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

С.И. Кашкевич, В.В. Климович

ЗАДАЧИ ШКОЛЬНЫХ ОЛИМПИАД ПО ИНФОРМАТИКЕ

Учебно-методическое пособие для студентов факультета прикладной математики и информатики

В трёх частях

Часть 1

Задачи районных олимпиад по информатике г. Минска (2005 – 2012 годы)

МИНСК
2013

УДК 004(075.3)
ББК 32.81к2
К-31

Утверждено на заседании кафедры информационных систем управления
факультета прикладной математики и информатики
20 марта 2013 г., протокол № 10

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

Кашкевич С.И.

К-31 Задачи школьных олимпиад по информатике. В 3 ч. Ч. 1: Задачи районных олимпиад по информатике г. Минска (2005 – 2012 годы) / С. И. Кашкевич, В. В. Климович. – Минск : БГУ, 2013. – 58 с.

Пособие содержит условия, а также разборы решений задач школьных районных олимпиад по информатике г. Минска (2005 – 2012 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

Предисловие

Олимпиадное движение по информатике (т.н. *спортивное программирование*) приобрело в нашей стране большую популярность как среди школьников, так и среди студентов. Каждый год проводятся множество соревнований по спортивному программированию в различных форматах. Так, республиканская олимпиада по информатике среди школьников состоит из четырёх этапов: школьный, районный, областной и республиканский. Эта олимпиада проводится по правилам, принятым на международных олимпиадах по информатике (IOI).

Соревнования в формате IOI состоят из одного или нескольких туров (в настоящее время первый и второй этапы республиканской олимпиады проводятся в один тур, а третий и четвёртый – в два тура). На каждом туре участникам олимпиады предлагается для решения несколько задач (от трёх до пяти).

Решением классической задачи (а именно такие задачи представлены в предлагаемом пособии) является исходный текст программы на одном из языков программирования высокого уровня. Этот текст компилируется членами жюри и проверяется на наборе тестов, неизвестных участнику. Каждый тест из набора оценивается своим количеством баллов (чаще всего это количество одинаково для всех тестов набора).

Каждый тест из набора оценивается независимо от других, позволяет авторам объединить в одном условии несколько задач, указав различные ограничения для различных наборов тестов. Например, для 50 % тестов могут задаваться меньшие ограничения, которые позволяют использовать более простые (хотя и менее эффективные) алгоритмы. Участник олимпиады может, таким образом сдать не совсем эффективное решение, которое, тем не менее, наберёт ненулевое количество баллов.

Сборники олимпиадных задач прошлых лет являются серьёзным подспорьем при подготовке к будущим олимпиадам. Особенно это полезно тогда, когда условия задач сопровождаются авторским разбором, как это сделано в предлагаемом пособии.

В течение ряда лет авторы составляют задачи для олимпиад различного уровня. В пособии представлены задачи для районных школьных олимпиад по информатике (второй этап республиканской олимпиады), которые проводились в г. Минске в 2005 – 2012 годах. В него включена 31 задача; для каждой из них написан разбор решения. Наборы тестов для каждой задачи можно найти в интернете по адресу:

https://docs.google.com/file/d/0Bymq5SA5EB3NRFJUUmE1QjVRenc/edit?usp=sharing

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.

В дальнейшем планируется выпустить еще два сборника олимпиадных задач:

- задачи командных чемпионатов школьников г. Минска;
- задачи районных олимпиад Минской области.

Авторы благодарят А.Д. Субача, который внимательно прочёл рукопись и сделал 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.

Условия задач

2005/2006 учебный год

Автор задач «Период времени» и «Мат в один ход» – С.И. Кашкевич. Задача «Правые части» взята из хорватских интернет-олимпиад (http://hsin.hr/coci). Решения к задаче «Правые части» писал также С.В. Гафуров.

Задача 1: Период времени

Ограничения на время и память не задавались
Максимальное количество баллов: 60

Информация о дате и времени наступления какого-либо события, определенных с точностью до секунды, записывается в виде

```
dd.mm.yyyy hh:nn:ss
```

где `dd` – номер дня (от 01 до последнего дня месяца), `mm` – номер месяца (от 01 до 12), **yyyy** – год (от 1801 до 2100), `hh`, `nn`, `ss` – соответственно час (от 00 до 23), минута (от 00 до 59) и секунда (от 00 до 59) наступления события. Записи о дате и времени разделяются одним пробелом. Например, информация о событии, наступившем в полночь с 1 на 2 февраля 2005 года, будет записана как

```
02.02.2005 00:00:00
```

На основании двух корректных записей в формате, описанном выше, вам требуется определить, сколько секунд прошло между соответствующими событиями. Известно, что первое событие наступило не позже второго. Переходы с летнего времени на зимнее и обратно, а также от старого стиля к новому следует игнорировать. Для 50 % тестов обе записи будут относиться к одной и той же дате.

При решении задачи необходимо учитывать правила определения високосных лет: год считается високосным, если его номер делится на 4. Однако если номер года делится на 100, но не делится на 400, год не является високосным. Так, 2000 год – високосный, а 1900 и 2100 годы – нет.

Входные данные читаются из стандартного ввода и состоят из двух строк. Каждая строка содержит запись о наступлении одного события.

Выходные данные помещаются в стандартный вывод и содержат единственную строку с найденной величиной.

Пример входных и выходных данных

16.08.1980 23:12:40 17.08.1980 09:45:10	37950

Задача 2: Правые части

Ограничения на время и память не задавались
Максимальное количество баллов: 60

Правой частью натурального числа N назовем число K, полученное из двоичной записи числа N отбрасыванием всех цифр, стоящих левее крайней правой единицы, и преобразованное обратно в десятичную систему. Так, правой частью числа 6 является число 2, а правой частью числа 1088 – число 16.

Вам требуется определить сумму правых частей всех натуральных чисел из интервала $[A, B]$, включая границы этого интервала, где $1 \le A \le B \le 10^{15}$. Гарантируется, что результат будет помещаться в 64-битное целое число. Для 50 % тестов $1 \le A \le B \le 10^{8}$, а результат будет помещаться в 32-битное целое число.

Входные данные читаются из стандартного ввода и содержат два числа – значения A и B.

Выходные данные помещаются в стандартный вывод и содержат десятичное представление найденной суммы.

Пример входных и выходных данных

5 9	13

Задача 3: Мат в один ход

Ограничения на время и память не задавались
Максимальное количество баллов: 100

На квадратной шахматной доске размером $K \times K$ стоят три фигуры: белые король и ферзь, а также чёрный король. В начальной позиции ни один из королей не находится под боем. Следующий ход должны делать белые. Можете ли Вы, играя за белых, поставить этим ходом мат чёрным?

Следует заметить, что тесты этой задачи разбиты на группы, и участник олимпиады получает ненулевые баллы за группу тестов только в случае, когда все тесты этой группы пройдены!

Напомним шахматные правила, знание которых необходимо для решения задачи. В шахматы играют два игрока («белые» и «чёрные»), которые делают ходы по очереди. Ход заключается в перемещении одной из фигур своего цвета на новое место, не совпадающее с первоначальным. На каждой клетке доски в любой момент может находиться не более одной фигуры.

В частности, ферзь может двигаться по свободным клеткам вдоль горизонтали, вертикали или диагоналей и остановиться на любой из этих клеток. Ферзь может также снять неприятельскую фигуру, стоящую на его пути, и стать на ее место. Однако ферзь не может перепрыгивать через занятые клетки.

Король может перейти на одну из свободных соседних клеток, расположенных по вертикали, горизонтали или диагоналям (за исключением клеток, *находящихся под боем*). Король может также снять неприятельскую фигуру, стоящую на одной из соседних клеток, встав на ее место (если только после такого хода король не будет *находиться под боем*).

С точки зрения одного из игроков, его фигура *находится под боем*, если его соперник своим следующим ходом может снять эту фигуру. Свободная клетка находится под боем, если соперник своим следующим ходом может занять эту клетку одной из своих фигур.

Игрок получает мат (и проигрывает партию), если при его очереди хода король этого игрока находится под боем, и у него нет ходов, выводящих своего короля из-под боя.

Входные данные находятся в текстовом файле CHESS.IN, описывающем одну группу тестов и состоящем из нескольких строк. Первая строка файла содержит величины N ($1 \le N \le 5$) – количество тестов в группе, и K ($3 \le K \le 1\ 000\ 000$, для 50 % тестов $K = 8$). Каждая из последующих N строк описывает один тест группы и содержит положение белого короля, белого ферзя и черного короля, разделенные одним пробелом. Положения фигур задаются двумя числами, также разделенными пробелом: номерами вертикали и горизонтали (от 1 до K). Вертикали нумеруются слева направо, а горизонтали – снизу вверх.

Выходные данные помещаются в текстовый файл CHESS.OUT и содержат N строк, по одной строке на каждый тест группы. Если мат в один ход поставить нельзя, соответствующая строка содержит текст

'impossible'. В противном случае Вы должны указать один из возможных ходов белых в формате: обозначение фигуры (*R* – король, *Q* – ферзь) и номера вертикали и горизонтали, на которую эта фигура переходит. Данные в строках выходного файла должны разделяться ровно одним пробелом.

Пример входных и выходных данных

2 8 1 5 2 2 4 5 3 3 6 5 3 1	impossible Q 6 1

2006/2007 учебный год

*Автор задач «Игра», «Пустые прямоугольники» и «Поиск анаграмм» – С.И. Кашкевич. Идея задачи «Поиск анаграмм» родилась после чтения условий задач международной олимпиады по информатике, проходившей в Мексике. Задача «Постулат Бертрана» взята из интернет – олимпиад для школьников, проводящихся в Санкт-Петербурге (*http://neerc.ifmo.ru/school/io*). Решения к задачам писал также С.В. Гафуров.*

Задача 1: Игра

Ограничения на время и память не задавались
Максимальное количество баллов: 100

Вася и Петя играют в следующую игру на прямоугольной доске, имеющей *M* горизонтальных и *N* вертикальных рядов. В начале игры в левой нижней клетке доски стоит одна фишка. Игроки делают ходы по очереди, причем первым ходит Вася. Во время хода игрок может либо передвинуть фишку на одну, две или три клетки вверх, либо перенести ее на одну или две клетки вправо. После хода фишка должна оставаться на доске! Игрок, поставивший фишку в правую верхнюю клетку доски, проигрывает. Вам требуется определить, кто из детей выиграет, в зависимости от размеров доски, если в процессе игры никто из них не будет совершать ошибок.

Ограничения на данные: $1 \le M, N \le 1\,000\,000$; $M+N > 1$. Для 50 % тестов $1 \le M, N \le 10\,000$.

Входные данные находятся в текстовом файле GAME.IN, описывающем одну группу тестов и состоящем из нескольких строк. Первая строка файла содержит величину K ($2 \le K \le 5$) – количество тестов в

группе. Каждая из последующих *K* строк описывает один тест группы и содержит значения *M* и *N*, разделенные одним пробелом.

Выходные данные помещаются в текстовый файл GAME.OUT и содержат *K* чисел, записанных в одной строку через пробел. Каждое число соответствует одному тесту группы и равно 1, если победить должен Вася, и 2 – если победителем должен быть Петя. Пробелы в начале и конце строки не допускаются.

Пример входных и выходных данных

2 5 3 6 4	1 2

Примечание: в группе, соответствующей входному файлу, есть хотя бы один тест, когда побеждает Вася, и хотя бы один тест, когда побеждает Петя. Участник олимпиады получает ненулевые баллы за группу тестов только в случае, когда все тесты этой группы пройдены.

Задача 2: Постулат Бертрана

Ограничения на время и память не задавались
Максимальное количество баллов: 60

Постулат Бертрана (теорема Бертрана-Чебышёва, теорема Чебышёва) гласит, что для любого $n \geq 2$ найдётся простое число p в интервале $n < p < 2n$. Такая гипотеза была выдвинута в 1845 году французским математиком Жозе Бертраном (проверившим её до $n = 3000000$) и доказана в 1850 Пафнутием Чебышёвым. Раманужан в 1920 году нашёл более простое доказательство, а Эрдёш в 1932 – ещё более простое.

Вы должны решить несколько более общую задачу - а именно по числу n найти количество простых чисел p из интервала $n < p < 2n$.

Напомним, что натуральное число называется простым, если оно делится только само на себя и на единицу. Единица не считается простым числом.

Входные данные читаются из стандартного ввода и содержат значение числа n ($2 \leq n \leq 1\ 000\ 000$; для 50 % тестов $n \leq 5000$).

Выходные данные помещаются в стандартный вывод и содержат искомое количество простых чисел.

Примеры входных и выходных данных

2	1
4	2
8	2

Задача 3: Пустые прямоугольники

Ограничения на время и память не задавались
Максимальное количество баллов: 80

Задано множество из N несовпадающих точек на плоскости ($2 \le N \le 1000$). Выберем любую пару точек из этого множества с координатами ($x1$, $y1$) и ($x2$, $y2$). Если $x1 \ne x2$ и $y1 \ne y2$, то для этой пары точек можно построить прямоугольник со сторонами, параллельными осям координат, так что выбранные точки будут находиться в противоположных углах прямоугольника.

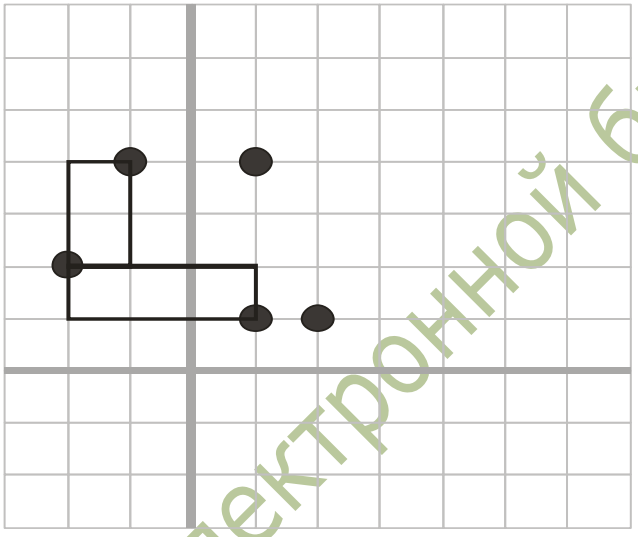

Рис. 1. Иллюстрация примера из условия (пустые прямоугольники выделены)

Прямоугольник, построенный описанным выше образом, назовем *пустым*, если внутри него и на его границе нет других заданных точек. Вам необходимо определить, сколько различных пустых прямоугольников можно построить из заданного множества точек.

Входные данные находятся в текстовом файле WIDERECT.IN, состоящем из нескольких строк. Первая строка файла содержит величину ***N***. В каждой из последующих ***N*** строк через пробел задаются координаты X и Y одной точки множества – целые числа, не превосходящие по модулю 1000000 (в 50 % тестов координаты не превосходят по модулю 10000).

Выходные данные помещаются в текстовый файл WIDERECT.OUT и содержат единственную строку с найденным числом пустых прямоугольников.

Пример входных и выходных данных

5 -1 4 1 4 1 1 2 1 -2 2	2

Задача 4: Поиск анаграмм

Ограничения на время и память не задавались
Максимальное количество баллов: 60

Две непустые строки одинаковой длины называются *анаграммами* друг друга, если вторая строка составлена из символов первой, и каждый символ используется только один раз. Так, пары строк «апельсин» и «спаниель», «дереза» и «резеда» являются анаграммами, а пары строк «каток» и «откат», «тропа» и «пират» – не являются.

Вы должны ввести две непустых строки и определить, являются ли они анаграммами друг друга. Строки содержат только символы латинского алфавита, причем прописные и строчные буквы считаются различными.

Входные данные находятся в текстовом файле ANAGRAM.IN, описывающем одну группу тестов и состоящем из нескольких строк. Первая строка файла содержит величину K ($2 \le K \le 5$) – количество тестов в группе. Далее следуют K пар строк – каждая пара соответствует одному тесту. Длина одной строки не превосходит 3000 символов (в 50 % групп тестов эта длина не превосходит 200).

Выходные данные помещаются в текстовый файл ANAGRAM.OUT и содержат единственную строку из K чисел, разделенных одним пробелом. Каждое число соответствует одному тесту и должно быть равно 1, если введенные строки являются анаграммами, и 0 в противном случае. Пробелы в начале и конце строки не допускаются.

Пример входных и выходных данных

4 Acad cAda AbRa arBA duda adua termo metro	1 0 0 1

Примечание: в группе, соответствующей входному файлу, есть хотя бы один тест, правильный ответ на который – 1, и хотя бы один тест с правильным ответом 0. Участник олимпиады получает ненулевые баллы за группу тестов только в случае, когда все тесты этой группы пройдены.

2007/2008 учебный год

Автор задач – С.И. Кашкевич. Идея задачи «Экспрессные маршруты» не нова, в той или иной формулировке её можно найти на любом сайте, где разбираются задачи по основам динамического программирования. Решения к задачам писали также И.Г. Филипович и В.С. Кашкевич.

Задача 1: Экспрессные маршруты

Ограничения на время и память не задавались

Максимальное количество баллов: 100

Между городом *A* и городом *B* проложена единственная дорога, на которой построено *N* остановочных пунктов. Обычный автобусный маршрут из *A* в *B* предусматривает остановки на каждом из оборудованных пунктов. Экспрессный маршрут пропускает некоторые (не менее одного) остановочных пунктов, но ни один экспрессный маршрут не пропускает более двух пунктов подряд.

Сколько различных экспрессных маршрутов можно организовать между городом *A* и городом *B*? Два маршрута считаются различными, если множества остановочных пунктов, которые они пропускают, различны.

Входные данные читаются из стандартного ввода и содержат значение числа N ($1 \le N \le 70$; для 50 % тестов $N \le 20$).

Выходные данные помещаются в стандартный вывод и содержат искомое количество экспрессных маршрутов.

Примеры входных и выходных данных

1	1
2	3
4	12

Задача 2: Отрезки на прямой

Ограничения по времени: 3 секунды
Ограничения по памяти: 32 мегабайта
Максимальное количество баллов: 80

На числовой прямой задано N отрезков $[a_1; b_1]$, …, $[a_N; b_N]$, где $a_i \le b_i$. Объединение этих отрезков образует K новых отрезков $[c_1; d_1]$, …, $[c_K; d_K]$, где $1 \le K \le N$.

Определите величину K и длину самого большого отрезка $[c_i; d_i]$

Входные данные читаются из текстового файла SEGMENT.IN. Первая строка этого файла содержит величину N ($1 \le N \le 1\ 000\ 000$; для 50 % тестов $N \le 5000$). Каждая из последующих N строк содержит величины a_i и b_i, разделенные одним или несколькими пробелами. Эти числа – вещественные, не превосходят по модулю 100000 и содержат не более 5 цифр в дробной части.

Выходные данные помещаются в файл SEGMENT.OUT. Единственная строка этого файла содержит два искомых числа, разделенные одним или несколькими пробелами.

Пример входных и выходных данных

5 4 4.5 8 11.2 -3 2 0.00001 0.00001 3 9.1	2 8.2

Задача 3: Проблема Гольдбаха

Ограничения на время и память не задавались

Максимальное количество баллов: 80

Проблема Гольдбаха — одна из самых старых проблем математики, не разрешённых до сих пор. В 1742 году прусский математик Кристиан Гольдбах послал Леонарду Эйлеру письмо, в котором было высказано предположение: *любое чётное число, большее двух, можно представить в виде суммы двух простых чисел*.

Напомним, что простое число – это натуральное число, большее единицы, имеющее ровно два натуральных делителя: единицу и само себя.

К настоящему времени это утверждение ни доказано, ни опровергнуто, хотя на март 2004 года известно, что оно выполняется для всех чётных чисел, не превышающих $2 \cdot 10^{17}$.

Вам необходимо представить заданное чётное число N, большее двух, в виде суммы двух простых чисел a_1 и a_2. При этом a_1 должно быть минимальным из возможных чисел, и $a_1 \le a_2$.

Входные данные читаются из стандартного ввода и содержат значение числа N ($4 \le N \le 100\,000$; для 50 % тестов $N \le 5000$).

Выходные данные помещаются в стандартный вывод и содержат строку из двух чисел a_1 и a_2, разделенных одним или несколькими пробелами.

Примеры входных и выходных данных

4	2 2
10	3 7
32	3 29

Задача 4: Изменить Строку!

Ограничения по времени: 1 секунда

Ограничения по памяти: 16 мегабайт

Максимальное количество баллов: 40

Строка S состоит из нескольких (не менее одного) *слов*, разделенных одним или несколькими пробелами. Все другие символы считаются частями слова. В начале и конце строки также могут быть пробелы.

Вам необходимо во всех словах, начинающихся с маленькой латинской буквы, заменить первый символ этих слов на соответствующую прописную букву. Остальные символы строки должны остаться без изменений.

Входные данные находятся в текстовом файле CAPITAL.IN. Единственная строка этого файла содержит исходную строку *S*. Длина этой строки не превосходит 30000 (в 50 % тестов эта длина не превосходит 255).

Выходные данные помещаются в текстовый файл CAPITAL.OUT и содержат преобразованную строку. Длины исходной и преобразованной строк должны совпадать.

В приведенных далее примерах пробелы для наглядности заменены символами '_' (подчеркивание).

Примеры входных и выходных данных (каждая строка соответствует одному примеру)

Wish_you_were_here ___Hello,world!_ shine_on_you_crazy_diamond_(part_1) __Превед,_медвед!
Wish_You_Were_Here ___Hello,world!_ Shine_On_You_Crazy_Diamond_(part_1) __Превед,_медвед!

2008/2009 учебный год

Автор задач – С.И. Кашкевич. Решения к задачам писал также С.В. Гафуров.

Задача 1: Тарабарская грамота

Ограничения по времени: 5 секунд
Ограничения по памяти: 16 мегабайт
Максимальное количество баллов: 60

Тарабарская грамота, или простая литорея – один из способов шифрования текстов, применявшийся в древней Руси. Суть его, приме-

нительно к современному русскому алфавиту, заключается в следующем. Поставив согласные буквы в два ряда, в порядке:

б в г д ж з к л м н

щ ш ч ц х ф т с р п

употребляют в письме верхние буквы вместо нижних и наоборот, причем гласные остаются без перемены; так, например, лсошамь = словарь.

Применим этот же принцип к латинскому алфавиту и расставим согласные буквы в два ряда:

b c d f g h j k l m

z x w v t s r q p n

Вам требуется зашифровать текст, содержащий любые символы с кодами от 32 до 127 (других символов в тексте нет). Заменяться должны только согласные латинские буквы, причем прописные буквы заменяются прописными, а строчные – строчными. Остальные символы остаются без изменений.

Входные данные читаются из текстового файла CIPHER.IN, содержащего исходный текст. Этот файл состоит из нескольких строк, причем длина одной строки не превышает 30000 символов (в 50 % тестов эта величина не превосходит 255 символов), а общий объем файла не превосходит 5 мегабайт.

Выходные данные. Зашифрованный текст должен быть помещен в текстовый файл CIPHER.OUT. Количество строк и размер каждой строки выходного файла должны соответствовать данным входного файла. Если исходный файл пуст, Вы также должны создать пустой файл. В противном случае последняя строка выходного файла должна заканчиваться символами перевода строки.

Пример входных и выходных данных

Deep Purple ”Who Do We Think We Are”	Weel Lujlpe ”Dso Wo De Gsimq De Aje”

Задача 2: Сглаживание матриц

Ограничения по времени: 6 секунд

Ограничения по памяти: 32 мегабайт

Максимальное количество баллов: 100

Соседями элемента двумерной матрицы назовем элементы, имеющие с ним общую сторону или угол. Элемент матрицы называется ло-

кальным максимумом, если он строго больше всех своих соседей. Операция сглаживания матрицы заменяет каждый локальный максимум матрицы средним арифметическим его соседей. Другие элементы матрицы не изменяются.

Выполните операцию сглаживания для заданной матрицы!

Входные данные читаются из текстового файла MATRIX.IN. Первая строка этого файла содержит числа M и N – количество строк и столбцов матрицы ($1 \le M, N \le 5000$, $1 < M \times N \le 500000$). Далее следуют M строк, каждая из которых содержит N действительных чисел и соответствует одной строке матрицы. Элементы матрицы не превосходят по модулю 100000 и записываются не более чем с тремя цифрами в дробной части. В 70 % тестов размеры матрицы не превосходят 100.

Выходные данные должны быть помещены в текстовый файл MATRIX.OUT, содержащий полученную матрицу в том же формате, что и входной файл. Значения элементов матрицы должны быть выведены с точностью до 10^{-6}.

Пример входных и выходных данных

```
4 5
1 12.483 5 8 4
0 -10 2 9 9
3 11.001 5 7 1
2 10 9 8 9
```

```
1.000000 -0.400000 5.000000 8.000000 4.000000
0.000000 -10.000000 2.000000 9.000000 9.000000
3.000000 2.625000 5.000000 7.000000 1.000000
2.000000 10.000000 9.000000 8.000000 5.333333
```

Задача 3: Где-то недостаток, а где-то – избыток...

Ограничения по времени: 3 секунды

Ограничения по памяти: 16 мегабайт

Максимальное количество баллов: 40

Вам, наверное, известно определение совершенного числа (*perfect number*): совершенное число - натуральное число, равное сумме всех своих собственных делителей (т. е. всех положительных делителей, отличных от самого́ числа). Числа, не являющиеся совершенными, делятся на две категории: недостаточные и избыточные числа. Избыточное число (*abundant number*) – положительное целое число n, сумма положитель-

ных делителей (отличных от n) которого превышает n. Аналогично определяется недостаточное число (*deficient number*). Единица относится к недостаточным числам.

Число 48, например, является избыточным, поскольку 1 + 2 + 3 + 4 + 6 + 8 + 12 + 16 + 24 = 76, 76 > 48. Число 45 – недостаточное, поскольку 1 + 3 + 5 + 9 + 15 = 33, 33 < 45. Число 28 – совершенное, так как 1 + 2 + 4 + 7 + 14 = 28.

Вам требуется определить, к какой категории относится каждый элемент заданной последовательности натуральных чисел.

Входные данные читаются из текстового файла DPA.IN. Первая строка этого файла содержит величину K – количество чисел в последовательности ($1 \le K \le 500$). Каждая из последующих K строк соответствует одному элементу последовательности и содержит натуральное число, не превосходящее 10^8. Сумма элементов последовательности также не превышает 10^8. В 50 % входных файлов $K \le 250$, а числа не превосходят 50000.

Выходные данные должны быть помещены в текстовый файл DPA.OUT, содержащий единственную строку из K символов. i - й символ ($1 \le i \le K$) этой строки должен быть равен прописной латинской букве `'D'`, `'P'` либо `'A'` в зависимости от того, является ли соответствующий элемент последовательности недостаточным, совершенным или избыточным числом.

Пример входных и выходных данных

3 28 45 48	PDA

Задача 4: Черепаха

Ограничения по времени: 1 секунда
Ограничения по памяти: 16 мегабайт
Максимальное количество баллов: 80

Тропический остров представляет собой круг радиуса R и состоит из песчаного пляжа и заросшего травой луга. Луг также представляет собой круг радиуса P, и центры этих двух кругов совпадают.

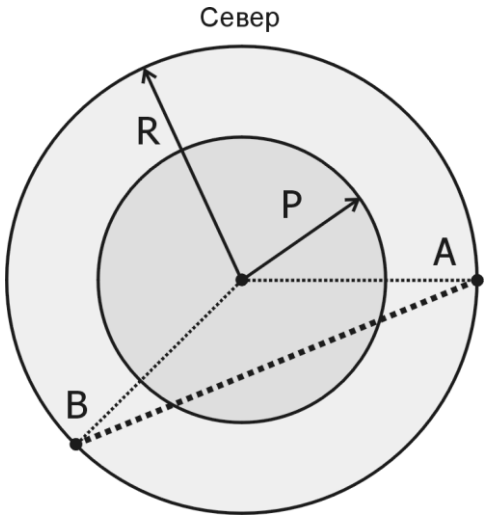

Рис. 2. Иллюстрация примера из условия задачи

Морская черепаха выползла на остров для кладки яиц в точке A, расположенной на берегу. Место для кладки яиц также расположено на берегу острова в точке B. Черепаха может ползти только по прямой, причем скорость ее движения по песку равна V, а по траве – U.

Определите время, за которое черепаха доберется из точки A в точку B.

Входные данные находятся в текстовом файле TURTLE.IN. Первая строка этого файла содержит четыре целых числа – величины R, P, V, U ($0 \le P < R \le 1000$; $1 \le V, U \le 100$). Во второй строке записываются азимуты точек A и B, рассчитанные относительно центра острова (целые числа от 0 до 359). Напомним, что азимутом называется угол между направлением на север и направлением на соответствующую точку. Азимут измеряется в градусах и отсчитывается по ходу часовой стрелки.

Выходные данные помещаются в текстовый файл TURTLE.OUT и содержат единственное число – искомое время, рассчитанное с точностью до 10^{-4}.

Пример входных и выходных данных

10 6 20 17 90 225	1.0054

2009/2010 учебный год

Автор задач – С.И. Кашкевич. Решения к задачам писал также А.М. Статкевич.

Задача 1: Литорея

Ограничения по времени: 1 секунда
Ограничения по памяти: 16 мегабайт
Максимальное количество баллов: 100

Литорея – система тайнописи, применявшаяся в древнерусских текстах. Известно несколько разновидностей литореи, и мы рассмотрим одну из них применительно к текстам из латинских букв.

Текст шифруется с помощью ключа, представляющего собой слово из строчных латинских букв. В шифруемом тексте заменяются только латинские буквы, остальные символы остаются неизменными. Латинские буквы разделяются на блоки так, что длина всех блоков (кроме, может быть, последнего) равна длине ключа. Пусть a_1 – номер первой буквы блока в латинском алфавите, b_1 – номер первой буквы ключа. Тогда первая буква блока заменяется буквой, чей номер в алфавите равен a_1+b_1 (если $a_1+b_1 > 26$, то берется величина a_1+b_1-26). При этом прописная буква заменяется на прописную, а строчная – на строчную. Вторая и последующие буквы блоков шифруются с помощью соответствующих букв ключа аналогичным образом.

Пусть шифруемое слово – `"Crusader"`, а ключом является слово `"bow"`. Тогда первая буква заменяется буквой `'E'` (номер первой буквы в латинском алфавите – 3, а номер первой буквы ключа – 2; следовательно, она заменяется буквой с номером 5). Аналогично, вторая буква заменяется буквой `'g'` (18+15=33, 33-26=7). Зашифрованный текст, таким образом, выглядит как `"Egrupagg"`. Обратите внимание на то, что в этом примере одинаковые буквы заменяются одинаковыми, однако это случайное совпадение – если расстояние между одинаковыми буквами не кратно длине ключа, такого не произойдет!

Выполните шифрование заданного текста методом литореи.

Входные данные читаются из текстового файла LITTERA.IN, содержащего хотя бы одну строку. Первая, непустая строка этого файла содержит ключ шифрования, а остальные строки - шифруемый текст (символы с кодами от 32 до 255). Длина одной строки не превышает 30000 символов (в 50 % тестов эта величина не превосходит 255 символов), а общий объем файла не превосходит 5 мегабайт. При переходе на следующую строку шифрование продолжается с позиции ключа, следующей за той, с помощью которой было выполнено шифрование предыдущей части текста (как показано во втором примере).

Выходные данные. Зашифрованный текст должен быть помещен в текстовый файл LITTERA.OUT. Количество строк и размер каждой строки выходного файла должны соответствовать шифруемым данным входного файла, за одним исключением: последняя строка выходного файла должна заканчиваться символами перевода строки, даже если это-

го нет во входном файле. Если исходный файл содержит только строку с ключом, выходной файл должен быть пустым (иметь длину 0 байт).

Примеры входных и выходных данных

bow Crusader	Egrupagg
rock Deep Purple ”Who Do We Think We Are”	Vtha Hjuadt "Zsg Sr Hw Iktfz Zp Sgh"

Задача 2: Две окружности

Ограничения по времени: 1 секунда
Ограничения по памяти: 16 мегабайт
Максимальное количество баллов: 40

На плоскости расположены две окружности, которые заданы координатами своего центра и радиусами. Определите, сколько общих точек имеют эти две окружности. Точки считаются совпадающими, если расстояние между ними не превосходит 10^{-6}.

Входные данные читаются из текстового файла CIRCLES.IN, состоящего из двух строк. Каждая строка этого файла соответствует одной окружности и содержит три числа – координаты центра окружности и её радиус. Все числа – целые, не превосходящие по модулю 10^8 (в 50 % тестов эти числа не превосходят 10^5). Радиусы окружностей больше нуля.

Выходные данные должны быть помещены в текстовый файл CIRCLES.OUT, единственная строка которого содержит искомое количество общих точек двух окружностей. Если окружности совпадают, выведите в качестве результата –1.

Примеры входных и выходных данных

0 0 10 0 0 20	0
0 5 9 0 4 10	1
0 5 9 0 5 9	-1

Задача 3: Квадраты цифр числа

Ограничения по времени: 0.5 секунд
Ограничения по памяти: 16 мегабайт
Максимальное количество баллов: 100

Задано натуральное число N. Заменим это число суммой квадратов его цифр, и последовательно выполним K таких замен. Какое число получится в результате этих операций?

Входные данные находятся в текстовом файле QUAD.IN, единственная строка этого файла содержит значения величин N и K ($1 \le N, K \le 10^9$, в 50 % тестов эти величины не превосходят 50000).

Выходные данные помещаются в текстовый файл QUAD.OUT. Единственная строка этого файла должна содержать результат вычислений.

Примеры входных и выходных данных

5 3	85
912 4	1

Задача 4: «Жизнь»

Ограничения по времени: 3 секунды
Ограничения по памяти: 16 мегабайт
Максимальное количество баллов: 100

«Жизнь» – компьютерная игра, придуманная английским математиком Джоном Конвеем (John Horton Conway). Впервые описание этой игры было опубликовано в октябрьском выпуске (1970) журнала Scientific American, в рубрике «Математические игры» Мартина Гарднера (Martin Gardner). Рассмотрим одну из разновидностей этой игры.

Место действия игры – *«вселенная»* – размеченный на единичные квадраты (*клетки*) прямоугольник размером M строк и N столбцов. Каждая клетка может находиться в одном из двух состояний: быть живой или мёртвой. Клетка имеет, в зависимости от ее расположения, от трех до восьми *соседей* (т.е. клеток, имеющих с ней общую сторону или угол). Распределение живых клеток в начале игры называется *первым поколением*. Каждое следующее поколение рассчитывается на основе предыдущего по следующим правилам:

- вначале наступает фаза рождения. Каждая пустая (мёртвая) клетка, рядом с которой есть ровно три живые клетки-соседки, оживает;
- затем наступает фаза смерти. Если у живой клетки есть две или три живые соседки, то эта клетка продолжает жить; в противном случае (если соседей меньше двух или больше трёх) клетка умирает от «одиночества» или «перенаселённости». Клетки, ожившие на фазе рождения этого же поколения, не учитываются при расчёте количества живых соседок.

По карте вселенной с первым поколением клеток определите карту с *P*-м поколением.

Входные данные находятся в текстовом файле LIFE.IN. Первая строка этого файла содержит три целых числа – величины *M*, *N*, *P* ($1 \le M, N \le 100$, $1 \le P \le 1000$, в 50 % тестов эта величина не превосходят 100). Далее следуют *M* строк по *N* символов каждая – описание первого поколения. Символ `'*'` (звёздочка) в этом описании соответствует живой, а символ `'.'` (точка) – мёртвой клетке.

Выходные данные помещаются в текстовый файл LIFE.OUT и содержат описание *P*-го поколения в том же формате, что и во входном файле. Последняя строка файла должна заканчиваться символами перевода строки.

Примеры входных и выходных данных

10 9 4 ....**...... .**...... ....*. ....*. ....*. ....*.... ....*.... .....*...	....**...... .**...... ....*** ....

10 10 10 ....*..... ...***....	....***.... ....*.....*.. .*.....*.. .*.....*.. ....***.... ....

2010/2011 учебный год

Авторы задач – С.И. Кашкевич и А.А. Толстиков. Идея задачи «Гири и весы» принадлежит Ю.Г. Стёпину.

Имена входных файлов для всех задач: input.txt

Имя выходных файлов для всех задач: output.txt

Задача 1: Перевернуть строку!

Ограничения по времени: 1 секунда

Ограничения по памяти: 16 мегабайт

Максимальное количество баллов: 50

Задана непустая строка S и два целых числа A и B ($1 \le A \le B$, B не превосходит длины строки). Вам требуется «перевернуть» часть строки, поменяв местами символы с номерами A и B, A+1 и B-1, …, $A+n$ и $B-n$ до тех пор, пока $A+n < B-n$.

Формат входных данных. Входной файл состоит из двух строк. Первая строка содержит значения A и B, вторая – строку S. Длина строки не превосходит 30000 (в 50% тестов длина не превосходит 255).

Формат выходных данных. Единственная строка выходного файла должна содержать преобразованную строку.

Примеры входных и выходных данных

2 5 Перевернуть строку!	Пвереернуть строку!
1 10 1234567890	0987654321

5 5 abracadabra	abracadabra

Задача 2: Гири и весы

Ограничения по времени: 1 секунда
Ограничения по памяти: 64 мегабайт
Максимальное количество баллов: 100

Комплект гирь состоит из N гирь различного веса; в Вашем распоряжении имеются два таких комплекта. Сможете ли Вы уравновесить рычажные весы гирями из этих комплектов, положив на каждую из чашек по две гири? При этом гири на левой чашке должны иметь одинаковый вес, а на правой – разный.

Формат входных данных. Первая строка входного файла содержит величину N ($1 \le N \le 10000$, для 50% тестов эта величина не превышает 1000). Вторая строка содержит N целых положительных чисел, не превосходящих 10^9 – веса очередной гири.

Формат выходных данных. В первой строке запишите `'Yes'` или `'No'` (без кавычек), в зависимости от того, имеет ли задача решение. В случае положительного ответа вторая строка должна содержать веса гирь, лежащих на правой чашке весов (в произвольном порядке), а третья – вес одной из гирь, лежащих на левой чашке. Если ответ отрицательный, вторая строка должна содержать наименьший и наибольший вес гирь из комплекта (в произвольном порядке).

Если задача допускает несколько вариантов решения, выведите любой из них.

Примеры входных и выходных данных

5 10 20 30 40 50	Yes 20 40 30
7 20 50 1 10 5 2 100	No 100 1

Задача 3: Электронная мишень

Ограничения по времени: 1 секунда
Ограничения по памяти: 16 мегабайт
Максимальное количество баллов: 100

Для тренировки спортсменов-стрелков всё чаще используются электронные мишени, вместо традиционных бумажных. Электронная мишень представляет собой доску, покрытую специальным составом, реагирующим на луч лазера. На мишени отображаются K концентрических окружностей различного радиуса. Окружности пронумерованы, начиная с единицы, в порядке убывания радиусов. Попадание вовнутрь и на границу самой маленькой окружности приносит стрелку K очков. Попадание в кольцо, ограниченное окружностями i и i+1 (включая границу окружности i, но исключая границу окружности i+1) даёт i очков. Наконец, если стрелок не попадает в окружность 1, ему засчитывается промах и начисляется 0 очков.

Стрельба по таким мишеням ведётся из специального оружия, которое вместо пуль выпускает луч лазера.

Программное обеспечение электронной мишени должно, во-первых, настраивать количество и размеры окружностей, и во-вторых, рассчитывать результаты стрельбы.

Вам поручена реализация второй части этой задачи, а именно: по информации о числе окружностей и их радиусах, а также о координатах попаданий N выстрелов, сделанных одним стрелком, рассчитать количество очков, полученных этим спортсменом.

Формат входных данных. Первая строка входного файла содержит величины K и N ($1 \le K \le 10^4$, $1 \le N \le 10^5$, в 50% тестов N, $K \le 100$, а в 80% тестов $K \le 100$, $N \le 1000$). Вторая строка содержит K целых чисел - радиусы окружностей R_1, …, R_K. Ограничения на радиусы: $1 \le R_K < \ldots < R_1$. Считается, что центры всех окружностей находятся в точке (0, 0).

Затем следуют N строк, каждая из которых описывает результаты одного выстрела и содержат два целых числа – координаты очередного попадания. Абсолютные величины координат и радиусов окружностей не превосходят 50000 (в 60% тестов они не превосходят 20000).

Формат выходных данных. Выведите единственное число – рассчитанный результат стрельбы.

Пример входных и выходных данных

5 6 10 8 6 4 2 0 0 1 1 2 2 3 3 4 4 5 5	22

Задача 4: Числа-близнецы

Ограничения по времени: 2 секунды
Ограничения по памяти: 256 мегабайт
Максимальное количество баллов: 100

Пару чисел-близнецов составляют два целых положительных числа A_1 и A_2, удовлетворяющие следующим условиям:

- $A_1 = A_2 + 2$;
- оба этих числа – простые.

Определите, сколько различных пар чисел-близнецов находится в интервале от M до N. Пары считаются различными, если их меньшие элементы не равны. Напомним, что единица не считается простым числом.

Формат входных данных. Единственная строка содержит величины M и N ($1 \le M \le N \le 10^9$, $N–M \le 10^{\$}$; для 20% тестов $N \le 10^3$, а для 80% тестов $N \le 10^7$).

Формат выходных данных. Выведите искомое количество чисел-близнецов.

Пример входных и выходных данных

1 13	3

Пояснение: результатом будут пары (3, 5), (5, 7), (11, 13).

2011/2012 учебный год

Авторы задач – С.И. Кашкевич и А.А. Толстиков.

Полное решение каждой задачи оценивалось в 100 баллов.

Имена входных файлов для всех задач: input.txt

Имя выходных файлов для всех задач: output.txt

Задача 1: Метрическое время

Ограничения по времени: 1 секунда

Ограничения по памяти: 16 мебибайт

Предположим, что первоапрельская шутка, прозвучавшая четверть века назад в одной из программ австралийского телевидения, обернулась кошмарной явью…

Принято решение для измерения промежутков времени отказаться от привычных часов, минут и секунд и ввести новые метрические единицы измерения. Согласно этому проекту, сутки делятся на 10 равных долей – *децидней*. Децидень, в свою очередь, делится на сто *миллидней*, а миллидень – на 100 *микродней*. Запись времени, прошедшего с начала суток (полуночи), ведётся в виде a:bb:cc, где a, b, c ($0 \le a \le 9$, $0 \le b, c \le 99$) – соответственно число деци-, милли- и микродней. Заметьте, что для записи величин b и c всегда используются две цифры, например, 3:05:86.

Напомним, что запись времени в «старой» 24-часовой системе выглядит как dd:mm:ss, где d, m, s – соответственно число часов, минут и секунд, прошедших с полуночи. Для записи часов, минут и секунд всегда используются две цифры, например, 05:20:00.

Во время переходного периода необходимо быстро переводить время из 24-часовой в метрическую систему и обратно. Вам поручено разработать соответствующее программное обеспечение…

Формат входных данных. Первая строка входного файла содержит величину K – количество подтестов ($2 \le K \le 10^5$, в 80 % тестов $K \le 10^4$, в 50% тестов $K \le 100$). Каждая из последующих K строк описывает один подтест и содержит тип системы исчисления времени (1 – 24-часовая, 2 – метрическая), за которым следует единственный пробел, и информацию о времени, записанную в этой системе. Вам необходимо записать это же время в другой системе исчисления времени. Дробную часть секунд и микродней следует отбрасывать, а не округлять эти величины!

Формат выходных данных. Выходной файл должен содержать K строк, каждая из которых соответствует одному подтесту и содержит результат перевода.

Пример входных и выходных данных

4 1 06:04:17 1 00:00:00 2 5:18:71 2 9:99:99	2:52:97 0:00:00 12:26:56 23:59:59

Задача 2: Проверка на нечётность

Ограничения по времени: 2 секунды

Ограничения по памяти: 16 мебибайт

Заданы два целых положительных числа A и B ($A \le B$). Определите, для скольких целых чисел из интервала $[A, B]$ их двоичное представление содержит нечётное количество единиц.

Формат входных данных. Первая строка входного файла содержит величину K – количество подтестов ($2 \le K \le 5$). Каждая из последующих K строк соответствует одному подтесту и содержит величины A и B ($1 \le A, B \le 10^{18}$). Для 50 % тестов величина $B - A$ не превышает 10^6, а для 25 % тестов – 10^5 во всех подтестах таких тестов.

Формат выходных данных. Выведите K строк, каждая из которых должна содержать ответ на очередной подтест – искомое количество чисел.

Примеры входных и выходных данных

2 10 16 20 25	4 3
5 1 4 10 40 100 400 1000 4000 10000 40000	3 15 151 1500 15001

4 10 99 100 999 1000 9999 10000 99999	45 450 4500 45000

Задача 3: Химия, химия...

Ограничения по времени: 1 секунда
Ограничения по памяти: 16 мебибайт

Названия химических элементов состоят либо из одной прописной латинской буквы (например, **H** – водород, **K –** калий), либо из пары латинских букв, первая из которых является прописной, а вторая – строчной (например, **Ca –** кальций, **Hg –** ртуть).

Задана непустая строка *S*, содержащая названия всех химических элементов, атомы которых входят в состав молекулы некоего химического вещества. Если в состав молекулы входят несколько атомов одного химического элемента, название этого элемента повторяется соответствующее количество раз, причём не обязательно подряд. Никаких разделителей между названиями элементов нет. Так, строка **OKOMnOO** (перманганат калия) означает, что в состав этой молекулы входят четыре атома кислорода (**O**) и по одному атому калия (**K**) и марганца (**Mn**)[1].

Определите количество различных химических элементов, атомы которых входят в состав описанного вещества.

Формат входных данных. Входной файл содержит единственную строку *S*. Длина строки не превосходит 30000 (в 50 % тестов длина не превосходит 255). Кроме того, в 25 % тестов строка *S* содержит только прописные латинские буквы.

Формат выходных данных. Единственная строка выходного файла должна содержать искомое число различных химических элементов.

Примеры входных и выходных данных

OKOMnOO	3
ABABAB	2
AbABaB	4

[1] Химические элементы и вещества, описанные в примерах и тестах, могут не существовать в природе.

Задача 4: День пряника

Ограничения по времени: 1 секунда
Ограничения по памяти: 16 мебибайт

В школах страны Байтландии прижился интересный обычай. Один раз в году, в т.н. «День пряника» девочки пекут (под руководством старших, конечно!) пряники, печенье и другие вкусные вещи, а потом угощают ими своих одноклассников. Выпечка, приготовленная к Дню пряника, обычно имеет форму треугольников или прямоугольников.

Маша приготовила на этот праздник N пряников различной формы и размера и подготовила соответствующее число одинаковых круглых коробок, чтобы упаковать угощение, по одному прянику в коробку. Но вот незадача! Оказывается, некоторые пряники не помещаются в подготовленные коробки…

Помогите Маше определить, что из приготовленного угощения поместится в подготовленные упаковочные коробки, а что – нет.

Формат входных данных. В первой строке входного файла записан диаметр упаковочной коробки, а во второй – количество приготовленных Машей угощений N ($2 \le N \le 100$). Каждая из последующих N строк содержит описание одного пряника. Если пряник имеет треугольную форму, то в начале строки записывается 1, а затем – длины сторон этого треугольника (треугольник невырожденный). Для прямоугольного пряника в начале строки записывается 2, а затем – длины смежных сторон прямоугольника. Числа в строках разделяются единственным пробелом. Все размеры – целые положительные числа, не превосходящие 10^{17} (для 80 % тестов эта величина не превосходит 10^3).

Для 25 % тестов угощение имеет только форму прямоугольников.

Формат выходных данных. Выведите в выходной файл строку из N символов. Каждый символ строки соответствует одному прянику (в порядке, заданном во входном файле). Символ `'Y'` означает, что пряник можно поместить в коробку, а символ `'N'` – что пряник поместить нельзя.

Примеры входных и выходных данных

20 2 2 15 17 2 19 5	NY

20 4 1 20 12 16 1 10 10 10 1 20 20 20 2 13 15	YYNY

2012/2013 учебный год

Авторы задач – С.И. Кашкевич и А.А. Толстиков. Идея второй задачи взята из соревнований ВКОШП (г. Санкт-Петербург).

Полное решение каждой задачи оценивалось в 100 баллов.

Имена входных файлов для всех задач: input.txt

Имя выходных файлов для всех задач: output.txt

Задача 1: Построить римское число

Ограничения по времени: 1 секунда

Ограничения по памяти: 64 мебибайта

> *One of the main causes of the fall of the Roman Empire was that, lacking zero, they had no way to indicate successful termination of their C programs.*
>
> *Robert Firth*

Напомним прежде всего правила записи натуральных чисел в римской системе счисления. В этой системе может быть представлено любое натуральное число, не превосходящее 3999. Для записи отдельных (т.н. *атомарных*) чисел используются следующие заглавные латинские буквы и их комбинации:

1 – I	4 – IV	5 – V	9 – IX
10 – X	40 – XL	50 – L	90 – XC
100 – C	400 – CD	500 – D	900 – CM
1000 – M			

Число *P*, не входящее в приведенный список, записывается следующим образом: находим наибольшее из атомарных чисел *K*, не превосходящее *P*, и записываем его представление в римской системе счисления. Затем повторяем эту же процедуру для числа *P*-*K* и т.д. до тех пор, пока после вычитания не получим ноль. Представления атомарных чисел записываются слева направо без каких-либо промежутков. Так, число 999 в римской системе счисления будет записано как CMXCIX (а не IM, как может показаться!).

Задана непустая строка, состоящая из символов `'M'`, `'D'`, `'C'`, `'L'`, `'X'`, `'V'`, `'I'`. Переставьте символы в этой строке так, чтобы они образовали правильное число в римской системе счисления.

Формат входных данных. В первой строке входного файла задаётся число N –количество подтестов ($1 \le N \le 5$). Каждая из последующих N строк соответствует одному подтесту и содержит исходную последовательность символов. Длина последовательности не превосходит 15. В 50 % тестов длина последовательности не превосходит 5 для всех подтестов этого теста.

Формат выходных данных. Выходной файл содержит N строк, каждая из которых соответствует одному подтесту. Она должна содержать полученное число в десятичной системе счисления. Если построить число невозможно, выведите ноль (отсутствие которого и стало причиной краха Римской империи). Если задача допускает несколько решений, выведите максимальное из возможных чисел.

Пример входных и выходных данных

5 IXIC CMMMM CMMMCM LILI D	112 3900 0 0 500

Задача 2: Две матрицы

Ограничения по времени: 1 секунда

Ограничения по памяти: 64 мебибайта

Заданы две матрицы $\boldsymbol{A}$ и $\boldsymbol{B}$, состоящие из M строк и N столбцов каждая. Матрица $\boldsymbol{A}$ заполнена числами от 1 до $M \times N$ сначала по строкам, а затем по столбцам, а матрица $\boldsymbol{B}$ – вначале по столбцам, а потом по строкам. Так, для $M = 3$ и $N = 5$ эти матрицы будут иметь вид

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

Найдите количество различных пар (i, j), $1 \le i \le M$, $1 \le j \le N$ таких, что $\boldsymbol{A}_{ij} = \boldsymbol{B}_{ij}$.

Формат входных данных. Единственная строка файла содержит числа M и N ($1 \le M, N \le 10^{18}$, для 20 % тестов $M, N \le 20$, а для 50 % тестов $M, N \le 1000$) – размерности матриц.

Формат выходных данных. Выведите одно число – искомое количество пар.

Примеры входных и выходных данных

10 5	2
3 5	3

Задача 3: Резисторы

Ограничения по времени: 0,5 секунды
Ограничения по памяти: 8 мебибайт

В Вашем распоряжении имеется N резисторов, сопротивление которых неизвестно. Единственное, что вы можете сделать – подключить эти резисторы по одному к достаточно мощному[2] источнику постоянного напряжения и замерить величину тока (в микроамперах), проходящего по полученной цепи. Для i-го резистора ($1 \le i \le N$) эта величина равна T_i.

Выберите пять из имеющихся резисторов так, чтобы после их параллельного соединения и подключения полученной схемы к тому же источнику постоянного напряжения сила тока в неразветвлённой части цепи была максимальной. Из всех физических эффектов следует принимать во внимание только закон Ома для цепи постоянного тока.

Формат входных данных. Первая строка входного файла содержит величину N ($5 \le N \le 10^6$, в 20 % тестов эта величина не превосходит 100, а в 50 % тестов – 10000). Вторая строка содержит значения T_i – целые положительные числа, не превосходящие 10000.

Формат выходных данных. Единственная строка выходного файла должна содержать пять чисел - номера выбранных Вами резисторов (резисторы нумеруются, начиная с единицы, в порядке появления информации о них во входном файле). Если задача допускает несколько решений, выберите любое из них.

Пример входных и выходных данных

10 3 1 3 1 3 1 1 1 1 2	1 2 3 5 10

[2] Это означает, что напряжение, вырабатываемое этим источником, будет неизменным при любой конфигурации цепи из имеющихся резисторов.

Задача 4: Юный чертёжник

Ограничения по времени: 1 секунда
Ограничения по памяти: 64 мебибайта

У Коли не так давно начались уроки черчения… Его любимое упражнение – рисовать прямые линии с помощью линейки. Но просто прикладывать линейку к ватману и проводить карандашом линию ему не нравится, поэтому он любит рисовать прямые по двум точкам.

Однажды он отметил 4 точки, и решил провести через них 2 прямые (одну через первую и вторую точки, а вторую через третью и четвертую), но до этого момента он решил угадать, как будут соотноситься эти прямые между собой, т.е. будут ли они пересекающимися, параллельными или совпадающими. Коля уже записал свое предположение на листке бумаги, помогите ему получить правильный ответ.

Формат входных данных. В первой строке входного файла записано целое число T ($1 \le T \le 10$) – количество подтестов. Далее в каждой из T строк записано по 8 целых чисел x_1, y_1, x_2, y_2, x_3, y_3, x_4 и y_4 ($-1{,}000{,}000 \le x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \le 1{,}000{,}000$, в 50 % тестов эти величины не превосходят 1000 для всех подтестов) – координаты четырёх отмеченных Колей точек. Гарантируется, что первая и вторая, а также третья и четвёртая точки различны в каждом подтесте.

Формат выходных данных. В выходной файл выведите T строк – по одной для каждого подтеста. Если прямые из i-ого подтеста совпадают, то выведите `'COINSIDE'`; если они пересекаются в одной точке, выведите `'CROSS'`; если параллельны – `'PARALLEL'`.

Пример входных и выходных данных

7 1 2 4 8 16 32 64 128 1 2 3 4 5 6 7 8 0 0 1 1 1 0 2 1 1 2 2 1 0 4 4 0 1 1 2 2 -1 1 -2 2 1 2 3 4 4 3 2 1 1 9 8 3 4 5 2 7	COINSIDE COINSIDE PARALLEL PARALLEL CROSS PARALLEL CROSS

Разбор задач

Авторы разбора – С.И. Кашкевич и В.В. Климович.

2005/2006 учебный год

Задача 1: Период времени

Наилучшим, по мнению авторов, способом решения этой задачи, было бы использование специальных типов, предназначенных для работы с датой и временем (наподобие типа TDateTime систем Free Pascal и Delphi). В этом случае необходимо перевести входные строки в тип TDateTime, а затем с помощью функции SecondsBetween получить результат.[3]

Если участники не знают этих типов или не умеют с ними работать, им придётся выполнять соответствующие преобразования вручную. В этом случае оптимальным (опять-таки, по мнению автора) является следующий подход: подсчитать число секунд, прошедших с полуночи 1 января 1801 года до момента времени, указанного во входном файле, а затем выполнить вычитание полученных величин.

Приведём пример функции, считающей число секунд:

```
function Str2Secs(J: string): extended;
var
 Y, M, D, H, N, S: longint;
 Days: extended;
 i: word;
begin
 Y := StrToInt(copy(J, 7, 4));
 M := StrToInt(copy(J, 4, 2));
 D := StrToInt(copy(J, 1, 2));
 H := StrToInt(copy(J, 12, 2));
 N := StrToInt(copy(J, 15, 2));
 S := StrToInt(copy(J, 18, 2));
 Days := 365*(Y-1801)+(Y-1801) div 4;
 if Y>1900 then
  Days := Days-1;
 for i := 1 to M-1 do
  Days := Days+DayInMonth[i];
```

[3] Последние версии системы программирования Free Pascal некорректно обрабатывают даты, относящиеся к XIX столетию… ☹

```
    // DayInMonth - массив дней в месяце
    // в невисокосном году
 if (M>2) and (Y mod 4 = 0) and (Y <> 1900) and
(Y<>2100) then
  Days := Days+1;
 Days := Days+D-1;
 Str2Secs := 3600*H+60*N+S + 86400*Days;
end;
```

Наконец, участники, запутавшиеся в вычислении числа дней, или не знающие, как это сделать, могут получить 50 % баллов, решая задачу для тестов, относящихся к одной дате. В этом случае им достаточно определить количество секунд, прошедших с начала дня, по формуле

```
3600*H+60*N+S
```

Задача 2: Правые части

Обозначим через $F(x)$ сумму правых частей для чисел от 1 до x (и положим $F(0) = 0$). Тогда ответ задачи будет выражаться формулой $F(B) – F(A–1)$.

Для нахождения $F(x)$ разобьём числа из этого интервала на несколько групп. В частности, группу S_i будут образовывать все числа, которые не делятся нацело на 2^i, но делятся на меньшие степени двойки. В частности, все нечётные числа будут образовывать группу S_0. Правые части для каждого числа из группы S_i равны 2^i, осталось оценить количество чисел в группе. Эта величина почти равна $x \text{ div } 2^{i+1}$ (но если i-й двоичный разряд числа x содержит единицу, то к ней надо добавить единицу).

Теперь расчёт суммы правых частей не представляет сложностей. Учитывая, что основные операции при вычислениях – это умножение и деление на степени двойки, логично было бы использовать побитовые операции shr, shl, and вместо умножения и деления. Однако это не принципиально, поскольку ограничения на время обработки одного теста не накладывались.

Задача 3: Мат в один ход

Сейчас авторы вынуждены признать, что допустили ошибку, предлагая на второй этап олимпиады столь сложную задачу. Кроме того, из-за ручной проверки, выполненной в отдельных районах, некоторые пра-

вильные ответы не были засчитаны, хотя автоматическая проверка, предложенная авторами, дала бы правильный результат. Но, как говорится, что есть, то есть…

Анализ возможностей поставить мат королю чёрных показывает, что это возможно сделать, только если чёрный король находится на крайней вертикали или горизонтали, а белый король препятствует его уходу с этой линии. Тогда мат ставит белый ферзь.

Таким образом, необходимо в первую очередь определить, находятся ли короли в положении, предшествующем мату, а затем – может ли ферзь без нарушения правил перейти в одну из матующих позиций. Возможные матующие позиции показаны на рис. 3.

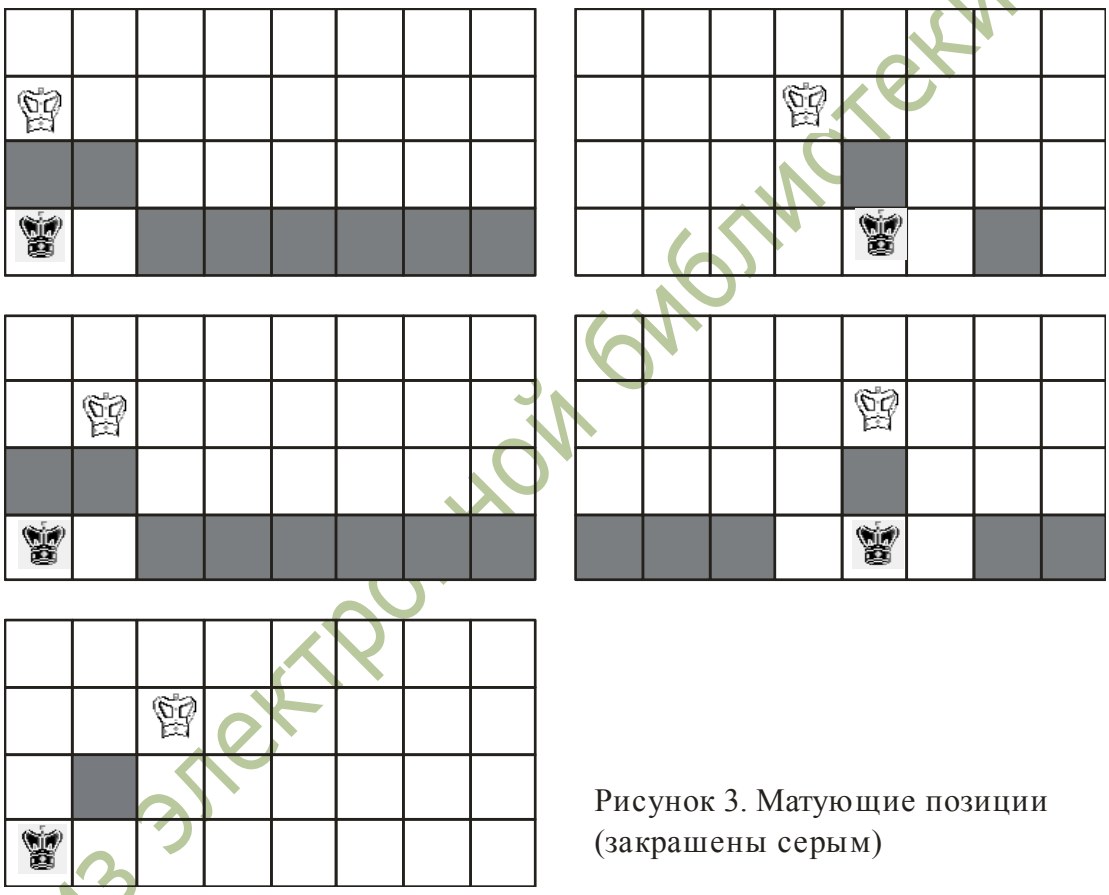

Рисунок 3. Матующие позиции (закрашены серым)

2006/2007 учебный год

Задача 1: Игра

Это – типичная задача, относящая к классу антагонистических игр. Ее решение заключается в следующем. Введем на игровом поле систему

координат так, что левая нижняя клетка будет иметь координаты (1, 1), а правая верхняя – (*M*, *N*).Все клетки игрового поля делятся на три вида:

- *финальная* клетка с координатами (*M*, *N*);
- *выигрышные* клетки, из которых можно сделать хотя бы один ход в проигрышную клетку;
- *проигрышные* клетки, переход из которых возможен только в выигрышную либо финальную клетку.

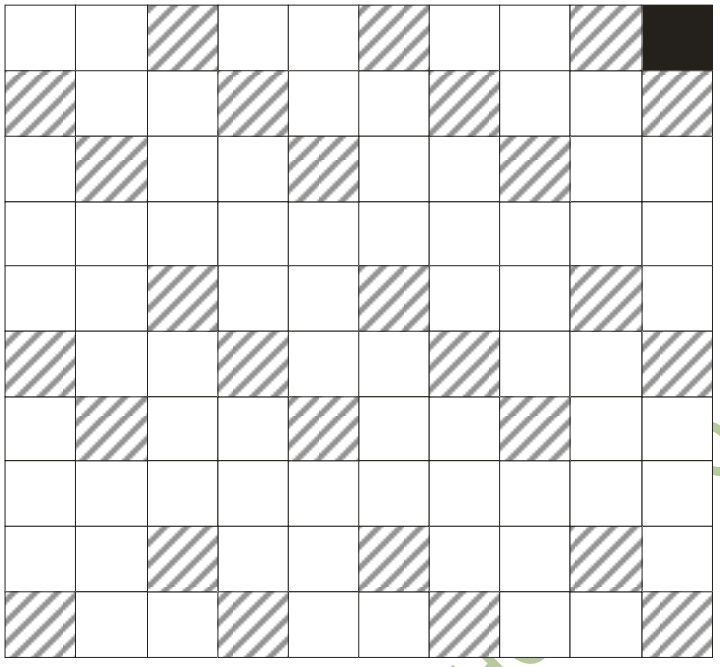

Рис. 4. Выигрышные и проигрышные (залитые штриховкой) клетки для доски 10 × 10.

При правильной стратегии игрок, при ходе которого фишка стоит в выигрышной клетке, помещает фишку в проигрышную. Тогда его противник вынужден либо пойти в финальную клетку и проиграть, либо вновь вернуть фишку в выигрышную клетку.

Итак, для того, чтобы решить поставленную задачу, достаточно определить, является ли клетка (1, 1) выигрышной или проигрышной.

В нашем случае справедливо следующее утверждение: клетка с координатами (*i*, *j*) является проигрышной, если клетки (*i*, *j*+1), (*i*, *j*+2), (*i*+1, *j*), (*i*+2, *j*), (*i*+3, *j*) либо не существуют, либо являются выигрышными. В противном случае клетка (*i*, *j*) является выигрышной.

Первый, нерациональный способ решения задачи заключается в том, что формируется битовый массив размером *M* × *N*, который заполняется в следующем порядке: последняя строка, последний столбец, предпоследняя строка и т.д. После того, как заполнена клетка (1, 1), задача решена. Из-за больших ограничений на *M* и *N* этот способ может взять только 50 % тестов. Кроме того, при этом способе требуется большой объем памяти для хранения массива, что требует дополнительных усилий при решении задачи на Borland Pascal (например, можно хранить в памяти не всю матрицу, а только 4 строки и 3 столбца).

Второй способ решает задачу за время, не зависящее от M и N. Заполним вручную несколько последних строк и столбцов игрового поля, как показано на рис. 4, чтобы проследить закономерность (серым цветом закрашены проигрышные клетки). Теперь очевидно, что при $M \bmod 4 = 0$ Вася всегда выигрывает, в других случаях нетрудно вывести формулу, определяющую вид клетки (1, 1) в зависимости от величин $M \bmod 4$ и $N \bmod 3$.

Задача 2: Постулат Бертрана

Казалось бы, определение того, является ли натуральное число A простым, не представляет трудностей – достаточно находить остаток от его деления на все предыдущие натуральные числа, превосходящие единицу. Однако такое лобовое решение крайне неэффективно. Ускорить работу можно, останавливая процесс деления, если делитель становится большим, чем sqrt(A) – в этом случае A гарантированно является простым числом. Кроме того, нет смысла делить на составные числа – следует хранить в памяти массив простых чисел, не превосходящих sqrt(A), и делить только на элементы этого массива. Оказывается, при A=2000000 необходимо хранить только 223 первых простых числа, что снимает проблемы с выделением памяти для программирующих в среде Borland Pascal.

Еще одно усовершенствование связано с тем, что нет необходимости просматривать все числа в промежутке [2, $2 \times N$); достаточно исследовать два промежутка: [2, int(sqrt($2 \times N$))] и (N, $2 \times N$).

Задача 3: Пустые прямоугольники

Самый простой метод решения этой задачи заключается в том, что перебираются все допустимые пары точек i и j (т. е. такие пары, что $x_i \neq x_j$ и $y_i \neq y_j$) и для каждой такой пары просматриваются оставшиеся точки. Если хотя бы одна из оставшихся точек лежит внутри или на границе прямоугольника, образованного точками i и j, мы должны перейти к анализу следующей пары.

Это алгоритм имеет трудоемкость O(N^3). Ограничения по времени не устанавливались, так что даже этот алгоритм при правильном переборе (так, необходимо потребовать, чтобы выполнялось условие $i < j$, чтобы по крайней мере вдвое сократить число вычислений) должен брать все тесты.

Оптимизация алгоритма заключается в том, что исходные точки предварительно сортируются по возрастанию значений одной из координат (например, координаты X). Тогда для выбранной пары точек i и j в большинстве случаев можно существенно уменьшить количество просматриваемых точек. Однако этот приём требует высокой техники кодирования, и автор задачи считает, что вводить ограничения по времени, требующие применения подобных алгоритмов, на уровне районной олимпиады нецелесообразно.

Задача 4: Поиск анаграмм

Заметим, что две строки S_1 и S_2 являются анаграммами друг друга, если количество вхождений каждого символа в строку S_1 и S_2 одинаково. Поэтому для решения задачи достаточно определить два целочисленных массива из 52 элементов каждый, и, просматривая исходные строки, накапливать в этих массивах число соответствующих символов. После окончания просмотра необходимо сравнить эти массивы поэлементно: если массивы равны, то строки являются анаграммами друг друга, и наоборот. Трудоёмкость такого решения – O (N), где N – длина строки.

Другие подходы к решению этой задачи:

- формировать один массив, и для строки S_1 увеличивать соответствующий элемент массива, а для строки S_2 – уменьшать. Строки являются анаграммами, если после их анализа все элементы массива будут равны нулю.
- сортировать по возрастанию символов первую строку, а каждый символ второй строки искать в первой с использованием дихотомии. В случае повторяющихся символов такой подход требует аккуратного анализа таких символов. Кроме того, его трудоёмкость – O(N log N), что может привести к нарушению лимита времени.
- отсортировать обе строки, а затем сравнить их. Алгоритм также не является оптимальным по времени.

2007/2008 учебный год

Задача 1: Экспрессные маршруты

Пусть $D(n)$ – число всех маршрутов (обычных и экспрессных), соединяющих города A и B при условии n промежуточных остановок. Тогда ответ будет равен $D(n)$-1.

Составим рекуррентное соотношение для нахождения $D(n)$:

```
D(n) := D(n-1) //маршруты, делающие последнюю
            //  промежуточную остановку в пункте n
      + D(n-2) // то же для последней остановки
            // в пункте n-1
      + D(n-3) // то же для последней остановки в
            // пункте n-2
```

Начальные значения для $D(n)$:

```
D(0) = 1; D(1) = 2; D(2) := 4;
```

Остается написать программу для реализации этого рекуррентного соотношения ☺.

Входные данные для не менее, чем 50 % тестов подобраны так, чтобы результаты помещались в 4-байтовое целое число (тип longint в Паскале или long в C++), для остальных тестов результаты помещаются в 8-байтовое число.

Задача 2: Отрезки на прямой

Отсортируем все отрезки по неубыванию их начальных точек.

Просматривая отсортированную последовательность отрезков, мы можем получить ответ по следующему алгоритму:

1. первый из новых отрезков совпадает с первым отрезком из условия;
2. если начало очередного отрезка из условия выходит за границы последнего нового отрезка, это – сигнал к тому, что количество новых отрезков увеличилось. В этом случае, как и на начальном шаге, новый отрезок совпадает с очередным отрезком из условия;
3. если условие п. 2 не выполняется, остаёмся в том же новом отрезке, при необходимости увеличивая его длину.

Псевдокод приводится ниже.

```
c[1] := a[1]; d[1] := b[1]; k := [1];
for i := 2 to N do begin
  if a[i]>d[k] then begin
   // рассчитать длину максимального из просмотрен-
ных отрезков
   k := k+1;
   c[k] := a[i]; d[k] := b[i];
  end
  else
   d[k] := max (d[k], b[i]);
```

```
end;
```

Для не менее чем 50 % тестов размерность задачи позволяет выполнить сортировку за время O (n^2), однако для полного решения задачи требуется умение программировать более эффективные алгоритмы сортировки.

Задача 3: Проблема Гольдбаха

Заводим логический массив P из N элементов, и заполняем его значениями true или false в зависимости от того, является ли число i ($1 \le i \le N$) простым. Метод заполнения – решето Эратосфена (те участники, которые незнакомы с этим алгоритмом, могут, конечно, проверять простоту числа другими методами, но это может привести к нарушению предела времени).

После этого выполняем поиск решения: просматриваем элементы массива P в порядке уменьшения индексов. Пусть $P[i]$ – простое число, тогда проверяем на простоту число $N–P[i]$. Если и это число – простое, задача решена. В противном случае продолжаем просмотр элементов массива P.

Не менее 50 % тестов позволяют решить задачу с учетом ограничений на объем статической памяти, накладываемых системой программирования Borland Pascal, для прохождения остальных тестов требуется либо работать в других средах, либо выделять память динамически.

Задача 4: Изменить Строку!

Определим условие, когда текущий символ строки стоит в начале слова: текущий символ – не пробел и либо он стоит в начале строки, либо перед ним находится пробел. Порядок проверки условий важен, поэтому необходимо включить режим неполного вычисления логических выражений (по умолчанию он включен в системах программирования для Pascal).

Режим неполного вычисления логических выражений, применительно к этой задаче, состоит в следующем. Если при вычислении выражения

```
(a) or (b)
```

оказалось, что значение a - истина, то b можно и не вычислять, а иногда и нельзя вычислить... Если символ стоит в начале строки, то проверять, не стоит ли перед ним пробел, нельзя.

Просматриваем строку и для каждого символа, стоящего в начале слова и являющегося маленькой латинской буквой, выполняем соответствующую замену…

Не менее 50 % тестов позволяют решить задачу с учетом ограничений на длину строки, накладываемых системой программирования Borland Pascal, для прохождения остальных тестов требуется либо работать в других средах, либо переходить к типу PChar.

2008/2009 учебный год

Задача 1: Тарабарская грамота

Относительно простая задача, для решения которой не требуется никаких специальных знаний. Достаточно лишь построить два массива – с исходным и «перевернутым» порядком согласных букв, и заменять все символы, которые были найдены в первом массиве, на соответствующие элементы из второго массива.

Единственная сложность состоит в обработке длинных строк. Чтение из входного файла по одному символу приведет к длительному выполнению программы, поэтому для того, чтобы взять максимальное количество баллов, необходимо уметь работать с типами AnsiString или PChar в системе программирования Free Pascal. Сниженные ограничения соответствуют типу string в этой системе программирования; для остальных систем проблем возникнуть не должно.

Затруднение у участников вызвала фраза из условия: «…в противном случае последняя строка выходного файла должна заканчиваться символами перевода строки». Многие восприняли это, как необходимость выводить в конце файла пустую строку. На самом деле это означает, что во входном файле последняя строка может оканчиваться символами перевода строки, а может и не оканчиваться ими (в тесте 03). Выходной же файл ВСЕГДА должен завершаться этими символами (это необходимо для правильной работы чекера).

Задача 2: Сглаживание матриц

Самая сложная, на мой взгляд, задача соревнования. В ней заложены два подводных камня.

1) Необходимо правильно определить количество соседей каждого элемента матрицы (от 1 до 8) и просмотреть всех соседей. Одно из возможных решений этой задачи состоит в следующем. Пусть нам надо

определить всех соседей элемента A[i, j]. Напишем следующий фрагмент программы на Паскале:

```
for di := -1 to 1 do
  for dj := -1 to 1 do
    if ((di<>0) or (dj<>0)) and
       (i+di>0) and (i+di<=M) and
       (j+dj>0) and (j+dj<=N) then begin
 {элемент A[i+di, j+dj] является соседом, обработать
его}
    end;
```

2) Ограничения на размеры матрицы таковы, что невозможно выделить память для ее хранения по максимуму. Поэтому надо либо выделять память динамически, либо хранить в памяти только три очередных строки матрицы.

Сниженные ограничения дают возможность хранить матрицу в памяти целиком без динамического выделения памяти.

Задача 3: Где-то недостаток, а где-то – избыток…

Самая простая задача, для решения которой необходимо лишь умение определять все делители натурального числа *N* путем полного перебора всех чисел от 1 до *N* и проверки делимости.

Сниженные ограничения дают возможность использовать тип string в системе программирования Free Pascal для хранения результатов работы.

Задача 4: Черепаха

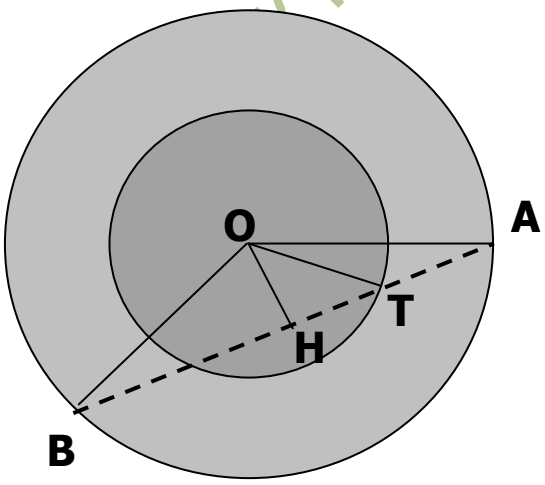

Рис. 5. Прямоугольные треугольники для расчёта

Задача является упрощенным двумерным вариантом задачи «Червяк на яблоке», предложенной на четвертьфинале студенческого чемпионата мира по программированию, проходившем в БГУ в 2008 году.

Задача требует умения рассчитывать стороны прямоугольных треугольников. Обозначим через φ_A и φ_B азимуты точек *A* и *B* соответственно. Тогда угол *AOB*, равный φ, вычисляется по формуле

```
φ := abs(φA - φB);
if φ >180 then
  φ := 360- φ;
```

Расстояние *AB*, которое проползет черепаха, равно 2**R**sin(*φ* /2). Для того, чтобы найти, какую часть этого пути составит луг, найдем *OH* – высоту треугольника *AOB*. Она равна $h = R \times \cos(\varphi/2)$. Если $h \geq P$, весь путь черепахи проходит по песку. В противном случае расстояние, которое черепаха проползет по траве, вычисляется по теореме Пифагора из треугольника *OHT*.

2009/2010 учебный год

Задача 1: Литорея

Задача является упрощением одноимённой задачи, представленной на четвертьфинале ACM в ноябре 2008 года. В представленном варианте она не требует особых знаний алгоритмики. Однако, как и любая олимпиадная задача, она содержит требования, выходящие за рамки школьной программы по информатике. В частности, требуется уметь правильно читать длинную строку, а также организовывать два независимых друг от друга цикла – по длине строки и по длине ключа, с учётом того, что переход на новую строку не сбрасывает счётчик цикла по ключу!

Ограничения, накладываемые на 50 % тестов, дают возможность работать со стандартным типом string системы программирования Free Pascal.

Задача 2: Две окружности

Задача, которая, по мнению авторов, должна была быть утешительной, но на самом деле такой не явилась… ☹

Пусть D – расстояние между центрами окружностей, R_1 и R_2 – их радиусы. Будем без ограничения общности полагать, что $R_1 \geq R_2$. Тогда:

- если центры окружностей и их радиусы равны, то ответ -1;
- если ($R_1 + R_2 = D$) или ($R_1 - R_2 = D$), то ответ 1;
- если ($R_1 + R_2 > D$) или ($R_1 + R_2 < D$), то ответ 0;
- в противном случае ответ 2.

При этом не играет роли, как расположены окружности – находится ли центр одной из окружностей внутри другой, либо нет.

В 50 % тестов для вычислений достаточно использовать тип double, в противном случае требуется работа с типом extended.

Задача 3: Квадраты цифр числа

Задача взята из классической книги Гуго Штейнгауза "Сто задач" (М.: «Наука», 1976). В этой книге доказывается, что искомая величина после нескольких начальных вычислений (не более 12) либо становится равной единице, либо циклически выбирается из последовательности (145, 42, 20, 4, 16, 37, 58, 89).

Но даже если не знать этого факта, достаточно заметить, что искомая величина не превосходит 81*9 = 729 для любого начального *N*, и выполнять *K* вычислений совершенно необязательно – достаточно выполнить несколько начальных и искать повторения среди полученных результатов... Полный цикл вычислений не укладывается в ограничения по времени.

В 50 % тестов для вычислений достаточно использовать тип integer системы программирования Free Pascal, в противном случае требуется работа с типом longint.

Задача 4: «Жизнь»

Задача требует умения работать с двумерными массивами и, в частности, правильно обрабатывать соседние элементы матрицы. Алгоритм такой обработки приведен в разборе задач за 2008/2009 год.

Пусть нам надо определить всех соседей элемента A[i, j]. Как и в задаче «Сглаживание матриц», напишем фрагмент программы на Паскале, реализующий это:

```
for di := -1 to 1 do
  for dj := -1 to 1 do
    if ((di<>0) or (dj<>0)) and
       (i+di>0) and (i+di<=M) and
       (j+dj>0) and (j+dj<=N) then begin
{элемент A[i+di, j+dj] является соседом, обработать
его }
    end;
```

Для того, чтобы отличать только что родившиеся клетки от влияющих на умирание своих соседей, рекомендуется давать им временные пометки. Временную пометку клетки [*i*, *j*] можно заменить на постоянную после определения судьбы клетки [*i*+1, *j*+1].

2010/2011 учебный год

Задача 1: Перевернуть строку!

Самая простая задача в наборе, именно поэтому за её решение можно получить 50 баллов, а не 100, как за решение других. Всё, что нужно уметь для решения этой задачи – организовывать совместную работу с двумя индексами.

Определяем два целочисленных индекса i и j. Изначально $i = A$, $j = B$. Основной цикл обработки исходной строки S выглядит следующим образом:

```
while i<j do begin
  Tmp := S[i];
  S[i] := S[j];
  S[j] := Tmp;
  i := i+1;
  j := j-1;
end;
```

Те участники, которые умеют работать лишь с типом string Free Pascal, могут взять 50 % тестов.

Задача 2: Гири и весы

Для решения этой задачи также требуется умение одновременно работать с несколькими индексами, хотя такая работа должна быть организована более хитро…

Обозначим через P массив весов гирь из одного комплекта, отсортированный по возрастанию. Будем подбирать такие индексы i, j, k, что $P[i] + P[k] = 2 \times P[j]$, и $i < j < k$.

Средний индекс j, очевидно, пробегает значения от 2 до N-1. Индекс k для каждого нового значения j изначально устанавливаем в j+1. Меньший индекс i будет изменяться от 1 до j-1, а вот больший индекс для каждого нового значения i должен, в зависимости от знака выражения $P[i] + P[k] - 2 \times P[j]$, либо уменьшаться, либо увеличиваться (оставаясь, тем не менее, в интервале $[j+1, N]$). Такой подход обеспечивает трудоёмкость $O(N^2)$ для определения ситуации, когда уравновесить весы нельзя. Очевидно, положительный ответ на вопрос задачи будет достигнут быстрее.

Задача 3: Электронная мишень

Казалось бы, для решения этой задачи требуется умение рассчитывать расстояние от точки до начала координат и выполнение поиска в упорядоченном массиве. Однако…

Для прохождения всех тестов этой задачи требуется умение выполнять ещё две вещи:

- вместо расчёта расстояния от каждой точки попадания до центра мишени оперировать с квадратом этой величины (возводя исходные радиусы в квадрат);
- использовать дихотомию для определения нужного кольца и, соответственно, количества очков, засчитанных за выстрел.

Игнорирование хотя бы одного из этих требований может привести к нарушению предела времени для теста.

В 60 % тестов достаточно использовать тип longint для вычисления квадратов расстояний; остальные тесты требуют типа int64.

Задача 4: Числа - близнецы

Для определения того, является ли заданное число простым, могут быть использованы два алгоритма:

- деление этого числа на предыдущие найденные простые числа (или, что хуже, на все числа, меньшие заданного);
- построение т.н. «решета Эратосфена».

Второй способ является более быстрым (доказано, что его трудоёмкость равна O (N log log N), где N – заданное число). Однако он не изучается в рамках школьной программы, и не все участники знают о нём...

Более того, даже построение решета Эратосфена для заданного N приводит к нарушению лимита времени из-за больших N (применение метода деления позволяет взять 20 % тестов, построение решета – 80 %).

Для того, чтобы прошли все тесты, необходимо заметить, что поиск следует производить только в интервале от M до N, который не столь велик. С другой стороны, для вычёркивания в решете Эратосфена элементов, не превосходящих N, используются простые числа, не превосходящие sqrt(N), в ограничениях задачи – не более 32000.

Следовательно, можно организовать два массива – решета. Первый содержит 32000 элементов и служит для поиска простых чисел, используемых для работы со вторым массивом. Второй массив содержит N-M+1 элемент, и именно в нём ищутся числа-близнецы.

2011/2012 учебный год

Задача 1: Метрическое время

Для решения этой задачи необходимо воспользоваться типовым приёмом: перевести время в заданной системе исчисления времени, в ко-

личество минимальных единиц в этой же системе (секунд или микродней). Затем это количество требуется перевести в другую систему (умножением или делением на 86400/100000), а потом восстановить результат в более крупных единицах.

Первый подтест примера:

```
1 06:04:17
```

даёт 21857 секунд (6*3600 + 4*60 + 17), что соответствует 25297.45 микродням (21857*100000/86400).

Большое количество подтестов позволяет отсечь возможные неэффективные вычисления.

Задача 2: Проверка на нечётность

Задача, которая допускает множество подходов к её решению. Авторы, естественно, стремились оценить каждый из таких подходов.

Простейшее решение «в лоб» предполагает, что для каждого числа из интервала [*A*, *B*] мы будем подсчитывать количество единиц, не обращая внимания на результаты предыдущих расчётов.

Если для определения количества единиц используются операции получения остатка от деления на 2 и последующего деления на 2, такое решение берёт 35-40 % тестов.

Можно предположить, что побитовые операции shr и and будут работать быстрее, чем стандартное деление, но давать тот же эффект. Действительно, проверка

```
(X mod 2)=1
```

эквивалентна

```
(X and 1)=1,
```

а вычисление

```
X := X div 2
```

можно заменить оператором

```
X := X shr 1.
```

Использование побитовых операций позволяет взять от 50 до 60 % тестов.

Для полного решения задачи необходимо придумать формулу для функции *F*(*X*) – количества чисел, не превосходящих *X*, содержащих не-

чётное количество единиц в своём двоичном представлении. Тогда ответом на задачу будет величина $F(B) - F(A\text{-}1)$.

Величина $F(X)$ для нечётных X равна $(X+1)$ div 2, это доказывается по индукции.

При $X=1$ $F(X)$ также равно единице. Пусть формула верна для $X = 2k–1$ и $F(2k–1) = k$. Далее следуют два числа: $2k$ (чётное) и $2k+1$ (нечётное). Их двоичные представления отличаются только младшим битом, так что одно из них содержит в этом представлении нечётное количество единиц. Таким образом, $F(2k+1) = k+1$, что и требовалось доказать.

Для чётных X нам необходимо оценить, содержит ли X нечётное количество единиц. Если да, то $F(X) = F(X–1)+1$; в противном случае $F(X) = F(X\text{-}1)$.

Наконец, авторы сочли возможным оценить и подход «половина чисел, входящих в интервал». Явно это не прописано в условии, однако последний пример показывает, что и такое может быть… ☺ Такое решение оценивается в 10 баллов.

Задача 3: Химия, химия...

Задача требует умения работать с одномерными и двумерными массивами и использовать эти массивы для хранения результатов вычислений.

Определяем два булевских массива: один – двумерный, размерностью 26 × 26, и второй – одномерный из 26 элементов. Первый из этих массивов предназначен для хранения информации о химических элементах с двухсимвольным обозначением, второй – с односимвольным.

На Паскале, где допускается использование типа char в качестве индексного, описание массива выглядит особенно красиво:

```
var
   A: array ['A'..'Z', 'a'..'z'] of boolean;
   B: array ['A'..'Z'] of boolean;
```

Изначально значения элементов этих массивов устанавливаются в false. Затем, сканируя входную строку S, определяем, является ли текущий символ с индексом i признаком односимвольного элемента или первым символом двухсимвольного. В первом случае записываем

```
B[S[i]] := true;
```

во втором –

```
A[S[i], S[i+1]] := true;
```

После анализа строки достаточно подсчитать количество элементов, равных true, в обоих массивах. Такое решение имеет трудоёмкость O (N), где N – длина строки.

Участники, которые не умеют использовать массивы таким образом, могут находить количество различных элементов строки общим алгоритмом, имеющим трудоёмкость O (N^2), однако это решение, очевидно, не наберёт полного балла. Наконец, те, кто не умеет выделять односимвольные и двухсимвольные химические элементы, могут попробовать написать решение для 25 % тестов.

Задача 4: День пряника

Ещё одна задача, которая допускает множество частичных решений, в зависимости от подготовки участников позволяющих набрать неполное количество баллов.

Для прямоугольных пряников критерий того, что они поместятся в коробку, прост – диагональ пряника не должна превосходить диаметр коробки. Однако для того, чтобы не потерять точность, имеет смысл рассчитывать квадраты расстояний (которые будут целыми числами):

$$A^2 \leq D^2 - B^2 \quad (1)$$

где A, B – стороны прямоугольника, D – диаметр коробки.

В случае треугольного угощения необходимо рассмотреть два случая. Если треугольник – тупоугольный, то достаточно, чтобы большая сторона не превосходила диаметра коробки (в этом случае диаметр окружности, описанной около треугольника, может быть и больше диаметра коробки, как показано на рис. 6).

В противном случае требуется, чтобы диаметр окружности, описанной около треугольника, не превосходил диаметра коробки. Пусть P – диаметр описанной окружности:

$$P = \frac{ABC}{2S}$$

Здесь A, B, C – длины сторон треугольника, S – его площадь.

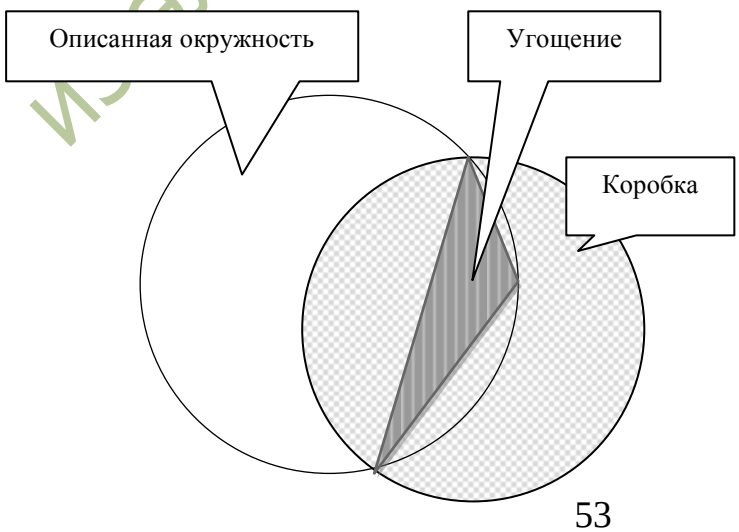

Рис. 6. Возможный вариант расположения тупоугольного треугольника

Поскольку $P \le D$, эту формулу можно преобразовать:

$$4S^2D^2 \ge A^2B^2C^2$$

Возведение в квадрат, как и ранее, позволяет избежать потери точности.

Восьмиклассники, не изучавшие соответствующие формулы, могут решить задачу только для прямоугольных тестов.

В 80 % тестов для выполнения вычислений достаточно использовать тип int64. Для получения полного балла требуется умение либо реализовывать длинную арифметику с операцией умножения на целое число, либо, как указала А. И. Лапо, работать с типом extended (её решение также берёт все тесты). В качестве иллюстрации приведём пример реализации длинной арифметики из авторского решения:

Описание типа «длинное число»:

```
type
  BigInt = record
    Dig: array[1..1000] of int64;
  // каждый элемент массива – одна десятичная цифра
    Size: integer;
  // количество значащих цифр
  end;
```

Процедура умножения:

```
procedure Mul(var A:BigInt; B: int64);
var
  i: integer;
  C: int64;
begin
  for i := 1 to A.Size do
    A.Dig[i] := A.Dig[i]*B;
  C := 0;
  for i := 1 to A.Size do begin
    A.Dig[i] := (A.Dig[i]+C);
    C := A.Dig[i] div 10;
    A.Dig[i] := A.Dig[i] mod 10;
  end;
  while C>0 do begin
    A.Size := A.Size+1;
    A.Dig[A.Size] := C mod 10;
    C := C div 10;
  end;
```

```
end;
```

Несложное преобразование формул для расчёта позволяет обойтись без реализации сложения и вычитания. Так, вместо формулы (1) можно использовать формулу

$$A^2 \le (D - B) \times (D + B)$$

Для других случаев выводятся аналогичные формулы.

2012/2013 учебный год

Задача 1: Построить римское число

Возможны не менее двух подходов к решению этой задачи.

Первый подход. Заводим двумерный массив Q размерности 3999×7. Элемент $Q[i, j]$ будет содержать количество символов, соответствующих j-й допустимой латинской букве в римской записи числа i. Напомним, что допустимыми буквами являются 'M', 'D', 'C', 'L', 'X', 'V', 'I'.

Заполняем этот массив ещё до чтения входного файла. Это позволит не выполнять многократно одну и ту же работу.

Для каждого прочитанного подтеста аналогичным образом заполняем одномерный массив P из 7 элементов. Затем в цикле от 3999 до 1 пытаемся найти такую строку с номером i в массиве Q, что для всех j $Q[i, j] = P[j]$. Если такая строка найдена, выводим соответствующее i, в противном случае, по завершении цикла выводим ноль.

Второй подход. После чтения очередного подтеста и заполнения массива P так же, как и в первом подходе, пытаемся построить максимальное из возможных римских чисел. Максимальное количество допустимых букв, используем для римской записи числа, образует массив

```
F: array [1..7] of integer = (4, 1, 4, 1, 4, 1, 3);
```

Если $F[j] > P[j]$ хотя бы для одного j, число построить нельзя. В противном случае строим вначале римское представление числа: в цикле от 1 до 7 переносим $P[i]$ букв в результирующую строку. Однако перед переносом четвёртой такой буквы мы должны убедиться, что $P[i+1] = 0$, а $P[i+2] = 1$. Если приведенное условие неверно, число построить нельзя, иначе вначале переносим $(i+2)$-ю букву, а затем – i-ю. После завершения построения римского числа остаётся преобразовать его в десятичную систему счисления и вывести полученный результат.

Задача 2: Две матрицы

Решение «в лоб», которое предполагает заполнение обеих матриц в соответствии с условиями задачи и последующее попарное сравнение каждого элемента этих матриц, из-за ограничений по памяти работает только для $M, N \leq 1000$ и берёт только 50 % тестов.

Следовательно, надо искать решение, которое позволило бы обойтись без явного построения матриц.

Значение элементов матриц A_{ij} и B_{ij} вычисляется по формулам

$$A_{ij} = N \times (i-1) + j, \;\; B_{ij} = M \times (j-1) + i$$

Приравняв эти величины и выполнив несложные преобразования, получим (предполагая, что $N \leq M$)

$$i = 1 + \frac{M-1}{N-1} \times (j-1)$$

Левая часть равенства – целая, поэтому найти нужное j можно только, если дробь – тоже целая. Это достигается, когда она равна D = НОД(M–1, N–1). Тогда мы получаем D+1 серию из различных пар (i, j).

Случай M=1 или N=1 следует рассмотреть отдельно.

Тесты подобраны таким образом, что попытка вывести ответ 2 для любых входных данных наберёт только 10 баллов.

Задача 3: Резисторы

Прежде всего, построим модель ситуации.

Как известно, закон Ома может быть записан в виде формулы

$$U = T_i \times R_i$$

где нам известна лишь величина тока T_i. Но поскольку напряжение одинаково для получения информации о каждом из резисторов, можно сделать вывод, что чем больше величина тока, тем меньше сопротивление соответствующего резистора.

С другой стороны, суммарное сопротивление схемы из пяти параллельно соединённых резисторов, рассчитывается по формуле

$$\frac{1}{R} = \sum_{i=1}^{5} \frac{1}{R_{s_i}}, 1 \leq s_i \leq N$$

Отсюда видно, что чем меньшие сопротивления мы выбираем в правой части формулы, тем меньшей будет величина R, и, следовательно,

тем большей будет сила тока, протекающего по неразветвлённой части цепи.

Рассмотрим обоснование этого на примере из условия задачи.

Пусть, напряжение, вырабатываемое источником, равно 10 (можно взять, конечно же, любое другое значение). Тогда сопротивления имеющихся транзисторов составляют 10/3 (три резистора), 10/2 (один резистор), 10/1 (шесть резисторов).

При применении резисторов с наименьшим сопротивлением получаем

$$\frac{1}{R} = 3 \times \frac{3}{10} + 1 \times \frac{2}{10} + 1 \times \frac{1}{10};\ R = \frac{10}{12}$$

При применении резисторов с наибольшим сопротивлением получаем

$$\frac{1}{R} = 5 \times \frac{1}{10};\ R = 2$$

В первом случае ток, протекающий по неразветвлённой части цепи, равен 12, во втором – 5.

Таким образом, задача свелась к нахождению индексов пяти элементов массива с наибольшими значениями T_i. Однако большие допустимые значения для N вместе с жёсткими ограничениями на время и память не дают возможности отсортировать этот массив. Так сортировка «пузырьком» набирает в авторском решении 50 баллов, использование алгоритма sort из библиотеки STL – 76 баллов, и даже использование гораздо более быстрого алгоритма nth_element из этой же библиотеки не набирает полный балл.

Для полного решения задачи можно либо воспользоваться следующим подходом: создаём двумерный массив P размерностью 5 × 2, первый столбец которого содержит значения T_i, а второй – индексы соответствующих резисторов в исходном массиве. Считываем из входного файла первые пять значений, заполняем массив P, сортируем его по убыванию T_i. Для каждого из последующих значений определяем, можно ли его занести его в массив P с соблюдением упорядоченности, и в случае положительного ответа заносим его в этот массив, «вытесняя» последнюю строку. После завершения ввода достаточно вывести второй столбец массива P.

Такой алгоритм имеет трудоёмкость O (N), что достаточно для полного решения задачи при заданных ограничениях на время и память.

Можно также применить алгоритм partial_sort из библиотеки STL, хотя авторское решение, основанное на его использовании, требует 0.45

– 0.47 секунды на больших тестах. Значит, любая небрежность в написании программы может быть чревата потерей баллов…

Задача 4: Юный чертёжник

Известно, что коэффициенты прямой $Ax + By + C = 0$, проходящей через две точки (x_1, y_1) и (x_2, y_2), можно рассчитать по формулам

$$A = y_1 - y_2$$
$$B = x_2 - x_1$$
$$C = x_1 y_2 - x_2 y_1$$

Однако две совпадающие прямые могут иметь различные наборы коэффициентов. Так, прямые

$$3x + 9y - 12 = 0$$

и

$$-x - 3y + 4 = 0$$

на самом деле совпадают. Для того, чтобы исключить двусмысленность, во-первых, делим полученные коэффициенты на НОД(A, B), и во-вторых, изменением знака всех коэффициентов добиваемся того, что $A > 0$ или ($A = 0$ и $B > 0$).

Пусть описанным выше образом получены коэффициенты A_1, B_1, C_1, A_2, B_2, C_2 для каждой из двух прямых. Тогда

- если $A_1 = A2$, $B_1 = B_2$, $C_1 = C_2$, то прямые совпадают;
- если $A_1 = A2$, $B_1 = B_2$, $C_1 \neq C_2$, то прямые параллельны;
- в противном случае прямые пересекаются.

Содержание

Учебное издание

Кашкевич Сергей Иванович
Климович Владимир Владимирович

Задачи школьных олимпиад по информатике

Учебно-методическое пособие для студентов факультета прикладной математики и информатики

В трёх частях

Ч а с т ь 1

Задачи районных олимпиад по информатике г. Минска (2005 – 2012 годы)

В авторской редакции

Ответственный за выпуск *В.В. Климович*

Подписано в печать ____.04.2013. Формат 60×84/16. Бумага офсетная.
Гарнитура Таймс. Усл. печ. л. 3,49. Уч.-изд. л. 2,76. Тираж 50 экз. Зак. ____.

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.