минимальное значение, возвращаемое датчиком на абсолютно черной поверхности, - 0.

В какой-то момент времени устройство начинает двигаться над участком поверхности, представляющим из себя набор из параллельных черных линий, одинаковой длины. Черные линии формируют двоичный код следующим образом:

- Первая и десятая линии являются калибровочными. Они всегда есть на поверхности поля.
- Линии со второй по девятую формируют число в двоичной системе. Если соответствующая линия есть, то она кодирует двоичную 1, если линия отсутствует, то двоичный 0.
- Вторая линия - старший бит числа, девятая линия - младший бит числа.
- Минимальное число в десятичной системе, которое может быть закодировано с помощью, данного кода - 0, максимальное - 255.



Все черные линии - одинаковой ширины.

Необходимо найти десятичное число, заданное двоичным кодом, нанесенным на участок поверхности. Угол, под которым подъезжает устройство к первой калибровочной линии, меньше 45 градусов (на рисунке обозначен α). Смещение центра сенсора черной линии относительно центра первой калибровочной линии неизвестно. Длина черных линий также неизвестна, но известно, что робототехническое устройство проезжает над кодом так, что информации, считанной со всех датчиков за время движения по поверхности с кодом, достаточно для определения кода.

Формат входных данных:

Первая строчка содержит целое число - количество замеров N $(10 \leq N \leq 100)$, выполненных сенсором черной линии.

Следующие N строчек содержат восемь целых чисел l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8 $(0 \leq l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8 \leq 100)$. Первое число l_1 - значение на крайнем левом датчике отраженного света, восьмое число l_8 - значение на крайнем правом датчике отраженного света.

Формат выходных данных:

Выведите единственное число, заданное двоичным кодом.

Пример:

stdin:

35
65 72 70 69 72 69 57 47
72 64 71 57 48 38 37 34
61 40 32 39 36 32 33 36
36 41 35 36 41 39 47 46
36 35 32 38 52 63 68 71
39 47 64 69 66 68 67 68
69 69 74 65 69 67 71 71
64 67 66 66 73 65 72 66
73 69 70 74 66 69 65 68
64 65 73 65 69 70 67 68
64 68 74 69 70 67 74 72
66 66 65 74 69 73 65 69
66 72 72 66 69 73 67 69
66 72 69 71 71 64 64 70
73 70 70 66 67 68 67 65
65 74 67 68 70 64 73 71
70 66 65 64 74 74 59 41
66 67 71 58 54 40 37 31
68 58 44 33 40 36 39 40
36 32 35 31 37 41 39 37
40 40 33 38 38 35 37 38
37 39 37 39 38 38 32 36
40 39 41 40 41 41 40 47
36 34 38 39 36 46 65 70
35 32 48 67 64 74 67 70
59 66 66 67 66 64 66 72
74 67 71 67 66 63 50 35
65 64 62 58 38 41 31 34

63 53 34 35 33 34 39 32
60 32 39 33 32 36 37 33
69 42 33 39 38 37 31 34
67 58 31 31 31 37 37 38
68 72 34 37 37 35 40 52
74 68 53 36 47 57 70 72
71 71 72 65 69 64 68 64
stdout:
13

Решение:

Решение основывается на предположении, что хотя бы один датчик из восьми прошел над всеми десятью линиями. Очевидно, что если для каждого датчика игнорировать показания, встреченные до первого и после последнего черных показаний, то датчик, в массиве которого осталось больше всего элементов – как раз тот датчик, прошедший через все линии черного цвета.

Например:

Входные данные	**Представление в виде белое/черное**
65 72 70 69 72 69 57 47	B
72 64 71 57 48 38 37 34	B B B B
61 40 32 39 36 32 33 36	B B B B B B B
36 41 35 36 41 39 47 46	B B B B B B B B
36 35 32 38 52 63 68 71	B B B B B W W W
39 47 64 69 66 68 67 68	B B W W W W W W
69 69 74 65 69 67 71 71	W W W W W W W W
64 67 66 66 73 65 72 66	W W W W W W W W
73 69 70 74 66 69 65 68	W W W W W W W W
64 65 73 65 69 70 67 68	W W W W W W W W
64 68 74 69 70 67 74 72	W W W W W W W W
66 66 65 74 69 73 65 69	W W W W W W W W
66 72 72 66 69 73 67 69	W W W W W W W W
66 72 69 71 71 64 64 70	W W W W W W W W
73 70 70 66 67 68 67 65	W W W W W W W W
65 74 67 68 70 64 73 71	W W W W W W W W
70 66 65 64 74 74 59 41	W W W W W W W B
66 67 71 58 54 40 37 31	W W W W B B B B
68 58 44 33 40 36 39 40	W W B B B B B B
36 32 35 31 37 41 39 37	B B B B B B B B
40 40 33 38 38 35 37 38	B B B B B B B B
37 39 37 39 38 38 32 36	B B B B B B B B
40 39 41 40 41 41 40 47	B B B B B B B B
36 34 38 39 36 46 65 70	B B B B B B W W
35 32 48 67 64 74 67 70	B B B W W W W W
59 66 66 67 66 64 66 72	W W W W W W W
74 67 71 67 66 63 50 35	W W W W W B B
65 64 62 58 38 41 31 34	W W W B B B B
63 53 34 35 33 34 39 32	B B B B B B B
60 32 39 33 32 36 37 33	B B B B B B B

69 42 33 39 38 37 31 34 67 58 31 31 31 37 37 38 68 72 34 37 37 35 40 52 74 68 53 36 47 57 70 72 71 71 72 65 69 64 68 64	B B B B B B B B B B B B B B B B B B B B B B

Буквой W обозначены показания выше серого (белые), B – ниже серого (черные). Серое значение следует рассчитывать как среднее арифметическое между максимальным (самым белым) и минимальным (самым черным) замерами, причем для каждого датчика отдельно.

В W/B представлении удалены лишние показания – белые замеры до встречи с первой контрольной полосой черного цвета.

Стало заметно, что максимальное полезное число замеров у датчиков под номерами 5 и 8. Можем работать с любым из них. Для примера рассмотрим датчик 8.

Количество полезных замеров – k = 33. Так как количество полос штрихкода – 10, то для успешного распознавания необходимо данные интерполировать до достижения массива длиной, кратной 10. Оптимально привести массив к длине НОК(k, 10), в нашем случае – НОК(33, 10) = 330.

При интерполяции в массив между соседними элементами записывается среднее арифметическое причем НОК(k,10)/k-1 раз, чтобы массив длиной k преобразовать в массив длиной НОК(k, 10). В нашем случае начало и конец нового массива будут выглядеть так:

0	1-9	10	11-19	20	...	320-329
47	40,5	34	35	36	...	52

После, беря среднее арифметическое для подмассивов длиной НОК(k, 10)/10 и определяя, больше ли оно серого, или нет – определяем цвет каждой полоски. Отсекаем контрольные и выводим результат, приведенный из двоичной системы счисления.

Пример программы, реализующей данный алгоритм на языке Python:

```
from fractions import gcd
# Deprecated gcd from fractions.
# gcd from math should be used
# Stepic has old version of Python,
# so has no gcd in math module.

# Arrays of white
white = [0 for k in range(8)]
# gray,
gray = [0 for k in range(8)]
# and Black values of different sensors.
black = [100 for k in range(8)]

n = int(input())
sensors = [[] for i in range(8)]

for i in range(n):
    curr_sensors = [int(x) for x in input().split()]
    for j in range(8):
        # Adding sensors data.
        sensors[j].append(curr_sensors[j])
```

```
        # Calculating white, gray and black for
        # each sensor separately.
        white[j] = max(curr_sensors[j], white[j])
        black[j] = min(curr_sensors[j], black[j])
        gray[j] = (white[j] + black[j]) / 2

# Getting rid of unwanted white values that
# preceding and succeeding important values in
# the middle. The important values start from
# the first black and ands with the last one.
max_len = 0
max_len_sens = 0

for i in range(8):
    # Going through dataset and getting
    # rid of unwanted white values.
    i_top = 0
    i_end = len(sensors[i]) - 1
    offset_top = offset_end = 0
    saw_black_top = saw_black_end = False
    while not (saw_black_top and saw_black_end):
        saw_black_top = True if sensors[i][i_top] <= gray[i] or
            i_top == len(sensors[i]) - 1 else saw_black_top
        offset_top += 1 if not saw_black_top else 0
        i_top += 1

        saw_black_end = True if sensors[i][i_end] <= gray[i] or
            i_end == 0 else saw_black_end
        offset_end += 1 if not saw_black_end else 0
        i_end -= 1

    sensors[i] = sensors[i][offset_top:len(sensors[i]) -
        offset_end]

    # Determining longest dataset, that we suppose to
    # use as dataset that came through all the stripes.
    if max_len < len(sensors[i]):
        max_len = len(sensors[i])
        max_len_sens = i

# Finding lcm of max_len and 10 helps us to
# interpolate in case max_len is not divisible by 10.
curr_len = max_len * 10 // gcd(max_len, 10)
# This array will contain the last interpolated
# (if necessary) data.
curr_data = []

# Interpolating if necessary.
if curr_len > max_len:
    for i in range(max_len):
        curr_data.append(sensors[max_len_sens][i])
        if i != max_len - 1:
            aver = (sensors[max_len_sens][i] +
                sensors[max_len_sens][i+1]) / 2
        else:
            aver = sensors[max_len_sens][i]
```

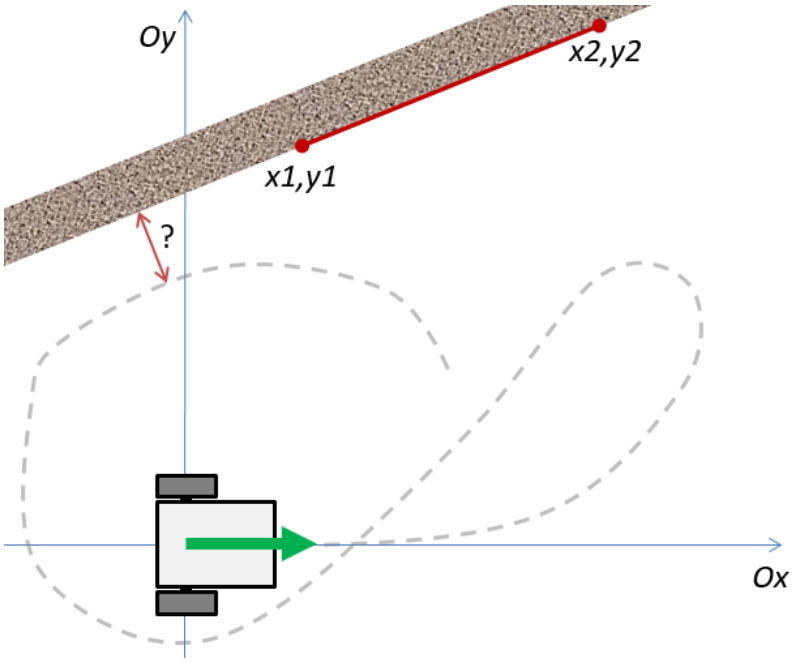
```
            for j in range(curr_len // max_len - 1):
                curr_data.append(aver)
    else:
        curr_data = sensors[max_len_sens]

    ans = ""

    # Making an array of "boolean" data.
    # 1 - black stripe, 0 - white.
    for i in range(0, curr_len, curr_len // 10):
        sum = 0
        for j in range(i, i + curr_len // 10):
            sum += curr_data[j]
        ans += str(1 if sum / (curr_len // 10) <
            gray[max_len_sens] else 0)

    # Printing answer, dropping out 1st and last stripe.
    # Converting binary to decimal.
    print(str(int(ans[1:len(ans)-1], 2)))
```

Задача 2.3.2 (10 баллов)

Условие:

Робототехническое устройство с колеей 170 мм. и радиусом колес 28 мм, собранное по дифференциальной схеме, выезжает из точки (0,0) в направлении положительной полуоси Ox. В ходе перемещения устройство проезжает мимо стены, плоскость которой пересекает координатную плоскость по прямой, проходящей через точки (x1,y1) и (x2,y2).

Найти на каком минимальном расстоянии до стены находился во время перемещения центр робота. Ответ указать в милиметрах. Допустимая погрешность определение расстояния ± 10 мм.



Траектория перемещения робота задается, как последовательность пар чисел, определяющих показания инкрементного энкодера в градусах для левого и правого

колеса через равные промежутки времени. Первое число в паре - показание энкодера левого колеса, второе число в паре - показание энкодера правого колеса.

Считать, что сцепление колес устройства с поверхностью поля - постоянное и без проскальзываний. Центр устройства совпадает с точкой, лежащей на одинаковом расстоянии от центров пятен контактов левого и правого колес. В промежутках между измерениями показаний энкодеров скорости вращения колес постоянны, а угол поворота колеса соответствует углу поворота энкодера. Физическими силами, воздействующими на робота и его элементы, можно пренебречь.

Формат входных данных:

Первая строка входных данных содержит два целых числа x1, y1 $(-2^{31} \leq x1, y1 \leq 2^{31} - 1)$ - координаты первой точки, через которую проходит прямая.

Вторая строка содержит два целых числа x2, y2 $(-2^{31} \leq x2, y2 \leq 2^{31} - 1)$ координаты второй точки.

Третья строка содержит одно целое число n - количество показаний энкодеров $(1 \leq n \leq 2^{31} - 1)$.

В следующих n строках заданы пары чисел l и r - показания энкодера в градусах для левого и правого колеса $(-2^{31} \leq l, r \leq 2^{31} - 1)$.

Формат выходных данных:

Выведите единственное число - минимальное расстояние до стены, где во время перемещения находился центр робототехнического устройства.

Пример:

stdin:

500 -1900
-1510 -1490
41
0 0
7 7
83 90
173 150
269 208
364 266
459 325
552 383
647 441
742 500
838 559
933 617
1025 675
1122 734
1215 806
1308 867
1401 928
1490 990
1582 1051
1675 1113
1768 1174
1854 1239
1946 1302

2039 1363
2131 1425
2222 1484
2314 1546
2408 1610
2505 1675
2595 1739
2686 1804
2777 1863
2869 1924
2961 1988
3052 2050
3142 2111
3232 2176
3325 2237
3418 2298
3507 2359
3603 2433
stdout:
918

Решение:

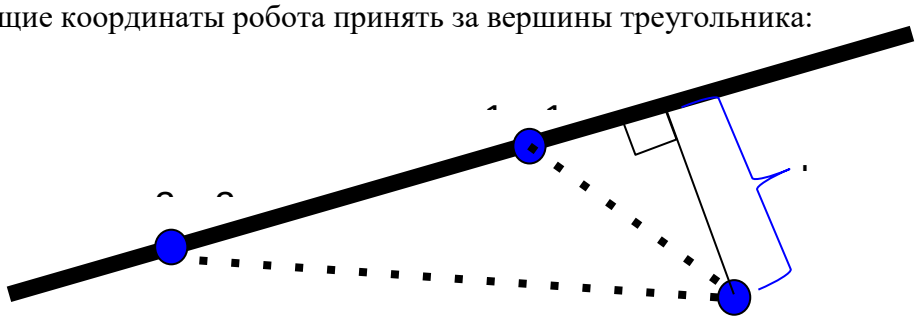
Процесс одометрии позволяет восстанавливать координаты робота относительно места его старта. Так как он стартует в точке 0,0, то эти координаты будут соответствовать абсолютным.

Зная координаты робота в любой момент времени и используя формулу расстояния между прямой и точкой на плоскости можно решить эту задачу.

Для определения расстояния между прямой и точкой на плоскости используем общую формулу площади треугольника в декартовых координатах на плоскости:

$$S = \frac{|(xB - xA)(yC - yA) - (xC - xA)(yB - yA)|}{2}$$

Если точки (x1,y1); (x2,y2) – задающие прямую, обозначающую стену , и точку (x,y) – текущие координаты робота принять за вершины треугольника:



Высота треугольника h будет обозначать кратчайшее расстояние от точки (x,y) до стены, так как падает на нее под углом 90 градусов. Так как площадь треугольника также вычисляется по формуле:

$$S = \frac{b \cdot h}{2},$$

где h – высота треугольника, а b – сторона, на которую падает высота.

Выражая искомую нами h с помощью двух уравнений и формулы расстояния между двумя точками в декартовой плоскости, получаем:

$$h = \frac{2 \cdot S}{b} = \frac{|(x2 - x1)(y - y1) - (x - x1)(y2 - y1)|}{\sqrt{(x2 - x2)^2 - (y2 - y1)^2}}$$

Для вычисления текущих координат мобильного наземного устройства, собранного по дифференциальной схеме, с воспользуемся формулами:

$$\theta_n = \theta_{n-1} + \frac{(S_n^r - S_{n-1}^r) - (S_n^l - S_{n-1}^l)}{w}$$

$$x_n = x_{n-1} + \frac{(S_n^r - S_{n-1}^r) + (S_n^l - S_{n-1}^l)}{2} \cos(\theta_{n-1} + \frac{(S_n^r - S_{n-1}^r) - (S_n^l - S_{n-1}^l)}{2 \cdot w})$$

$$y_n = y_{n-1} + \frac{(S_n^r - S_{n-1}^r) + (S_n^l - S_{n-1}^l)}{2} \sin(\theta_{n-1} + \frac{(S_n^r - S_{n-1}^r) - (S_n^l - S_{n-1}^l)}{2 \cdot w}),$$

где S^r и S^l - расстояния, пройденные правым и левым колесом соответственно, w - колея, а θ - угол между осью Ox и текущим направлением робота.

Пример программы, реализующей данный алгоритм на языке Java:

```
import java.io.IOException;
import java.util.Scanner;

public class Main {
    private static long w = 170; // mm - distance between wheels
    private static long r = 28; //mm  - diameter of wheels

    public static void main(String[] args) throws IOException {
        Scanner sc =new Scanner(System.in);
        long x0 = sc.nextInt();
        long y0 = sc.nextInt();
        long x1 = sc.nextInt();
        long y1 = sc.nextInt();
        int n = sc.nextInt();
        long[] encL = new long[n];
        long[] encR = new long[n];
        for (int i = 0; i < n; i++) {
            encL[i] = sc.nextInt();
            encR[i] = sc.nextInt();
        }

        // founding parameters of line( wall )
        double a = y1 - y0;
        double b = x0 - x1;
        double c = y0 * x1 - y1 * x0;
        // founding x,y and minimum distance to wall
        double x, y;
        double dist, min;

        x = 0;
        y = 0;
        min = Math.abs(c) / Math.sqrt(a * a + b * b);

        double teta = 0;
        double dSl, dSr, dTeta, dS, dX, dY;
        long dEncL, dEncR;
        for (int i = 1; i < n; i++) {
            dEncL = encL[i] - encL[i - 1];
            dEncR = encR[i] - encR[i - 1];
            dSl = (dEncL * 2 * 3.14 * r) / 360;
            dSr = (dEncR * 2 * 3.14 * r) / 360;
            dTeta = (dSr - dSl) / w;
            dS = (dSl + dSr) / 2;

            dX = dS * Math.cos(teta + dTeta / 2);
```

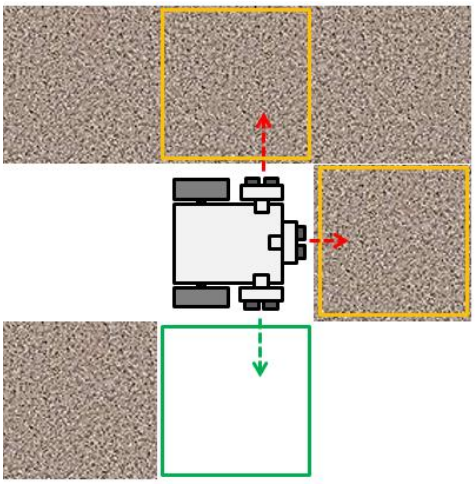
```
            x = x + dX;
            dY = dS * Math.sin(teta + dTeta / 2);
            y = y + dY;
            teta += dTeta;
            dist = Math.abs(a * x + b * y + c) /
                   Math.sqrt(a * a + b * b);
            if (dist < min) {
                min = dist;
            }
        }

        // writing minimum distance
        System.out.println(Math.round(min));
    }
}
```

Задача 2.3.3 (15 баллов)

Условие:

После проведения ремонтных работ автоматизированный погрузчик был активирован в одном из помещений в логистическом центре. Во время ремонтных работ оперативная память погрузчика была сброшена, поэтому устройство управления данного погрузчика не имеет информации о том, в каком помещении погрузчик активирован. Тем не менее, устройство управления имеет доступ к карте всего логистического центра, которое сохранилось в постоянной памяти погрузчика.

Для навигации в помещении на устройстве установлено три датчика обнаружения препятствий: один из датчиков определяет наличие препятствия прямо перед погрузчиком, другие с левой и правой стороны, что позволяет определять препятствие слева и справа соответственно.



Если датчик видит препятствие, он возвращает 1, иначе возвращает 0.

Помещения представляют из себя квадратные секции. За одну фазу перемещения погрузчик переходит прямо из одного помещения в другое либо выполняет поворот на 90 градусов внутри текущего помещения.

Погрузчик начинает двигаться из одного помещения в другое. Необходимо определить из какого помещения началось перемещение устройства после активации.

Формат входных данных:

Первая строка входных данных содержит три целых числа n, k, m - количество помещений $(2 \leq n \leq 1000)$, количество переходов, соединяющих помещения

$(1 \le k \le n \cdot \frac{(n-1)}{2})$ и количество перемещений, выполненных погрузчиком $(2 \le m \le 10000)$.

Далее идет k строк, содержащих три целых числа u, v, w - номера помещений, которые соединены двусторонним переходом $(1 \le u \le v \le n)$ и направление, по которым соединены эти два помещения: 0 - проход между первым и вторым помещениями расположен по направлению на север, 1 - на восток, 2 - на юг, 3 - на запад $(w \in [0, 1, 2, 3])$. Никакие два помещения не соединены двумя переходами, и переход не соединяет помещение с самим с собой.

Следующие m строк описывают перемещения, выполненные роботом. Каждая строка содержит три числа l, c, r - показания датчиков слева, по центру и справа погрузчика $(0 \le l, c, r \le 3)$. Если все три числа равны 2, то данная строчка описывает поворот налево, если все три числа равны 3, то данная строчка описывает поворот погрузчика направо. Сразу после строк с цифрами 2 или 3 (кроме последней строки) идет строка с показаниями датчиков сразу после поворота. Всего строк предоставляется столько, что ими описано перемещения устройства по всем помещениям логистического центра.

Формат выходных данных:

Выведите единственное число - номер помещения, из которого началось перемещение робота после активации. −1, если определить номер помещения невозможно (симметричная структура логистического центра).

Пример:

stdin:

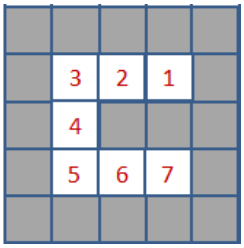
```
7 6 13
1 2 3
2 3 3
3 4 2
4 5 2
5 6 1
6 7 1
0 1 1
2 2 2
1 0 1
1 0 1
1 1 0
3 3 3
1 0 0
1 0 1
1 1 0
3 3 3
1 0 0
1 0 1
1 1 1
```

stdout:

```
7
```

Пояснение:

В примере приводится описание логистического центра, имеющего следующую структуру:



Номерами указаны помещения, по которым перемещался погрузчик.

Решение:

Для решения данной задачи можно использовать метод схожий с методом “Фильтр частиц”: мы будем итеративно оценивать все возможные положения робота с учетом текущих измерений и предыдущих перемещений:

- Шаг 1: Определить все возможные положения (размещение и направление) робота на карте с учетом первого набора показаний датчиков;
- Шаг 2: Выполнить перемещение (поворот или движение вперед) всех положений, определенных на предыдущем шаге;
- Шаг 3: Отфильтровать все положения, которые стали неактуальны из-за отсутствия возможности перемещения в соседнее помещение на предыдущем шаге.
- Шаг 4: Отфильтровать все возможные положения (расположение и направление) робота на карте с учетом текущего набора показаний датчиков;
- Повторять шаги с 2го по 4ый пока не останется только одно возможное положение. Если перемещения закончились, а возможных положений - больше чем одно, то сообщить о невозможности локализации.

Рассмотрим алгоритм действий наглядно, используя данный пример (для описания структур данных будет использоваться нотация языка Python).

Считаем v_num, e_num, path_len – кол-во комнат, переходов и длину пути робота соответственно.

Последовательно заполним массив graph[v_num + 1][4], в котором на месте i,j запишем информацию о комнате, находящейся в направлении j от комнаты i, если между ними есть переход:

Массив graph		**Номер комнаты**						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Направление перехода	**0**	-	-	-	3	4	-	-
	1	-	1	2	-	6	7	-
	2	-	-	4	5	-	-	-
	3	2	3	-	-	-	5	6

Из этого массива составим sensors_data[8][] – массив возможных показаний сенсоров робота в любом положении в любой комнате:

Показания сенсора	**Индекс в массиве**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
000	**0**								
001	**1**	3, 2	5, 1						
010	**2**	2, 0	2, 2	4, 1	4, 3	6, 0	6, 2		
011	**3**	1, 0	3, 3	5, 2	7, 0				
100	**4**	3, 1	5, 0						
101	**5**	1, 3	2, 1	2, 3	4, 0	4, 2	6, 1	6, 3	7, 3
110	**6**	1, 2	3, 0	5, 3	7, 2				
111	**7**	1, 1	7, 1						

В массиве в строке i хранится информация о местоположениях (комната, направление), находясь в которых робот будет видеть датчиками определенную ситуацию, при этом i складывается из записанных слева направо показаний датчиков, переведенных из двоичной системы счисления в десятичную. Например: левый датчик видит стену (1), средний тоже (1), правый не видит (0) – полученные показания (110) при переводе из двоичной системы дают 6 – значит за всеми местоположениями, при которых датчик видит такую картину нужно обращаться в sensors_data[6].

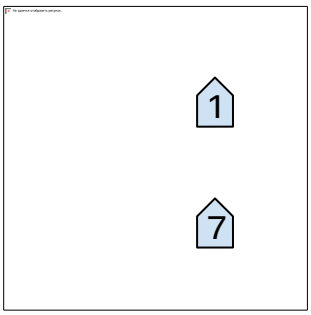
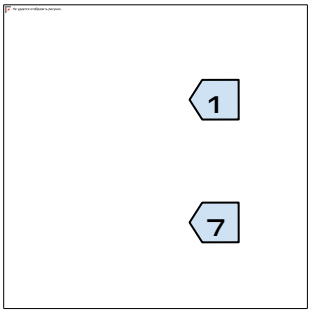
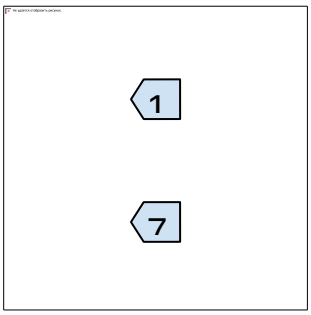
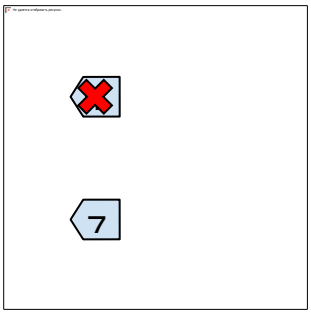
В массиве c_pos будем хранить список возможных позиций робота на текущем шаге и стартовую позицию робота. При первом запуске алгоритма заполняем его всеми возможными позициями из sensors_data[c_sens], где c_sens – первые показания сенсоров робота.

На всех остальных итерациях, пока длина c_pos не будет равна 1 выполняем следующее:

- Учитываем последние считанные данные составляем массив f_pos – массив местоположений, в которых может оказаться робот после осуществления последнего действия. В конце итераций переписываем f_pos в c_pos.

Рассмотрим пример, приведенный в условии:

№ итерации	Считанные данные	Визуализация c_pos	Комментарий

1	011		В массиве всего 2 возможных случая. В центре фигур, обозначающих возможные положения, сохраняем номер стартовой комнаты
2	222 101		Изменяем наш массив после поворота робота налево
3	101		Робот едет вперед
4	110		Ключевая точка. Показания 110 недостижимы из позиции (3, 3), а значит эту точку мы удаляем.

- Выполнять итерации дальше не имеет смысла, так как в массиве остался один элемент, задающий единственно возможное положение робота на данный момент. Так как мы хранили и стартовое положение робота, то мы легко можем дать правильный ответ – 7.

Пример программы, реализующей данный алгоритм на языке Python:

```
# Directions: 0 - North, 1 - East,
#             2 - South, 3 - West

# Return direction after turning
# on turn_on from dir_from.
def turn(dir_from, turn_on):
    return (dir_from + turn_on + 4) % 4

# Return position after one step forward,
# if possible. Else return 0.
```

```
def step(a_v, a_dir):
    return [graph[a_v][a_dir], a_dir]
        if graph[a_v][a_dir] != 0 else 0

################################
# Program execution starts here #
################################

v_num, e_num, path_len =
    [int(x) for x in input().split()]

# graph[v][direct] = u -> vert. v has neighbour u on
# direct from it.
graph = [[0]*4 for i in range(v_num + 1)]
# List of all possible outcomes of sensors.
sensors_data = [[] for i in range(8)]

# Filling graph[v][direct] contains u - vert. num
# that is on direct from v e.g. graph[1][0] = 4 means
# there is 4th vert. on the North from the 1st vert.
for i in range(e_num):
    v, u, direct = [int(x) for x in input().split()]
    graph[v][direct] = u
    graph[u][turn(direct, 2)] = v

for v in range(1, v_num+1):      # Filling sensors_data.
    for direct in range(0, 4):
        c = "1" if graph[v][direct] == 0 else "0"
        l = "1" if graph[v][turn(direct, -1)] == 0 else "0"
        r = "1" if graph[v][turn(direct, 1)] == 0 else "0"
        sensors_data[int(l+c+r, 2)].append([v, direct])

# Current possible locations and directions: [[loc, dir],
start_loc].
# c_pos after next step of robot.
# start_loc - location, where the robot started following the
path.
c_pos = []
f_pos = []

c_step = 1

# Main algorithm starts here.
while len(c_pos) != 1 or c_step == path_len + 1:
    # Works until has 1 possible position or, if determination
failed,
    # until reaching the end of array with robots's movements.
    c_sens = "".join(input().split(" ")).replace("\r", "").\
        replace("\n", "")    # Current sensors state.

    if len(c_pos) == 0 and (c_sens != "222" and c_sens != "333"):
        c_pos = [[x, x[0]] for x in sensors_data[int(c_sens, 2)]]
    else:
        f_pos = []

        if c_sens == "222" or c_sens == "333":
            # If last action was turning, get next sensors data.
```

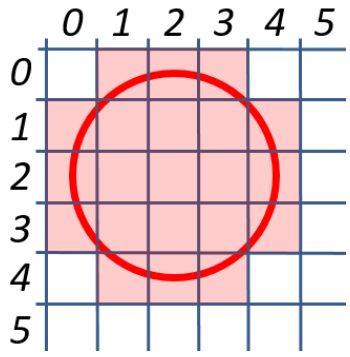
```
            t = -1 if c_sens == "222" else 1
            c_sens = "".join(input().split(" ")).replace("\r",
"").\
                replace("\n", "")

            for j in range(len(c_pos)):
                t_pos = [[c_pos[j][0][0], turn(c_pos[j][0][1],
t)],
                         c_pos[j][1]]
                if t_pos[0] in sensors_data[int(c_sens, 2)]:
                    f_pos.append(t_pos)
        else:
            # Updating f_pos with positions with c_sens
visibility,
            # available from c_pos positions.
            for j in range(len(c_pos)):
                t_pos = [step(c_pos[j][0][0], c_pos[j][0][1]),
                         c_pos[j][1]]

                if t_pos[0] != 0 and t_pos[0] in \
                        sensors_data[int(c_sens, 2)]:
                    f_pos.append(t_pos)

        c_pos = f_pos   # Updating c_pos after 1 step.
    c_step += 1

if c_step <= path_len + 1:
    print(c_pos[0][1])
else:
    print(-1)
```

Задача 2.3.4 (15 баллов)

Условие:

Мобильное робототехническое устройство перемещается по полю разбитому на ячейки. Устройство из каждой клетки может перемещаться либо на одну ячейку вверх, либо на одну ячейку влево, либо на одну ячейку вниз, либо на одну ячейку вправо. Ни одна часть устройства не может выезжать за пределы поля.

На поверхности поля нанесено N окружностей про каждую окружность известна координата ячейки - центра окружности и ее радиус, также измеряемый в ячейках. Робототехническое устройство может проезжать по тем ячейкам поля, которые заняты окружностями.



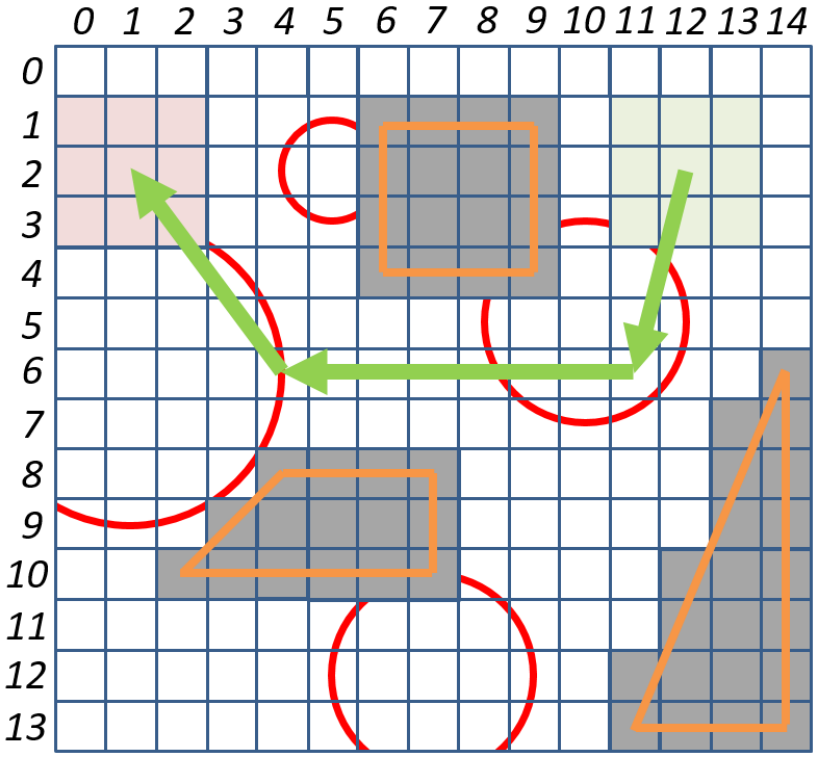
Окружность с центром в координатах (2,2) и радиусом 2.

Также на поле размещено M препятствий, проекции на поле которых можно представить в виде замкнутых ломаных кривых. Никакая часть проекции устройства на поле не может пересекаться с ячейкой занятой проекцией препятствия.

Для робототехнического устройства заданы координаты ячейки старта и ячейки, где устройство должно финишировать. Координаты ячейки старта и ячейки финиша совпадают с центром устройства.

Известно, что устройство - голономное, в любой момент времени перемещения оно занимает квадрат 3 на 3 ячейки, стороны квадрата всегда параллельны осям X и Y.

Необходимо найти кратчайший путь перемещения робототехнического устройства от места старта к месту финиша.



Если перемещение из координаты старта в координату финиша для устройства невозможно, то вывести -1.

Формат входных данных:

Первая строка содержит размер поля $K \times L\,(5 \le K, L \le 32000)$. K - количество ячеек по горизонтали, L - количество ячеек по вертикали. Отсчет начинается с ячейки с координатами (0, 0).

Вторая строка содержит координаты S_1, S_2 места старта робототехнического устройства $(1 \le S_1 \le K-2), (1 \le S_2 \le L-2)$.

Третья строка содержит координаты F_1, F_2 места финиша робототехнического устройства $(1 \le F_1 \le K-2), (1 \le F_2 \le L-2)$.

Четвертая строка содержит два целых числа N и M - количество окружностей и количество препятствий $(3 \le N, M \le 100)$.

Следующие N строк описывают окружности. Каждая строчка содержит четыре целых числа H, C_1, C_2, R - координаты окружности и радиус $(1 \le R \le 100), (-99 \le C_1, C_2 \le 32099), (0 \le H \le N-1)$.

Дальше следует M блоков, состоящих из нескольких строк. Первая строка блока содержит два целых числа O и P - номер препятствия и количество вершин $(0 \le O \le M-1), (3 \le P \le 1000)$. Последующие P строк блока содержат по два целых числа U_1, U_2 - координаты вершин препятствия под номером O

$(0 \leq U_1 \leq K-1), (0 \leq U_2 \leq L-1)$. Вершины одной ломаной перечисляются последовательно.

Формат выходных данных:

Выведите номера всех окружностей, разделенных пробелом, по которым проходить кратчайший путь перемещения из координаты начала в координату финиша. Считается, что путь робототехнического устройства, проходит через окружность, если какая-то часть устройства пересекается с одной из ячеек, принадлежащей окружности. Номера окружностей должны идти в том же порядке, в каком они будут посещены робототехническим устройством.

Пример:

stdin:

15 14
12 2
1 2
4 3
0 5 2 1
1 10 5 2
2 1 6 3
3 7 12 2
0 4
9 1
9 4
6 4
6 1
1 3
11 13
14 13
14 6
2 4
4 8
2 10
7 10
7 8

stdout:

1 2

Решение:

Решение состоит из 3 этапов.

На первом этапе необходимо аппроксимировать препятствия и окружности в целочисленные координаты, например, используя алгоритмы растровой дискретизации Брезенхема.

На втором этапе необходимо преобразовать поле в граф со следующими свойствами: вершины графа – те точки куда может попасть центр робота; у вершин графа указан номер окружности, которую робот будет касаться(пересекать) если окажется в данной точке; ребра графа имеют одинаковый вес.

На третьем этапе необходимо воспользоваться поиском пути по графу (например, поиск в ширину) и найти кратчайший путь, при этом определяя пересекаемые окружности для каждой вершины графа в процессе определения оптимального пути. В

случае если имеется несколько путей, то следует выбирать тот, где окружностей меньше, т.е. возможно переопределение пути до любой точки в процессе поиска пути в графе.

Пример программы, реализующей данный алгоритм на языке Java:

```
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Main {

  private static boolean notObstacle(String s) {
    if (s.length() > 0)
      return !s.contains(".");
    else
      return false;
  }

  public static void main(String[] args)
      throws FileNotFoundException {
    double accuracy = 0.1;

    Scanner sc = new Scanner(System.in);
    int x = sc.nextInt();
    int y = sc.nextInt();
    String[][] field = new String[y][x];

    int[][] graph = new int[y][x];
    int[][][] graphA = new int[y][x][4];
    String[][] way = new String[y][x];
    for (int i = 0; i < y; i++) {// y
      for (int j = 0; j < x; j++) {// x
        way[i][j] = "";
        graph[i][j] = 0;
        graphA[i][j][0] = 0;// f(x)
        graphA[i][j][1] = 0;// g(x)
        graphA[i][j][2] = 0;// h(x)
        graphA[i][j][3] = 0;// nothing(0) / open (1) / close(2)

      }
    }

    for (int i = 0; i < y; i++) {
      for (int j = 0; j < x; j++) {
        field[i][j] = "--";
      }
    }
    int xBegin = sc.nextInt();
    int yBegin = sc.nextInt();
    field[yBegin][xBegin] = "bb";
    graph[yBegin][xBegin] = 2;

    int xEnd = sc.nextInt();
    int yEnd = sc.nextInt();
    field[yEnd][xEnd] = "ee";
    graph[yEnd][xEnd] = 3;

    int numOfCircle = sc.nextInt();
    int numOfObstacle = sc.nextInt();
```

```
for (int i = 0; i < numOfCircle; i++) {
  int ni = sc.nextInt();
  int xi = sc.nextInt();
  int yi = sc.nextInt();
  int ri = sc.nextInt();
  for (int j = 0; j < y; j++) { // y
    for (int k = 0; k < x; k++) { // x
      if (((k - xi) * (k - xi)
          + (j - yi) * (j - yi)) <= (ri * ri))
        field[j][k] = " " + Integer.toString(ni);
    }
  }

}

for (int i = 0; i < numOfObstacle; i++) { // reading obstacles
  int ni = sc.nextInt();
  int hmi = sc.nextInt();
  int xiOld = sc.nextInt();
  int yiOld = sc.nextInt();
  int x0 = xiOld;
  int y0 = yiOld;
  field[yiOld][xiOld] = "." + Integer.toString(ni);
  for (int j = 1; j < hmi; j++) {
    int xi = sc.nextInt();
    int yi = sc.nextInt();
    field[yi][xi] = "." + Integer.toString(ni);

    for (int k = 0; k < y; k++) { // y
      for (int l = 0; l < x; l++) { // x
        double a1 = Math.sqrt(
            Math.pow(l - xi, 2) + Math.pow(k - yi, 2));
        double a2 = Math.sqrt(Math.pow(l - xiOld, 2)
            + Math.pow(k - yiOld, 2));
        double a = Math.sqrt(Math.pow(xi - xiOld, 2)
            + Math.pow(yi - yiOld, 2));
        if (((a1 + a2 - a) < accuracy)
            && (l >= Math.min(xi, xiOld))
            && (l <= Math.max(xi, xiOld))
            && (k >= Math.min(yi, yiOld))
            && (k <= Math.max(yi, yiOld)))
          field[k][l] = "." + Integer.toString(ni);
      }
    }
    xiOld = xi;
    yiOld = yi;
  }

  for (int k = 0; k < y; k++) { // y
    for (int l = 0; l < x; l++) { // x
      double a1 = Math.sqrt(
          Math.pow(l - x0, 2) + Math.pow(k - y0, 2));
      double a2 = Math.sqrt(Math.pow(l - xiOld, 2)
          + Math.pow(k - yiOld, 2));
      double a = Math.sqrt(Math.pow(x0 - xiOld, 2)
          + Math.pow(y0 - yiOld, 2));
      if (((a1 + a2 - a) < accuracy)
```

```
          && (l >= Math.min(x0, xiOld))
          && (l <= Math.max(x0, xiOld))
          && (k >= Math.min(y0, yiOld))
          && (k <= Math.max(y0, yiOld)))
        field[k][l] = "." + Integer.toString(ni);
    }
  }
}

for (int i = 1; i < (y - 1); i++) {// y
  for (int j = 1; j < (x - 1); j++) {// x
    if ((graph[i][j] == 0)
        && notObstacle(field[i - 1][j - 1])
        && notObstacle(field[i - 1][j])
        && notObstacle(field[i - 1][j + 1])
        && notObstacle(field[i][j - 1])
        && notObstacle(field[i][j])
        && notObstacle(field[i][j + 1])
        && notObstacle(field[i + 1][j - 1])
        && notObstacle(field[i + 1][j])
        && notObstacle(field[i + 1][j + 1]))
      graph[i][j] = 1;
  }
}

for (int i = 0; i < y; i++) {// y
  for (int j = 0; j < x; j++) {// x
    if (graph[i][j] == 0)
      graphA[i][j][3] = 2;
  }
}

int xi = xBegin;
int yi = yBegin;
int minFx = 300000000;
int xiMin = xBegin;
int yiMin = yBegin;
graphA[yi][xi][1] = 0;
graphA[yi][xi][2] = 10
    * (Math.abs(xEnd - xi) + Math.abs(yEnd - yi));
graphA[yi][xi][0] = graphA[yi][xi][2];
graphA[yi][xi][3] = 2;

String circles = "";
String s;
int count = 0;

int xiOld = -1;
int yiOld = -1;
boolean found = true;
while (((xi != xEnd) || (yi != yEnd)) && found) {
  graphA[yi][xi][3] = 2;

  if ((xiOld == xi) && (yiOld == yi)) {
    found = false;
    for (int i = 1; i < y - 1; i++) {
      for (int j = 1; j < x - 1; j++) {
```

```
      if (!found && graphA[i][j][3] == 1) {
        xi = j;
        yi = i;
        found = true;
        circles = way[i][j];
        graphA[yi][xi][3] = 2;
        break;
      }
    }
  }
}
minFx = 300000000;

for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    if (graphA[yi + i - 1][xi + j - 1][3] != 2) {
      if (graphA[yi + i - 1][xi + j - 1][3] == 0) {
        way[yi + i - 1][xi + j - 1] = circles;
        graphA[yi + i - 1][xi + j - 1][3] = 1;

        if ((j == 1) || (i == 1))
          graphA[yi + i - 1][xi + j
              - 1][1] = graphA[yi][xi][1] + 10;
        else
          graphA[yi + i - 1][xi + j
              - 1][1] = graphA[yi][xi][1] + 14;
        graphA[yi + i - 1][xi + j - 1][2] = Math
            .toIntExact(Math.round(10 * Math.sqrt(Math
                .pow(xEnd - (xi + j - 1), 2)
                + Math.pow(yEnd - (yi + i - 1), 2))));
        graphA[yi + i - 1][xi + j
            - 1][0] = graphA[yi + i - 1][xi + j
                - 1][1]
            + graphA[yi + i - 1][xi + j - 1][2];
      } else { // graphA[yi + i - 1][xi + j - 1][3] == 1
        if ((j == 1) || (i == 1)) {
          if ((graphA[yi][xi][1]
              + 10) < graphA[yi + i - 1][xi + j
                  - 1][1]) {
            graphA[yi + i - 1][xi + j
                - 1][1] = graphA[yi][xi][1] + 10;
            graphA[yi + i - 1][xi + j
                - 1][0] = graphA[yi + i - 1][xi + j
                    - 1][1]
                    + graphA[yi + i - 1][xi + j
                        - 1][2];

            way[yi + i - 1][xi + j - 1] = circles;
          }
        } else {
          if ((graphA[yi][xi][1]
              + 14) < graphA[yi + i - 1][xi + j
                  - 1][1]) {
            graphA[yi + i - 1][xi + j
                - 1][1] = graphA[yi][xi][1] + 14;
            graphA[yi + i - 1][xi + j
                - 1][0] = graphA[yi + i - 1][xi + j
```

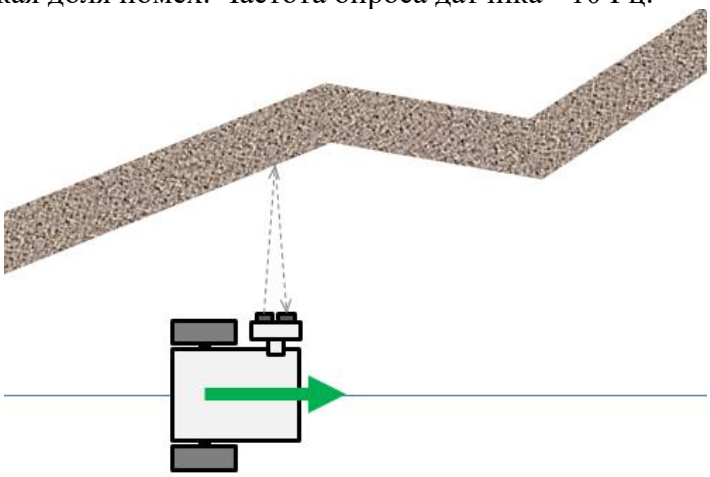
```
                        - 1][1]
                        + graphA[yi + i - 1][xi + j
                            - 1][2];

                  way[yi + i - 1][xi + j - 1] = circles;
                }
              }
            }
          }
        }

        for (int i = 1; i < y - 1; i++) {
          for (int j = 1; j < x - 1; j++) {
            if ((graphA[i][j][3] == 1)
                && (minFx > graphA[i][j][0])) {
              circles = way[i][j];
              minFx = graphA[i][j][0];
              xiMin = j;
              yiMin = i;
            }
          }
        }

        xiOld = xi;
        yiOld = yi;

        xi = xiMin;
        yi = yiMin;

        for (int i = 0; i < 3; i++) {
          for (int j = 0; j < 3; j++) {
            s = field[yi + i - 1][xi + j - 1];
            if (s.contains(" ") && !circles.contains(s))
              circles += s;
          }
        }

      }

      if (!((xi == xEnd) && (yi == yEnd)))
        circles = "";

      if (!circles.isEmpty())
        System.out.println(circles);
      else
        System.out.println("-1");

    }
}
```

Задача 2.3.5 (15 баллов)

Условие:

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

Слева от робототехнического устройства располагается стена, состоящая из нескольких сегментов S $(1 \leq S \leq 5)$. Каждый сегмент имеет свою протяженность и расположен под углом относительно предыдущего сегмента. Минимальная длина сегмента - 380 мм, максимальная длина сегмента 760 мм, угол между плоскостью сегмента и плоскостью, проходящей перпендикулярно через ось траекторию движения робота, не больше 45 градусов. Между сегментами нет зазоров. Во время движения датчик расстояния возвращает расстояние до сегмента стены, мимо которого проезжает робот. Известно, что показания ультразвукового датчика не идеальны, в них всегда присутствует высокая доля помех. Частота опроса датчика - 10 Гц.



Нужно найти количество сегментов в стене S.

Формат входных данных:

Первая строка входных данных содержит два целых числа d, n - диаметр колес d $(20 \leq d \leq 100)$ в миллиметрах и количество показаний ультразвукового датчика расстояния $(10 \leq n \leq 10^3)$. В следующих n строках перечисляются пары чисел E, L - среднее арифметическое показаний энкодеров левого и правого колеса в градусах $(0 \leq E \leq 2^{31} - 1)$ и показание датчика расстояния в миллиметрах $(30 \leq L \leq 2550)$. Угол поворота колеса соответствует углу поворота энкодера.

Формат выходных данных:

Выведите единственное число - количество сегментов в стене.

Пример:

stdin:

28 325
7 198
17 197
29 190
40 197
52 205
63 198
76 197
88 197
100 190

112 190
123 190
136 199
147 195
159 190
171 190
183 195
. . .
3626 184
3638 211
3650 191
3661 189
3673 184
3685 191
3696 195
3708 190
3719 179
3729 183
3740 184
3750 184
stdout:
3

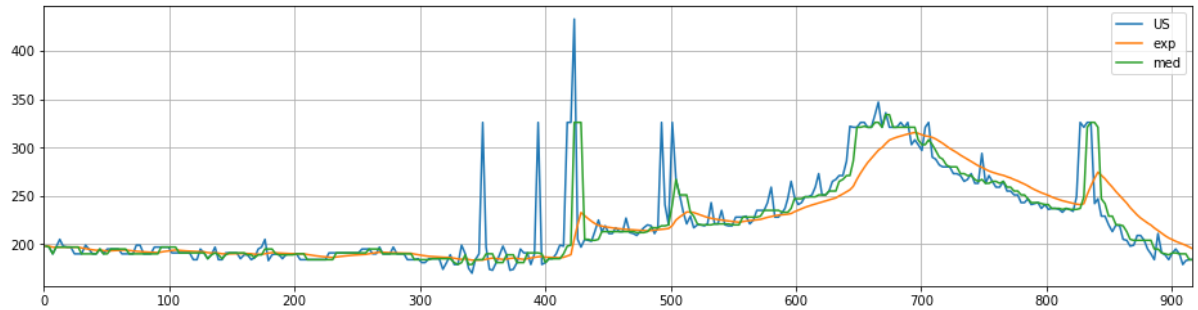
Решение:

Задача относится к классу задач на анализ данных.

Чтобы восстановить количество сегментов, из которых составлена стена, определить в последовательности измерений границы участков, где происходит изменение прироста в значениях датчика. При этом возможны следующие варианты:

- соседние показания датчика были примерно одинаковые, потом начали линейно возрастать;
- соседние показания датчика линейно возрастали, потом начали линейно убывать;
- соседние показание датчика линейно убывали, потом стали примерно равны друг другу;
- и т.п.

Поскольку данные зашумлены, мы не можем сравнивать соседние показания датчиков друг с другом. Поэтому необходимо использовать фильтрацию для уменьшения шума. Параметры фильтрации нужно выбирать так, чтобы, с одной стороны, чрезмерные всплески исчезли, но, с другой стороны, переходы между сегментами однозначно идентифицировались.

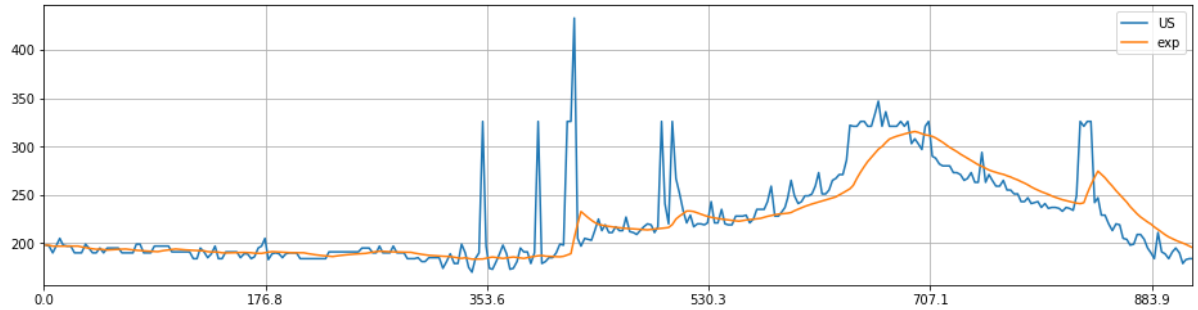


Для получения графиков, представленных выше использовались следующие фильтры:

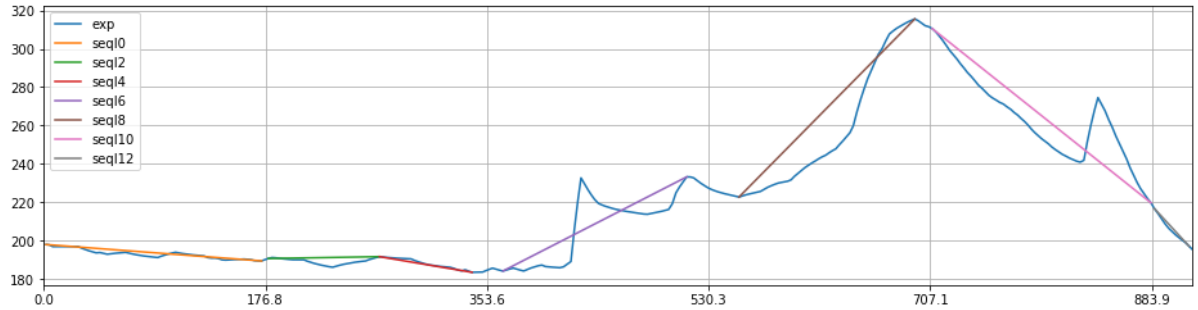
- зеленая кривая - медианный фильтр с окном в 5 измерений;

- оранжевая кривая - последовательное применение медианного фильтра с окном в 5 измерений и экспоненциальное скользящее среднее с коэффициентом стабилизации 0.88.

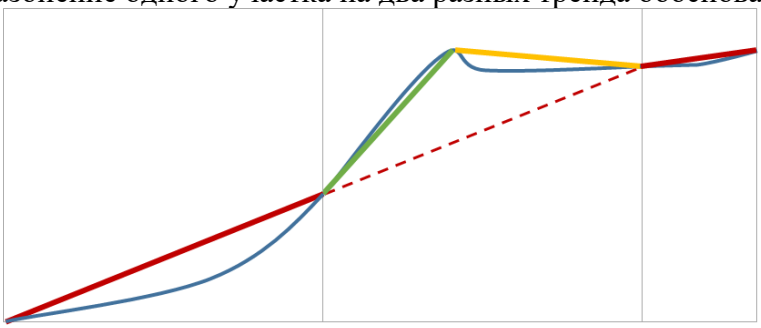
После фильтрации полученные данные все равно не готовы к сравнению между собой - на небольшом участке данных они все еще могут изменять направление, хотя принадлежат одному сегменту. Поэтому нужно сделать укрупнение исследуемых участков: сравнивать между собой не соседние показания, а тренды поведения графика на соседних участках. Для этого нужно сделать разбиение всех показаний на группы. Размер группы разумно сделать соизмеримым с величиной сегмента. Но поскольку длина сегмента строго не зафиксирована, согласно условию задачи, то нужно опираться на его минимальное возможное значение. В ходе решения задачи выяснилось, что удобно оперировать сегментами длиной в 250 мм. Тогда проекция сегмента на ось X, определяющая количество измерений принадлежащих этому сегменту, будет равна $l_{segment} \cdot \cos\frac{\pi}{4}$.



Теперь, если на каждом участке взять крайнее левое и крайнее правое значения, то можно определить тренд прироста показаний. При этом, такое построение тренда в случаях значительного искривления графика будет приводить к неправильному определению частного направления изменения показаний. Следовательно, нужно использовать комплексный подход, основанный на получении экстремумов - максимума и минимума показаний на каждом участке.



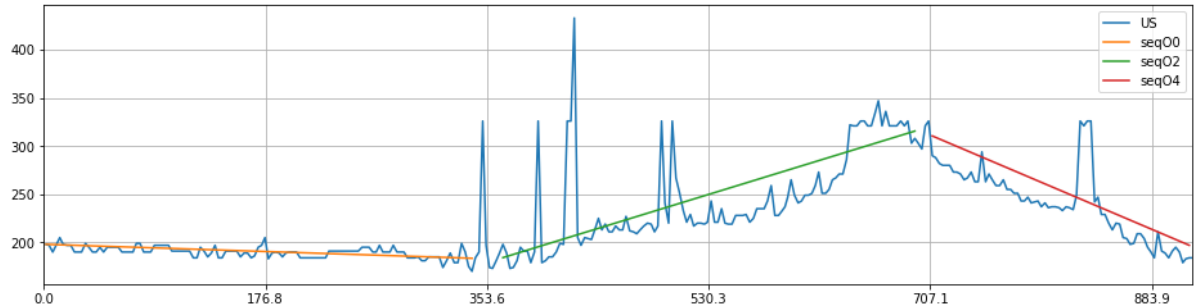
Вариант, когда разбиение одного участка на два разных тренда обосновано:



Появление линий трендов позволяет сравнивать участки между собой. Для этого для каждой линии определяется ее угловой коэффициент:

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

- Если коэффициенты соседних участков равны по знаку, то можно эти участки объединить, предполагая, что они принадлежат одному и тому же сегменту, а разница в значении коэффициентов вызвана сглаживанием.
- Если коэффициенты соседних участков отличаются по знаку, то они принадлежать двум разным сегментам. Исключение можно делать для участков, угол между линий трендов которых меньше 10 градусов, поскольку такое отклонение может быть также вызвано сглаживанием.



Количество участков, полученных после объединения, определяет количество сегментов в стене, рядом с которой ехал робот.

Пример программы, реализующей данный алгоритм на языке Python:

```
import math
import numpy as np

MEDIAN_WINDOW_SIZE=5
K_ema = 0.88
SLOPE_MIN_THRESHOLD=10
SLOPE_MAX_THRESHOLD=45-SLOPE_MIN_THRESHOLD

#Calculate how many degrees corresponds the slope of a segment
def calc_deg_of_slope(pos1, pos2):
    return math.atan2(pos2[1]-pos1[1],
                      pos2[0]-pos1[0])*180/math.pi

#minimal length of the segment
seg_length=250.0
seg_proj=seg_length*math.cos(math.pi/4)

(diameter, num_of_measurements) = input().split()

#Read measurements and filter them
window = []
values = []
for i in range(0, int(num_of_measurements)):
    (enc, value) = input().split()
    #Use millimeters instead of encoder ticks
    x = float(enc)*math.pi*int(diameter)/360
    us = float(value)
    window.append(us)
    if i >= MEDIAN_WINDOW_SIZE-1:
        #median filter implementation
        win_sorted = window[:]
        win_sorted.sort()
        us = win_sorted[2]
```

```
        del window[0]
    if i != 0:
        #exponential moving average filter implementation
        us = values[i-1][1] * K_ema + us * (1 - K_ema)
    values.append([x, us])
    i += 1

#Identification of all meaningful segments
pos_list = []
start_pos = 0
pos_b = values[0][:]
pos_e = []
pos_min = pos_b[:]
pos_max = pos_b[:]
for row in values:
    #initial segmentation is based on minimal segment size:
    #looking for min and max measurements to make
    #an assumption about the slope of the segment
    if (row[0] < start_pos + seg_proj) and (row != values[-1]):
        pos_e = row[:]
        if row[1] > pos_max[1]:
            pos_max = row[:]
        if row[1] < pos_min[1]:
            pos_min = row[:]
    else:
        if pos_max[0] < pos_min[0]:
            pos1 = pos_max[:]
            pos2 = pos_min[:]
        else:
            pos1 = pos_min[:]
            pos2 = pos_max[:]
        #if the distance between min and max is small
        #it is not reliable since it could be impacted by noise
        #so the parts belonging to the segment but outside
        #the min-max range could be used if they are not short
        if (pos2[0] - pos1[0] < seg_proj/2):
            seg_left=pos1[0] - pos_b[0]
            seg_right=pos_e[0] - pos2[0]
            if (pos1[0] - pos_b[0]) > (pos_e[0] - pos2[0]):
                if seg_left > seg_proj/2:
                    pos_list.append(pos_b)
                    pos_list.append(pos1)
                pos_list.append(pos1)
                pos_list.append(pos2)
            else:
                pos_list.append(pos1)
                pos_list.append(pos2)
                if seg_right > seg_proj/2:
                    pos_list.append(pos2)
                    pos_list.append(pos_e)
        else:
            pos_list.append(pos1)
            pos_list.append(pos2)

        #ready to consider the next segment
        start_pos += seg_proj
        pos_b = row[:]
```

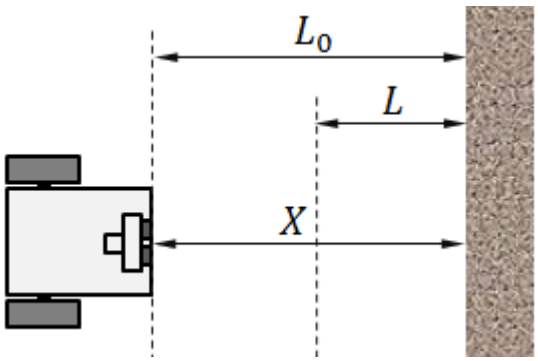
```
        pos_e = []
        pos_min = pos_b[:]
        pos_max = pos_b[:]

#investigate slopes of the segments:
#1. consider two segements as one if slopes of both
#   segments are in the same direction
#   except the cases when one segment is steeply
#   sloping and another is gently sloping
#2. consider two segments as one if slopes are
#   in different direction but both are gentle
new_list = [pos_list[0]]
last_elem = 1
for i in range(1,len(pos_list)-1,2):
    k_left  = calc_deg_of_slope(new_list[-1], pos_list[i])
    k_right = calc_deg_of_slope(pos_list[i+1], pos_list[i+2])
    if np.sign(k_left) != np.sign(k_right):
        if (np.fabs(k_left-k_right)>SLOPE_MIN_THRESHOLD):
            new_list.append(pos_list[i])
            new_list.append(pos_list[i+1])
    elif (np.fabs(k_left-k_right)>SLOPE_MAX_THRESHOLD):
        new_list.append(pos_list[i])
        new_list.append(pos_list[i+1])
    last_elem=i+2
new_list.append(pos_list[last_elem])

#calculate number of assumed segments excluding short
#segments
c = 0
for i in range(0,len(new_list),2):
    if new_list[i+1][0] - new_list[i][0] >= seg_proj/2:
        c += 1
print(c)
```

Задача 2.3.6 (15 баллов)

Условие:

Перед стеной по направлению к стене установлено робототехническое устройство массой 1 кг. Устройство спроектировано так, что оно приводится в движение только одним мотором, который вращает два колеса диаметром D мм., расположенных слева и справа от устройства. Колеса вращаются синхронно. Угол поворота вала мотора совпадает с углом поворота колеса. Таким образом, устройство может либо приближаться, либо удаляться от стены. При движении на устройство действует сила сопротивления среды (сопротивление воздуха), которая пропорциональна текущей скорости перемещения. Коэффициент пропорциональности силы сопротивления среды $K_{friction}$.

На передней части устройства расположен датчик дальномер. Изначально устройство установлено у стены на таком расстоянии, что датчик показывает L_0 мм. После начала движения информация с датчика считывается каждые 50 миллисекунд. Датчик работает идеально точно - в любой момент времени показывает кратчайшее расстояние от передней части робота до стены.



Перед устройством ставится задача - переместиться на расстояние L мм. от стены. Устройство управления роботом запрограммировано таким образом, что оно генерирует значение крутящего момента M на моторе согласно закону

где k - коэффициент пропорциональности при управлении моментом силы мотора, а X - Текущее показание датчика дальномера. Момент силы меняется дискретно, одновременно с измерением расстояния датчиком.

Необходимо найти значение коэффициента k с точностью 0.1, при котором робототехническое устройство остановится на заданное расстояние за минимальное время. Критерием остановки считать условие $|L - X| + |v| < 10$, где X- текущее показание датчика дальномера, а v - Текущая скорость устройства в миллиметрах в секунду.

При решении задачи необходимо учитывать, что устройство разрушится при столкновении со стеной и не сможет продолжать движение.

Подготовьте решение задачи в общем виде и загрузите файл с входными данными, чтобы найти конкретное значение k и соответствующее ему минимальное время остановки t, Для подготовки решения задачи в общем виде можете использовать информацию о крайних значениях параметров и шаг их изменения:

- $0.1 \leq K_{friction} \leq 4$ с шагом 0.1, Коэффициент используется для скорости, выраженной в м/с.;
- $20 \leq D \leq 100$ (в миллиметрах) с шагом 5;
- $100 \leq L_0 \leq 600$ (в миллиметрах) с шагом 50;
- $L = L_0 + a$, где $-400 \leq a \leq 100$ (в миллиметрах) с шагом 50, при этом L не будет меньше 50

Формат входных данных:

Файл содержит строчки:

- Первая строчка содержит значение $K_{friction}$
- Вторая строчка содержит значение D
- Третья строчка - значение L_0
- Четвертая строчка - значение L

Формат выходных данных:

Введите два числа в одну строку:

- первое число содержит значение коэффициента k с точностью до десятых;
- второе число содержит время остановки t с точностью до сотых.

Числа разделены одним пробелом, целая часть от дробной отделена точкой.

Решение:

Для решения задачи нужно построить математическую модель описанной системы.

Из условия задачи известно, что робот движется только в одном направлении, а, следовательно, движение - прямолинейное, так же, известно, что на систему действует коэффициент пропорциональности силы сопротивления среды, поэтому можно сказать, что движение - равноускоренное.

Из общей формулы равноускоренного-прямолинейного движения можно определить положение робота в каждый момент времени, а значит и крутящий момент, который будет задаваться мотором.

$$x = x_0 + v_{0x}t + \frac{a_x t^2}{2}$$

Поскольку и скорость и ускорение будут меняться со временем, то удобнее преобразовать формулу к

$$x_n = x_{n-1} + v_{xn-1}\Delta t + \frac{a_x \Delta t^2}{2}$$

где Δt - фиксировано и равно 0,05, x_0 - равно L_0, v_{x0} равно 0. При этом в каждый момент времени:

$$v_n = v_{xn-1} + a_x \Delta t$$

$$a = \frac{F}{m}$$

$$F = F_{motor} - F_{friction}, F_{motor} = \frac{M}{r}, r = \frac{D}{2}, F_{friction} = K_{friction} \cdot v$$

$$F = \frac{2M}{D} - K_{friction} v$$

В формуле выше момент силы M вычисляется в зависимости от расстояния робота до препятствия x_{n-1} и зависит от неизвестного коэффициента k.

Чтобы найти ответ на задачу нужно перебирать все возможные значения коэффициента k, затем моделировать процесс изменения X в зависимости от изменяющихся значений момента силы. При этом нужно отслеживать, что значение X не стало меньше 0, что означает столкновение с препятствием. Если устройство в какой-то момент времени оказывается близко к требуемой позиции (L) и скорость его перемещения мала, то данный коэффициент и время, необходимое для достижения условия остановки запоминается.

После окончания перебора всех возможных значений k, нужно из запомненных значений выбрать то, которое приводило к остановке быстрее всего.

Пример программы, реализующей данный алгоритм на языке Python:

```
def regulation(l,x,k):
    motor = (l-x)*k
    if motor >= 10 :
        motor = 10
    elif motor <= -10 :
        motor = -10
    return motor

def findK(dataset):
    ll =dataset.split()
    l = int(ll.pop())/ 1000
    l0 = int(ll.pop())/ 1000
    d = int(ll.pop())/ 1000
    kf = float(ll.pop())
    k = 0
    t = 0.05
    kmin = 200
    tmin = -1
```

```
    flag = False
    while k <= 200:
        k += 0.1
        v = 0
        x = l0
        tm = 0
        v0 = 0
        count = 0
        while (abs(l-x) + abs(v))*1000 > 10:
            M = regulation(l, x, k)
            v0 = v
            a = 2 * M / d - kf * v
            v = v0 + a * t
            x = x + v0 * t + a * t * t / 2
            tm += t
            count += 1
            if x < 0 or count > 1000000:
                flag = True
                break
        if not flag and ( tmin < 0 or tmin > tm) :
            tmin = tm
            kmin = k
    return '%.1f %.2f' % (kmin, tmin)

print(findK(sys.stdin.read()))
```

2.4 Критерии определения призеров и победителей

Количество баллов, набранных при решении всех задач суммируется. Призерам второго отборочного этапа было необходимо набрать 5 баллов (для участников из 9 класса) и 10 (для участников из 10-11 класса). Победители первого отборочного этапа должны были набрать 30 баллов.

§3 Финальный этап

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

3.1 Задачи индивидуального тура

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

Решение каждой задачи по математике дает определенное количество баллов (см. критерии оценки). При этом каждая задача делилась на 3 подзадачи таким образом, что решение каждой подзадачи подводило к решению следующей. За каждую подзадачу можно получить от 0 до указанного количества баллов.

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

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

3.1.1 Задачи по математике (9 класс)

Задача 3.1.1.1 (33 балла)

Условие:

На складе детали упакованы в ящики по 17 или 20 штук.

a. **(3 балла)** Может ли робот-погрузчик загрузить 299 деталей, не вскрывая ящики?
b. **(10 баллов)** Может ли робот-погрузчик загрузить 263 детали, не вскрывая ящики?
c. **(20 баллов)** Какое наибольшее количество деталей робот-погрузчик не сможет загрузить, не вскрывая ящики?

Решение:

a. Да, может. 9·20+7·17=299. Сперва заметим, что 6·20-7·17=1 и 15·20=300. В последнем равенстве 6 ящиков по 20 деталей заменим на 7 ящиков по 17 деталей и общее количество деталей уменьшится на 1. (Если пример получен, то необязательно объяснять как именно.)
b. Нет. Пусть робот-погрузчик загрузит x ящиков по 20 деталей и y ящиков по 17. Тогда общее количество загруженных деталей с удовлетворяет равенству

$$20x + 17y = c.(*)$$

Представим с в виде $20 \cdot (6c) + 17 \cdot (-7c)$ и подставим в уравнение

$$20x + 17y = 20 \cdot (6c) + 17 \cdot (-7c);$$
$$20 \cdot (x - 6c) + 17 \cdot (y + 7c) = 0;$$
$$\begin{cases} x - 6c = -17n, \\ y + 7c = 20n, \ n \in ; \end{cases}$$

откуда получим все решения (*) в целых числах

$$\begin{cases} x = 6c - 17n, \\ y = 20n - 7c, \ n \in ; \end{cases}$$

Но по определению переменные x и y принимают только неотрицательные значения. Поэтому $6c - 17n \geq 0$ и $20n - 7c \geq 0$, что равносильно в свою очередь $\frac{7c}{20} \leq n \leq \frac{6c}{17}$. При c=263 в промежутке

$$92 < \frac{7 \cdot 263}{20} \leq n \leq \frac{6 \cdot 263}{17} < 93$$

целых значений n нет.

c. 303 детали получить нельзя, т.к. в промежутке

$$106 < \frac{7 \cdot 303}{20} \leq n \leq \frac{6 \cdot 303}{17} < 107$$

целых значений n нет. При $c \geq 340$ длина указанного промежутка

$$\frac{6c}{17} - \frac{7c}{20} = \frac{c}{340} \geq 1,$$

поэтому промежуток содержит целое число и найдется решение уравнения (*) в неотрицательных числах. 304=20·5+17·12, меняя ящик с 17-ю деталями на ящик с 20-ю деталями 12 раз получаем распределение по ящикам для 307, 310, 313, 316, 319, 322, 325, 328, 331, 334, 337, 340 деталей. 305=20·11+17·5, из которого легко получаем 308, 311, 314, 317, 320 деталей заменой меньшего ящика большим. 323=17·19, из которого получаем 326, 329, 332, 335, 338 деталей. 306=17·18, откуда получаем кратные трем количество деталей до 20·18=360. Следовательно, любое количество деталей, большее 303, можно загрузить ящиками по 17 и 20 деталей.

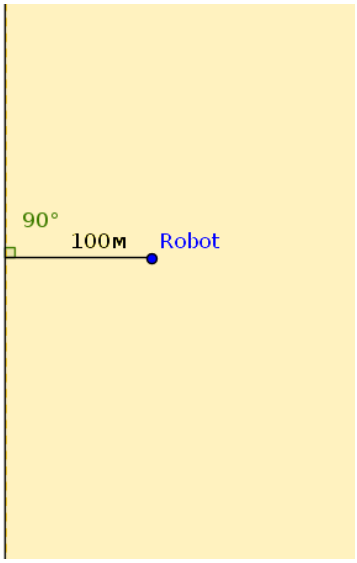
Критерии оценки:

За задачу (b) может быть начислено 10 баллов, если обоснование ответа построено только на базе последней цифры числа 263.

Задача 3.1.1.2 (33 балла)

Условие:

Робот находится в поле пшеницы. В памяти сохранилось, что расстояние до границы 100 метров, но направление потеряно. Считать, что поле представляет собой полуплоскость, т.е. ресурса батареи не хватит выйти из поля в другом направлении, и робот не “увидит” границу поля, пока не достигнет её.

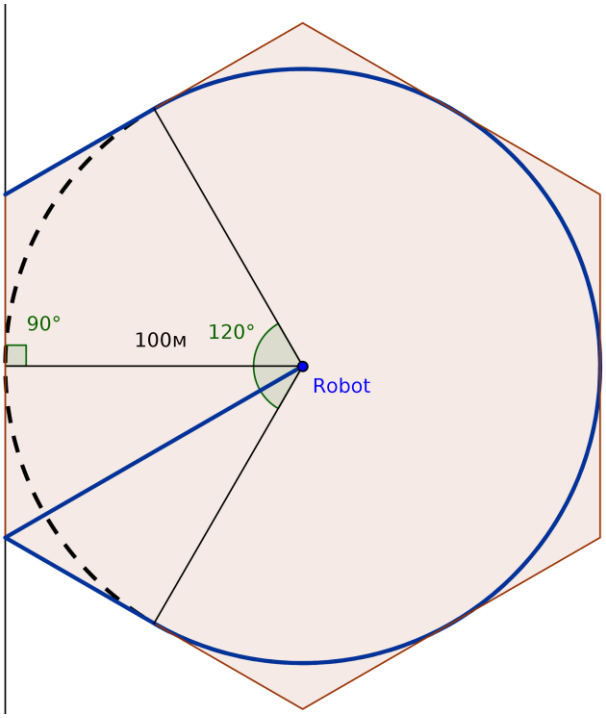


a. **(3 балла)** Существует ли траектория движения робота, которая позволяет выйти из поля, пройдя не более 800 метров;
b. **(13 баллов)** Существует ли траектория движения робота, которая позволяет выйти из поля, пройдя не более 700 метров;
c. **(17 баллов)** Существует ли траектория движения робота, которая позволяет выйти из поля, пройдя не более 650 метров.

Решение:

Граница поля является касательной к окружности с центром в точке нахождения робота и радиусом 100 метров. Поэтому достаточно придумать траекторию, которая пересекает все касательные к этой окружности. Достаточно двигаться по периметру многоугольника, описанного вокруг окружности. Меняя количество сторон, можем оптимизировать траекторию.

Во всех пунктах ответ утвердительный. Причем решение пункта в) подходит и для первых двух. Поэтому приведем путь короче 650 метров, которая обязательно выведет робота к границе поля.



Длина синей траектории равна $\frac{2}{\sqrt{3}} \cdot 100 + \frac{1}{\sqrt{3}} \cdot 100 + \frac{2}{3} \cdot 2\pi \cdot 100 + \frac{1}{\sqrt{3}} \cdot 100,$ что меньше 650.

Критерии оценки:

За задачу (а) может быть начислен 1 балл, если приведен пример траектории без обоснования.

Задача 3.1.1.3 (34 балла)

Условие:

a. **(6 баллов)** Какое преобразование будет результатом последовательного применения трех осевых симметрий относительно биссектрис треугольника ABC?
b. **(12 баллов)** Через центр O окружности проведены три прямые. При помощи циркуля и линейки построить описанный около окружности треугольник, вершины которого лежат на этих прямых.
c. **(16 баллов)** Через центр O окружности проведены 2017 прямых. Как построить описанный около окружности 2017-угольник, вершины которого лежат на этих прямых?

Решение:

a. Пусть O - центр вписанной окружности треугольника ABC, OD - высота, опущенная на AC. Рассмотрим композицию $F=S_{CO} \circ S_{BO} \circ S_{AO}$ трех осевых симметрий относительно биссектрис углов A, B, C в указанном порядке. Это преобразование является движением, меняет ориентацию треугольника ABC, оставляет на месте точки O и D, переводит в себя вписанную окружность и прямую AC. Нетрудно проверить, что F является осевой симметрией относительно прямой OD. Действительно, движение задается тремя точками, и если две из них переходят в себя (O и D), то третья перейдет в себя или в симметричную точку (относительно OD), т.к. расстояния сохраняются. В первом случае сохраняется ориентация (тождественное преобразование), во втором меняется (осевая симметрия).
b. Прямые AO, BO и CO нам известны, необходимо выяснить местоположение самих точек A, B и C. Применяя преобразование F к некоторой точке X получим точку Y, серединный перпендикуляр отрезка XY есть прямая OD в обозначениях предыдущего пункта. Касательная к окружности, перпендикулярная OD, пересекает прямые AO и CO в точках A и C соответственно. Вершина B восстанавливается элементарно. Задача имеет два решения, так как существуют две касательные к окружности, перпендикулярные OD.
c. Композиция нечетного количества осевых симметрий относительно прямых с общей точкой является осевой симметрией относительно прямой, проходящей через ту же точку. Доказательство по индукции.
Обозначим $A_1A_2...A_{2017}$ искомый 2017-угольник. Композиция 2017 осевых симметрий относительно прямых A_1O, A_2O, ..., $A_{2017}O$ переведет прямую A_1A_{2017} в себя, следовательно, ось симметрии соответствующей этой композиции перпендикулярна прямой A_1A_{2017}, её можно восстановить по образу и прообразу некоторой точки. Берем произвольную точку X_0, применяем симметрии
$$S_{A_kO}(X_{k-1} = X_k,\ 1 \le k \le 2017,$$

строим серединный перпендикуляр к X_0X_{2017}, в точке пересечения этой прямой с окружностью строим касательную, она и есть прямая A_1A_{2017}. Остальные стороны 2017-угольника получаем последовательно отражая симметрично предыдущую сторону относительно биссектрисы угла при этой стороне. Задача имеет два решения.

3.1.2 Задачи по математике (10-11 класс)

Задача 3.1.2.1 (16 баллов)

Условие:

Саша и Рустам захотели попить кофе. Саша сделал полный стакан кофе (200 мл), а Рустам налил половину стакана молока (100 мл), и тут выяснилось, что в аппарате закончился кофе. Тогда Саша решил поделиться с другом. Он перелил кофе в стакан Рустама из своего, пока тот не заполнился, при этом тщательно перемешивал. Потом это действие повторил в отношении к своему стакану. Эту процедуру Саша повторил несколько раз. В итоге у них в стаканах оказалось столько же жидкости, что в начале, но концентрация кофе отличалась менее чем на 1%. Какое наименьшее количество переливаний сделал Саша?

Решение:

Пусть в некоторый момент времени доля молока в стакане у Саши была равна α_i (при этом у него в стакане 200 мл), а у Рустама доля молока равна β_i (в стакане 100 мл). Тогда после одной операции переливания в стакане Рустама окажется

$100 \cdot (1-\beta_i) + 100 \cdot (1-\alpha_i)$ мл кофе и $100 \cdot (\alpha_i + \beta_i)$ мл молока.

Доли молока и кофе в первом стакане не изменилась (то есть $\alpha_{i+1} = \alpha_i$), изменился только общий объем. А вот во втором стакане доля молока теперь равна

$$\beta_{i+1} = \frac{100 \cdot (\alpha_i + \beta i)}{200} = \frac{\alpha_i + \beta_i}{2}.$$

После обратного переливания в стакане Саши окажется

$$100 \cdot (1-\alpha_{i+1}) + 100 \cdot (1-\beta_{i+1}) = 100 \cdot \left(2 - \frac{3\alpha_i}{2} - \frac{\beta_i}{2}\right)$$ мл кофе

и

$$100 \cdot \alpha_{i+1} + 100 \cdot \beta_{i+1} = 100 \cdot \left(\frac{3\alpha_i}{2} + \frac{\beta_i}{2}\right)$$ мл молока.

Следовательно, доля молока равна

$$\alpha_{i+2} = \frac{100 \cdot \left(\frac{3\alpha_i}{2} + \frac{\beta_i}{2}\right)}{200} = \frac{3\alpha_i + \beta_i}{4},$$

а в стакане у Рустама не изменилась

$$\beta_{i+2} = \beta_{i+1} = \frac{\alpha_i + \beta_i}{2}.$$

Теперь посмотрим как изменялся модуль разности концентрации молока: изначально он был равен

$$\gamma_i = |\alpha_i - \beta_i|,$$

а после двух переливаний стал равен

$$\gamma_{i+2} = |\alpha_{i+2} - \beta_{i+2}| = \left|\frac{3\alpha_i + \beta_i}{4} - \frac{\alpha_i + \beta_i}{2}\right| = \frac{|\alpha_i - \beta_i|}{4} = \frac{\gamma_i}{4}.$$

В итоге мы получили, что после двух переливании модуль разности концентрации уменьшается ровно в 4 раза. Изначально он равен $\gamma_0 = 1$, а после k пар переливаний впервые стал менее чем $\gamma_{2k} < \frac{1}{100}$. Следовательно, нужно найти такое наименьшее k, что

$\frac{1}{4^k} < \frac{1}{100}$ (если разность концентрации кофе меньше 1, то и разность концентрации молока тоже меньше 1 и наоборот). Получаем, что k = 4. Поэтому всего переливаний было 8.

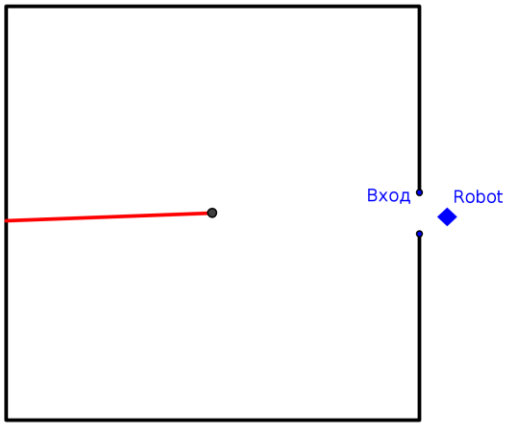
Критерии оценки:

- За полный и правильный расчет всех значений концентрации - 16 баллов;
- Если имелась вычислительная ошибка при расчет концентрации - не более 8 баллов;
- Если в какой-то момент времени происходило округление значений - не 14 баллов.

Задача 3.1.2.2 (34 балла)

Условие:

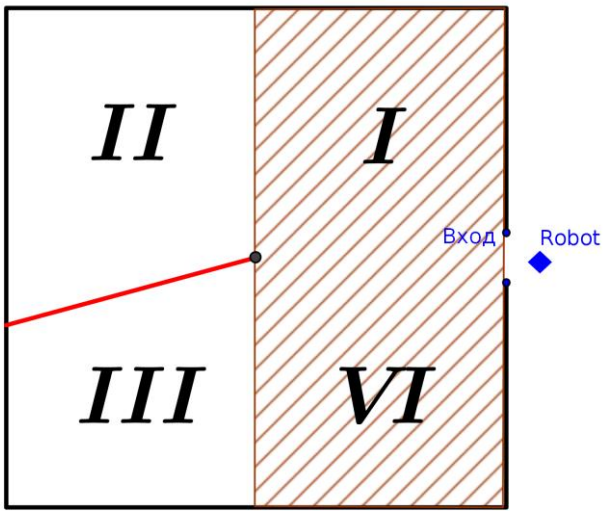
В центре квадратной комнаты со стороной 20 метров находится лазерный радар, который вращается со скоростью один оборот в минуту. Робот должен незаметно пробраться до центра, не попав под луч радара.



a. **(14 баллов)** Докажите, что робот не сможет достичь цели, если его скорость меньше 8 м/мин.
b. **(20 баллов)** Сможет ли робот достичь цели, если его скорость меньше 10 м/мин.

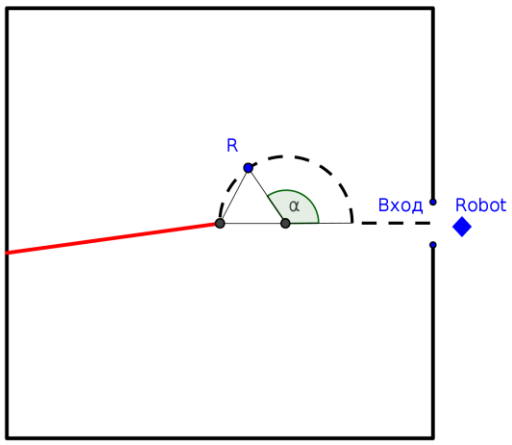
Решение:

a. Не умаляя общности будем считать, что радар вращается против часовой стрелки. Если радар в начальный момент времени (момент начала движения робота) в VI четверти, то он за половину минуты дойдет до конца I четверти и пересечет робота.



А если в I, II или III четверти, то за 1.25 минут он точно покроет заштрихованную область (непрерывно, начиная с начала IV и заканчивая концом I четверти). Так как скорость робота меньше 8 м/мин, то за 1.25 минут он удалится менее чем на 1.25·8 = 10 м от начальной точки, а значит будет в заштрихованной области. Следовательно, робот попадет под луч радара.

b. Пусть робот выехал ровно в тот момент, когда радар начинал движение в I четверти. Также будем считать, что робот двигался по траектории обозначенной пунктиром. Причем, весь путь он двигался с постоянной скоростью v = 9.5 м/мин, t = 0.75 мин двигался прямолинейно, а остаток пути по полуокружности. Докажем, что робот не попадет под радар. На прямолинейном участке пути радар не найдет робота, так как за 0.75 мин он не дойдет до конца IV четверти. Рассмотрим теперь произвольную точку R на полуокружности. Пусть центральный угол равен α, тогда точка R видна под углом $\frac{\alpha}{2}$ с центра поля (отсчет начинается с начала I четверти).



Радар не засечет робота в точке R если

$$t + \frac{(10 - vt)\pi}{2v} \cdot \frac{\alpha}{\pi} < 1 + \frac{\alpha}{4\pi}$$

Подставим значения

$$0.75 + \frac{23}{152} \cdot \alpha < 1 + 0.25 \cdot \frac{\alpha}{\pi} \iff \left(\frac{23\pi}{152} - \frac{1}{4}\right) \cdot \frac{\alpha}{\pi} < 0.25$$

Последнее неравенство очевидно верно для $\alpha \in [0, \pi]$. Следовательно, на полуокружности радар не засечет робота.

Критерии оценки:

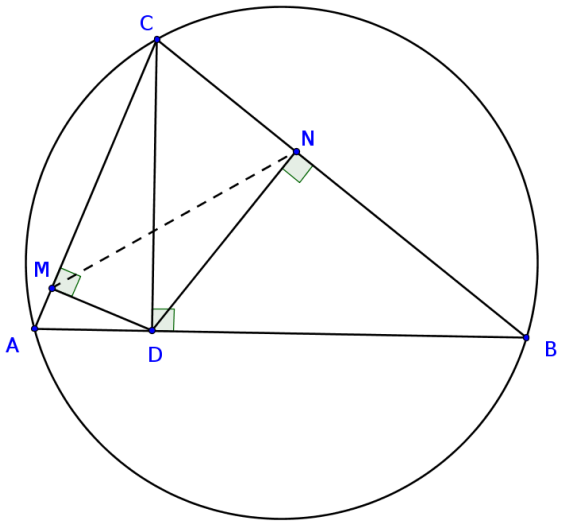
- Если в задаче (a) есть только утверждение о том, в какой момент времени должен поехать робот - 2 балла;
- Если в задаче (a) рассмотрена только окрестность расположения робота через некоторое время - 5 баллов;
- Если в задаче (b) указана только траектория движения робота без доказательства корректности - не более 10 баллов.

Задача 3.1.2.3 (50 баллов)

Условие:

a. **(14 баллов)** Точки A(a) и B(b) лежат на единичной окружности $z \cdot \overline{z} = 1$ с центром в начале координат комплексной плоскости. Докажите, что координата ортогональной проекции точки M(m) на прямую AB задается уравнением

$$z = \frac{1}{2}(a + b + m - ab\overline{m})$$



b. **(16 баллов)** Пусть дан треугольник ABC (см. рисунок), вершины которого соответствуют комплексным числам a, b и c, причём описанная около него окружность имеет уравнение $z \cdot \overline{z} = 1$. Из основания высоты CD треугольника опущены перпендикуляры DM и DN на две стороны. Докажите, что

$$m - n = \frac{(a-b)(a-c)(b-c)}{4ab},$$

где m и n - комплексные координаты точек M и N.

c. **(20 баллов)** Докажите, что длина MN не зависит от выбора высоты треугольника.

Решение:

a. Найдём координату ортогональной проекции точки M(m) на прямую, заданную точками A(a) и B(b). Если Z(z) -- искомая точка, то имеем

$$\begin{cases} z(\overline{a} - \overline{b}) + a(\overline{b} - \overline{z}) + b(\overline{z} - \overline{a}) = 0 \ (1) \\ (a-b)(\overline{m} - \overline{z}) + (\overline{a} - \overline{b})(m - z) = 0 \ (2) \end{cases}$$

где (1) - условие коллинеарности трех точек A, B, Z; (2) - условие ортогональности $AB \perp MZ$.

Откуда выражаем z и получаем

$$z = \frac{a(\overline{m} - \overline{b}) - b(\overline{m} - \overline{a})}{2(\overline{a} - \overline{b})} + \frac{m}{2}$$

В случае, когда точки A(a) и B(b) принадлежат единичной окружности с центром в начале координат $a\overline{a} = b\overline{b} = 1$. Поэтому

$$z = \frac{a(\overline{m} - \overline{b}) - b(\overline{m} - \overline{a})}{2(\overline{a} - \overline{b})} + \frac{m}{2} = \frac{\overline{a}b - a\overline{b}}{2(\overline{a} - \overline{b})} + \overline{m} \cdot \frac{a - b}{2(\overline{a} - \overline{b})} + \frac{m}{2} =$$
$$= \frac{a + b}{2} - \overline{m} \cdot \frac{ab}{2} + \frac{m}{2} = \frac{1}{2}(a + b + m - ab\overline{m})$$

b. Так как $a\overline{a} = b\overline{b} = c\overline{c} = 1$, то воспользуемся три раза предыдущим пунктом для точки D с отрезками AC, BC и для точки C с отрезком AB

$$\begin{cases} m = \frac{1}{2}(a + c + d - ac\overline{d}) \\ n = \frac{1}{2}(b + c + d - bc\overline{d}) \\ d = \frac{1}{2}(a + b + c - ab\overline{c}) \end{cases} \Longrightarrow$$

$$\Longrightarrow m - n = \frac{1}{2}(a + c + d - ac\overline{d}) - \frac{1}{2}(b + c + d - bc\overline{d}) = \frac{1}{2}(a - b)(1 - c\overline{d}) =$$
$$= \frac{(a - b)}{4}(2 - c(\overline{a} + \overline{b} + \overline{c}) + \overline{a}\overline{b}cc) = \frac{(a - b)}{4}\frac{(ab - cab(\overline{a} + \overline{b}) + cc)}{ab} =$$
$$\frac{(a - b)(a - c)(b - c)}{4ab}$$

c. Так как $|a| = |b| = 1$, то

$$|m - n| = \frac{|a - b| \cdot |b - c| \cdot |c - a|}{4|a| \cdot |b|} = \frac{|a - b| \cdot |b - c| \cdot |c - a|}{4}$$

Это выражение симметрично относительно a, b и c, поэтому длина MN не зависит от выбора высоты треугольника.

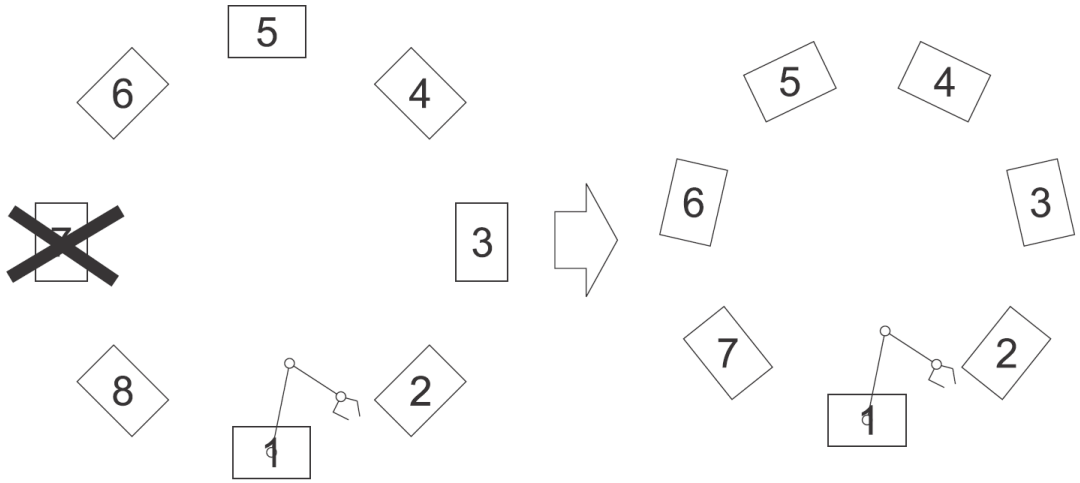
3.1.3 Задачи по информатике

Задача 3. 1.3.1 (30 баллов)

Условие:

В цехе по упаковке конечной продукции на заводе по производству мебели робот-манипулятор подготавливает товар к отгрузке. Контейнеры с элементами, из которых упаковывается та или иная единица мебели, расположены так, что робот вместе с ними образует круг. Каждая позиция, где может быть расположен контейнер, пронумерована против часовой стрелки. Манипулятор установлен в позиции 1, справа от него находится контейнер с позицией 2, слева — контейнер с позицией n.

В каждом i-ом контейнере находится c_i определенного типа элементов. Во время операций по отгрузке робот может доставать из контейнеров по одному элементу. Длины звеньев манипулятора хватает только, чтобы достичь k-го контейнера по правую или по левую сторону от робота. Если во время операций по отгрузке какой-то контейнер становится пустым, он отправляется в производственных цех, а круг контейнеров сужается.



Операция извлечения элемента и перемещения его в зону упаковки манипулятором занимает 1 минуту.

В какой-то момент времени в цех по упаковке пришел запрос из производственного цеха освободить контейнер с определенным типом элементов. Необходимо определить, какое минимальное количество времени необходимо манипулятору, чтобы освободить запрошенный контейнер, находящийся на текущий момент в позиции под номером m.

Формат входных данных:

- Первый набор тестов:
  В первой строке входных файлов содержится три разделенных пробелом целых числа n, k, m $(2 \le n \le 25,\ 1 \le kn,\ 2 \le m \le n)$.
  Во второй строке содержится n разделенных пробелом чисел, где на i-й позиции стоит h_i $(1 \le h_i \le 10\,000)$ — количество элементов в i-ом контейнере.
- Второй набор тестов:
  В первой строке входных файлов содержится три разделенных пробелом целых числа n, k, m $(2 \le n \le 500,\ 1 \le kn,\ 2 \le m \le n)$.
  Во второй строке содержится n разделенных пробелом чисел, где на i-й позиции стоит hi $(1 \le h_i \le 10\,000)$ — количество элементов в i-ом контейнере.
- Третий набор тестов:
  В первой строке входных файлов содержится три разделенных пробелом целых числа n, k, m $(2 \le n \le 10\,000,\ 1 \le kn,\ 2 \le m \le n)$.
  Во второй строке содержится n разделенных пробелом чисел, где на i-й позиции стоит hi $(1 \le h_i \le 10\,000)$ — количество элементов в i-ом контейнере.

Формат выходных данных:

Выведите одно число — минимальное количество времени, необходимое для освобождения контейнера под номером m.

Пример №1:

stdin:
6 2 4
10 3 2 5 4 4
stdout:
7

Пример №2:

stdin:

5 1 5
1 4 8 2 6
stdout:
6

Примечание:

В первом примере манипулятор не может сразу достигнуть до контейнера, поэтому для начала ему нужно потратить 2 минуты, чтобы освободить контейнер под номером 3. После этого манипулятор уже может выгрузить нужный контейнер, потратив 5 минут.

Во втором примере манипулятор сразу может за 6 минут выгрузить контейнер.

Критерий оценки:

За решение задачи начислялось:

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

Для проверки результата используется следующий код на языке Python:

```
def solve(dataset):
    dataset = dataset.splitlines()
    n, k, m = map(int, dataset[0].split())
    m = m - 1
    health = list(map(int, dataset[1].split()))

    heap = []
    r_ans = 0
    last = k
    for i in range(1, last + 1):
        heappush(heap, health[i])
    while last < m:
        r_ans += heappop(heap)
        last = last + 1
        heappush(heap, health[last])

    heap = []
    l_ans = 0
    last = n - k
    for i in range(last, n):
        heappush(heap, health[i])
    while last > m:
        l_ans += heappop(heap)
        last = last - 1
        heappush(heap, health[last])

    return str(min(l_ans, r_ans) + health[m])
```

Решение:

Немножко изменим нашу задачу, пусть контейнеры стоят в ряд, а не создают круг. Манипулятор может достать контейнеры от 1 до k. Какой контейнер выгоднее опустошить, чтобы манипулятор доставал до контейнера k+1? Очевидно, что контейнер с

наименьшим количеством элементов в нем, при этом другие контейнеры мы не трогаем. А если хотим достать до контейнера k+2? Тогда требуется выбрать контейнер с минимальным количеством элементов от 1 до k и еще один минимум от 1 до k+1, но без учета контейнера, который мы уже обработали.

К чему свелась наша задача? Каждый раз нам требуется находить минимум среди k элементов, удалять его, и повторять это действие до тех пор, пока не достанем до нужного контейнера. Структура данных двоичная куча позволяет добавлять элементы за O(log(k)) и убирать минимум за такую же асимптотику. Итоговое время работы: O(n·log(k)).

Так как наши контейнеры образуют круг, требуется еще запустить алгоритм, но уже брать контейнеры с другой стороны.

Пример программы, реализующей данный алгоритм на языке C++:

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

using namespace std;

int main() {
    int n, k, m;
    cin >> n >> k >> m;
    vector <int> health(n + 1);
    for (int i = 1; i <= n; i++) {
        scanf("%d ", &health[i]);
    }

    multiset <int> heap;
    int r_ans = 0;
    int last = k + 1;
    for (int i = 2; i <= last; i++) {
        heap.insert(health[i]);
    }
    while (last < m) {
        r_ans += *(heap.begin());
        heap.erase(heap.begin());
        heap.insert(health[++last]);
    }

    heap.clear();
    int l_ans = 0;
    last = n - k + 1;
    for (int i = n; i >= last; i--) {
        heap.insert(health[i]);
    }
    while (last > m) {
        l_ans += *(heap.begin());
        heap.erase(heap.begin());
        heap.insert(health[--last]);
    }

    cout << min(l_ans, r_ans) + health[m] << endl;
}
```

Задача 3.1.3.2 (30 баллов)

Условие:

Передача информации от Центра Планирования Миссии (ЦПМ) до робота на Марсе происходит пакетами. Пакеты передаются в виде массива, каждый пакет представляет из себя целое положительное число. Пакеты доходят с разной скоростью, и из-за этого существует одна проблема — они могут поменяться местами. Но мы живём в будущем, и не всё уж так плохо: гарантируется, что пакет мог поменяться только с одним из соседних пакетов (после чего эти 2 пакета не могли дальше поменяться местами с новыми соседями).

Главе ЦПМ надо приготовить отчёт, поэтому ему надо узнать, какой может быть максимальное значение суммы, вычисленной по следующей формуле: a_1 xor a_2 + a_3 xor a_4 + ... + a_{n-1} xor a_n, где xor - побитовое исключающее ИЛИ. Именно вам предстоит вычислить это значение.

Формат входных данных:

- Первый набор тестов
  В первой строке дано целое положительное число n $(2 \leqslant n \leqslant 20)$ — количество элементов массива. Гарантируется, что оно чётное.
  Следующая строка содержит n чисел — изначально заданный массив.
  Все числа положительные и не превосходят 10^5.
- Второй набор тестов
  В первой строке дано целое положительное число n $(2 \leqslant n \leqslant 100\,000)$ — количество элементов массива. Гарантируется, что оно чётное.
  Следующая строка содержит n чисел — изначально заданный массив.
  Все числа положительные и не превосходят 10^5.

Формат выходных данных:

Выведите одно число — максимальное значение суммы, вычисленный вышеописанным способом.

Пример №1:

stdin:
4
1 1 2 6
stdout:
10

Пример №2:

stdin:
6
5 7 6 1 8 4
stdout:
23

Примечание:

Массив в первом пример, при котором достигается максимальная сумма: 1 2 1 6.
Во втором примере: 5 6 7 8 1 4.

Критерий оценки:

За решение задачи начислялось:

- **10 баллов**, если пройдена только первая группа тестов;

- **30 баллов**, если пройдены все тесты.

Для проверки результата используется следующий код на языке Python:

```
def solve(dataset):
    dataset = dataset.splitlines()
    n = int(dataset[0])
    a = dataset[1].split()

    for i in range(0, n):
        a[i] = int(a[i])

    a = [a[0]] + a + [a[n - 1]]
    n = n + 2

    dp = [[0 for x in range(2)] for y in range(n)]

    for i in range(3, n, 2):
        dp[i][0] = max(dp[i - 2][0] + (a[i - 2] ^ a[i - 1]),
                       dp[i - 2][1] + (a[i - 3] ^ a[i - 1]))
        dp[i][1] = max(dp[i - 2][0] + (a[i - 2] ^ a[i]),
                       dp[i - 2][1] + (a[i - 3] ^ a[i]))

    return str(max(dp[n - 1][0], dp[n - 1][1]))
```

Решение:

Заведем матрицы $dp_0[n]$, $dp_1[n]$ – максимальный ответ, если разрешено использовать только первые i элементов, при этом $dp_0[i]$ – мы не меняем местами (i-1)-й и i-й элементы, $dp_1[i]$ – меняем. Тогда как получить ответ для j, если мы знаем ответ для префикса?

$$dp_0[j] = max(dp_0[j-2]+(a[j-2]\oplus a[j-1]), dp_1[j-2]+(a[j-3]\oplus a[j-1]))$$
$$dp_1[j] = max(dp_0[j-2] + (a[j-2] \oplus a[j]), dp_1[j-2] + (a[j-3] \oplus a[i]))$$

Пример программы, реализующей данный алгоритм на языке C++:

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

using namespace std;

int main() {
    int n;
    scanf("%ld", &n);

    vector<long long> a(n + 2);

    for (int i = 1; i <= n; i++) {
        scanf("%lld", &a[i]);
    }

    n += 2;
    a[0] = a[1];
    a[n - 1] = a[n - 2];

    vector< vector<long long> > dp(n, vector<long long> (2, 0));

    for (int i = 3; i < n; i += 2) {
        dp[i][0] = max(dp[i - 2][0] + (a[i - 2] ^ a[i - 1]),
                       dp[i - 2][1] + (a[i - 3] ^ a[i - 1]));
```

```
        dp[i][1] = max(dp[i - 2][0] + (a[i - 2] ^ a[i]),
                       dp[i - 2][1] + (a[i - 3] ^ a[i]));
    }

    printf("%lld", max(dp[n - 1][0], dp[n - 1][1]));

    return 0;
}
```

Задача 3.1.3.3 (40 баллов)

Условие:

В одном из государств было решено создать сеть из n городов, между которыми бы курсировали беспилотные автомобили, но остается проблема возникновения внештатных ситуаций. Для решения этого вопроса было решено поставить маяки, которые бы управляли автомобилями, в случаях возникновения опасности. В каждом из n городов хотят поставить по одному маяку так, чтобы если какой-то один маяк выйдет из строя, то оставшиеся города тоже представляли из себя единую сеть, что из всех оставшихся существовал путь между любыми двумя городами.

Так как государство не хочет сильно тратиться, то решили произвести одинаковые маяки с наименьшем радиусом действия, но выполняющие все условия безопасности, которые были описаны ранее.

В нашем случае представим, что города — точки на плоскости, а дороги, как кратчайшее расстояние между точками. Заметим, что из города a в город b можно попасть и через другие города. В итоге мы хотим выбрать минимальный радиус маяка, чтобы получить такую сеть, что из любого города можно добраться до любого другого и, если какой-то маяк выйдет из строя, то оставшиеся города тоже представляли из себя единую сеть.

Формат входных данных:

- Первый набор тестов
  Первая строка содержит одно целое число n $(0 \leqslant n \leqslant 100)$ — количество городов.
  Следующие n строк содержат по два числа — координаты городов соответственно.
  Все координаты точек неотрицательные и не превосходят 10^9.
- Второй набор тестов
  Первая строка содержит одно целое число n $(0 \leqslant n \leqslant 300)$ — количество городов.
  Следующие n строк содержат по два числа — координаты городов соответственно.
  Все координаты точек неотрицательные и не превосходят 10^9.

Формат выходных данных:

Выведите одно число — минимальный радиус действия для всех маяков, удовлетворяющий всем вышеописанным условиям. Число выведите с точностью не менее 10^{-6}.

Пример №1:

stdin:

4
3 0
0 4

3 5
6 4
stdout:
2.500000

Пример №2:

stdin:
6
2 1
3 5
8 2
8 4
12 1
12 5
stdout:
3.041381

Критерий оценки:
За решение задачи начислялось:

- **15 баллов**, если пройдена только первая группа тестов;
- **40 баллов**, если пройдены все тесты.

Для проверки результата используется следующий код на языке Python:

```
def dfs(v, p, edges, used, tin, up):
    global t
    used[v] = 1
    up[v] = t
    tin[v] = t
    t = t + 1
    flag = 0
    was = 0
    child = 0
    for i in range(len(edges[v])):
        to = i
        if to == p or edges[v][i] == 0:
            continue
        if used[to] == 0:
            child = child + 1
            was = max(was, dfs(to, v, edges, used, tin, up))
            up[v] = min(up[v], up[to])
            if up[i] >= tin[v]:
                flag = 1
        else:
            up[v] = min(up[v], tin[to])
    if p == -1:
        flag = 0
    if flag == 1 or (p == -1 and child > 1):
        return 1
    else:
        return was


def solve(dataset):
    global t
    dataset = dataset.splitlines()
```

```
    n = int(dataset[0])
    a = [list(map(int, dataset[i + 1].split())) for i in range(n)]

    t = 0
    l = -1
    r = 10 ** 18
    while r - l > 1:
        edges = [[0 for i in range(n)] for j in range(n)]
        used = [0] * n
        up = [0] * n
        tin = [0] * n
        t = 0
        m = (l + r) // 2
        for i in range(n):
            for j in range(n):
                dist = (a[i][0] - a[j][0]) ** 2 +
                       (a[i][1] - a[j][1]) ** 2
                if dist <= m:
                    edges[i][j] = 1
        res = dfs(0, -1, edges, used, tin, up)
        flag = 0
        for i in range(n):
            if used[i] == 0:
                flag = 1
        if flag == 1 or res == 1:
            l = m
        else:
            r = m
    return str('%.6f' % (math.sqrt(r) / 2))
```

Решение:

Если радиус r удовлетворяет нашим условиям, то любой больший радиус тоже удовлетворяет. Поэтому мы можем воспользоваться двоичным поиском для поиска минимального подходящего радиуса. Для определения того, что при выходе из строя любого маяка, наша сеть все еще останется связной, можно воспользоваться методом поиска точек сочленения. Итоговая асимптотика: $O(N^2)$.

Пример программы, реализующей данный алгоритм на языке C++:

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

using namespace std;

typedef long long ll;

bool intersection(pair <int, int> &p1, pair <int, int> &p2, ll &r)
{
     ll dist = ((ll)p1.first - p2.first) * (p1.first - p2.first) +
     ((ll)p1.second - p2.second) * (p1.second - p2.second);
```

```
      return dist <= r;
}

int t;

bool dfs(int v, int p, vector < vector <int> > &edges,
         vector <bool> &used, vector <int> &in, vector <int> &up)
{
      used[v] = true;
      in[v] = up[v] = ++t;
      int child = 0;
      bool flag = false, was = false;
      for(auto i : edges[v]) {
            if(i == p) continue;
            if(!used[i]) {
                  child++;
                  was = max(was, dfs(i, v, edges, used, in, up));
                  up[v] = min(up[v], up[i]);
                  if(up[i] >= in[v]) flag = true;
            } else {
                  up[v] = min(up[v], in[i]);
            }
      }
      if(p == -1) flag = false;
      if(flag || (p == -1 && child > 1))
            return true;
      else
            return was;
}

int main() {
    int n;
    cin >> n;
    vector < pair <int, int> > v(n);
    for(auto &i : v) {
      cin >> i.first >> i.second;
    }
      ll l = -1, r = 2 * 1e18 + 1;
    while(r - l > 1) {
      ll m = (l + r) / 2;
      vector < vector <int> > edges(n);
      vector <bool> used(n, false);
      vector <int> in(n), up(n);
      t = 0;
      for(size_t i = 0; i < v.size(); i++) {
            for(size_t j = 0; j < i; j++) {
                  if(intersection(v[i], v[j], m)) {
                        edges[i].push_back(j);
                        edges[j].push_back(i);
                  }
            }
      }
      bool res = dfs(0, -1, edges, used, in, up);
      bool flag = false;
      for(auto i : used) {
            if(!i) flag = true;
      }
```

```
        if(flag || res)
              l = m;
        else
              r = m;
      }
      cout << fixed << setprecision(15) << sqrt((double)r) / 2;
        return 0;
}
```

3.2. Задача командного тура

В командной части заключительного этапа участники должны спроектировать автономную систему управления роботом-погрузчиком для решения задач навигации и локализации на модели логистического центра.

Командная часть заключительного этапа имеет продолжительность 3,5 дня (всего 24 астрономических часа), которые включают работу по оснащению роботов необходимыми датчиками, программированию, пробные заезды на макете логистического центра, зачетные попытки.

3.2.1. Легенда

Работники логистического центра с предыдущей смены оставили робот-погрузчик где-то внутри логистического центра. Система управления робота имеет информацию о структуре центра, но у нее нет информации о текущем расположении робота на его территории.

По внутренней системе распределения задач, погрузчик получает задачу самостоятельно прибыть в определенные сектора логистического центра, где он пройдет сервисное обслуживание, после чего он должен самостоятельно проследовать в сектор приписки данного робота.

3.2.2. Набор заданий

Решение командной задачи было разбито на 3 этапа. Первые три этапа итеративно подводили участников к решению полной финальной задачи, осуществляемому во время последнего четвертого этапа. На каждом этапе помимо полноты решения заданий данного этапа проверялась воспроизводимость результатов - робототехническое устройство участников должно было неоднократно выполнить требуемые действия.

Первый этап

Робототехническое устройство должно проехать по модели логистического центра по правилу правой или левой руки из сектора старта, пока не увидит линию черную линию, нанесенную на пол модели. Линия обозначает сектор финиша. Путь перемещения робота будет заранее выбран так, что сектор финиша достижим из сектора старта - в нем не будет циклических структур. Устройство должно заехать в сектор финиша, остановиться и вывести на экран количество, пройденных сегментов.

Включая содержательные задачи:

- Определение оптимального расположения датчиков сенсорной системы, расчет интенсивности принимаемого сенсорами сигнала, определение пороговых значений;

- Реализация навигации
    - Перемещение вдоль стены с использованием показаний датчиков расстояния или освещенности;
    - Выравнивание;
    - Перемещение по азимуту с использованием показаний датчиков гироскопа или акселерометра;
    - Поворот на заданный угол с использованием показаний датчиков гироскопа или акселерометра;
- Реализация счисления пути.

Второй этап

Робототехническое устройство после старта из заранее неизвестного расположения определяет свою текущую позицию после некоторого времени перемещения по лабиринту. При этом проверяется два возможных режима работы:

1. Определение циклических структур полигона: устройство должно двигаться вдоль циклической структуры до тех пор, пока не определит, что двигается по той же траектории. При этом должен быть пройден, как минимум один полный обход вдоль циклической структуры. После определения циклической структуры, устройство должно начать двигаться по секциям, лежащим вне начальной траектории.
2. Локализация: устройство должно двигаться до тех пор, пока не определит свою текущую позицию в лабиринте. После остановки, на экране должны отобразиться текущие координаты устройства.

Включая содержательные задачи:

- Реализация построения карты по данным одометрии;
- Реализация локализации.

Третий этап

Робототехническое устройство после старта из заранее неизвестного расположения (сектор старта находится не далее чем в пяти секторах до сектора сервисного обслуживания) доезжает до штрих-кода, останавливается и выводит на экран десятичное значение закодированного числа.

Включая содержательные задачи:

- Реализация алгоритмов распознавания штрих-кода.

Четвертый этап

Робототехническое устройство после старта из заранее неизвестного расположения проезжает через два сектора сервисного обслуживания и завершает свое перемещение в секторе приписки робота.

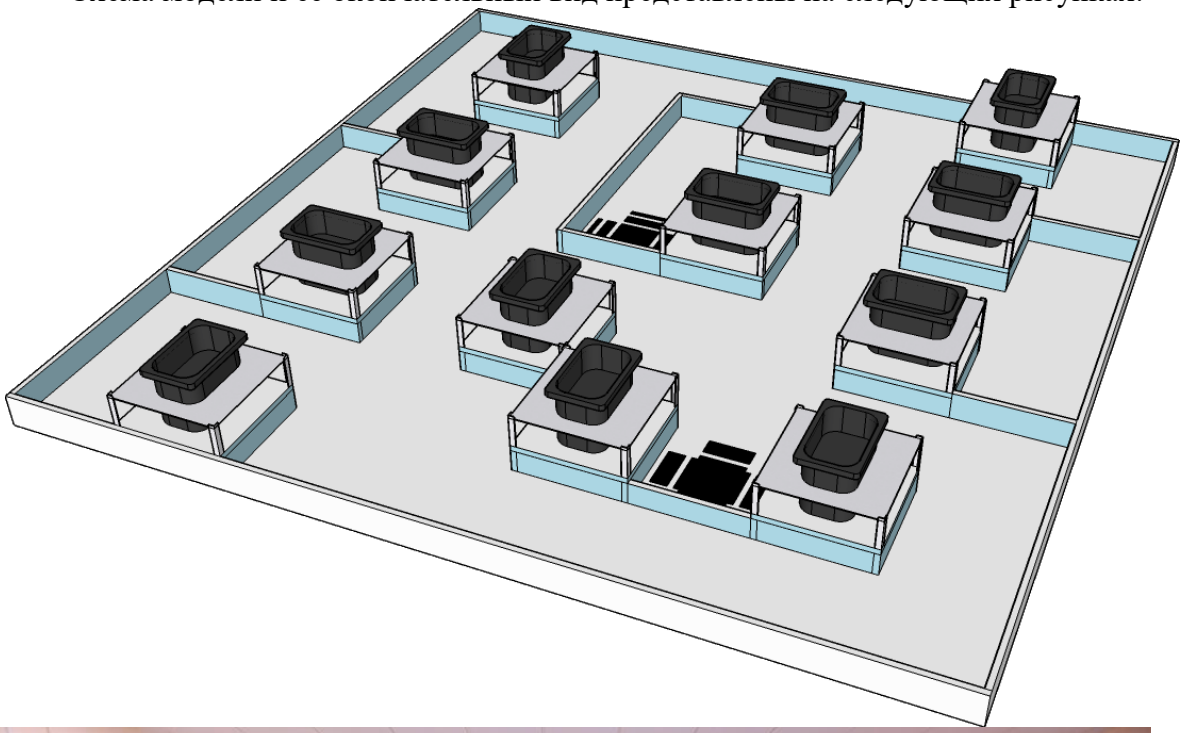
Включая содержательные задачи:

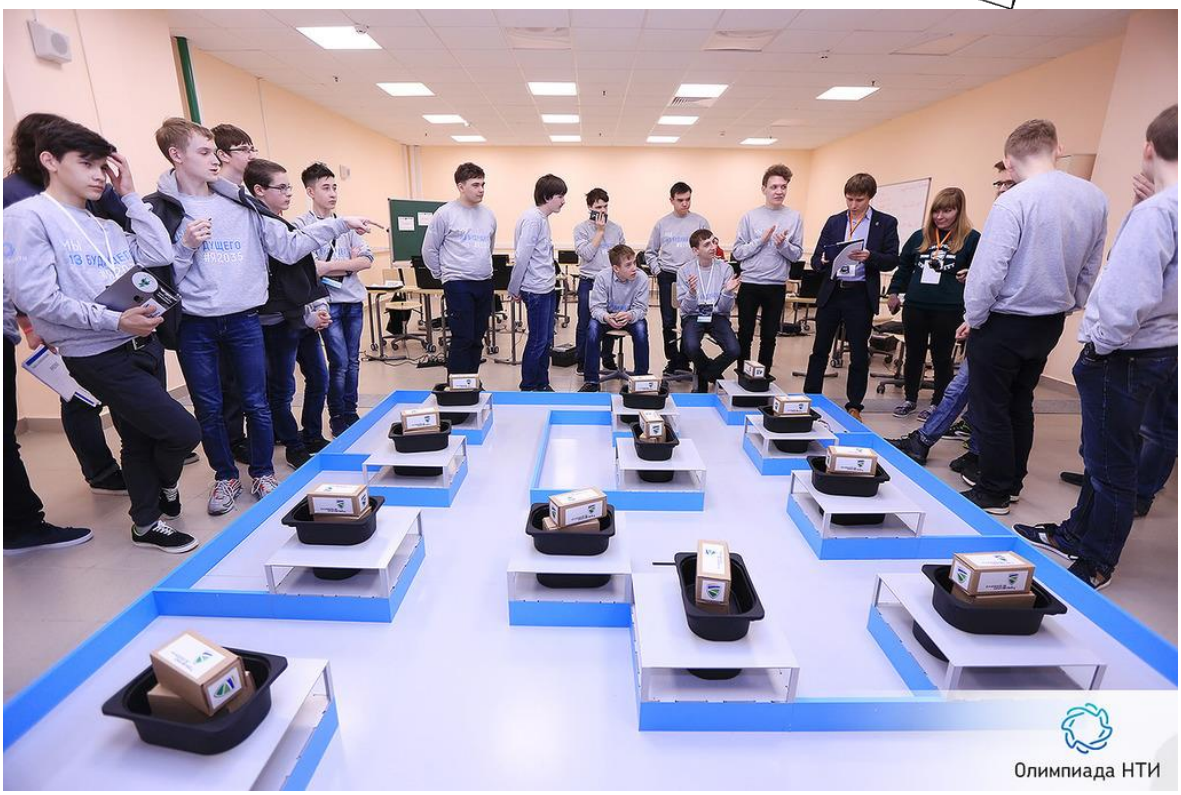
- Реализация алгоритмов построения оптимальных маршрутов до секторов сервисного обслуживания и до финального сектора;
- Реализация алгоритмов автоматического планирования перемещения.

3.2.3. Описание модели логистического центра

Полигон - квадратное поле 3200х3200 мм., разделенное на квадратные сектора 400х400 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 размером 200х300х100 мм.

Схема модели и ее окончательный вид представлены на следующих рисунках:





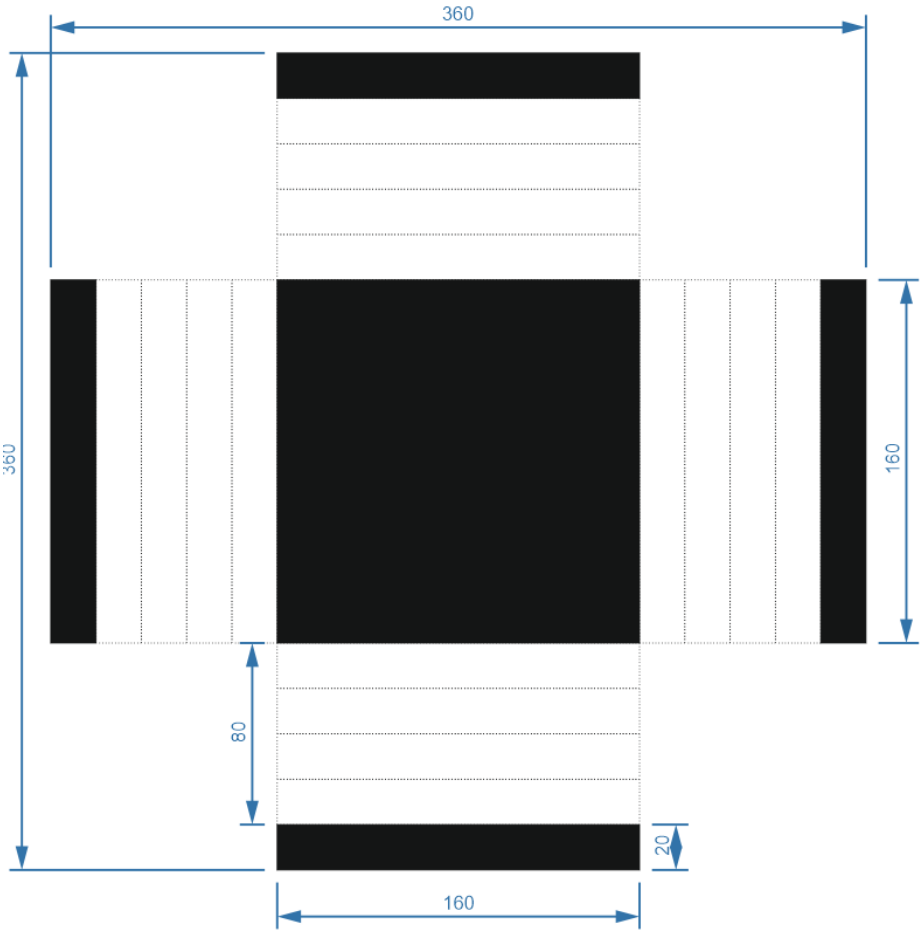
Полигон окружен бортом высотой 100 мм.

Конфигурация полигона определяется в первый день финального этапа и объявляется участникам. Данная конфигурация будет использоваться все дни финального тура.

В местах расположения секторов сервисного обслуживания расположен штрих-код, кодирующий сектор приписки робота-погрузчика - финальный сектор, куда должен прибыть робот в конце выполнения задачи.

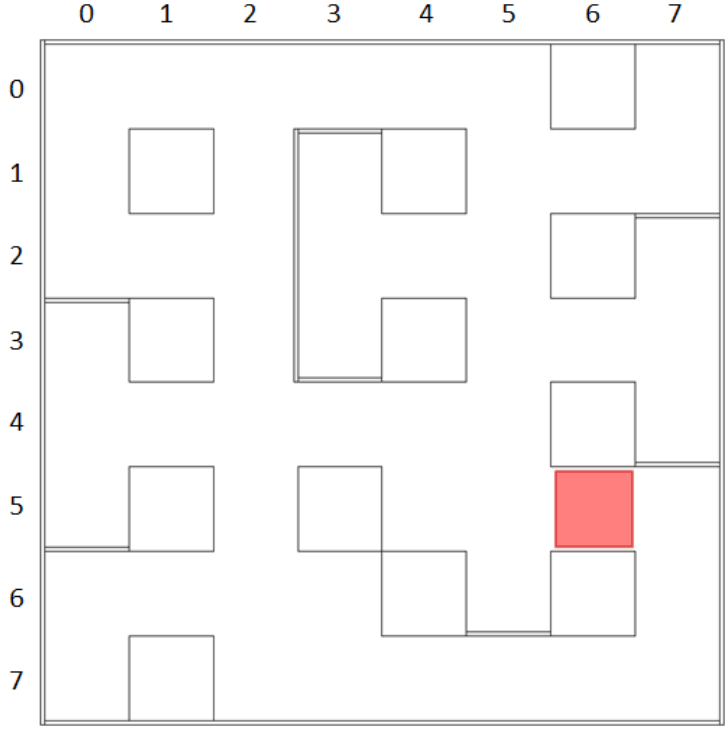
Штрих-код - симметричный, где черная линия обозначает единицу, отсутствие черной линии - 0. Старший бит, закодированного двоичного числа, располагается ближе к краю сектора. Младший бит - ближе к середине. Если закодированное двоичное число находится в диапазоне от 0_{10} до 7_{10}, то оно кодирует координату X финального сектора. Если двоичное число - в диапазоне от 8_{10} до 15_{10}, то оно кодирует Y координату финального сектора, при этом координата определяется вычитанием 8 из закодированного числа. Начало отсчета координат находится в левом верхнем углу поля.

Размеры штрих-кода:



Примеры штрих-кода изображены на рисунке:

В данном примере левый штрих-код задает число 110_2 - 6_{10}. Правый штрих-код задает число 1101_2 - 13_{10}. Финальный сектор находится в позиции с координатами (6, 5):



Финальный сектор никак не обозначается на поле.

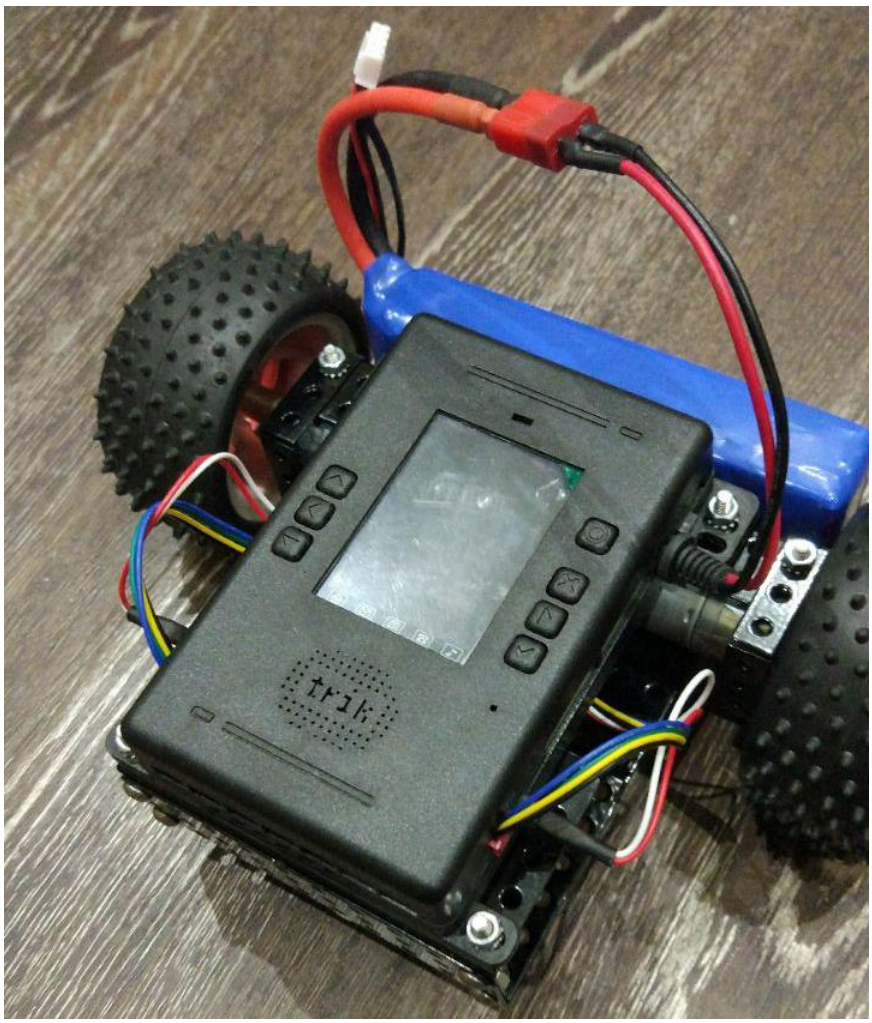
Местоположение секторов сервисного обслуживания определяется в первый день финального этапа и объявляется участникам. Оно остается постоянным все дни финального тура.

Финальный сектор может изменяться перед каждым заездом робота.

3.2.4. Описание конструктора

В первый день финального тура каждой команде выдается мобильная наземная платформа на базе конструктора ТРИК в сборе (блок управления TRIK, аккумулятор, два мотора с энкодерами на датчиках Холла, колеса), но без установленных датчиков. Мобильная платформа построена по принципу дифференциального управления. Физические размеры платформы позволяют совершать все маневры внутри одного сектора модели логистического центра без касания со стенками стеллажей или бортов.

Фотография собранной мобильной платформы:



В первый день финального тура каждой команде выдается набор датчиков:

- 1 датчик касания;
- 2 инфракрасных датчика дальности;
- 2 ультразвуковых датчика расстояния;
- 2 датчика освещенности;

Также команде выдается

- комплект дополнительных деталей из конструктора ТРИК;
- ноутбуки с установленной TRIK Studio (версия с поддержкой интерпретации QtScript в режиме отладки), на каждого члена команды по одному ноутбуку.

3.2.5. Условия проведения

1. Из полученного набора датчиков команды могут выбирать те, с помощью которых, на их взгляд, можно решить задачу наиболее эффективным способом.
2. Команды могут вносить любые изменения в мобильную наземную платформу.
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 должны выполнять решение задачи поэтапно. Критерии прохождения каждого этапа формулируются для каждого дня финального тура. За подзадачи,

решенные в конкретном этапе начисляются, баллы. Баллы за подзадачи можно получить только день, закрепленный за конкретным этапом.

6. Во время рабочего времени команды могут совершать неограниченное количество подходов к полигону для проведения испытаний.
7. Испытания на полигоне должны осуществляться так, чтобы не мешать другим командам, проводящим в это время испытания на полигоне.
8. Каждый день финального тура за 1,5 часа 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.
9. После момента, когда все роботы сданы в карантин, судьи вызывают команды по одной для определения решения подзадач, закрепленных за этапом конкретного дня финального тура. 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.
10. Робот должен выполнять задание полностью автономно. Удаленное управление не допускается. Касание робота участником команды после его старта во время приемочных запусков не допускается. Алгоритм, реализующий систему управления робота должен планировать свое выполнение, полагаясь только на информацию с датчиков. Если какая-то подзадача подразумевает считывание информации с элементов, расположенных на полигоне, запрещается при запуске робота вводить информацию о положении этих элементов или значениях, которые данные элементы определяют.
11. Если во время приемочных запусков у судьи возникли сомнения о том, что задачи подэтапа решены корректно (робот не выполняет задачу полностью автономно, участник вводит значения в робота перед запуском), то он вправе провести инспекцию кода. По результатам инспекции, судья вправе снять с команды баллы, набранные за данный этап.
12. 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.
13. Команда вправе обсуждать с судьей результаты приемочных запусков до вызова следующей команды, но финальное решение остается о начислении баллов остается за судьей.

3.2.6. Процедура проведения приемочных запусков и критерии оценки

Первый этап

1. Судья спрашивает команду об алгоритме обхода секторов макета логистического центра (по правилу правой руки или по правилу левой руки) и определяет 2 сектора запуска робота. Для всех команд, с одинаковым алгоритмом обхода, сектора запуска робота будут одинаковым. Место запуска для робота неизвестно до начала приемочных запусков. Первый сектор запуска используется для первой попытки. Второй сектор запуска используется для второй попытки.
2. Максимальное время выполнения одной попытки - 2 минуты. Время на доработку решения между попытками не выдается.
3. Критерии оценки:

№ п.п.	Критерий	Баллы
1	Робот покинул сектор старта	6
2	Робот проехал половину от всего количества секторов от сектора старта до сектора финиша	13
3	Робот проехал от сектора старта до черной линии (в ходе движения робот пересек черную линию или остановился таким образом, что черная линия пересекает проекцию робота на поле)	6
4	Робот проехал от сектора старта, заехал в сектор финиша (проекция робота на поле внутри сектора) и остановился	6
5	Робот проехал от сектора старта, остановился в секции финиша (проекция робота внутри сектора) и вывел на экран верное количество пройденных секций	12

4. Баллы за две попытки суммируются.
5. Выполнение всех критериев в каждой из двух попыток дает дополнительные 19 баллов.
6. Максимальное количество баллов за этап - 105 баллов.

Второй этап

1. Судья спрашивает команду об алгоритме обхода секторов макета логистического центра (по правилу правой руки или по правилу левой руки) и определяет 4 сектора запуска робота. Для всех команд, с одинаковым алгоритмом обхода, сектора запуска робота будут одинаковым. Место запуска для робота неизвестно до начала приемочных запусков. Первый, второй и четвертый сектора для запуска выбираются так, чтобы робот точно двигался вдоль циклической структуры на макете. Третий сектор выбирается так, чтобы на пути робота не было циклических структур (не считая цикла обхода роботом макета по периметру). Первый сектор запуска используется для первой попытки, второй сектор запуска используется для второй попытки и т.д. Первые две попытки для проверки решения задачи на определение циклических структур макета. Оставшиеся попытки для проверки решения задачи локализации.
2. Максимальное время выполнения первой или второй попытки - 2 минуты. Время на доработку решения между попытками не выдается.
3. Максимальное время выполнения третьей или четвертой попытки - 3 минуты. Время на доработку решения между попытками не выдается.
4. Критерии оценки:
   Определение циклических структур макета логистического центра:

№ п.п.	Критерий	Баллы
1	Робот проехал по периметру вокруг циклической структуры и остановился после определения цикла (допускается проезд по второму кругу до 3х секторов)	7
2	Робот после верного определения циклической структуры, остановился на 3 секунды, и продолжил движение по другим	12

	секторам макета, не принадлежащим циклической структуре (движение не менее, чем по 5 не принадлежащим циклической структуре)	

Локализация:

№ п.п.	Критерий	Баллы
1	Робот проехал не меньше 7 секторов по макету логистического центра и остановился; на экране отображена отображены координаты смещения относительно сектора старта: первое число – смещение по оси X, второе число – смещение по оси Y. При этом ось X совпадает с направлением робота в секторе старта, а ось Y направлена вправо. Данный критерий не будет оцениваться, если робот способен выполнять сразу следующий пункт списка критериев.	6
2	После остановки отображается 2 числа, определяющее позицию, где произошла остановка: первое число – сектор по оси X, второе число – сектор по оси Y	31

5. Баллы за четыре попытки суммируются.
6. Выполнение всех критериев в каждой из попыток для проверки задачи локализации дает дополнительные 19 баллов.
7. Максимальное количество баллов за этап - 119 баллов.

Третий этап

1. Судья спрашивает команду об алгоритме обхода секторов макета логистического центра (по правилу правой руки или по правилу левой руки) и определяет 2 сектора запуска робота. Для всех команд, с одинаковым алгоритмом обхода, сектора запуска робота будут одинаковым. Место запуска для робота неизвестно до начала приемочных запусков. Первый сектор запуска используется для первой попытки, второй сектор запуска используется для второй попытки.
2. Перед каждой попыткой определяется вид штрих-кода. Для всех команд в одной и той же попытки штрих-код будет одинаковым.
3. Максимальное время выполнения первой или второй попытки - 2 минуты. Время на доработку решения между попытками не выдается.
4. Критерии оценки:

№ п.п.	Критерий	Баллы
1	Робот доехал до сектора сервисного обслуживания (проекция робота внутри сектора) и остановился	6
2	После остановки робота на экран выведено закодированное значение.	19

5. Баллы за две попытки суммируются.
6. Выполнение всех критериев в каждой из попыток дает дополнительные 12 баллов.
7. Максимальное количество баллов за этап - 62 балла.

Четвертый этап

1. Перед первой и второй попыткой судья определяет сектора старта и направление робота в секторе страта, также определяется сектор финиша. Команда должна в присутствии судьи изменить программу, внеся данную информацию в программу. Допускается изменить только 5 переменных: 2 переменных (тип - целое), отвечающих за кодирование сектора старта, 1 переменная (одно целое число, либо один символ), отвечающая за кодирование направление в секторе старта, и две переменных (тип - целое), отвечающих за кодирование сектора финиша. После изменения программы, робот устанавливается в сектор старта, программа загружается на робота и должна запуститься. Никаких других изменений в программе не допускается.
   Перед третьей попыткой судья определяет сектор старта робота, его направление в секторе старта и финальный сектор. Для всех команд эти условия будут одинаковыми. Значения штрих-кодов в секторах сервисного обслуживания выставляются в соответствии с позицией финального сектора.
2. Максимальное время выполнения первой или второй попытки - 2 минуты. Время на доработку решения между попытками не выдается.
3. Максимальное время выполнения третьей попытки - 5 минут.
4. Критерии оценки:
   Определение пути перемещения:

№ п.п.	Критерий	Баллы
1	После запуска программы, роботе стоит на месте в течение на 10 секунд. На экран выведен путь от сектора старта до сектора финиша. При этом путь выведен в виде координат секций, через которые должен пройти робот. Данный критерий не будет оцениваться, если робот способен выполнять сразу следующий пункт списка критериев.	6
2	После запуска программы, роботе стоит на месте в течение на 10 секунд. На экран выведен путь от сектора старта до сектора финиша. При этом путь выведен в виде виде алгоритма перемещения (F - одна секция прямо, R- поворот направо, L - поворот налево) с учетом необходимых поворотов в секторе старта.	9
3	Путь, выведенный после запуска робота, является оптимальным по количеству секторов, необходимых для посещения.	3
4	Путь перемещения робота из сектора старта до сектора финиша является оптимальным по количеству секторов.	3
5	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	6

Полная задача:

№ п.п.	Критерий	Баллы
1	Робот выехал из сектора старта и остановился на 10 секунд в первом (по ходу движения робота) секторе сервисного	6

	обслуживания. Перед началом отсчета таймера робот подает звуковой сигнал.	
2	После остановки робота в секторе сервисного обслуживания выведена одна компонента координаты финального сектора вместе с идентификатором оси (буква X или Y). Выведенное число совпадает со штрих-кодом нанесенным в данном секторе сервисного обслуживания.	5
3	Робот продолжил движение и остановился на 10 секунд во втором секторе сервисного обслуживания. Перед началом отсчета таймера робот подает звуковой сигнал.	11
4	После остановки робота в секторе сервисного обслуживания выведены обе компоненты координаты финального сектора вместе с идентификаторами осей (буква X или Y). Координата совпадает со штрих-кодом нанесенным в данном секторе сервисного обслуживания.	5
5	Путь перемещения робота из сектора, где робот локализовался до очередного сектора сервисного обслуживания или финального сектора является оптимальным по количеству секторов.	5
6	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	28

5. Баллы за три попытки суммируются.
6. Выполнение всех критериев в каждой из попыток для проверки задачи определение пути перемещения дает дополнительные 19 баллов.
7. Максимальное количество баллов за этап - 114 баллов.
8.

3.2.7. Решение

Пример программы на языке QtScript, обеспечивающий решение задание задачи последнего этапа (без части считывания штрих-кода - см. следующий пример программы):

```
var pi = 3.1415926535897931;
var d = 5.6;//diameter of wheel, cm
var l = 17.5; // base of robot
var degInRad = 180 / pi;
var alpha = 0;

var readGyro = brick.gyroscope().read
function readYaw() { return  -readGyro()[6]; }

//кнопки
var stopKey = KeysEnum.Esc;
var startKey = KeysEnum.Left;

//диод
var led = brick.led();
```

```
//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;

//сенсоры ИК
sA1 = brick.sensor(A1).readRawData;
sA2 = brick.sensor(A2).readRawData;
sA3 = brick.sensor(A3).readRawData;

//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);

// количество сигналов на оборот
// Подбирается для каждого вида энкодеров
var cpr = 628;

//длина клетки
var cellLength = 40 * cpr/(pi*d);

var direction = 0; // absolute angle of direction movement
var directionOld = 0;
var azimut = 0; // we should go on azimut or turn to it

led.red();
eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);
print("Start");

//инициализация и калибровка гироскопа
print("gyro initialaize...");
brick.gyroscope().calibrate(12000);
script.wait(13000);
print("gyro inited");
brick.display().addLabel(30, 10, "Start!");
brick.display().redraw();
script.wait(1000);

var v = 50; // velocity

var ex = 0;
var ey = 0;
var encLeftOld = 0;
var encRightOld = 0;
var encLeft = 0;
var encRight = 0;
var n = 0;

// вычисление абсолютного угла относительно
// начального положения
function angle()
```

```
{
  var sgn = 0;
  var _direction = readYaw(); // mgrad
  var dtDirection= _direction - directionOld;

  sgn = directionOld == 0 ? 0 :
directionOld/Math.abs(directionOld);
  n += sgn*Math.floor(Math.abs(dtDirection/320000));
  direction = _direction + n * 360000;
  directionOld = _direction;
}

var t = 0;

print("Run, robot, run!");


// делаем препывание основной программы с частотой 200Гц,
// чтобы посчитать абсолютный угол с гироскопа
var mtimer = script.timer(50);
mtimer.timeout.connect(angle);

// вывод в консоль показаний гироскопа
function printGyro()
{
  print("direction=" + direction + " sA1 " + sA1() +
        " sA2 " + sA2() + " sA3 " + sA3());
}

var ptimer = script.timer(300);
ptimer.timeout.connect(printGyro);

//поворот на угол по гироскопу _angle - относительный
// угол на который необходимо повернуться
function turnDirection(_angle, _v){
  _angle = azimut + _angle;
  azimut = _angle;
  _angle = _angle * 1000; //toMGrad
  var _vel = _v == undefined ? 10 : _v;
  var angleOfRotate = _angle - direction;
  var sgn =  angleOfRotate == 0 ? 0 :
angleOfRotate/Math.abs(angleOfRotate);
  mLeft(_vel * sgn);
  mRight(-_vel * sgn);
  while (Math.abs(angleOfRotate = _angle - direction) > 8000){
    script.wait(50);
  }
  brick.motor(M3).forceBreak(500);
  brick.motor(M4).forceBreak(500);
  script.wait(500);
}

// проезд в перед на количество ячеек _kceel со
// скоростью _v выравниваясь по гироскопу на угол azimut
function forward1( _v, _kcell)
{
  _alpha = azimut;
```

```
  var _vel = _v == undefined ? 10 : _v;
  var u = 0;
  var el = Math.abs(eLeft.readRawData());
  while(Math.abs(eLeft.readRawData()) < el + (_kcell *
cellLength)){
    u = -1.5*(_alpha-direction/1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
  }
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);
}


// функция считывания сенсора в массив, при этом в массиве
// {левый датчик, передний датчик,правый датчик, задний датчик}
function readSensors(b){
  var sensorRead = [0,0,0,b];
  left = sA3();
  front = sA1();
  right = sA2();
  if(left>70)    sensorRead[0]  = 1;
  else       sensorRead[0]  = 0;
  if(front>70)  sensorRead[1]  = 1;
  else      sensorRead[1]  = 0;
  if(right>70)  sensorRead[2]  = 1;
  else      sensorRead[2]  = 0;
  return sensorRead;
}

// первая проверка гипотез
function firstCheck(sensor, cell)
{
  var s = 0;
for ( i = 0; i < 4; i++)
  s += sensor[i];
for ( i = 0; i < 4; i++)
  s -= cell[2+i];
return Math.abs(s) < 2;
}

// движение вперёд с учётом переноса гипотез
function forward(preHypotheses)
{
  forward1(v,1);
  var newHypotheses = go(preHypotheses);
  sensor[3]=1;
  return newHypotheses;
}

//поворот влево с учётом переноса гипотез
function turnLeft(preHypotheses)
{
  turnDirection(90,v);
  var newHypotheses = [];
```

```
  while(preHypotheses.length > 0)
  {
    var hypotisis = preHypotheses.pop();
    var k = hypotisis[4];
    hypotisis[2] = hypotisis[5];
    hypotisis[3] = hypotisis[2];
    hypotisis[4] = hypotisis[3];
    hypotisis[5] = k;
    hypotisis[6] += 3;
    hypotisis[6] %= 4;
    newHypotheses.push(hypotisis);
  }
  sensor[3]=sensor[2]
  return newHypotheses;
}

//поворот право с учётом переноса гипотез
function turnRight( preHypotheses)
{
  turnDirection(-90,v);
  var newHypotheses = [];

while(preHypotheses.length > 0)
  {
  var hypotisis = preHypotheses.pop();
  var k = hypotisis[2];
  hypotisis[2] = hypotisis[3];
  hypotisis[3] = hypotisis[4];
  hypotisis[4] = hypotisis[5];
  hypotisis[5] = k;
  hypotisis[6] += 1;
  hypotisis[6] %= 4;
  newHypotheses.push(hypotisis);
}
sensor[3]=sensor[0]
return newHypotheses;
}

//проверка на возможность гипотезы
function check( sensors, cell)
{
  return   (sensor[0]==cell[2] && sensor[1]==cell[3] &&
            sensor[2]==cell[4] && sensor[3]==cell[5] ) ||
           (sensor[0]==cell[3] && sensor[1]==cell[4] &&
            sensor[2]==cell[5] && sensor[3]==cell[2] ) ||
           (sensor[0]==cell[4] && sensor[1]==cell[5] &&
            sensor[2]==cell[2] && sensor[3]==cell[3] ) ||
           (sensor[0]==cell[5] && sensor[1]==cell[2] &&
            sensor[2]==cell[3] && sensor[3]==cell[4] );
}
//перенос гипотезы вперёд
function go(hypotheses){
  var newHypotheses = []

  while(hypotheses.length>0){
    var cell = hypotheses.pop();
    switch(cell[6]){
```

```
      case 0: cell[1]--;break;
      case 1: cell[0]++;break;
      case 2: cell[1]++;break;
      case 3: cell[0]--;break;
      }
  if(cell[0]>=0 && cell[1]>=0 && cell[0]<=7 && cell[1]<=7)
    if(lab[2*cell[1]+1][2*cell[0]+1]!=0)
    {
      cell = generateCell(cell[6],cell[1],cell[0])
      newHypotheses.push(cell);
    }
  }
  return newHypotheses
}

//создание описания клетки
function generateCell(k , i , j ){
  switch(k){
    case 0:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+0] = lab[2 * i + 1][2 * j];
          cell[2+1] = lab[2 * i ][2 * j +1];
          cell[2+2] = lab[2 * i  + 1][2 * j + 2];
          cell[2+3] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 0;
          return cell;
    case 1:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+3] = lab[2 * i + 1][2 * j];
          cell[2+0] = lab[2 * i ][2 * j +1];
          cell[2+1] = lab[2 * i  + 1][2 * j + 2];
          cell[2+2] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 1;
          return cell;
    case 2:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+2] = lab[2 * i + 1][2 * j];
          cell[2+3] = lab[2 * i ][2 * j +1];
          cell[2+0] = lab[2 * i  + 1][2 * j + 2];
          cell[2+1] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 2;
          return cell;
    case 3:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+1] = lab[2 * i + 1][2 * j];
          cell[2+2] = lab[2 * i ][2 * j +1];
          cell[2+3] = lab[2 * i  + 1][2 * j + 2];
          cell[2+0] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 3;
```

```
            return cell;
    }
  }

// карта
  var lab = [
    [0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0,0],
    [0,1, 1,1, 1,1, 1,1, 1,1, 1,1, 0,0, 0,1,0],

    [0,1, 0,0, 1,1, 0,0, 0,0, 1,1, 1,0, 1,1,0],
    [0,1, 0,0, 0,1, 0,1, 0,0, 0,1, 1,1, 1,1,0],

    [0,1, 0,0, 1,1, 0,1, 0,0, 1,1, 0,0, 0,0,0],
    [0,1, 1,1, 1,1, 0,1, 1,1, 1,1, 0,0, 0,1,0],

    [0,0, 0,0, 1,1, 0,1, 0,0, 1,1, 1,0, 1,1,0],
    [0,1, 0,0, 0,1, 0,1, 0,0, 0,1, 1,1, 1,1,0],

    [0,1, 1,0, 1,1, 1,0, 1,0, 1,1, 0,0, 0,1,0],
    [0,1, 1,1, 1,1, 1,1, 1,1, 1,1, 0,0, 0,1,0],

    [0,1, 0,0, 1,1, 0,0, 1,1, 1,1, 1,0, 1,0,0],
    [0,1, 0,0, 0,1, 0,0, 0,1, 1,1, 1,1, 1,1,0],

    [0,0, 1,0, 1,1, 1,0, 0,0, 1,1, 0,0, 1,1,0],
    [0,1, 1,1, 1,1, 1,1, 0,0, 0,1, 0,0, 0,1,0],

    [0,1, 0,0, 1,1, 1,1, 1,0, 1,0, 1,0, 1,1,0],
    [0,1, 0,0, 0,1, 1,1, 1,1, 1,1, 1,1, 1,1,0],

    [0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0,0]
    ]

function go_to_cell(x_begin,y_begin,directionOfRobot, x_end,
y_end){
  // creating array for path
  var path = [
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  ];


  path[x_begin][y_begin][0] = 0;
```

```
var x_temp = x_begin;
var y_temp = y_begin;
var queue = new Array();

// алгоритм a*
while(path[x_end][y_end][0] == -1){
  if((x_temp-1) >= 0){
    var coord_temp = x_temp-1;
    if(((path[coord_temp][y_temp][0]) == -1) &&
        (lab[y_temp*2+1][x_temp*2+1 -1] == 1)){
      path[coord_temp][y_temp][0] = path[x_temp][y_temp][0]+1;
      path[coord_temp][y_temp][1] = x_temp;
      path[coord_temp][y_temp][2] = y_temp;
      queue.push([coord_temp,y_temp]);
    }
  }


  if((x_temp+1) < 8){
    var coord_temp = x_temp+1;
    if(((path[coord_temp][y_temp][0]) == -1) &&
        (lab[y_temp*2+1][x_temp*2+1 +1] == 1)){
      path[coord_temp][y_temp][0] = path[x_temp][y_temp][0]+1;
      path[coord_temp][y_temp][1] = x_temp;
      path[coord_temp][y_temp][2] = y_temp;
      queue.push([coord_temp,y_temp]);
    }
  }

  if((y_temp-1) >= 0){
    var coord_temp = y_temp - 1;
    if(((path[x_temp][coord_temp][0]) == -1)  &&
        (lab[y_temp*2+1-1][x_temp*2+1] == 1)){
      path[x_temp][coord_temp][0] = path[x_temp][y_temp][0]+1;
      path[x_temp][coord_temp][1] = x_temp;
      path[x_temp][coord_temp][2] = y_temp;
      queue.push([x_temp,coord_temp]);
    }
  }

  if((y_temp+1) < 8){
    var coord_temp = y_temp + 1;
    if (((path[x_temp][coord_temp][0]) == -1) &&
         (lab[y_temp*2+1+1][x_temp*2+1] == 1)){
      path[x_temp][coord_temp][0] = path[x_temp][y_temp][0]+1;
      path[x_temp][coord_temp][1] = x_temp;
      path[x_temp][coord_temp][2] = y_temp;
      queue.push([x_temp,coord_temp]);
    }
  }
  var firstElementOfQueue = queue.shift();
  x_temp = firstElementOfQueue[0];
  y_temp = firstElementOfQueue[1];
  script.wait(3);
}
```

```
// зная предков каждой посещенной ячейки создаем
// стек необходимого перемещения робота
var stack = new Array();
x_temp = x_end;
y_temp = y_end;
while((x_temp != x_begin) || (y_temp !=y_begin)){
  stack.push([x_temp,y_temp]);
  var temp_cell = path[x_temp][y_temp];
  x_temp = temp_cell[1];
  y_temp = temp_cell[2];
}



// Проходим по стеку и перемещаем робота в
// соответствии с ним.
// В стеке у нас находятся последовательно
// ячейки в которые нам необходимо попасть.
// Последний элемент стека - ячейка в которую нам
// необходимо попасть используб данную функцию go_to_cell.
var len = stack.length
var prev_cell = [x_begin,y_begin];
for (var i = 0; i < len;i++){
  var temp_cell = stack.pop();
  var dx = temp_cell[0] - prev_cell[0];
  var dy = temp_cell[1] - prev_cell[1];

  if(dy == 0){
    if(dx > 0){
      if(directionOfRobot == 2){
        turnDirection(90,v);
        directionOfRobot --;

      }else{
        while(directionOfRobot != 1){
          turnDirection(-90,v);
          directionOfRobot ++;
          directionOfRobot %= 4;
        }
      }
    }else{
      if(directionOfRobot == 0){
        turnDirection(90,v);
        directionOfRobot = 3;

      }else{
        while(directionOfRobot != 3){
          turnDirection(-90,v);
          directionOfRobot ++;
          directionOfRobot %= 4;
        }
      }
    }
  }else{
    if(dy > 0){
      if(directionOfRobot == 3){
        turnDirection(90,v);
```

```
          directionOfRobot --;

        }else{
          while(directionOfRobot != 2){
            turnDirection(-90,v);
            directionOfRobot ++;
            directionOfRobot %= 4;
          }
        }
      }else{
        if(directionOfRobot == 1){
          turnDirection(90,v);
          directionOfRobot --;

        }else{
          while(directionOfRobot != 0){
            turnDirection(-90,v); // turn left
            directionOfRobot ++;
            directionOfRobot %= 4;
          }
        }
      }
    }
    forward1(v,1);


    prev_cell = temp_cell;
  }
  return directionOfRobot;
}

// ----------------- main part

// локализация
while(true){

  var sensor = [0,0,0,0];


  var hypotheses1 = [];
  var hypotheses = [];

  //создание гипотез
  for (i = 0; i < 8; i++)
  {
    for (j = 0; j < 8; j++)
    {
      if (lab[2 * i + 1][2 * j + 1]  = 1)
      {
         for(q = 0 ; q<4;q++)
        hypotheses.push(generateCell(q,i,j));
      }
    }
  }

  //чтение сенсоров
  sensor= readSensors(0);
```

```
  //первая проверка
  while (hypotheses.length>0)
  {
    var hypotisis = hypotheses.pop();
    if(firstCheck(sensor, hypotisis)){
      hypotheses1.push(hypotisis);
    }
  }

  //полная локализация
  while (hypotheses1.length>1)
  {
    if(sensor[0]==1){
      hypotheses = turnLeft(hypotheses1);
      hypotheses = forward(hypotheses);
      } else if(sensor[1] == 1)
      hypotheses = forward(hypotheses1);
      else {
      hypotheses = turnRight(hypotheses1);
        }

    sensor = readSensors(sensor[3]);

    while (hypotheses.length>0)
    {
      var hypotisis = hypotheses.pop();
      if (check(sensor, hypotisis))
        hypotheses1.push(hypotisis)
    }

    script.wait(1000);

  }
  if(hypotheses1.length == 1)break;
}

var x = hypotheses1[0][0];
var y = hypotheses1[0][1];
var dir = hypotheses1[0][6];
print(x + " " + y + " " + dir);
dir = go_to_cell(x,y,dir,7,1);
script.wait(5000);
var xOfTheEnd = 5;
dir = go_to_cell(7,1,dir,4,5);
script.wait(5000);
var yOfTheEnd = 7;
dir = go_to_cell(4,5,dir,xOfTheEnd,yOfTheEnd);
print("Prodram end!");
led.orange();
brick.stop();
script.quit();
```

Пример программы на языке QtScript, обеспечивающий решение задание третьего этапа:

```
var pi = 3.1415926535897931;
```

```
var readGyro = brick.gyroscope().read
function readYaw() { return  -readGyro()[6]; }

//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;

//сенсоры освещенности
sA5 = brick.sensor(A5);
sA6 = brick.sensor(A6);

//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);

eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);
print("Start");

//инициализация и калибровка гироскопа
print("gyro initialaize...");
brick.gyroscope().calibrate(12000);
script.wait(13000);
print("gyro inited");
brick.display().addLabel(30, 10, "Start!");
brick.display().redraw();
script.wait(1000);

var encLeftOld = 0;
var encRightOld = 0;
var encLeft = 0;
var encRight = 0;

var main = function(){
  mLeft(0);
  mRight(0);

  var _alpha = 0;
  var _vel = 25;
  var u = 0;

  // едем до калибровочной линии
  while((sA5.read() < 90) && (sA6.read() < 90)){
    u = -0.6*(_alpha-readYaw()/1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
  }

  _vel = 15;
  // количество тиков энкодера в одной полосе кода
```

```
  var _encOne = 27; // выяснено эмпирически
  encLeft = eLeft.readRawData();
  encLeftOld = encLeft;
  var twoInPower = 8;
  var numb = 0;
  var count = 1;
  var sv;

  // считываем значений 4х битов
  while(count < 5){
    u = -0.5*(_alpha-readYaw()/1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    encLeft = eLeft.readRawData();
    sv = sA5.read();
    // мы считываем значение белая/черная линия
    // с частотой дискретизации _encOne
    if(Math.abs(encLeft - encLeftOld) >= count * _encOne){
      count ++;
      if(sv > 90){
        numb += twoInPower;
      }
      twoInPower = twoInPower / 2;
    }
    script.wait(3);
  }

  print(numb);

  brick.stop();
  script.quit();
}
```

3.3. Критерий определения победителей и призеров заключительного этапа

В заключительном этапе олимпиады баллы участника складываются из двух частей: он получает баллы за индивидуальное решение задач по предметам (математика и информатика) и за командное решение практической задачи.

$$S = S_1 + S_2$$

где S_1 — количество баллов, набранное в рамках индивидуальной части заключительного этапа (максимум 200 баллов); S_2 — количество баллов, набранное в рамках командной части заключительного этапа (максимум 400 баллов).

Критерий определения победителей и призеров:

	Призеры	**Победители**
Набранные баллы	от 120 до 150 баллов	от 150 баллов и выше