БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра информационных систем управления

С.И. Кашкевич, А.А. Толстиков

ЗАДАЧИ ШКОЛЬНЫХ ОЛИМПИАД ПО ИНФОРМАТИКЕ

Учебно-методическое пособие для студентов факультета прикладной математики и информатики

В трёх частях

Часть 2

Задачи командных чемпионатов г. Минска по программированию (2008 – 2010 годы)

МИНСК
2013

УДК 004(075.3)
ББК 32.81к2
К31

Утверждено на заседании кафедры информационных систем управления
факультета прикладной математики и информатики
29 августа 2013 г., протокол № 1

Р е ц е н з е н т
доктор физико-математических наук, профессор *В. М. Котов*

Кашкевич, С.И.

К31 Задачи школьных олимпиад по информатике: учеб.-метод. пособие для студентов фак. прикладной математики и информатики. В 3 ч. Ч. 2: Задачи командных чемпионатов г. Минска по программированию (2008 – 2010 годы) / С. И. Кашкевич, А.А. Толстиков. – Минск : БГУ, 2013. – 60 с.

Пособие содержит условия, а также разборы решений задач командных чемпионатов г. Минска по программированию (2008 – 2010 годы).

Предназначено для студентов факультета прикладной математики и информатики, а также для школьников, готовящихся к поступлению в вузы.

УДК 004(075.3)
ББК 32.81к2

© Кашкевич С.И.,
Толстиков А.А., 2013
© БГУ, 2013

Предисловие

Вторая часть сборника школьных олимпиадных задач по информатике содержит задачи командных чемпионатов по программированию для школьников г. Минска, проводившихся в 2008 – 2010 годах.

В настоящее время соревнования по спортивному программированию проводятся в нескольких различных форматах. В первой части пособия рассматривались задачи, подготовленные для т.н. формата IOI, в котором проводятся международные олимпиады по информатике. Этот формат является официальным для соревнований, которые проводятся под эгидой Министерства образования Республики Беларусь.

Другим форматом для проведения олимпиад по спортивному программированию является т.н. формат ACM (Association for Computing Machinery), соревнования в котором проводятся среди студенческих команд, начиная с 1977 года. К настоящему моменту олимпиада превратилась во всемирное соревнование – студенческий чемпионат мира, в котором проводятся четвертьфинальные, полуфинальные и финальные соревнования. На финальные соревнования приглашаются более 100 команд из различных стран мира, причём один университет может выставить не более одной команды. Медали получают, как правило 12 лучших команд. Команды Белорусского государственного университета неоднократно становились победителями этих престижных соревнований (2004 год – золотая медаль, 2007 год – бронзовая медаль, 2012 и 2013 годы – серебряные медали). Кроме того, в 2012 году бронзовую медаль завоевала команда БГУИР.

Начиная с 1998 года, в Минске проводится один из четвертьфиналов этого чемпионата (Западный подрегион Северо-восточного европейского региона), где соревнуются студенты из Беларуси, стран Балтии и Калининградской области Российской Федерации.

Школьные соревнования в формате ACM также проводятся, хоть и не так широко, как олимпиады в формате IOI. Можно, в частности, отметить командные чемпионаты школьников г. Минска (2008 – 2013 годы), а также Гомельскую командную олимпиаду. Эти соревнования проводятся под эгидой региональных органов образования.

Соревнования в формате ACM проводятся как для команд (в состав команды входит не более трёх участников), так и индивидуально. Команде или участнику (в случае индивидуальной олимпиады) предоставляется один компьютер.

На соревнования предлагаются от 6 до 12 задач классического типа, которые необходимо решить за время олимпиады (от трёх до пяти часов). Задачи обычно имеют различную сложность – от весьма простых до практически нерешаемых. В разборе задач командных чемпионатов школьников г. Минска приводятся критерии оценки сложности предложенных задач, с точки зрения их авторов.

В отличие от формата IOI, для того, чтобы решение задачи было засчитано, оно должны пройти все тесты из предложенного набора тестов (частичные решения не засчитываются). Для каждой сданной задачи начисляется т.н. *штрафное время*. Оно состоит из двух частей: количества полных минут, прошедших от начала соревнования до момента отсылки правильного решения в тестирующую систему, и штрафа в 20 минут за каждую предшествующую неудачную посылку. Так, если задача была сдана на 35-й минуте соревнования, а перед этим были выполнены две неудачные попытки сдать эту задачу, то штрафное время по этой задаче составит 74 минуты (34 + 2 * 20). Последующие отсылки уже принятой задачи не изменяют её штрафное время. Штрафное время за нерешённые задачи не начисляется, даже если по этим задачам были неудачные посылки.

Участники олимпиады ранжируются в итоговой таблице соревнования по количеству решённых задач. В случае равенства этого показателя для нескольких участников они ранжируются по сумме полученных штрафных времён. Итоговая таблица имеет следующий вид (показаны результаты олимпиады БГУ 2010 года, проводившейся на одном наборе задач с третьим командным чемпионатом школьников г. Минска, приведена только верхняя часть таблицы):

#	Команда	path	permut	polyhedron	rally	rect	shield	shooting	summit	tickets	twocapitals	v_shape1	v_shape2	Задачи	Время
1	BSU 4 (L)aBSUrd	+ 1:28	+1 1:52	+1 3:41	+2 0:41	+ 0:17	+ 0:09	+ 0:39	+2 1:02	+2 1:08	+ 2:46	+ 2:23	+ 2:24	12	1270
2	BSU 5	+ 1:22	+ 0:50	-9	+ 0:31	+ 0:07	+ 0:13	+ 0:39	+1 2:30	+ 0:04	+ 3:05	+1 2:02	+ 2:03	11	846
3	BSU 21	+ 0:49	+ 2:37	-2	+ 0:26	+ 0:06	+ 0:11	+ 0:08	+2 2:50	+ 0:18	+ 3:40	+1 1:42	+ 1:45	11	932

Здесь знаком «+» отмечены задачи, сданные с первой попытки, число, возможно, стоящее за плюсом – количество предыдущих неудачных попыток, отрицательное число задаёт количество неудачных попыток сдать задачу, которая так и осталась в итоге нерешённой. Если

участники не предпринимали ни одной попытки сдать задачу, в соответствующей позиции ставится точка. Для каждой сданной задачи указывается также время, прошедшее от начала соревнования до момента сдачи.

Текущая таблица результатов видна участникам в течение всего соревнования, за исключением последнего часа. За час до окончания соревнования наступает т.н. «заморозка», и участники видят общую таблицу только на момент заморозки. Собственные результаты команда видит в течение всего времени турнира.

Особенности правил ACM накладывают свой отпечаток на тактику командной борьбы. Соревнование по времени разбивается на несколько этапов. Вначале определяются самые простые задачи, которые следует решить как можно быстрее (и не делать при этом неудачных попыток, поскольку такие попытки на этом этапе являются следствием глупых ошибок). Увы, для слабых команд этот первый этап зачастую оказывается и последним…

Далее члены команды рассматривают оставшиеся задачи и определяют, кто какую задачу способен решить… После этого участники поочерёдно меняются местами у единственного предоставленного им компьютера и работают над своими задачами.

Наконец, третий этап наступает, когда надо объединить усилия и решить хотя бы некоторые из оставшихся задач. Чем больше времени остаётся для этого, тем лучше. Сейчас уже неудачные попытки в расчёт не принимаются – очевидно, что сдать задачу даже с огромным штрафным временем лучше, чем не сдать её вообще!

Важным является и распределение ролей в команде. Как всякий командный вид спорта, соревнование по спортивному программированию по правилам ACM требует слаженности в действиях команды (не говоря уже о психологической совместимости между её членами).

В пособии представлены задачи для командных чемпионатов школьников по программированию, которые проводились в г. Минске в 2008 – 2010 годах. В него включено 30 задач; для каждой из них написан разбор решения. Наборы тестов для каждой задачи можно найти в интернете по адресу:

https://docs.google.com/file/d/0Bymq5SA5EB3NOHBBVExfZThQZGM/edit?usp=sharing

В дальнейшем планируется издание третьей части пособия, в которой будут рассмотрены задачи следующих трёх чемпионатов (2011 – 2013).

Пособие рекомендуется студентам факультета прикладной математики и информатики, школьникам, участвующим в олимпиадном

движении по информатике или планирующим принимать такое участие, а также всем, кто интересуется олимпиадным движением.

Авторы благодарят А.Д. Субача и А.Е. Климанского, которые внимательно прочли рукопись и сделали ряд замечаний и предложений (особенно касающихся разборов задач), улучшивших её качество.

Выражаем также благодарность доктору физико-математических наук, профессору В.В.Котову за рецензирование рукописи.

Замечания и предложения по усовершенствованию пособия просим присылать на электронный адрес kash@bsu.by.

Условия задач

Первый чемпионат (2008 год)

Чемпионат состоялся 11 октября 2008 года.
Автор задач – С.И. Кашкевич. Следует сказать, что большинство задач не являются оригинальными и ранее предлагались на соревнованиях различного уровня. Задача «Прижимистый дачник» сейчас активно используется при изучении темы «сортировка» для студентов и школьников. Решения писал также А.М. Статкевич.

Задача A: Наибольшая сумма цифр

Ограничения по времени: 2 секунды
Ограничения по памяти: 32 мебибайта

Задана последовательность ($\mathbf{S}_1$, $\mathbf{S}_2$, … $\mathbf{S}_N$) целых чисел. Найдите элемент этой последовательности с наибольшей суммой цифр. Если таких элементов несколько, выберите наибольший из них.

Входные данные читаются из текстового файла SUMDIGIT.IN. Первая строка этого файла содержит величину N ($1 \le N \le 10^6$). Далее следуют N чисел, каждое из которых не превосходит по модулю 10^9 – элементы последовательности. Эти числа разделяются между собой пробелами и/или символами перевода строки.

Выходные данные помещаются в текстовый файл SUMDIGIT.OUT и содержат единственную строку с найденным элементом последовательности.

Пример входных и выходных данных

8 15 100000009 91 118 34 109000000 700000000 0	109000000

Задача B: Пересечение отрезков

Ограничения по времени: 1 секунда
Ограничения по памяти: 32 мебибайта

На плоскости заданы два отрезка ненулевой длины $[a_1; b_1]$ и $[a_2; b_2]$. Вам требуется определить, имеют ли эти отрезки хотя бы одну общую точку.

Входные данные читаются из текстового файла SEGMENT.IN, содержащего две строки. Каждая из этих строк содержит четыре числа – координаты x и y точек a_i и b_i, разделенные одним или несколькими пробелами. Эти числа – целые и не превосходят по модулю 100000.

Выходные данные помещаются в файл SEGMENT.OUT и содержат единственную строку 'Yes', если отрезки имеют хотя бы одну общую точку, и 'No' в противном случае.

Примеры входных и выходных данных

0 2 6 6 0 0 5 5	No
0 2 6 6 0 0 6 6	Yes
0 2 0 0 -1 1 1 1	Yes

Задача C: Локальные экстремумы

Ограничения по времени: 2 секунды

Ограничения по памяти: 32 мебибайта

Задана последовательность целых чисел (a_1, a_2, …, a_N). Элемент a_i этой последовательности является *локальным минимумом*, если $a_i < a_{i-1}$, и $a_i < a_{i+1}$. Аналогичным образом определяется понятие *локального максимума*. Очевидно, первый и последний элемент последовательности не могут быть ни локальным минимумом, ни локальным максимумом.

Вам требуется определить количество локальных минимумов и максимумов в исходной последовательности

Входные данные читаются из текстового файла EXTREMUM.IN и состоят из нескольких строк. Первая строка содержит значение числа N ($1 \le N \le 1000000$). Далее следуют N чисел, каждое из которых не превосходит по модулю 10^9 – элементы последовательности. Эти числа разделяются между собой пробелами и/или символами перевода строки.

Выходные данные помещаются в текстовый файл EXTREMUM.OUT и содержат строку из двух чисел, разделенных одним или несколькими пробелами – количества найденных локальных минимумов и максимумов.

Пример входных и выходных данных

8 10 32 5 4 9 3 2 1	1 2

Задача D: Широко шагая

Ограничения по времени: 1 секунда
Ограничения по памяти: 32 мебибайта

Назовем шахматную фигуру (*n,k*)-конем, если она перемещается на *n* клеток по горизонтали или вертикали и на *k* клеток в перпендикулярном направлении. Обычный шахматный конь является, согласно этой терминологии, (2,1)-конем.

На шахматную доску помещена единственная фигура – (1,0)-конь. Первоначально она находится на клетке *A*. После первого хода эта фигура превращается в (1,1)-коня, затем – в коня с характеристиками (2,1), (2,2), (3,2) и т.д.

Вам необходимо определить, за какое минимальное количество ходов эта фигура достигнет клетки *B*, не совпадающей с *A*. Выходить за границы доски нельзя!

Входные данные находятся в текстовом файле CHEVAL.IN. Единственная строка этого файла содержит координаты клеток *A* и *B* в стандартной шахматной нотации, разделенные одним пробелом. Согласно этой нотации, столбцы доски обозначаются латинскими буквами от *a* до *h* слева направо, строки – цифрами от *1* до *8* снизу вверх, так что левая нижняя клетка обозначается как `a1`, а правая верхняя – как `h8`.

Выходные данные помещаются в текстовый файл CHEVAL.OUT, и содержат единственное число – минимальное количество ходов, необходимое для перемещения фигуры из клетки *A* в клетку *B*, либо -1, если решения задачи не существует.

Пример входных и выходных данных

c4 d5	3

Пояснение: один из возможных путей – c4, d4, c3, d5.

Задача E: По-русски, как по-французски...

Ограничения по времени: 1 секунда
Ограничения по памяти: 32 мебибайта

Запись чисел прописью на французском языке имеет свои особенности. Во-первых, особые слова для числительных от 17 до 19 отсутствуют, и число 17 записывается как dix-sept (буквально «десять семь»). Во-вторых, отсутствуют особые слова для записи чисел 70 и 90, так что счет идет «по двадцаткам». В-третьих, запись числа 80 выглядит как quatre-veint (буквально «четыре двадцать»).

Примеры записи французских числительных приведены в таблице 1:

Таблица 1

Число	*Французская запись*	*Буквальный перевод*
70	soixante-dix	шестьдесят десять
74	soixante quatorze	шестьдесят четырнадцать
81	quatre-veint un	четыре двадцать один
99	quatre-veint dix-neuf	четыре двадцать десять девять

Применим похожий подход к русскому языку. Пусть для записи прописью т.н. *базовых* чисел используются следующие слова или сочетания из двух слов, заданные в таблице 2:

Таблица 2

Базовое число	*Его запись*	*Базовое число*	*Его запись*
1	один	17	десять семь
2	два	18	десять восемь
3	три	19	десять девять
4	четыре	20	двадцать
5	пять	30	тридцать
6	шесть	40	сорок
7	семь	50	пятьдесят
8	восемь	60	шестьдесят
9	девять	80	четыре двадцать

Базовое число	*Его запись*	*Базовое число*	*Его запись*
10	десять	100	сто
11	одиннадцать	200	двести
12	двенадцать	300	триста
13	тринадцать	400	четыреста
14	четырнадцать	500	пятьсот
15	пятнадцать	600	шестьсот
16	шестнадцать	800	четыре двести

Запись любого числа N в интервале от 1 до 999 производится по следующему алгоритму. Находим максимальное базовое число K, не превосходящее N, и помещаем в строку результата его запись прописью. Если $N > K$, выполняем эти же действия для числа $N - K$.

Входные данные находятся в текстовом файле FRENCH.IN, состоящем из одной строки, содержащей десятичную запись натурального числа в интервале от 1 до 999.

Выходные данные помещаются в текстовый файл FRENCH.OUT и содержат одну строку – запись прописью заданного числа по приведенному выше алгоритму. Слова в строке должны разделяться единственным пробелом, пробелы в начале и конце строки недопустимы. Для вывода русских букв можно использовать кодовую страницу 866 (кодировку OEM), либо кодовую таблицу 1251 (кодировку ANSI).

Примеры входных и выходных данных

70	шестьдесят десять
74	шестьдесят четырнадцать
81	четыре двадцать один
899	четыре двести четыре двадцать десять девять

Задача F: Химическая реакция

Ограничения по времени: 1 секунда
Ограничения по памяти: 32 мебибайта

Молекула каждого химического *вещества* состоит из атомов, принадлежащих одному или нескольким химическим *элементам*. Общепринятое обозначение химических элементов состоит либо из одной заглавной латинской буквы (**O** – кислород, **H** – водород, **N** – азот

и т.д.), либо из двух латинских букв – заглавной и строчной (**Na** – натрий, **Cu** – медь и т.д.).

Состав и пространственная структура любого вещества описывается его *формулой*, представляющей собой строку из обозначений химических элементов, за которыми может указываться подстрочный индекс – количество атомов этого элемента в одной молекуле вещества (число, не превосходящее 99). Если это количество равно единице, индекс не ставится. Так, формула воды $\mathbf{H_2O}$ означает, что одна молекула воды содержит два атома водорода и один – кислорода. Кроме того, для информации о пространственной структуре молекулы в записи формулы этого вещества обозначения некоторых элементов могут повторяться. Например, молекула этилового спирта, имеющего формулу $\mathbf{C_2H_5OH}$, содержит шесть атомов водорода.

В результате химической *реакции* одно или несколько *исходных* веществ превращаются в *результаты* реакции. Запись информации о химической реакции – *уравнение реакции* представляет собой строку, разделенную символом равенства на *левую* и *правую* части. Левая часть описывает исходные вещества, вступающие в реакцию, правая – ее результаты. Каждая из частей уравнения реакции содержит формулы химических веществ, разделенные знаком «плюс». Перед любой формулой может стоять целочисленный коэффициент, не превосходящий 99, который задает количество молекул этого вещества, вступающих в реакцию. Как и при записи формулы, коэффициент не указывается, если это количество равно единице. Пробелы в записи уравнения реакций не допускаются.

Приведем примеры уравнений химических реакций (эти реакции могут никогда не происходить, а указанные в тестах и примерах вещества и химические элементы не существовать в природе!):

$2H_2+O_2=2H_2O$

$Na_2NO_4+3HNO_3=Na_2O+6O_2+3NH$

Закон сохранения массы требует, чтобы суммарное количество атомов каждого химического элемента было одинаковым в правой и левой части уравнения реакции. Первая из приведенных строк удовлетворяет этому закону, а вторая – нет (количество атомов азота в левой части – 4, а в правой – 3).

К сожалению, при подготовке задачи была допущена ошибка форматирования, и все подстрочные индексы преобразовались в обычный текст…

Вам требуется определить, удовлетворяют ли записи о химических реакциях закону сохранения массы.

Входные данные находятся в текстовом файле CHEM.IN, содержащем единственную строку с записью об одной химической реакции. Длина строки не превосходит 300 символов.

Выходные данные помещаются в текстовый файл CHEM.OUT. Единственная строка этого файла должна содержать слово ‘YES’ или ‘NO’ в зависимости от того, удовлетворяет ли запись из входного файла закону сохранения массы.

Примеры входных и выходных данных

2H2+O2=2H2O	YES
Na2NO4+3HNO3=Na2O+6O2+3NH	NO

Задача G: Прижимистый дачник

Ограничения по времени: 2 секунды

Ограничения по памяти: 32 мебибайта

Пенсионер Иван Иваныч вырастил на своей даче богатый урожай и упаковал его в N корзин. Вес i-й ($1 \le i \le N$) корзины равен P_i . Машины у Ивана Иваныча нет, но зато есть бесплатный сезонный билет на электричку. Нанимать автомобиль для вывоза урожая Иван Иваныч не хочет – жалко денег... Вот и решил наш дачник потихоньку перевезти все свои корзины электричкой.

Согласно железнодорожным правилам, каждый пассажир имеет право бесплатно не более двух мест ручной клади суммарным весом не более Q (каждая корзина считается отдельным местом). Платить за излишний багаж Иван Иваныч также не желает!

Определите минимальное число поездок, которые должен совершить дачник для перевозки всего урожая.

Входные данные находятся в файле GRATIS.IN. Первая строка файла содержит два целых числа – величины N и Q. Каждая из последующих N строк входного файла содержит одно целое число – величину P_i. Ограничения на данные: $1 \le N \le 100\,000$, $1 \le P_i, Q \le 10\,000$, $P_i \le Q$.

Выходные данные. В выходной файл GRATIS.OUT выведите искомое число поездок.

Пример входных и выходных данных

10 80 70 15 30 35 10 80 20 35 10 30	6

Задача H: Разность

Ограничения по времени: 2 секунды
Ограничения по памяти: 32 мебибайта

Целое число *A* содержит не более 1000 десятичных цифр. Обозначим через *B* и *C* соответственно максимальное и минимальное из чисел, составленных из цифр *A* так, что каждая цифра используется только один раз. Числа *B* и *C* должны иметь тот же знак, что и *A*, и их десятичная запись не должна содержать ведущих нулей. Найдите разность между *B* и *C*.

Входные данные находятся в текстовом файле DIFFER.IN. Единственная строка этого файла содержит десятичную запись числа *A*.

Выходные данные помещаются в текстовый файл DIFFER.OUT и содержат единственную строку с искомой разностью.

Примеры входных и выходных данных

418	693
-1032	2187
8814	7353

Второй чемпионат (2009 год)

Чемпионат состоялся 3 октября 2009 года. Набор задач чемпионата совпадает с набором задач студенческой олимпиады БГУ, проходившей в это же время. Второй командный чемпионат впервые получил статус

отборочного соревнования для Всероссийской командной олимпиады школьников по программированию (г. Санкт-Петербурге). По итогам чемпионата на ВКОШП было отобрано пять команд.
Автор задач – С.И. Кашкевич. Над задачами работали также Д.В. Веремейчик, П.А. Иржавский и В.В. Керус.

Задача A: Расчёт сопротивления цепи

Ограничения по времени: 3 секунды
Ограничения по памяти: 256 мебибайт

Последовательно-параллельное соединение (ППС) – одна из разновидностей электрической цепи, подключенной к источнику постоянного тока. Дадим его определение:

- одиночный резистор является ППС;
- два и более ППС, соединенных последовательно, являются ППС;
- два и более ППС, соединенных параллельно в одной паре точек, являются ППС.

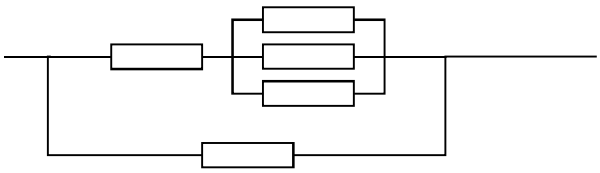

Рис. 1. Иллюстрация наличия ППС

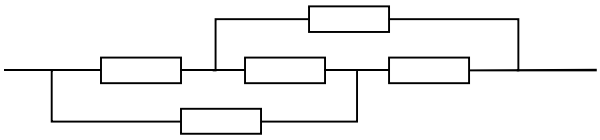

Рис.2. Иллюстрация отсутствия ППС

Таким образом, схема, изображенная на рисунке 1, является ППС, а схема с рисунка 2 – не является.

Задана электрическая цепь, являющаяся ППС, с M резисторами и N точками соединения, подключенная к источнику постоянного тока. Величины электрического сопротивления каждого резистора известны. Требуется рассчитать общее сопротивление всей цепи. При выполнении расчетов из всех физических эффектов достаточно принимать во внимание только закон Ома для цепи постоянного тока.

Входные данные находятся в текстовом файле PPS.IN. Первая строка этого файла содержит значения величин M и N ($1 \le M \le 10000$, $2 \le N \le 10000$). Далее следуют M строк, описывающий каждый резистор схемы и содержащие три целых числа – номера точек соединения, к которым подсоединен этот резистор (каждый резистор подсоединен к

различным точкам), и значение его электрического сопротивления (по-
(последнее число – натуральное, не превосходящее 100000). Точки соединения нумеруются, начиная с единицы. Вся схема подключена к источнику постоянного тока в точках 1 и *N*.

Выходные данные помещаются в текстовый файл PPS.OUT. В единственной строке этого файла записывается найденное сопротивление цепи, рассчитанное с точностью до 0.0001.

Пример входных и выходных данных

5 3 3 1 8 2 3 6 1 2 10 3 2 8 3 2 10	4.8861

Задача B: Проверка наличия ППС

Ограничения по времени: 3 секунды
Ограничения по памяти: 256 мебибайт

При подготовке материалов к предыдущей задаче оказалось, что построение правильных тестов – непростая работа. Авторам пришлось написать специальную программу для проверки корректности тестов. И сразу же возникла идея – предложить участникам соревнований написать такую же программу!

Входные данные находятся в текстовом файле PPS2.IN, формат которого совпадает с форматом входных данных предыдущей задачи, однако с другими ограничениями на M и N: $1 \le M \le 100000$, $2 \le N \le 100000$.

Выходные данные помещаются в текстовый файл PPS2.OUT. Первая строка этого файла должна содержать значение «Yes» или «No» (без кавычек) в зависимости от того, является ли предложенная схема ППС. В случае положительного ответа во второй строке файла записывается найденное сопротивление цепи, рассчитанное с точностью до 0.0001.

Примеры входных и выходных данных

5 3 3 1 8 2 3 6 1 2 10 3 2 8 3 2 10	Yes 4.8861
5 4 3 1 8 2 3 6 1 2 10 4 2 8 3 4 10	No

Задача C: Идите в баню!

Ограничения по времени: 1 секунда

Ограничения по памяти: 256 мебибайт

После многолетнего ремонта вот-вот откроется баня номер девять. Остались сущие мелочи… Одно из таких несделанных дел – закупить и прикрепить к шкафчикам для одежды таблички с номерами.

Администрация бани решила не заказывать отдельную табличку для каждого шкафчика, а купить таблички с изображением цифр и прикреплять к каждому шкафчику несколько табличек. Так, на шкафчик № 17 требуется прикрепить две таблички с цифрами 1 и 7.

Вам требуется определить, сколько табличек с каждой цифрой необходимо закупить. Имейте в виду, что в бане могут быть мужское и женское отделения, и нумерация шкафчиков в каждом отделении начинается с единицы!

Входные данные читаются из текстового файла BATH.IN. Единственная строка этого файла содержит два целых числа от 0 до 10^{18} – число мест в мужском и женском отделениях. Хотя бы одно из этих чисел должно быть положительным, а значение 0 для одной из величин соответствует бане из одного отделения.

Выходные данные должны быть помещены в текстовый файл BATH.OUT, содержащий единственную строку с записью десяти чисел. Первое число – количество табличек с изображением цифры 1, второе – с изображением цифры 2, наконец, последнее, десятое число – количество табличек с изображением цифры 0.

Пример входных и выходных данных

22 24	26 14 5 5 4 4 4 4 4 4

Задача D: Нетривиальные делители

Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Пусть задано натуральное число N. *Нетривиальным делителем* этого числа называется любое целое число M ($1 < M < N$) такое, что N делится на M без остатка.

По заданному числу N найдите количество его различных нетривиальных делителей.

Входные данные читаются из текстового файла DIVIDORS.IN. Единственная строка этого файла содержит величину N ($1 \le N \le 10^6$).

Выходные данные помещаются в текстовый файл DIVIDORS.OUT. Единственная строка этого файла должна содержать искомое количество различных нетривиальных делителей числа N.

Примеры входных и выходных данных

7	0
24	6

Задача E: Дрон на лестнице

Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Сценарий компьютерной игры – «стрелялки» предполагает, что игрок получит возможность управлять боевым роботом – дроном. Дрон может использоваться для решения различных задач, в частности, только с его помощью можно перейти на один из последующих уровней.

Дверь, за которой находится такой переход, открывается при попадании лазерного луча в датчик, расположенный на стене на высоте H. У дрона имеется лазер, установленный на высоте h относительно нижней части дрона. Луч, выпущенный горизонтально из этого лазера, заставит сработать датчик на стене.

На той же стене вертикально подвешена лестница длиной L, касающаяся пола. При касании дроном лестницы она начинает скользить так, что нижняя часть движется по полу перпендикулярно стене с

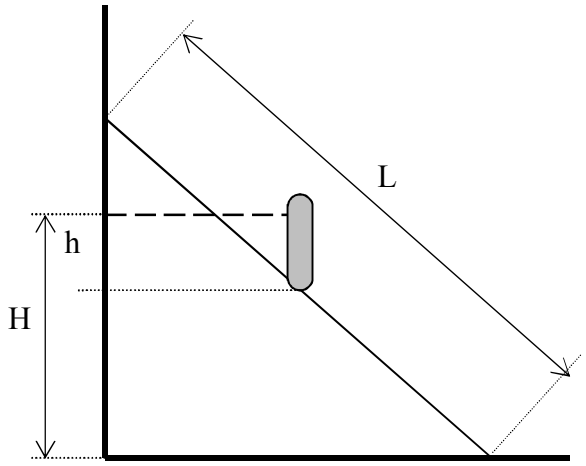

Рис. 3. Пояснение к условию задачи

постоянной скоростью *u*, а верхняя – касается стены. Дрон может двигаться по лестнице с постоянной скоростью *v*, независимо от наклона лестницы, и остается при этом в вертикальном положении. На рисунке 3 изображен момент попадания луча лазера в датчик открывания двери.

Вам требуется определить, можно ли перейти на следующий уровень при заданных параметрах *H, h, L, u, v*.

Входные данные читаются из текстового файла DRONE.IN. Единственная строка этого файла содержит записанные через один или несколько пробелов значения параметров *H, h, L, u, v*. Все числа – целые, $1 \le L, v \le 10000$, $0 \le H, h, u \le 10000$.

Выходные данные помещаются в текстовый файл DRONE.OUT. Первая строка этого файла должна содержать значение «Yes» или «No» (без кавычек) в зависимости от того, можно ли перейти на следующий уровень. В случае положительного ответа во второй строке файла записывается минимальное время от момента касания дроном лестницы до срабатывания лазера, рассчитанное с точностью до 0.0001

Примеры входных и выходных данных

21 20 100 20 20	Yes 0.0501
50 2 30 45 18	No

Задача F: Сортировка фамилий

Ограничения по времени: 2 секунды

Ограничения по памяти: 256 мебибайт

Фамилии зачастую начинаются с частиц, которые, например, указывают на аристократическое положение их владельца ☺ - д'Артаньян, фон дер Бек и т.д. Эти частицы не участвуют в определении того, с какой буквы начинается фамилия, и не должны учитываться при

сортировке фамилий по алфавиту. Так, первая из приведенных фамилий начинается на букву «А», вторая – на «Б».

Будем считать, что буква, с которой начинается фамилия (и позиция начала строки для сортировки) – первая прописная буква в записи фамилии. Таким образом, фамилия «`du Roys`» начинается с буквы «R», а фамилии «`Du Roys`» и «`Duroys`» – с буквы «D». Внутри сортируемой части фамилии могут быть пробелы, которые являются значимыми: фамилии «`Du Roys`» и «`Du  Roys`» считаются различными. Прописные и строчные буквы также различаются.

Отсортируйте по возрастанию список из N фамилий согласно приведенным выше правилам.

Входные данные находятся в текстовом файле NAMES.IN. Первая строка этого файла содержит значение величины N ($1 \le N \le 100000$) – количества сортируемых фамилий. Далее следуют N строк с фамилиями. Длина каждой фамилии не превосходит 100 символов, а в ее состав могут входить символы с кодами от 32 до 127. Среди этих символов есть хотя бы одна прописная латинская буква.

Выходные данные помещаются в текстовый файл NAMES.OUT, состоящий из N строк этого файла должны содержать отсортированный список фамилий. Если фамилии отличаются только начальными частицами, то они могут идти в произвольном порядке

Пример входных и выходных данных

`4`
`Atos`
`Portos`
`Aramis`
`d'Artagnan`
	`Aramis`
	`d'Artagnan`
	`Atos`
	`Portos`

Задача G: Касательные

Ограничения по времени: 5 секунд

Ограничения по памяти: 256 мебибайт

Прежде всего дадим несколько определений:

- две точки на плоскости считаются совпадающими, если расстояние между ними менее 10^{-6};
- отрезок считается невырожденным, если два его конца не совпадают;

- два невырожденных отрезка совпадают, если (возможно, после переупорядочивания концов одного из отрезков) их концы совпадают. Вырожденные отрезки всегда совпадают;
- две окружности совпадают, если их центры совпадают, а радиусы отличаются менее, чем на 10^{-6};
- невырожденный отрезок считается общей касательной для пары окружностей, если он лежит на прямой, касающейся обеих окружностей, а его концы лежат на этих окружностях.

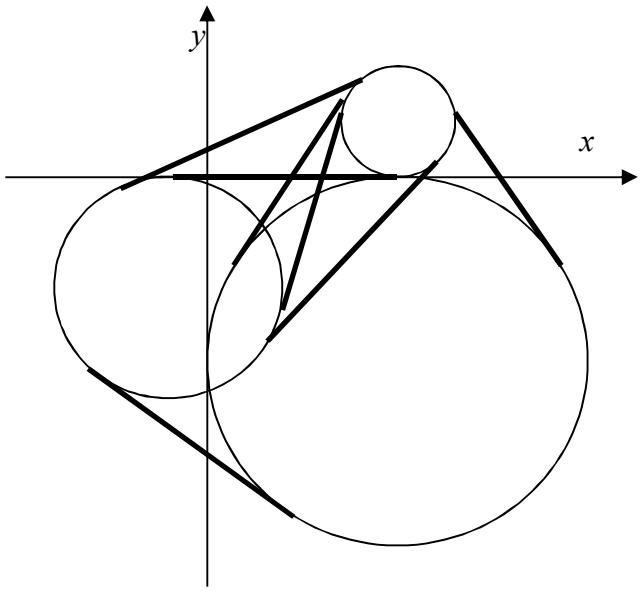

Рис. 4. Иллюстрация примера из условия задачи «Касательные»

На плоскости заданы N несовпадающих окружностей. Гарантируется, что:

- длина любой общей касательной будет больше 10^{-6};
- для любой пары несовпадающих общих касательных расстояние хотя бы от одного из концов хотя бы одной касательной до любого из концов другой касательной будет больше 10^{-6};
- не существует таких трех концов общих касательных, что первый совпадает со вторым, второй с третьим, но первый при этом не совпадает с третьим.

Определите, сколько несовпадающих отрезков являются общими касательными для хотя бы одной пары этих окружностей.

Входные данные находятся в текстовом файле TANGENT.IN. Первая строка этого файла содержит значение величины N ($2 \le N \le 70$). Далее следуют N строк, описывающих каждую окружность – координаты центра окружности (целые числа, не превосходящие по модулю 1000000) и ее радиус (целое положительное число, не превосходящее 1000000).

Выходные данные помещаются в текстовый файл TANGENT.OUT. Единственная строка этого файла должна содержать значение K – число найденных касательных.

Пример входных и выходных данных

3 -2 -6 6 5 -10 10 5 3 3	7

Задача H: Решить систему уравнений

Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Задана система уравнений

$$\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}$$

Требуется определить, сколько решений имеет эта система, и найти эти решения.

Входные данные читаются из текстового файла EQ_SYST.IN. Единственная строка этого файла содержит шесть целых чисел: коэффициенты a_1, b_1, c_1, a_2, b_2, c_2 (целые числа, не превосходящие по модулю 10000)

Выходные данные должны быть помещены в текстовый файл EQ_SYST.OUT, Первая строка этого файла должна содержать количество найденных решений: 0, 1 или -1 (в случае, если решений бесконечно много). Если найдено одно решение, вторая строка должна содержать найденные значения x и y, рассчитанные так, чтобы погрешность в вычислении каждого уравнения не превосходила 0.01. Если найдено бесконечно много решений, вторая и третья строки выходного файла должны содержать координаты любых двух решений системы; хотя бы одна из координат этих решений должна отличаться не менее чем на 1.

Примеры входных и выходных данных

1 2 3 2 4 7	0
1 2 3 2 4 6	-1 1.00000 1.00000 3.00000 0.00000
1 1 8 2 5 -4	1

	14.66667 -6.66667

Задача I: Книжная полка

Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

На книжной полке расставлены в один ряд N журналов. При этом на полке могут быть журналы с одним и тем же названием (но с разными номерами и годами выпуска). Читатель за один раз может снять с полки один или несколько журналов с одинаковым названием, стоящих подряд. После этого журналы на полке сдвигаются для того, чтобы они стояли плотно.

Определите минимальное количество таких операций, которое необходимо выполнить, чтобы снять с полки все журналы.

Входные данные находятся в текстовом файле SHELF.IN. Первая строка этого файла содержит значение величины N ($1 \le N \le 300$). Далее следуют N строк с названиями журналов, стоящих на полке. Порядок следования этих строк соответствует порядку расположения журналов на полке. Название каждого журнала не превосходит 100 символов и может содержать символы с кодами от 32 до 255.

Выходные данные помещаются в текстовый файл SHELF.OUT. Единственная строка этого файла должна содержать значение K – число операций по снятию журналов с полки.

Пример входных и выходных данных

10 Игромания Хакер Хакер Популярная механика Популярная механика Популярная механика Хакер Вокруг света Вокруг света Хакер	4

Задача J: Что больше?

Ограничения по времени: 1 секунда

Ограничения по памяти: 256 мебибайт

Заданы две непустые строки с десятичной записью неотрицательных действительных чисел. Как целая, так и дробная части содержат не более 100000 цифр и разделяются символом «точка». Дробная часть может отсутствовать, и точка в этом случае не записывается. Целая часть также может отсутствовать, и в этом случае точка стоит в начале строки. Дробные и целые части не могут отсутствовать одновременно. В начале и конце строки может быть записано произвольное количество нулей. Других символов, кроме описанных, в строках нет.

Определите, какое число больше…

Входные данные читаются из текстового файла COMPARE.IN. Две строки этого файла содержат два числа в формате, описанном выше.

Выходные данные помещаются в текстовый файл COMPARE.OUT. Единственная строка этого файла должна содержать значение -1, если первое из входных чисел меньше второго, 0, если эти числа равны, и 1, если первое число больше второго

Примеры входных и выходных данных

211.0000000000000001 211	1
15 00000000015.00000000	0
.15 00000000015.00000000	-1

Третий чемпионат (2010 год)

Чемпионат состоялся 2 октября 2010 года. Набор задач чемпионата совпадает с набором задач студенческой олимпиады БГУ, проходившей в это же время.

На Всероссийскую командную олимпиаду школьников по программированию, проходившую в Санкт-Петербурге, по итогам чемпионата было отобрано пять команд.

Авторы задач – С.И. Кашкевич (задачи A, D-I, K-L), П. А. Иржавский (задача B) , А.А. Толстиков (задачи C, J). Кроме них, в работе над задачами принимал участие В.В. Керус.

Задача A: Красно-чёрный путь

Имя входного файла: path.in
Имя выходного файла: path.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

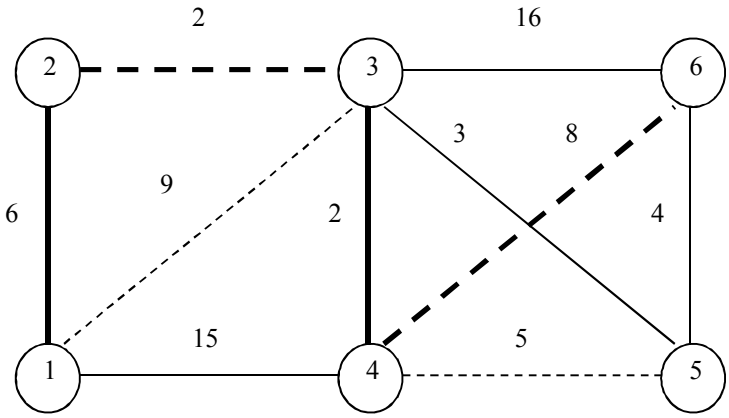

Рис. 5. Иллюстрация первого примера из условия задачи «Красно-чёрный путь».

Задан неориентированный граф с N вершинами и M рёбрами. Вершины пронумерованы, начиная с единицы. Каждое ребро характеризуется своим весом и цветом (красным либо чёрным). Вам необходимо найти путь с минимальным суммарным весом из вершины 1 в вершину N. При этом путь должен проходить вперемежку по красным и чёрным рёбрам. Это означает, что если первое ребро в найденном пути красное, то второе должно быть чёрным, третье – опять красным, и т.д. Если же первое ребро чёрное, то второе должно быть красным, и т.д.

Формат входных данных. Первая строка содержит величины N ($2 \le N \le 1000$) и M ($1 \le M \le 10000$). Каждая из последующих M строк содержит описание одного ребра и включает четыре числа: номера вершин, которое соединяет это ребро, вес ребра (неотрицательное целое число, не превосходящее 1000) и его цвет (0 – красный, 1 – чёрный).

Формат выходных данных. Единственная строка выходного файла должна содержать суммарный вес в найденном пути.

Примеры входных и выходных данных

6 10 1 2 6 1 2 3 2 0 1 3 9 0 1 4 15 1 3 4 2 1 3 5 3 1 3 6 16 1 4 5 5 0 4 6 8 0 5 6 4 1	18
6 10 1 2 6 1 2 3 2 1 1 3 9 1 1 4 15 1 3 4 2 1 3 5 3 1 3 6 16 1 4 5 5 1 4 6 8 1 5 6 4 1	-1

Пояснение: В первом примере, соответствующем рисунку 5, красные рёбра показаны пунктирной линией, а чёрные – сплошной. Найденный путь отмечен на рисунке жирными линиями. Во втором примере все рёбра одного цвета, поэтому путь построить нельзя.

Задача B: Перестановка

Имя входного файла: permut.in
Имя выходного файла: permut.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

По рзеульаттам илссеовадний одонго анлигйсокго унвиертисета, не иеемт занчнеия, в кокам пряокде рсапожолены бкувы в солве. Галвоне, чотбы преавя и пслоендяя бквуы блыи на мсете. Осатьлыне бкувы мгоут селдовтаь в плоонм бсепордяке, все рвано ткест чтаитсея без побрелм. Пичрионй эгото ялвятеся то,

что мы не чиатем кдаужю бкуву по отдьенотси, а все солво цли-цликеом.

Петя поделился с Васей программой, которую недавно написал забавы ради. Программа каким-то образом переставляет символы во входном файле. Вася запустил эту программу, подав ей на вход файл с очень важным документом. Но он не знал, что исходного файла у него не остается, программа просто записывает новые данные поверх старых. После некоторых экспериментов с другими файлами Вася заметил, что порядок, в котором переставляются символы, зависит только от длины входного файла. Но узнать этот порядок Вася не может. Может быть, его и не обязательно знать, чтобы восстановить исходные данные? Все, что Вася может сделать – запустить эту программу несколько раз. Документ не имеет определенной структуры, поэтому нельзя определить по файлу, восстановились данные или нет. Кроме того, свободного места на диске нет, поэтому сохранить копию текущего файла нельзя.

Какое наименьшее число запусков программы необходимо, чтобы восстановить исходный файл?

Формат входных данных. Целое число N ($1 \le N \le 100\ 000$) – длина исходного файла.

Формат выходных данных. Наименьшее количество запусков программы, позволяющее восстановить исходный файл. Если после любого числа запусков нет гарантии, что файл восстановится корректно, выведите "`Epic fail`" (без кавычек).

Примеры входных и выходных данных

1	0
2	1
3	5

Задача C: Странный многогранник

Имя входного файла: polyhedron.in
Имя выходного файла: polyhedron.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Вам дан выпуклый многогранник из N вершин, i-я вершина которого расположена в точке (x_i, y_i, z_i) ($1 \le x_i, y_i, z_i \le 100$). Вещество, из которого состоит многогранник, имеет довольно странную структуру – его плотность в точке (x, y, z) выражается формулой $\rho(x, y, z) = z$.

Вам необходимо найти массу этого многогранника.

Формат входных данных. Первая строка входного файла содержит число N ($4 \le N \le 100$), а каждая из последующих N строк – координаты i-й вершины многогранника (три целых положительных числа x_i, y_i, z_i). Вершины многогранника могут быть перечислены в произвольном порядке. Гарантируется, что все точки являются вершинами многогранника.

Формат выходных данных. Единственная строка выходного файла должна содержать массу многогранника с абсолютной погрешностью не более 10^{-3}.

Пример входных и выходных данных

8 1 1 1 1 2 1 1 1 2 2 1 1 2 2 1 1 2 2 2 1 2 2 2 2	1.5000

Задача D: Автомобильное ралли

Имя входного файла: rally.in
Имя выходного файла: rally.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Организаторы автомобильного ралли разместили на его маршруте N пунктов технического обслуживания, расположенных друг за другом. Стоимость обслуживания автомобиля в пункте с номером i ($1 \le i \le N$) равна A_i. Для уменьшения вероятности поломок на трассе в правила соревнований внесено следующее требование: каждый автомобиль, участвующий в соревновании, может проехать без обслуживания не более 3 пунктов подряд. При этом автомобиль должен пройти обслуживание хотя бы один раз.

Определите наименьшую сумму, которую должен потратить участник ралли на техническое обслуживание его автомобиля.

Формат входных данных. Первая строка входного файла содержит величину N ($1 \le N \le 1000$), а каждая из последующих N строк – величину A_i (положительное целое число, не превосходящее 10^9).

Формат выходных данных. Единственная строка выходного файла содержит искомое количество денег, которое надо потратить на техническое обслуживание автомобиля.

Пример входных и выходных данных

7 3 4 5 10 3 12 8	6

Пояснения. Техническое обслуживание необходимо выполнить в пунктах 1 и 5.

Задача E: Прямоугольник

Имя входного файла: rect.in
Имя выходного файла: rect.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

На координатной плоскости нарисован прямоугольник со сторонами, параллельными осям координат. Верхний левый угол прямоугольника имеет координаты (a, b), а нижний правый – координаты (c, d).

Определите количество точек с целочисленными координатами, которые находятся строго внутри прямоугольника.

Формат входных данных. Единственная строка входного файла содержит целые величины a, b, c, d, записанные через пробел ($-10000 \le a, b, c, d \le 10000$, $a < c$, $b > d$).

Формат выходных данных. Единственная строка выходного файла должна содержать искомое число точек, находящихся внутри прямоугольника.

Пример входных и выходных данных

-1 2 2 -1	4

Пояснение. Внутри прямоугольника находятся точки с координатами (0, 0), (0, 1), (1, 0), (1, 1).

Задача F: Дополнительная защита

Имя входного файла: shield.in
Имя выходного файла: shield.out
Ограничения по времени: 1 секунда
Ограничения по памяти: 256 мебибайт

Вася Пупкин на семинаре по криптографии предложил новый (как ему кажется) алгоритм шифрования текстовых файлов. В результате шифрования получается текстовый файл, соседние символы которого не совпадают. Иными словами, если в результате шифрования в выходной файл помещается строка S длиной в N символов, то $S[i] \neq S[i+1]$ для любых допустимых i.

Увы, стойкость предложенного алгоритма оказалась неудовлетворительной… Вася догадывается, что это происходит из-за того, что злоумышленник по структуре зашифрованного файла догадывается об алгоритме шифрования и далее действует целенаправленно. Поэтому он предложил выполнить дополнительную защиту зашифрованного файла, случайным образом «размножая» отдельные символы так, что в строке появляются несколько одинаковых символов, стоящих подряд. Естественно, перед дешифровкой повторяющиеся соседние символы необходимо убрать, оставив только один из них.

Вася считает себя выше этой рутинной работы и поручает её Вам…

Формат входных данных. Единственная строка входного файла содержит зашифрованный текст, подвергнутый дополнительной защите. В ней могут находиться только большие и маленькие латинские буквы, а также пробелы. Гарантируется, что файл заканчивается признаком конца строки. Длина строки не превосходит 200 символов.

Формат выходных данных. В выходной файл следует поместить строку входного файла со снятой дополнительной защитой. Строка должна завершаться признаком конца строки.

Примеры входных и выходных данных

committee↵	comite↵
↵	↵
↵	↵

Примечание: признак конца строки в примерах заменён символом ↵ .

Задача G: Стрельба с подвохом

Имя входного файла: shooting.in
Имя выходного файла: shooting.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Владелец небольшого тира предлагает посетителям поучаствовать в следующем соревновании. Участнику выдаётся N патронов на поражение такого же количества мишеней. По каждой мишени можно стрелять только один раз, причём в определённом порядке. За попадание в мишень с номером i ($1 \le i \le N$) заведение выплачивает стрелку сумму A_i, а за промах – взыскивает с него сумму B_i. Казалось бы, хороший стрелок всегда окажется в выигрыше, но…

Подвох обнаруживается, когда посетитель, согласившийся участвовать в соревновании, рассматривает выданные ему патроны. Оказывается, среди них K холостых, попасть которыми в мишень попросту нельзя. Отказ от участия влечёт крупный штраф, поэтому остаётся лишь минимизировать свои потери… Действительно, зная ценность каждой мишени, можно заранее выбрать, каким патроном стрелять по ней – боевым или холостым!

Предположим, что стрелок никогда не промахивается по мишеням, стреляя боевыми патронами. Определите его выигрыш или проигрыш по окончанию соревнования.

Формат входных данных. Первая строка содержит величины N и K ($1 \le K \le N \le 10000$), а каждая из последующих N строк – величины A_i и B_i (положительные целые числа, не превосходящие 10^6).

Формат выходных данных. Первая строка содержит результат соревнования (положительное число, если посетитель остался в выигрыше, отрицательное – если выиграло заведение). Вторая строка содержит N символов 1 или 0. Единица в i-й позиции означает, что для поражения мишени с номером i используется боевой патрон, ноль –

холостой. Если задача допускает несколько решений, выведите любое из них.

Пример входных и выходных данных

7 3 3 2 4 6 5 1 10 7 2 9 4 4 6 7	15 0101101

Задача H: Подняться на вершину

Имя входного файла: summit.in
Имя выходного файла: summit.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Два путешественника находятся на горизонтальной поверхности и одновременно собираются идти навстречу друг другу по прямой, не сворачивая в сторону от неё. Но между ними находится холм, который надо преодолеть… Сечение холма вертикальной плоскостью, проходящей через прямую, по которой идут путешественники, представляет собой треугольник с вершиной, обращенной вверх. Скорость движения пешеходов по горизонтальной поверхности постоянна (хотя и различна для разных людей), но при подъёме на холм она снижается в зависимости от крутизны дороги. Вам требуется определить, кто из путешественников поднимется на вершину холма первым.

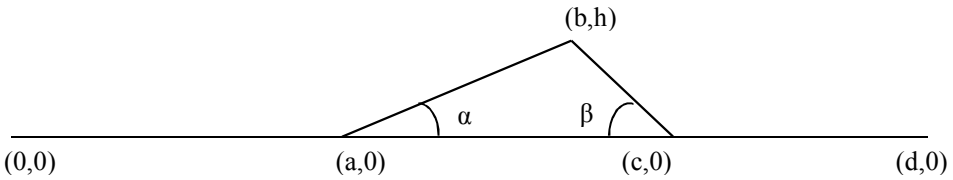

Рис. 6. Иллюстрация условия задачи «Подняться на вершину».

Формализуем постановку задачи (см. рисунок 6). Рассмотрим вертикальную плоскость, проходящую через линию движения путешественников. В момент начала движения первый путешественник находится в точке (0, 0), а второй – в точке (*d*, 0). Скорости их движения

равны соответственно v_1 и v_2. Координаты треугольника, который обра-образует сечение холма, равны соответственно (a, 0), (b, h) и (c, 0). Скорости пешеходов при подъёме на холм равны соответственно $v_1 \times \cos(\alpha)$ и $v_2 \times \cos(\beta)$.

Формат входных данных. Единственная строка содержит величины a, b, c, d, h, v_1, v_2, разделённые одним или несколькими пробелами ($0 \le a \le b \le c \le d \le 10000$; $a < c$; $0 < h, v_1, v_2 \le 10000$).

Формат выходных данных. Единственная строка содержит номер пешехода, первым добравшегося до вершины холма, или 0, если пешеходы дошли до неё одновременно.

Примеры входных и выходных данных

10 15 18 25 4 10 15	2
10 15 18 25 4 10 5	1

Задача I: Дешёвые поездки

Имя входного файла: tickets.in
Имя выходного файла: tickets.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

В некотором городе действует следующая система оплаты проезда в общественном транспорте. Для одной поездки в любом виде транспорта необходимо закомпостировать предварительно купленный талон. Талоны продаются как поодиночке (и в этом случае стоимость одного талона равна p_1), так и блоками по k штук (стоимость блока равна p_2).

Вам необходимо выполнить N поездок в транспорте этого города. Определите наименьшую сумму, которую Вы должны потратить на эти поездки. Предполагается, что ездить «зайцем» Вы не собираетесь… ☺

Формат входных данных. Единственная строка содержит величины N, k, p_1, p_2, разделённые одним или несколькими пробелами. Все величины – положительные целые числа, не превосходящие 10^9 (если $k = 1$, то $p_1 = p_2$).

Формат выходных данных. Искомое количество денег, которое надо затратить на все поездки.

Примеры входных и выходных данных

12 10 17 120	154
18 10 17 120	240

Пояснение. В первом примере Вам необходимо купить один блок и два отдельных талона. Во втором примере Вам необходимо купить два блока, а неиспользованные талоны выбросить.

Задача J: Две столицы

Имя входного файла: twocapitals.in
Имя выходного файла: twocapitals.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

В королевстве все хорошо. Все всем довольны, но… У короля два сына, и он не хочет делить между ними территорию. Тогда советники подсказали ему, что в стране просто нужно сделать две столицы, а мэрами назначить сыновей. Но как же выбрать столицы так, чтобы это было наиболее полезно для государства? А королевство-то у короля не простое, в нем между любой парой городов существует ровно один путь, их соединяющий. И тогда решил король выбрать столицы таким образом, чтобы максимальное расстояние от городов страны до ближайшей столицы было минимальным. Но решить этой задачи сам он не смог, да и не собирался, а поручил ее Вам, сказав, что у того, кто ее решит, шансов пройти дальше по карьерной лестнице будет намного больше…

Формат входных данных. В первой строке входного файла записано единственное число N – количество городов в государстве ($2 \le N \le 2222$). Далее в $N-1$ строке идут описания дорог в формате a, b, c ($1 \le a < b \le N$, $1 \le c \le 500\ 000$) – между городами a и b есть дорога протяженностью c.

Формат выходных данных. В первой строке выведите максимальное расстояние от одного из городов до ближайшей столицы (т.е. ту величину, которую необходимо минимизировать). Во второй строке через пробел выведите города, которые король может выбрать столицами. Если решений несколько, выведите любое.

Примеры входных и выходных данных

2 1 2 10	0 1 2

3 1 2 5 2 3 10	5 2 3

Задача K: Буква V

Имя входного файла: v_shape1.in
Имя выходного файла: v_shape1.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Будем говорить, что два отрезка ненулевой длины образуют букву V, если:

- они не параллельны;
- они пересекаются в одном из концов обоих отрезков;
- их длины одинаковы.

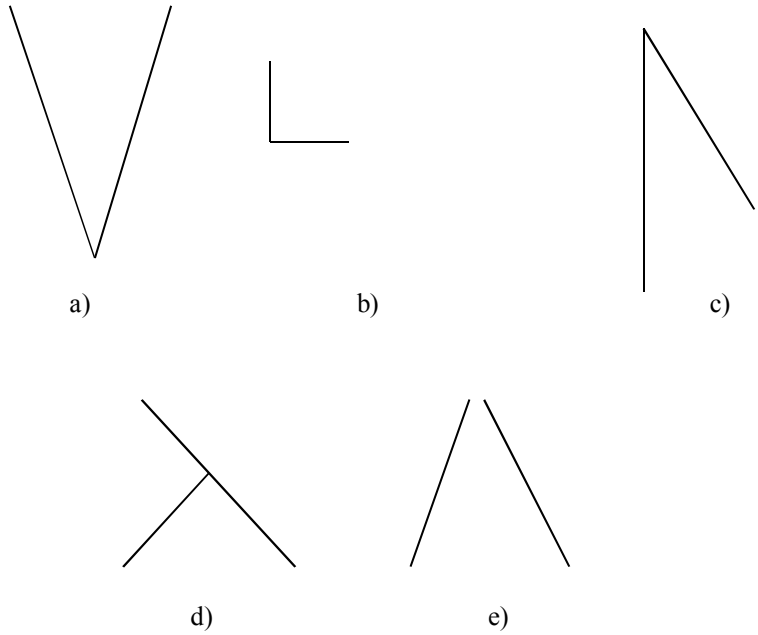

Рисунок 7. Примеры наличия и отсутствия буквы V.

Таким образом, отрезки a) – b) рисунка 7 образуют букву V, а отрезки на рис. c) – e) – не образуют.

На плоскости заданы N отрезков ненулевой длины, пронумерованных, начиная с единицы. Определите, сколько различных пар этих отрезков образуют букву V. Пары отрезков различаются, если в них входят отрезки с различными номерами.

Формат входных данных. Первая строка содержит величину N ($1 \le N \le 10000$). Далее следуют N строк с координатами концов каждого отрезка в формате "$x_1\ y_1\ x_2\ y_2$" (без кавычек). Все координаты – целые числа, не превосходящие по модулю 10^9.

Формат выходных данных. Искомое количество различных пар отрезков.

Пример входных и выходных данных

5 0 0 0 4 0 0 5 0 -4 0 0 0 0 4 0 0 0 4 0 0	3

Пояснение: букву V образуют пары отрезков (1, 3), (3, 4) , (3, 5).

Задача L: Буква V, часть вторая

Имя входного файла: v_shape2.in
Имя выходного файла: v_shape2.out
Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мебибайт

Решите предыдущую задачу в предположении, что число отрезков не превосходит 100000.

Входные и выходные данные имеют тот же формат, что и в предыдущей задаче.

Разбор задач

Первый чемпионат (2008 год)

Сейчас, по прошествии более чем пяти лет, можно сказать, что набор задач первого чемпионата получился, по сравнению с последующими соревнованиями, облегчённым. Причин тому много… Первая и главная – пресловутый «первый блин». При составлении набора задач совершенно неясно было, на что настраиваться. Вторая причина – «засвеченные» задачи, которые были знакомы многим участникам соревнования. Именно поэтому первой команде, решившей все задачи, понадобились чуть больше четырёх часов для решения всех задач этого соревнования. Увы, она заняла только второе место по штрафному времени.

Тем не менее, в одной части чемпионат удался – «никто не ушёл обиженным», каждая команда решила хотя бы одну задачу. Больше такого не было…

В соревновании приняло участие 19 команд.

Составители задач для командных соревнований обычно стараются включить в них задачи разного уровня сложности. Для команды составителей задач из БГУ используется следующая шкала уровней сложности: 1 – простая задача, которую должны решить все команды-участники, 5 – практически нерешаемая задача.

Конечно, оценки составителей субъективны и могут не совпадать с реальной сложностью, которая оценивается в количестве правильных решений. Тем не менее, корреляция между этими понятиями прослеживается, что показано в таблице 3. Подобного рода таблицы будут составляться для всех командных соревнований школьников.

Таблица 3

Задача	Оценка сложности	Количество решений
Наибольшая сумма цифр	1.0	15
Пересечение отрезков	3.0	3
Локальные экстремумы	1.0	18
Широко шагая	3.0	3
По-русски, как по-французски…	2.0	16
Химическая реакция	3.5	5
Прижимистый дачник	2.0	8
Разность	2.5	7

Задача A: Наибольшая сумма цифр

Одна из самых простых задач соревнования. Для её решения достаточно уметь выделять десятичные цифры числа, считать их сумму и находить элемент последовательности по сложному критерию, состоящему из проверки двух условий (максимальная сумма цифр, а в случае равенства – максимальное число).

Опишем алгоритм выделения десятичных цифр числа. Младшая цифра – это остаток от деления числа на 10. Для выделения оставшихся цифр достаточно разделить число на 10 нацело и продолжать процесс до тех пор, пока число не станет равным нулю. Например, для числа 175 младшая цифра равна 5. Далее, для числа 17 (результата деления на 10) младшая цифра – 7, и наконец, для числа 1 его единственная цифра также равна 1.

Задача B: Пересечение отрезков

Как правило, геометрические задачи вызывают чувство страха и неуверенности у школьников, решающих олимпиадные задачи по программированию. На самом деле, эта задача скорее относится к учебным. Умение определять, пересекаются ли два заданных отрезка, требуется при решении более сложных олимпиадных задач.

Перейдём к решению задачи…

Вначале при необходимости меняем местами координаты концов отрезков, чтобы первая точка находилась левее (а для вертикальных отрезков – ниже) второй точки.

После этого рассматриваем два случая. В первом случае отрезки лежат на одной прямой. Для этого достаточно определить, что точки a_2 и b_2 одновременно лежат на прямой, образуемой отрезком $[a_1, b_1]$. Если это так, то критерием того, что отрезки пересекаются, будет выполнение хотя бы одного из двух условий: точка a_1 лежит на отрезке $[a_2, b_2]$ или же точка a_2 лежит на отрезке $[a_1, b_1]$.

Во втором случае, когда отрезки не лежат на одной прямой, необходимо просчитать расстояния от концов одного отрезка до прямой, определяемой другим отрезком.

Пусть D_1 и D_2 – соответственно отклонения[1] от точек a_1 и b_1 до прямой, образуемой отрезком [a_2, b_2]. Вторую пару отклонений обозначим через D_3 и D_4. Теперь отрезки не пересекаются, если хотя бы одна пара отклонений (D_1 и D_2, либо D_3 и D_4) – ненулевые числа, имеющие одинаковый знак.

Заметим, что теперь нам нет необходимости разделять варианты, когда прямые, на которых лежат отрезки, пересекаются либо параллельны.

При выполнении вычислений следует заменять операцию деления операцией умножения, чтобы избежать потери точности. При этом вместо отклонений следует (для того, чтобы не извлекать квадратный корень) хранить их квадраты, не забывая, конечно, про знак этой величины.

Ограничения на координаты позволяют ограничиться типом int64 для задания координат и типом extended – для отклонений.

Задача C: Локальные экстремумы

Для решения этой задачи необходимо организовать цикл от второго до предпоследнего элемента последовательности и определять, выполняются ли для текущего элемента условия локального минимума либо локального максимума.

Случаи $N = 1$ или $N = 2$ можно рассмотреть отдельно, чтобы не организовывать ненужный цикл (хотя при его правильной организации никаких проблем не возникнет).

Задача D: Широко шагая

Задача решается с использованием механизма поиска в ширину. Заносим в очередь информацию о клетках, в которые мы можем попасть из текущей клетки, а также о номере шага, который должен быть выполнен после такого перехода. Впоследствии, при извлечении информации из очереди мы можем по номеру шага восстановить тип коня, который оказался в этой клетке. Заметим, что одна и та же клетка может быть включена в очередь несколько раз, поэтому размер очереди

[1] Отклонение численно равно расстоянию от точки до прямой, но может быть положительным либо отрицательным, в зависимости от того, в какой полуплоскости относительно прямой находится эта точка. В частности, если прямая определяется точками с координатами (x_1, y_1) и (x_2, y_2), то отклонение точки (x_0, y_0) от этой прямой равно $(x_0 - x_1) * (y_2 - y_1) - (y_0 - y_1) * (x_2 - x_1)$.

должен быть большим. Однако ограничения по памяти сделаны шикар-шикарными для того, чтобы организовать такую очередь (учитывая размер доски). Так, в авторском решении очередь состоит из 10000 элементов, что вполне достаточно для решения задачи.

В условии задачи имеется подвох. На самом деле, из любой клетки шахматной доски можно перейти в любую другую, так что ответ -1 никогда не должен получиться. Возможно, этот подвох усложнил задачу…

Задача E: По-русски, как по-французски…

Для решения этой задачи достаточно выделить очередное максимальное базовое число, не превосходящее N, и записать в выходную строку его словесное обозначение. После этого вычитаем из N это базовое число и продолжаем работу, пока $N > 0$. Не забываем ставить пробелы между словесными обозначениями соседних базовых чисел!

Задача является упрощенным вариантом задачи, предложенной автором для студенческого чемпионата Гродненского государственного университета – там требовалось по словесному обозначению числа восстановить его. Попытайтесь решить задачу и в таком варианте!

Задача F: Химическая реакция

Для решения этой задачи можно воспользоваться несколькими различными подходами. Одним из таких подходов является реализация конечного автомата, включающего следующие состояния:

1 – начальное состояние;
2 – ввели начальную букву названия химического элемента;
3 – ввели вторую букву названия химического элемента;
4 – вводим коэффициент при молекуле;
5 – закончили ввод молекулы;
6 – вводим количество атомов в составе молекулы;
7 – завершили ввод левой части формулы.

Разбор строки прекращается по её завершении, поэтому вводить конечное состояние не обязательно.

Символы, входящие в состав строки, делятся на пять групп:

1 – прописная латинская буква;
2 – строчная латинская буква;

3 – цифра;
4 – знак «плюс»;
5 – знак равенства.

Матрица переходов реализуемого конечного автомата будет иметь вид, показанный в таблице 4 (пустыми клетками отмечены запрещённые переходы, но поскольку формула предполагается верной, обрабатывать такие клетки не надо):

Таблица 4

Состояние	*Группа символов*				
	1	2	3	4	5
1	2		4		
2	2	3	6	5	7
3	2		6	5	7
4	2		4		
5	2		4		
6	2		6	5	7
7	2		4		

Расчёт количества атомов каждого химического элемента ведётся при переходе в состояния 2, 5 и 7, а также после завершения разбора входной строки.

Задача G: Прижимистый дачник

Для минимизации общего количества поездок Ивану Иванычу нужно максимизировать количество поездок, в которых ему удается взять сразу две корзины. Опишем один из способов достичь необходимой цели. Для этого отсортируем веса корзин по возрастанию. Теперь в каждую поездку Ивану Иванычу следует брать самую тяжёлую из оставшихся корзин. Если суммарный вес самой тяжёлой и самой лёгкой корзины не превосходит Q, то в поездку следует захватить и самую лёгкую корзину. Конечно, можно было бы искать и самую тяжелую корзину, которую можно взять второй в поездку, но для минимизации общего количества поездок этого делать не обязательно.

Для сортировки следует использовать эффективные алгоритмы, так как сортировка с трудоёмкостью $O(N^2)$ приведёт к нарушению лимита времени на тест.

Задача H: Разность

В условии задачи также имеется подвох: нам не надо рассматривать отдельно положительные и отрицательные исходные числа, поскольку искомая разность, во-первых, всегда неотрицательна, а во-вторых, её значение одинаково как для числа A, так и для числа $-A$.

Заводим два одинаковых массива из 10 элементов (обозначим их через S и T), каждый элемент этих массивов будет содержать количество повторений соответствующей цифры в числе A. Заполнение массивов можно вести одновременно с посимвольным чтением исходных данных, что позволяет не отводить память для хранения числа A.

Ситуацию с $A = 0$ следует рассмотреть отдельно, в этом случае, очевидно, результат также равен нулю.

Находим минимальную из цифр исходного числа из массива S, а также максимальную из цифр из массива T. Если в массиве T осталась только одна ненулевая цифра, а остальные – нули, вначале выбираем нули – это необходимо для реализации условия об отсутствии ведущих нулей у чисел B и C. Затем вычитаем (по правилам вычитания «столбиком», т.е. с учётом заёма) из минимальной цифры максимальную, получая младшую цифру результата. Из массивов S и T удаляем информацию об использованных цифрах.

Описанную процедуру повторяем N раз, где N – количество цифр исходного числа.

Второй чемпионат (2009 год)

Требование, выдвинутое организаторами ВКОШП для того, чтобы соревнование приобрело статус отборочного (задачи должны быть оригинальными) изменило подход к организации второго командного чемпионата. Было принято решение об использовании единого набора задач для школьного командного чемпионата по программированию и студенческой олимпиады БГУ. Это, естественно, привело к тому, что набор задач для школьников получился заметно усложнённым. В результате победители решили 6 задач из десяти (я считаю этот показатель невысоким), а шесть команд не смогли решить ни одной задачи.

В соревновании приняло участие 35 команд из заявившихся 39.

Как и ранее, в таблице 5 приводится оценка сложности задач, по мнению организаторов и по результатам соревнований.

Таблица 5

Задача	Оценка сложности	Количество решений
Расчёт сопротивления цепи	4.5	0
Проверка наличия ППС	5.0	0
Идите в баню!	3.0	0
Нетривиальные делители	1.0	28
Дрон на лестнице	3.0	2
Сортировка фамилий	2.0	11
Касательные	4.0	0
Решить систему уравнений	3.0	2
Книжная полка	4.0	1
Что больше?	2.0	13

Задача A: Расчёт сопротивления цепи; задача B: Проверка наличия ППС

Типичный пример т.н. «двойных» задач, объединённых общим условием. Двойные задачи могут предлагаться в двух случаях:

1) вторая задача является усложнённым вариантом первой. Тогда участники соревнования, решившие более сложную задачу, могут использовать свои наработки для более простой. Тем самым достигается своего рода компромисс между правилами IOI, предполагающими зачёт частичных решений, и гораздо более жёсткими правилами ACM. Именно этот мотив использовался в данном соревновании;

2) Совершенно разные задачи, объединённые лишь общим условием. Развитием этого подхода является использование одной идеи для условий всех задач соревнования, как это практикуется при подготовке задач для Открытого Кубка им. Е.В. Панкратьева по программированию. Однако авторы считают, что в этом случае тексты условий зачастую оказываются «притянутыми за уши»…

Итак, для решения обеих задач предлагается использовать единый подход:

а) Ищем все группы одиночных резисторов, параллельно соединённые в одной паре точек и заменяем каждую такую группу одним резистором с сопротивлением, вычисленным по закону Ома:

$$\frac{1}{R} = \sum_{i=1}^{t} \frac{1}{R_i}$$

b) Просматриваем все точки соединения, за исключением 1 и N, и ищем точки, в которых соединены ровно два одиночных резистора. Заменяем каждую такую пару резисторов одним с сопротивлением

$$R = R_1 + R_2$$

c) Выполняем последовательно шаги a) – b) до тех пор, пока схема не преобразуется в единственный резистор с рассчитанным сопротивлением, или пока не сможем выполнить ни одной такой замены. В последнем случае ППС нет.

На рисунке 8 показан пример разбора имеющегося соединения (первый пример задачи A). Тёмным цветом выделены резисторы, объединяемые на каждом шаге.

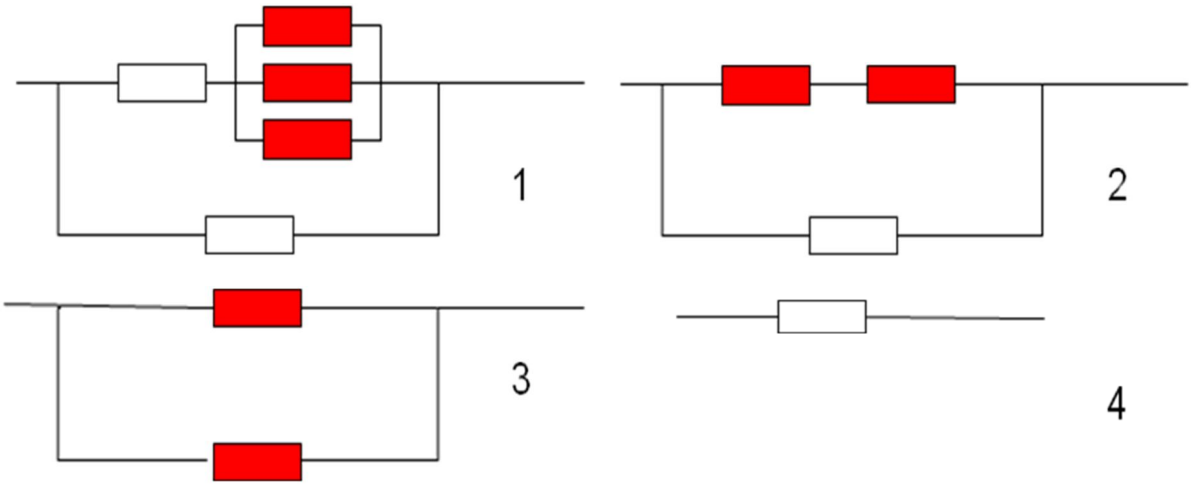

Рис. 8. Последовательный разбор ППС

При реализации этой задачи сложность представляет также выбор подходящей структуры данных для хранения заданной электрической схемы. В частности, автор использовал т.н. двойные списки смежности, когда каждый резистор (представляющий, по сути, ребро в графе, описывающем электрическую схему) включается в списки смежности для обеих вершин, которые он соединяет.

Задача C: Идите в баню!

Автор не ожидал, что эту задачу не решит никто из участников, и оценил её только на «троечку». Более того, рассматривался и вопрос о том, чтобы предложить её и на районную олимпиаду по информатике ☹.

Выполняем расчёт отдельно для каждого отделения бани, чтобы затем лишь просуммировать полученные количества табличек для каждой цифры. Пусть N – количество шкафчиков в текущем отделении, и это число имеет p десятичных цифр: d_p, d_{p-1}, …, d_1 (d_p – старшая цифра числа N).

Будем искать величину S(k, z), где $1 \le k \le p$, $0 \le z \le 9$ – количество цифр z, стоящих в k-й с конца позиции всех чисел из интервала [1; N]. Тогда, очевидно, общее количество табличек с цифрой z будет рассчитываться как сумма всех возможных величин S(k, z).

Будем вести цикл по k. Обозначим через A число, полученное из цифр d_p, …, d_{k+1}, а через P – число, полученное из цифр d_{k-1}, …, d_1 (для $k = 1$ будем полагать $P = 1$).

Если $d_k \neq 0$, то

$$S(k,z) = \begin{cases} (A+1)\cdot 10^{k-1} & для\ 1 \le z < d_k \\ A\cdot 10^{k-1} + P & для\ z = d_k \\ A\cdot 10^{k-1} & для\ z > d_k\ или\ z = 0 \end{cases}$$

Если же $d_k = 0$, формулы для расчёта S(k, z) принимают вид

$$S(k,z) = \begin{cases} A\cdot 10^{k-1} & для\ 1 \le z \le 9 \\ (A-1)\cdot 10^{k-1} + P & для\ z = 0\ и\ k \neq 1 \\ A\cdot 10^{k-1} & для\ z = 0\ и\ k = 1 \end{cases}$$

Читателям предлагается самостоятельно доказать правильность этих формул.

Задача D: Нетривиальные делители

Скорее учебная, чем олимпиадная задача… Неудивительно, что её решили почти все участники, показавшие ненулевой результат на чемпионате.

Напомним, что число k является делителем числа N, если N делится на k без остатка (иными словами, если $N \bmod k = 0$). Теперь нам достаточно организовать цикл от 2 до $N - 1$, чтобы исключить из рассмотрения делители тривиальные, и на каждой итерации накапливать счётчик делителей.

Случай $N \le 2$ отдельно рассматривать не стоит – при правильной организации цикла мы всё равно получим ответ, равный нулю.

Задача E: Дрон на лестнице

Прежде всего определим случаи, когда задача заведомо не имеет решения: $H < h$ или $H > h + L$. Эти случаи необходимо обработать отдельно.

Кроме того, при $H = h$ или $u = 0$ решение существует, и его можно найти «в одно действие»:

$$T = (H - h)/V$$

В остальных случаях одним из подходов к решению этой задачи является использование метода вычисления функции по сетке. Пусть F(t) – высота, на которую поднимется лазер дрона в момент времени t, где t находится в интервале

$$X = \left[0;\ \frac{\mathrm{L}}{\max(v,u)}\right]$$

Выведем формулу для вычисления F(t). Прежде всего определим величину угла φ наклона лестницы относительно угла пола в момент времени t. Как видно из рисунка 9, эта величина равна

$$\arccos\left(\frac{u * t}{L}\right)$$

Тогда

$$F(t) = h + v * t * \sin\varphi$$

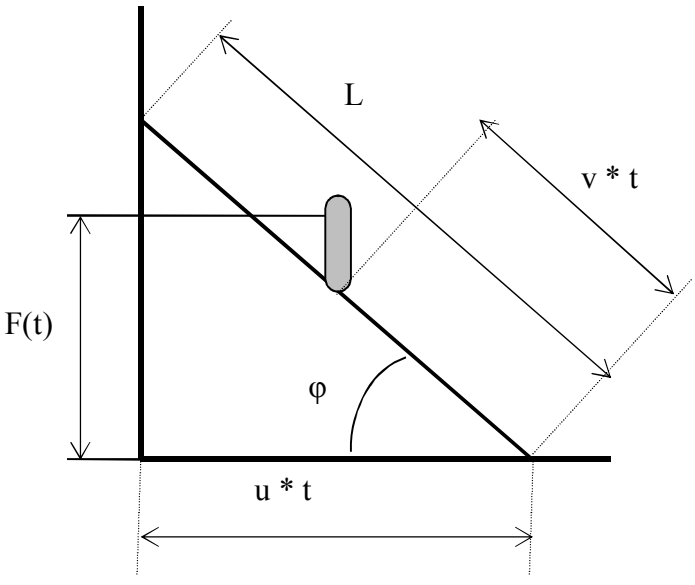

Рис. 9. Положение дрона в момент времени t

Функция F(t) вначале возрастает, а потом убывает. Нам необходимо найти первое значение этой функции, равное H, что позволяет использовать метод сеток.

Наложим на интервал X сетку из R равноотстоящих точек и вычислим значение F(t) в каждом из узлов этой сетки. Тогда возможны следующие случаи (в порядке перечисления их проверки):

а) $F(t_k) > H$, но $F(t_{k-1}) < H$ для некоторого k. Это означает, что решение задачи есть, и оно обязательно находится в интервале $[t_{k-1}, t_k]$.

Накладываем уменьшенную сетку на этот интервал и повторяем вычис-вычисления до тех пор, пока шаг сетки не станет меньше требуемой точности;

b) $F(t_k) \le F(t_{k-1})$ для некоторого k, но $F(t_{k-1}) > F(t_{k-2})$. Это означает (смотри рисунок 10), что решение если и есть, то только на интервалах $[t_{k-2}, t_{k-1}]$ либо $[t_{k-1}, t_k]$. Объединяем эти интервалы. Как и ранее, уменьшаем шаг сетки до тех пор, пока он не станет меньше требуемой точности. Однако, если мы так и не перешли к случаю a), то решения,

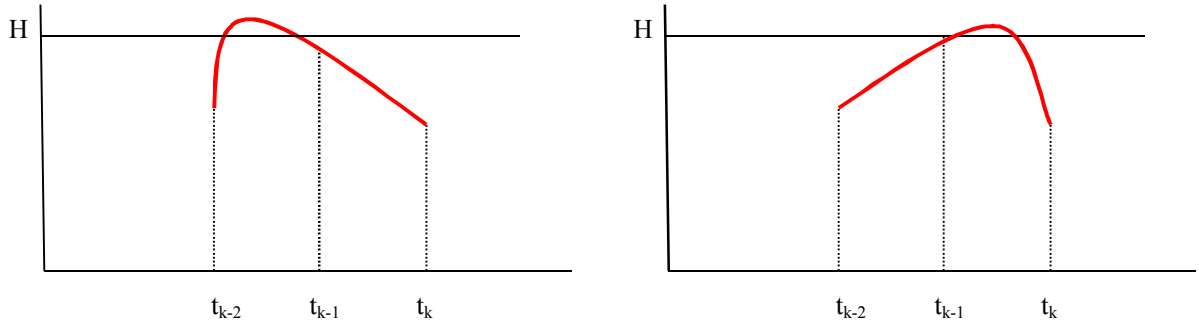

Рисунок 10. Варианты расположения максимума F(t) для случая b)

увы, нет...

c) $F(t_0) \le F(t_1)$ либо $F(t_{R-1}) < F(t_R)$. В этом случае решение продолжаем искать, как и в случае b), но только на первом и последнем отрезке сетки соответственно.

Задача F: Сортировка фамилий

Формируем массив входных данных, и в процессе этого формирования определяем позицию первой прописной латинской буквы в этом массиве. Затем сортируем полученный массив по подстрокам, начинающимся с найденной позиции. Для сортировки может применяться любой метод со скоростью $O(N \log N)$.

Для решения программы на языке C++ можно предложить написать свой функциональный класс для сравнения и воспользоваться алгоритмом sort из библиотеки STL.

Задача G: Касательные

Для каждой пары окружностей, в зависимости от их взаимного расположения, определяем число нужных касательных, которые относятся к этой паре (R_1, R_2 – радиусы окружностей, H – расстояние между центрами):

a) если окружности расположены одна внутри другой или касаются внутренним образом ($H \le \max(R_1, R_2)$), касательных нет;

b) если окружности касаются внешним образом или пересекаются ($\max(R_1, R_2) < H \le R_1 + R_2$), строим две касательных;

c) если окружности расположены одна вне другой и не пересекаются ($R_1 + R_2 < H$), строим четыре касательных.

Формируем массив построенных касательных, и каждый вновь построенный отрезок проверяем на наличие дубликатов в этом массиве.

Сложность алгоритма – $O(N^4)$.

Задача H: Решить систему уравнений

Для определения количества решений системы проще всего воспользоваться методом определителей. Находим определители

$$D_0 = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}, D_x = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}, D_y = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}$$

Если $D_0 = 0$, то система либо не имеет решений (при $D_x \neq 0$ или $D_y \neq 0$ или ($a_i = 0$ и $b_i = 0$ и $c_i \neq 0$)), либо имеет бесконечно много решений.

В последнем случае решения, удовлетворяющие условию задачи, находятся, например, при $x = 0$ и $x = 1$ (если $b_1 \neq 0$ или $b_2 \neq 0$) либо при $y = 0$ и $y = 1$ (если b_1 и b_2 равны нулю одновременно).

Если $D_0 \neq 0$, единственное решение находим по формулам

$$x = D_x / D_0 , y = D_y / D_0$$

Подводный камень: для того, чтобы погрешность вычисления каждого уравнения не превышала 0.01, требуется вывод значений x и y с гораздо большей точностью (до 10^{-8}).

Задача I: Книжная полка

Решение задачи основано на идеях динамического программирования.

Вначале избавляемся от нескольких журналов с одинаковым названием, стоящих подряд, заменяя их одним журналом. Пусть после этой операции осталось K журналов ($1 \le K \le N$).

Обозначим через R(a, b) количество операций по снятию с полки журналов, начиная с a-го и заканчивая b-м ($1 \le a \le b \le K$). Тогда ответом на задачу будет R(1, K).

Очевидно, R(a, a) = 1. Для других значений a и b составим рекуррентное соотношение.

Рассмотрим три случая.

В первом случае разобьём интервал (a, b) на две непустые части и будем снимать левую и правую часть журналов отдельно:

$$T_1(a,b) = \min_{a \le i < b} (R(a,i) + R(i+1,b))$$

Если на позициях a и b стоит один и тот же журнал, можно снять все журналы, находящиеся между ними, а затем за одну операцию убираем крайние журналы:

$$T_2(a,b) = R(a+1,b-1) + 1$$

Наконец, рассмотрим случай, когда на позициях a и b стоит один и тот же журнал, и внутри на позициях c_1, ..., c_p есть такой же журнал. В этом случае разбиваем интервал (a, b) на две части (a, c_i) и (c_i, b) и выполняем расчёт для каждого из этих участков, используя формулу для T_2. Не забываем уменьшить результат на единицу!

$$T_3(a,b) = \min_{i=1,\dots,p} (R(a,c_i) + R(c_i,b) - 1)$$

Тогда

$$R(a,b) = \min(T_1(a,b), T_2(a,b), T_3(a,b))$$

Задача J: Что больше?

В каждом числе выделяем две подстроки: с целой и дробной частью (одна из подстрок может быть пустой).

Избавляемся от начальных нулей в подстроке с целой частью и от конечных – в подстроках с дробной частью (при этом некоторые подстроки могут стать пустыми).

Сравниваем вначале подстроки с целой, а потом, в случае их равенства – подстроки с дробной частью. При этом подстроки с целой частью сравниваются по особым правилам:

- более длинная строка считается большей;
- строки одинаковой длины сравниваются лексикографически.

Строки с дробной частью сравниваются только лексикографически.

Третий чемпионат (2010 год)

Как и в 2009 году, для школьного командного чемпионата по программированию и студенческой олимпиады БГУ использовался единый набор задач. Количество задач было увеличено по сравнению с прошлым годом, что позволило сделать соревнование более интересным, чем предыдущее. Победители школьного чемпионата смогли решить 9 задач – прекрасный результат. С другой стороны, все 12 задач смогла решить только одна студенческая команда.

В соревновании приняло участие 39 команд из заявившихся 44. Увы, то, что организаторы принимали абсолютно все заявки, привело к появлению большого числа команд, которые оказались не подготовленными к соревнованиям в таком формате…

Как и ранее, в таблице 6 приводится оценка сложности задач, по мнению организаторов и по результатам соревнований.

Таблица 6

Задача	Оценка сложности	Количество решений
Красно-чёрный путь	3.67	6
Перестановка	4.00	0
Странный многогранник	5.00	0
Автомобильное ралли	2.33	15
Прямоугольник	1.00	30
Дополнительная защита	1.00	26
Стрельба с подвохом	2.33	7
Подняться на вершину	3.00	12
Дешёвые поездки	1.67	20
Две столицы	3.67	0
Буква V	2.33	8
Буква V, часть вторая	4.33	3

Задача A: Красно-чёрный путь

Для решения этой задачи можно использовать модифицированный алгоритм Дийкстры поиска кратчайшего пути в графе. Модификация заключается в следующем:

- для каждой вершины графа задаются пометки двух цветов – красные и чёрные;

• информация, заносимая в кучу, содержит, кроме информации о конечной вершине и длины пути к этой вершине, информацию о цвете ребра, по которому мы пришли в неё;

• при извлечении из кучи очередного элемента следует сделать постоянной только одну из двух пометок для соответствующей вершины;

• просмотр рёбер для выхода из текущей вершины предполагает фильтрацию их по цвету. Рассматриваются только рёбра, цвет которых не совпадает с цветом ребра, по которому мы пришли в текущую вершину;

• алгоритм завершает работу, когда постоянной стала любая из пометок для вершины N.

Корректность данного алгоритма можно строго доказать, построив другой граф из $2 \times N$ вершин, в котором задача о кратчайшем пути будет эквивалентной описанной выше модификации алгоритма Дийкстры. Построение такого графа авторы оставляют читателю в качестве упражнения.

Задача B: Перестановка

Случай $N = 1$ рассматривается отдельно – очевидно, в этом случае никакой порчи исходного текста не произойдёт. Поэтому последующие рассуждения будут вестись в предположении, что $N > 1$.

Формализуем постановку задачи. Задана неизвестная нам перестановка чисел от 1 до N P = (p_1, p_2, …, p_N). Операция S, применённая к другой, известной перестановке T = (t_1, t_2, …, t_N), перемещает i- й член перестановки T в позицию p_i. Требуется определить минимальное число последовательных выполнений операции S для исходной перестановки $T_0 = (1, 2, \ldots, N)$, после которых мы вновь гарантированно получим перестановку T_0. Для вывода ответа нам потребуется отнять единицу (ведь одна операция, испортившая исходный файл, уже была выполнена!).

Известно, что любую перестановку можно разложить на циклы. Предположим, что в разложении перестановки P содержится k циклов и их длины равны (l_1, l_2, …, l_k). Тогда минимальное число последовательных выполнений операции S для данной перестановки P будет равно $НОК(l_1, l_2, \ldots, l_k) - 1$. Данное соотношение следует из того, в каждом цикле длины L все становится на свои места каждые L применений операции S. Однако мы не знаем перестановки P, поэтому длины циклов могут быть любыми, но не превосходящими N.

Нетрудно видеть, что результат будет равен

$$HOK(2, 3, \dots, N) - 1$$

Для вычисления этого выражения можно использовать следующий алгоритм. Выделим все простые числа, не превосходящие N – числа a_1, a_2, …, a_m. Для каждого числа a_i найдём величину b_i – максимальное число, являющееся степенью a_i и не превосходящее N. Тогда

$$HOK(2, 3, \dots, N) = \prod_{i=1}^{m} b_i$$

Так, для $N = 6$ НОК(2, 3, …, 6) будет равен $2^2 \times 3 \times 5 = 60$. Интересно, что такой же результат получится и для $N = 5$…

Заметим, что для больших N результат будет очень большим (для $N = 100000$ он содержит больше 40000 десятичных цифр), поэтому для полного решения задачи требуется умение писать длинную арифметику с реализацией операции умножения.

Задача C: Странный многогранник

Задача, в первую очередь предназначена для студентов, понимающих термин «тройной интеграл» и умеющих рассчитывать такие интегралы для простейших фигур. Для школьников, очевидно, эта задача – неподъёмная… Именно поэтому разбор этой задачи будет дан схематично.

Масса тела, плотность которого в точке (x, y, z) численно равна z, задаётся интегралом

$$\int_x \int_y \int_z z \, dz \, dy \, dx$$

Сложность представляет вычисление пределов интегрирования. Здесь можно предложить следующий подход.

Разрежем фигуру на «блины» плоскостями, параллельными плоскости Oxy и проходящими через все вершины фигуры с различными координатами z (обозначим эти координаты через z_0, z_1, …, z_k). Тогда оказывается, что масса каждого «блина» может быть рассчитана с помощью уже не тройного, а одинарного интеграла

$$\int_{z_i}^{z_{i+1}} (a\tau^2 + b\tau + c)\tau \, d\tau$$

Здесь значение функционала $a\tau^2 + b\tau + c$ численно равно площади сечения многогранника плоскостью $z = \tau$. Для того, чтобы

показать, что численное значение площади сечения представляется квадратичной функцией, заметим, что вершины сечения, порождаемые ребрами многогранника, в одном «блине» не меняют своего набора и порядка и движутся по отрезку (т.е. по линейному закону). Также нужно отметить, если вершины треугольника движутся по линейному закону, то в силу того, что площадь можно вычислить через косое произведение векторов, то получим квадратичный закон изменения площади треугольника. Осталось вспомнить, что выпуклый многоугольник тривиально разбивается на треугольники.

Коэффициенты a, b, c, естественно, различны для различных «блинов». Для каждого «блина» они могут быть рассчитаны вычислением площади трёх различных сечений, например сечениями

$z = z_i$,

$z = (z_i + z_{i+1})/2$

$z = z_{i+1}$

и последующим решением системы линейных уравнений.

Задача D: Автомобильное ралли

Задача решается с использованием механизма динамического программирования. Пусть S_k – минимальная стоимость проезда первых k пунктов технического обслуживания в предположении, что последнее обслуживание обязательно выполняется в пункте k. Построим рекуррентное соотношение для вычисления S_k:

$$S_k = A_k, \quad при\ k \leq 4$$

$$S_k = \min(S_{k-1}, S_{k-2}, S_{k-3}) + A_k, \qquad при\ k > 4$$

Но, поскольку обслуживаться в трёх последних пунктах не обязательно, ответом будет выражение

$$\min(S_N, S_{N-1}, S_{N-2}, S_{N-3})$$

Конечно же, при $N < 4$ в результирующем выражении будет участвовать меньшее количество операндов!

Задача E: Прямоугольник

Задача, которая решается «в одно действие»: искомое количество точек равно

$$(c - a - 1) \times (b - d - 1)$$

Задача F: Дополнительная защита

Ещё одна простая задача (хоть и не настолько простая, как предыдущая). Она требует умения работать с двумя индексами, один из которых больше либо равен второму.

Пусть i – индекс последнего символа формируемой строки со снятой дополнительной защитой, j – индекс текущего символа исходной строки. Первоначально $i = 1$, $j = 2$. Тогда, если $S[i] \neq S[j]$, увеличиваем значение i и переносим $S[i]$ в $S[j]$ (эту операцию имеет смысл выполнять только тогда, когда $i < j$). Теперь можно переходить к следующему символу, увеличивая j.

Такой алгоритм выполняется за линейное относительно длины строки время и не требует дополнительной памяти.

Задача G: Стрельба с подвохом

Очевидно, что стрелять по мишени боевым патроном имеет смысл тогда, когда игрока ждёт либо крупный выигрыш за попадание, либо крупный штраф за промах. Обобщённым критерием, который позволяет определить, каким патроном стрелять по мишени с номером i, является величина $A_i + B_i$. Чем больше эта величина, тем выгоднее стрелять по соответствующей мишени боевым патроном.

Докажем это утверждение для случая двух мишеней и одного боевого патрона. Если игрок стреляет боевым патроном по первой мишени, то его выигрыш равен $A_1 - B_2$, в противном случае – $A_2 - B_1$. Сравним эти величины, и перенесём значения B_i в противоположную сторону. Получается, что стрелять боевым патроном по первой мишени выгоднее, если $A_2 + B_2 \leq A_1 + B_1$.

Итак, отсортируем мишени по убыванию величины $A_i + B_i$. После этого по первым K мишеням стреляем боевыми патронами, по остальным – холостыми.

В процессе сортировки нельзя терять информацию об исходном порядке мишеней – это необходимо для правильного вывода результата.

Задача H: Подняться на вершину

Относительно простая задача, в которой, тем не менее, содержится подвох. Для того, чтобы не потерять точность, мы должны избегать операций деления, заменяя их операциями с целочисленными

результатами. Кроме того, несмотря на небольшие ограничения на вход-входные данные, при вычислениях необходимо использовать тип int64. Возможно, именно эти «подводные камни» не позволили ни одной из команд сдать задачу с первого раза…

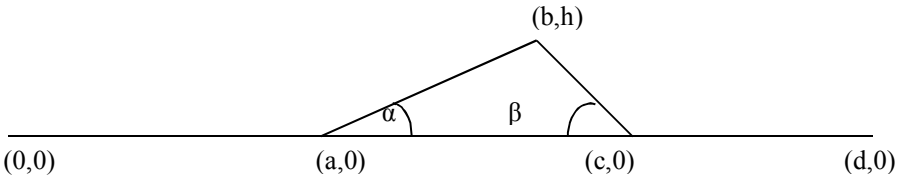

Рис. 11. Описание маршрута путешественников

Отдельно рассмотрим случаи, когда a = b или b = c (обозначения показаны на рисунке 11). Ответ в таких ситуациях очевиден.

Рассчитаем время подъёма на вершину первого путешественника.

$$\cos\alpha = \frac{b-a}{\sqrt{(b-a)^2+h^2}}$$

Время движения по горизонтали равно

$$\frac{a}{v_1}$$

Время движения на подъём равно

$$\frac{\sqrt{(b-a)^2+h^2}}{v_1 \cdot \cos\alpha} = \frac{(b-a)^2+h^2}{v_1 \cdot (b-a)}$$

Общее время движения первого путешественника равно

$$\frac{1}{v_1} \cdot \left(a + \frac{(b-a)^2+h^2}{(b-a)}\right)$$

Аналогично рассчитывается общее время движения второго путешественника:

$$\frac{1}{v_2} \cdot \left((d-c) + \frac{(c-b)^2+h^2}{(c-b)}\right)$$

Сравним эти два выражения и избавимся от операций деления, домножая обе части на положительные величины

$$\frac{1}{v_1} \cdot \left(a + \frac{(b-a)^2+h^2}{(b-a)}\right) > \frac{1}{v_2} \cdot \left((d-c) + \frac{(c-b)^2+h^2}{(c-b)}\right)$$

$$v_2 \cdot \left(\frac{a \cdot (b-a) + (b-a)^2+h^2}{(b-a)}\right)$$

$$> v_1 \cdot \left(\frac{(d-c) \cdot (c-b) + (c-b)^2+h^2}{(c-b)}\right)$$

$$v_2 \cdot (c-b) \cdot (a \cdot (b-a) + (b-a)^2 + h^2)$$
$$> v_1 \cdot (b-a) \cdot ((d-c) \cdot (c-b) + (c-b)^2 + h^2)$$

Последнее соотношение и требуется запрограммировать.

Задача I: Дешёвые поездки

Задача навеяна воспоминаниями от поездки автора в Париж, где в то время действовала именно такая система оплаты в общественном транспорте…

Отдельно рассмотрим случай, когда стоимость талона в блоке превосходит стоимость талона, проданного отдельно (такое не запрещено условиями задачи, хоть и маловероятно в реальной жизни). Ответ в этом случае, очевидно, будет равен $p_1 \times N$.

Пусть a и b – соответственно частное и остаток от целочисленного деления N на k. Тогда для выполнения $a \times k$ поездок следует покупать блоки и потратить на это $a \times p_2$. Для выполнения оставшихся b поездок следует сравнить величины p_2 и $b \times p_1$, выбирая меньшую из них.

Задача J: Две столицы

Фраза из условия «…между любой парой городов существует ровно один путь, их соединяющий» означает, что структура дорог королевства представляет собой взвешенное дерево.

Предположим, что столицы расположены в городах X и Y, тогда города, для которых каждая из столиц будет ближайшей, будут образовывать связные области, соединенные одним ребром (если какой-то город равноудален от столиц X и Y, то его можно отнести к любой из связных областей). Следовательно, удаляя в дереве любое ребро, мы разбиваем его на два несвязных поддерева. Тем самым мы разбиваем королевство на две части, в каждой из которых будем искать свою оптимальную столицу. Лучшее из разбиений и будет оптимальным выбором столиц для исходной задачи.

Переберём все рёбра дерева, поочерёдно удаляя каждое из них. Пусть $\boldsymbol{x}$ и $\boldsymbol{y}$ – вершины, которые соединяет очередное удаляемое ребро. Эти вершины выбираем в качестве корней соответствующих поддеревьев. Теперь для каждой вершины $\boldsymbol{z}$ рассчитываем наибольшее расстояние от неё до какого-нибудь листа в ее поддереве, используя поиск в глубину.

После этого вновь запускаем поиск в глубину для определения вершины, максимальное расстояние от которой до любой другой достижимой вершины минимально (для этого в процессе поиска в ширину при заходе в вершину z необходимо передавать длину наибольшего пути из z до вершин не из ее поддерева). Эта вершина и будет одной из столиц. Продолжаем перебор удаляемых рёбер для нахождения оптимальной пары столиц.

Рассмотрим работу этого алгоритма на примере, приведённом на рисунке 12.

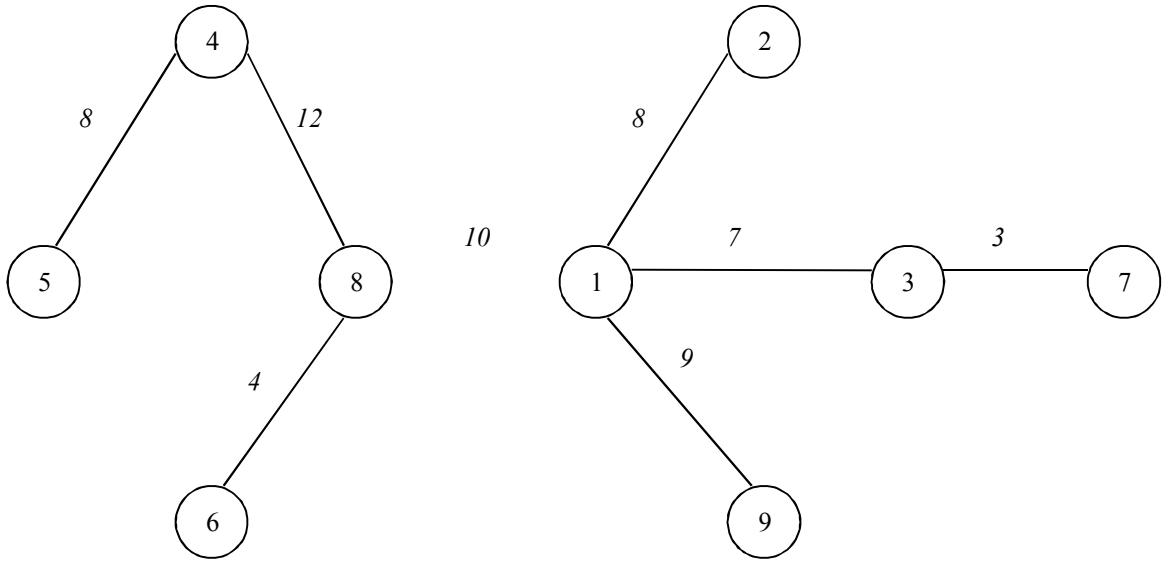

Рис. 12. Попытка удаления ребра (8, 1)

Пусть очередное удаляемое ребро соединяет вершины 1 и 8, как показано на рисунке 12. Рассчитываем наибольшие расстояния и занесём их в таблицу 7:

Таблица 7

Номер вершины	*Расстояние*
1	10
2	0
3	3

Номер вершины	*Расстояние*
4	8
5	0
6	0

Номер вершины	*Расстояние*
7	0
8	20
9	0

Изменение корня в правом поддереве не приводит к уменьшению максимального расстояния до листьев, а вот в левом поддереве имеет смысл перенести корень в вершину 4, при этом максимальное расстояние от корней до одного из листьев соответствующего поддерева уменьшится до 16.

Однако удаляемое в примере ребро не приводит к оптимальному решению, как видно из таблицы 8.

Таблица 8

Удаляемое ребро	*Вершины-столицы*	*Максимальное расстояние*
(1, 8)	1, 4	16
(1, 9)	8, 9	20
(1, 3)	8, 3	20
(1, 2)	8, 2	20
(3, 7)	8, 7	20
(4, 8)	**1, 4**	**14**
(4, 5)	8, 5	20
(6, 8)	6, 8	20

Оптимальное решение выделено полужирным шрифтом.

Задача K: Буква V

Ещё один пример «двойных» задач, где для решения второй из них требуется придумывать более эффективный алгоритм либо более подходящую структуру данных. Конечно же, такая программа решит обе задачи, однако необходимо дать возможность использовать и более простые подходы для решения только первой из них.

Каждую пару отрезков проверяем сначала на непараллельность, а затем – на то, чтобы они соприкасались концами, и, наконец – на равенство длин.

Такой подход работает с трудоёмкостью по времени $O(N^2)$.

Задача L: Буква V, часть вторая

Для N, достигающих 100000, предыдущий алгоритм будет слишком медленным. Надо искать алгоритм, решающий эту задачу с трудоёмкостью $O(N \log N)$…

Попытаемся каким-то образом установить отношение сравнения для вводимых отрезков. Тогда мы можем сначала упорядочить массив отрезков, а потом для каждого отрезка искать подходящие, используя, например, дихотомию или древесные структуры.

Один из способов упорядочивания предложен в решении П.А. Иржавского.

Будем считать, что для двух точек на плоскости a и b выполняется соотношение $a < b$, если $a.x < b.x$ или ($a.x = b.x$ и $a.y < b.y$). Иными словами, из двух точек та считается меньшей, которая лежит левее, а в случае расположения на одной вертикали – ниже другой.

Зададим похожий критерий упорядочения для отрезков. Прежде всего, вместо одного отрезка, описанного во входном файле, будем хранить информацию о паре противоположно направленных отрезков. Одна из точек таких направленных отрезков будет называться *начальной*, а вторая – *конечной*.

Отсортируем массив по возрастанию начальных точек, а в случае их равенства – по возрастанию длин. Тогда все отрезки с одинаковой начальной точкой и длиной (т.е. те отрезки, которые могут образовывать букву V) будут находиться в массиве рядом, т.е. образовывать *группу*, и такую группу можно определить, например, дихотомией. Выполняем такие действия для каждого элемента массива – отрезка P.

Однако это ещё не всё… Из полученных отрезков необходимо исключить те, которые лежат на одной прямой с отрезком P (по определению, они не могут образовывать букву V). Для этого можно воспользоваться следующим подходом.

Строим для каждого отрезка P вспомогательную точку q со следующими свойствами:

- её координаты целочисленные и неотрицательные;
- вектор q параллелен отрезку P;
- вектор q – кратчайший из всех векторов, удовлетворяющих предыдущим условиям.

Так, если отрезок P образован точками (2, 2) и (6, 8), то в качестве точки q берём точку (2, 3).

Массив направленных отрезков сортируем, как и ранее, по начальной точке и длине, а в случае совпадения этих величин – по возрастанию вспомогательной точки q. Теперь параллельные отрезки с одинаковыми начальной точкой, длиной и вспомогательной точкой также будут образовывать группу, которую опять-таки ищем с помощью дихотомии.

Содержание

Учебное издание

Кашкевич Сергей Иванович
Толстиков Алексей Александрович

Задачи школьных олимпиад по информатике

Учебно-методическое пособие для студентов факультета прикладной математики и информатики

В трёх частях

Часть 2

Задачи командных чемпионатов г. Минска по программированию (2008 – 2010 годы)

В авторской редакции

Ответственный за выпуск *С.И. Кашкевич*

Подписано в печать 26.09.2013. Формат 60×84/16. Бумага офсетная.
Усл. печ. л. 3,49. Уч.-изд. л. 3,22. Тираж 50 экз. Заказ

Белорусский государственный университет.
ЛИ № 02330/0494425 от 08.04.2009.
Пр. Независимости, 4, 220030, Минск.

Отпечатано с оригинала-макета заказчика
на копировально-множительной технике
факультета прикладной математики и информатики
Белорусского государственного университета.
Пр. Независимости, 4, 220030, Минск.