

Муниципальный этап Всероссийской олимпиады  
школьников по информатике  
в 2014–2015 учебном году  
8 класс

Время выполнения задач — 4 часа  
Ограничение по времени — 2 секунды на тест  
Ограничение по памяти — 64 мегабайта

Муниципальный этап  
Всероссийской олимпиады школьников  
по информатике

в 2014–2015 учебном году

Разборы решений и идеи тестов

**8.1. «Носим кирпичи».** Семья Торопыжкиных строит на даче новый сарай, для чего привезено  $n$  кирпичей. Петя Торопыжкин за один раз переносит не более  $m$  кирпичей, а его брат — не более  $k$  кирпичей. Переноской кирпичей будет заниматься кто-то один из них. Какое минимальное количество ходок нужно будет сделать для переноски всех имеющихся кирпичей?

**Формат входа:** В единственной строке через пробел заданы три целых числа  $n$ ,  $m$  и  $k$  ( $1 \leq n, m, k \leq 10^4$ ).

**Формат выхода:** Выдайте единственное целое число — минимальное количество ходок, за которое один из братьев перенесёт все кирпичи.

**Пример**

| <u>Вход:</u> | <u>Выход:</u> |
|--------------|---------------|
| 13 7 4       | 2             |

**8.2. «Решаем задачи».** В течение октября, готовясь к районному туру олимпиады по информатике, Петя Торопыжкин решает задачи. В  $n$ -й день октября ( $1 \leq n \leq 31$ ) он прорешивает  $k(n) = an^2 + bn + c$  задач. Если коэффициенты таковы, что в какой-то день значение  $k(n)$  отрицательно, то это означает, что в этот день он не решил ни одной задачи. По заданным целым коэффициентам  $a$ ,  $b$ ,  $c$  найдите, сколько задач прорешал Петя за тридцать один день октября.

**Формат входа:** В единственной строке через пробел заданы три целых числа  $a$ ,  $b$ ,  $c$ ; каждое по модулю не превосходит 1000.

**Формат выхода:** Выведите единственное целое неотрицательное число — количество задач, решённых Петей.

**Пример**

| <u>Вход:</u> | <u>Выход:</u> |
|--------------|---------------|
| 1 -30 200    | 1901          |

**8.3. «Книги в макулатуру».** В результате сбора макулатуры, проведённого в школе у Пети Торопыжкина, кроме всей прочей бумажной продукции было собрано  $n$  книг. Таская их в машину, Петя обнаружил, что их нечётное количество, и задумался о том, сколько всего страниц в сумме в самой тонкой книге, в самой толстой и в самой «средней» (то есть в книге, оказавшейся бы в середине ряда,

упорядоченного по толщине). Конечно, таская книги, он узнал, сколько страниц в каждой из них.

**Формат входа:** В первой строке задано единственное число  $n$  — количество книг ( $n$  — нечётное целое число,  $3 \leq n \leq 99999$ ). В следующей строке через пробел перечислены количества страниц в каждой из книг (в каком-то порядке). Число страниц в книге заключено в диапазоне от 1 до 1500.

**Формат выхода:** Выведите единственное целое число — сумму количества страниц в самой тонкой, самой толстой и «средней» книгах.

**Пример**

|                   |               |
|-------------------|---------------|
| <u>Вход:</u>      | <u>Выход:</u> |
| 5                 | 433           |
| 100 23 100 310 70 |               |

**8.4. «Прогулка по магазинам».** Родители поручили Пете Торопыжку купить к празднику колбасу и торт, выдав на руки некоторую достаточно большую сумму денег и сказав, что сдачу Петя может забрать себе. Поэтому интерес Пети состоит в том, чтобы купить нужные продукты подешевле (конечно же, не в ущерб качеству!). Около его дома расположено  $n$  магазинов, цены на колбасу и торты в которых Пете известны. Ему нужно узнать длину кольцевого маршрута, по которому ему придётся пройти от дома до магазина (или двух магазинов, если дешёвая колбаса продаётся не там же, где дешёвые торты) и обратно до дома. Петя не ограничен во времени, так что ему нет нужды находить самый короткий маршрут.

Следует отметить, что поскольку Петя живёт в городе, то он вынужден перемещаться между двумя точками не напрямик, а по прямоугольной сетке улиц. Поэтому для вычисления расстояния между двумя точками  $A(x_a, y_a)$  и  $B(x_b, y_b)$  применяется формула так называемого *манхэттенского расстояния*:  $d_1(A, B) = |x_a - x_b| + |y_a - y_b|$ . Действительно: сначала надо по улице, параллельной оси абсцисс, пройти расстояние  $|x_a - x_b|$ , а потом по улице, параллельной оси ординат, — расстояние  $|y_a - y_b|$ . Сочетание «манхэттенское расстояние» происходит от того, что такой сеткой улиц обладает Манхэттен, район города Нью-Йорка.

**Формат входа:** В первой строке через пробел заданы целые числа  $x_0$  и  $y_0$  — абсцисса и ордината дома Пети Торопыжкина. Во второй строке задано целое число  $n$  — количество магазинов ( $1 \leq n \leq 10^4$ ). В каждой из следующих  $n$  строк четвёркой целых чисел, разделённых пробелами, задаётся информация об очередном магазине: абсцисса и ордината положения магазина, цена на колбасу и цена на торт. Все координаты целые, по модулю не превосходящие  $10^4$ . Гарантируется, что никакие два магазина не расположены в одной точке и никакой магазин не совпадает с домом Пети. Цены принадлежат диапазону от 1 до 1000.

**Формат выхода:** Выдайте в одной строке через пробел три целых числа: номер магазина, в котором Пете стоит покупать колбасу, номер магазина, в котором Пете

стоит покупать торт, и длину маршрута, который придётся пройти Пете. Номера магазинов должны соответствовать исходному списку; нумерация начинается от 1. Если возможных ответов несколько, выдайте любой из них.

**Пример**

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| 1 2          | 3 4 24        |
| 4            |               |
| 3 7 30 70    |               |
| -1 -10 60 60 |               |
| 5 0 30 100   |               |
| 0 7 40 60    |               |

**8.5. «Степень строки».**  $n$ -й степенью строки в математике называют строку, получаемую  $n$ -кратным записыванием данной строки. Например, строка ABCD во второй степени — это ABCDABCD, а XY в четвёртой — это XYXYXYXY. Заметим, что строку XYXYXYXY также можно рассматривать как квадрат строки XYXY. Для данной строки определите, является ли она какой-либо степенью какой-либо более короткой строки.

**Формат входа:** В единственной строке задана исходная непустая строка, имеющая длину не более 255 символов и состоящая только из заглавных символов латиницы.

**Формат выхода:** Если данная строка не является никакой степенью никакой более короткой строки, то нужно вывести -1. Если она является степенью, то нужно вывести эту степень и через пробел — строку-основание. Если есть несколько ответов, то нужно вывести тот, у которого степень наибольшая.

**Пример 1**

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| ABCABD       | -1            |

**Пример 2**

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| XYXYXYXY     | 4 XY          |

**Муниципальный этап Всероссийской олимпиады  
школьников по информатике  
в 2014–2015 учебном году  
9 класс**

*Время выполнения задач — 4 часа  
Ограничение по времени — 2 секунды на тест  
Ограничение по памяти — 64 мегабайта*

**9.1. «Собираем урожай».** В огороде на даче у Пети Торопыжкина выросло  $n$  кочанов капусты, которые нужно перетаскать в дом. Петя за одну ходку относит не более  $k_1$  кочанов, что занимает у него  $t_1$  минут. Его брат может унести не более  $k_2$  кочанов, причём у него ходка занимает  $t_2$  минут. Уборкой капусты займётся один из братьев (другой будет убирать морковь). За какое наименьшее время можно убрать всю капусту?

**Формат входа:** В единственной строке через пробел заданы пять целых чисел  $n, k_1, t_1, k_2, t_2$ , каждое из диапазона от 1 до  $10^4$ .

**Формат выхода:** Выведите единственное целое число — количество минут, которое займёт наискорейшая уборка капусты одним из братьев.

**Пример**

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| 15 4 3 5 10  | 12            |

**9.2. «Нужная сумма».** Иногда Пете Торопыжкину приходится решать задачи, которые имеют формализованную постановку. Вот одна из них: напишите программу, которая из заданного набора натуральных чисел выделяет два элемента, кратных 17, так, чтобы их сумма была кратна 3.

**Формат входа:** В первой строке задано единственное целое число  $n$  — размер набора ( $2 \leq n \leq 10^5$ ). В следующей строке через пробел заданы элементы этого набора — целые числа из диапазона от 1 до  $10^5$ . Числа в наборе могут повторяться.

**Формат выхода:** Если искомую пару чисел выделить можно, выведите эти числа через пробел в любом порядке. Если ответов несколько, выведите любой из них. Если выделить пару невозможно, выведите единственное число 0.

|                 |                 |                 |
|-----------------|-----------------|-----------------|
| <b>Пример 1</b> | <b>Пример 2</b> | <b>Пример 3</b> |
| <u>Вход:</u>    | <u>Вход:</u>    | <u>Вход:</u>    |
| <u>Выход:</u>   | <u>Выход:</u>   | <u>Выход:</u>   |
| 4               | 4               | 4               |
| 17 51 102 1     | 17 2 85 17      | 17 51 5 17      |

**9.3. «Обеспечение продовольствием».** Во время, свободное от учёбы и решения задач по информатике, Петя Торопыжкин иногда играет. В том числе и в

стратегии. Однажды он решил аккуратно проанализировать свой любимый экономический симулятор. В нём имеется  $n$  складов и  $m$  магазинов, в которых можно продавать продовольствие, хранящееся на складах. На  $i$ -м складе имеется запас в  $s_i$  единиц продовольствия. В  $j$ -м магазине потребность в продовольствии составляет  $d_j$  единиц, которые могут быть проданы по  $p_j$  денежных единиц каждая. Какую максимальную прибыль может получить Петя в этой игре? В один магазин продовольствие может доставляться с любого количества складов и с одного склада в любое количество магазинов.

**Формат входа:** В первой строке задано единственное целое число  $n$  — количество складов ( $0 \leq n \leq 10^5$ ). В следующей строке через пробел перечислены запасы продовольствия  $s_i$  на складах ( $0 \leq s_i \leq 10^4$ ). Общий запас продовольствия не превышает  $10^6$  единиц. В третьей строке задано целое число  $m$  — количество магазинов ( $0 \leq m \leq 10^5$ ). В четвёртой строке через пробел перечислены цены  $p_j$  на продовольствие в магазинах ( $0 \leq p_j \leq 10^3$ ). В пятой строке перечислены потребности магазинов в продовольствии  $d_j$  ( $0 \leq d_j \leq 10^4$ ).

**Формат выхода:** Выведите единственное целое число — максимальную прибыль от продажи продовольствия, которую может получить Петя в заданных условиях.

|                 |                 |
|-----------------|-----------------|
| <b>Пример 1</b> | <b>Пример 2</b> |
| <u>Вход:</u>    | <u>Вход:</u>    |
| <u>Выход:</u>   | <u>Выход:</u>   |
| 3               | 3               |
| 5 100 10        | 5 50 10         |
| 4               | 4               |
| 20 30 40 10     | 20 30 40 10     |
| 30 10 20 10     | 30 10 20 10     |

**9.4. «Переносы в расписании».** Расписание дел Пети Торопыжкина очень плотное. И все дела важные. Но тут внезапно возникло ОЧЕНЬ ВАЖНОЕ дело, которое (возможно) пересекается с частью уже запланированных мероприятий. По имеющемуся петиному расписанию на день и срокам начала и конца очень важного дела определите, с каким количеством дел пересекается по времени очень важное дело? (Если очень важное дело примыкает спереди или с конца к другому делу, считается, что они не пересекаются.)

**Формат входа:** В первой строке через пробел заданы моменты начала и окончания очень важного дела. Во второй строке записано единственное неотрицательное целое число  $n$  — количество дел в петином расписании ( $n \leq 1440$ ). В следующих  $n$  строках через пробел заданы сроки начала и окончания очередного дела. Дела заданы, вообще говоря, в произвольном порядке. Никакие два дела из расписания по времени не пересекаются между собой. Формат задания момента времени — hh:mm ( $0 \leq hh \leq 23, 0 \leq mm \leq 59$ ). Если час или минуты меньше десяти, то они могут снабжаться ведущим нулём, то есть допустимы записи 9:00, 09:00, 9:0 и 09:0. Петя считается занятым с начала минуты момента начала и

до конца минуты момента конца. Гарантируется, что время окончания каждого дела не раньше времени его начала.

**Формат выхода:** Выведите единственное целое число — количество дел, с которыми пересекается очень важное дело.

**Пример**

Вход:                      Выход:

10:00 11:59    2

4

12:00 12:29

10:20 10:39

8:30 09:59

11:30 11:59

**9.5. «Делимость на 11».** На парте в кабинете математики Петя обнаружил несколько длинных неотрицательных целых чисел. Он решил исследовать каждое число на делимость на 11: делится ли оно на 11 и, если нет, можно ли переставить в нём одну пару соседних цифр так, чтобы оно стало кратным 11. Помогите ему, напишите соответствующую программу. Напомним, что ведущие нули в записи числа не допускаются.

**Формат входа:** В первой строке задано целое число  $n$  — количество чисел, обнаруженных Петей ( $1 \leq n \leq 1000$ ). В следующих  $n$  строках по одному на строке даны десятичные записи целых чисел, больших нуля; десятичная запись состоит из не менее чем из 1 и не более чем из 255 цифр.

**Формат выхода:** На выходе должно быть столько же строк, сколько было чисел на входе. Если очередное число делится на 11, в соответствующей строке выведите 0. Если не делится, но можно переставить две соседних цифры так, чтобы оно стало кратным 11, выведите позицию первой цифры из пары, которую нужно поменять местами. Если число не делится на 11 и его нельзя сделать кратным 11 перестановкой пары соседних цифр, выведите -1. Если допустимых перестановок несколько, выведите минимальную позицию.

Отсчёт позиций в числе идёт от старшего разряда и начинается с 1.

**Пример**

Вход:                      Выход:

3                            0

396                        2

369                        -1

397

## Муниципальный этап Всероссийской олимпиады школьников по информатике в 2014–2015 учебном году 10 класс

Время выполнения задач — 4 часа

Ограничение по времени — 2 секунды на тест

Ограничение по памяти — 64 мегабайта

**10.1. «Приятная покупка».** На Новый Год родители пообещали Пете Торопыжкину купить не более двух предметов из трёх: (1) новый телевизор; (2) новый ноутбук; (3) новую игровую приставку. Петя оценил удовольствие, которое он получит от обладания каждым предметом в отдельности, а также от обладания каждой парой предметов (эти значения не учитывают удовольствие от каждой вещи в отдельности). Основываясь на этих знаниях, он высказал свои пожелания по покупке. Напишите программу, которая автоматизирует этот выбор.

**Формат входа:** В единственной строке перечислены через пробел шесть целых чисел — удовольствия от обладания телевизором, ноутбуком, приставкой, телевизором и ноутбуком, телевизором и приставкой, ноутбуком и приставкой (в указанном порядке). Каждое число принадлежит диапазону от  $-1000$  до  $+1000$ , включая крайние значения (и да, удовольствие может быть отрицательным!).

**Формат выхода:** Выведите один или два номера предметов, которые Петя рекомендует купить. Если наилучшая покупка будет состоять из двух предметов, разделите номера пробелом. Если покупок с одинаковым суммарным удовольствием несколько, выведите любую из них. Кроме того, хотя бы один предмет Петя выбрать должен, чтобы не обижать родителей.

**Пример 1**

Вход:                                      Выход:

10 10 10 -20 -20 -20    1

**Пример 2**

Вход:                                      Выход:

10 10 10 -20 5 10    3 2

**10.2. «Наибольшее подчисло».** Назовём натуральное число  $k$  *подчислом* натурального числа  $n$ , если десятичная запись числа  $k$  входит в десятичную запись числа  $n$ . Например, 123 является подчислом 671235, а 27 не является подчислом 529725. Напишите программу, которая по заданному натуральному числу находит его наибольшее двузначное подчисло. Напомним, что ведущие нули в записи числа не допускаются.

**Формат входа:** Задано единственное целое число из диапазона от 10 до  $10^9$ .

**Формат выхода:** Выведите наибольшее двузначное подчисло исходного числа.

**Пример**

Вход:                      Выход:

529725                    97

**10.3. «Подготовка к олимпиаде».** Готовясь к областной олимпиаде по информатике, Петя Торопыжкин решает задачи. Пока что он взялся за задачи не слишком сложные, так что решение каждой задачи занимает ровно  $t$  минут. Но помимо решения задач у него есть и другие дела, расписание которых известно. К сожалению, решение задачи должно идти непрерывно, так что если промежуток между двумя делами равен 40 минутам, а решение одной задачи занимает 30 минут, то в этот промежуток удастся решить только одну задачу, а ещё 10 минут потратить на другие дела. По имеющемуся расписанию дел Пети найдите, сколько задач в день он сможет решить. Гарантируется, что в начале суток и в конце их Петя занят — он спит (в расписании обязательно присутствует дело, начинающееся в 00:00, и присутствует дело, оканчивающееся в 23:59).

**Формат входа:** В первой строке через пробел заданы длительность  $t$  (в минутах) решения одной задачи ( $t$  — целое число из диапазона от 1 до 1440) и  $n$  — количество дел у Пети в расписании ( $1 \leq n \leq 1000$ ). В следующих  $n$  строках заданы пары моментов времени в формате `hh:mm` ( $0 \leq hh \leq 23$ ,  $0 \leq mm \leq 59$ ) — времена начала и окончания очередного дела. Дела заданы, вообще говоря, в произвольном порядке. Никакие два дела из расписания по времени не пересекаются между собой. Если час или минуты меньше десяти, то они могут снабжаться ведущим нулём, то есть допустимы записи `9:00`, `09:00`, `9:0` и `09:0`. Петя считается занятым с начала минуты момента начала и до конца минуты момента конца.

**Формат выхода:** Выведите единственное целое число — количество задач, которое Петя сможет решить при указанном расписании.

#### Пример

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| 5 3          | 11            |
| 11:00 23:59  |               |
| 00:00 9:59   |               |
| 10:30 10:30  |               |

**10.4. «Разбор предложения».** Петя Торопыжкин наполовину выполнил домашнее задание по английскому языку: набрал (естественно, латиницей) и распечатал предложения, которые нужно разобрать по составу. Набор осуществлялся в одну строку моноширинным шрифтом, то есть все символы имеют одинаковую ширину, включающую в себя свободное пространство вокруг символа и равную в данном случае 6 мм. Слова в предложении разделены в точности одним пробелом; в начале и конце текста пробелов нет. Каждое предложение завершается точкой.

Дальше нужно было сделать разбор этих предложений по составу, что не являлось слишком сложной задачей, поскольку все предложения имели стандартную структуру: в начале предложения шли определения, за ними — подлежащие, потом — сказуемые, а остальные слова до конца предложения были дополнениями. При этом однородные члены предложения разделялись запятой, то есть если в

конце предыдущего слова стояла запятая, то следующее слово являлось тем же членом предложения, что и предыдущее. Пробелы между предыдущим словом и запятой не ставятся, а после запятой ставится в точности один пробел.

Тут Петю заинтересовал вопрос: а какова суммарная длина линий, которые ему нужно будет нарисовать?

Напомним, что подлежащее подчёркивается одной прямой линией, сказуемое — двумя, дополнение — штриховой линией так, что штрихи длиной в полсимвола центрированы относительно символов. Определение подчёркивается волнистой линией, но Петя вместо неё рисует «прямоугольную волну»: в начале слова рисует вертикально вниз отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вверх отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вниз отрезок на 1 мм, и т.д. Прямоугольная волна всегда заканчивается вертикальным отрезком. Знаки препинания не подчёркиваются. Например:

Beautiful flowers smell tasty, delicately.

(Везде между словами стоит ровно по одному пробелу.)

**Формат входа:** Единственная строка содержит текст, удовлетворяющий условию задачи. Длина текста лежит в диапазоне от 0 до 255 символов.

**Формат выхода:** Выведите единственное целое число — суммарную длину (в миллиметрах) линий, которые нарисует Петя.

#### Пример

Вход:

Beautiful flowers smell tasty, delicately.

Выход:

211

**10.5. «Делимость на 4».** У Пети Торопыжкина есть набор из  $n$  целых чисел. Он выбирает из них несколько так, чтобы их сумма была кратной 4. Каково максимальное значение такой суммы?

**Формат входа:** В первой строке через пробел задано целое число  $n$  из диапазона от 4 до  $10^4$ . Во второй строке через пробел задано  $n$  целых чисел, не превосходящих по модулю  $10^4$ .

**Формат выхода:** Выведите единственное целое число — максимальное значение суммы некоторых исходных чисел, кратной 4.

#### Пример

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| 5            | 8             |
| 1 2 3 4 -5   |               |

**Муниципальный этап Всероссийской олимпиады  
школьников по информатике  
в 2014–2015 учебном году  
11 класс**

*Время выполнения задач — 4 часа  
Ограничение по времени — 2 секунды на тест  
Ограничение по памяти — 64 мегабайта*

**11.1. «Модель Эйфелевой башни».** Петя Торопыжкин решил построить модель Эйфелевой башни, для чего ему нужно купить  $p$  металлических прутков, стоимостью  $r$  рублей каждый. В  $n$ -й день от начала покупок ( $n \geq 1$ ) родители выдают ему на карманные расходы  $k(n) = an^2 + bn + c$  рублей, из которых он максимальное количество тратит на покупку заготовок, а оставшуюся неистраченную сумму помещает в копилку. Если коэффициенты таковы, что в какой-то день значение  $k(n)$  отрицательно, то это означает, что в этот день ему карманных денег не дали, и он ничего не покупает и ничего в копилку не кладёт. Сколько денег окажется в копилке по окончании покупки нужного количества заготовок?

**Формат входа:** В первой строке через пробел заданы целые числа  $p$  и  $r$  ( $1 \leq p, r \leq 1000$ ). Во второй строке через пробел заданы три целых числа  $a, b, c$ ;  $a > 0$ , каждое число по модулю не превосходит 1000.

**Формат выхода:** Выведите единственное целое число — финальные петины накопления.

| Пример 1     |               | Пример 2     |               |
|--------------|---------------|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> | <u>Вход:</u> | <u>Выход:</u> |
| 16 5         | 7             | 17 5         | 23            |
| 1 -14 45     |               | 1 -14 45     |               |

**Примечание:** В примерах деньги даются Пете с 1-го по 4-й дни и затем с 10-го и далее. В первом примере он осуществляет закупку точно на 11-й день. Во втором примере ему нужно купить ещё одну заготовку на 12-й день, что он и делает, помещая оставшиеся деньги ( $21 - 5 = 16$  рублей) в копилку в дополнение к тем 7-ми, что там есть на конец 11-го дня.

**11.2. «Неубывающая последовательность».** Имеется упорядоченный набор из  $n$  целых чисел, не превосходящих по модулю 1000. Петя Торопыжкин может прибавлять к любому числу любую неотрицательную величину. Он хочет достичь того, чтобы этот набор представлял собой неубывающую последовательность (каждый последующий член должен быть не меньше предыдущего). Каково наименьшее значение суммы величин, которые необходимо прибавить к элементам набор для достижения этой цели?

**Формат входа:** В первой строке задано целое число  $n$  ( $1 \leq n \leq 1000$ ). В следующей строке через пробел заданы  $n$  целых чисел, по модулю не превосходящих 1000.

**Формат выхода:** Выдайте единственное целое неотрицательное число — сумму величин, которые надо добавить к членам набора, чтобы получить неубывающий набор.

**Пример**

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| 5            | 6             |
| 2 5 1 3 8    |               |

**11.3. «Странный максимум».** Пете Торопыжкину разонравились нечётные цифры (то есть цифры 1, 3, 5, 7 и 9). И теперь он сравнивает неотрицательные целые числа следующим образом: сначала из чисел вычёркиваются все нечётные цифры, затем вычёркиваются ведущие нули (если они появились), наконец, если от числа ничего не осталось, то оно полагается равным нулю. После такого «прореживания» числа сравниваются по обычным правилам. Если два числа получились равными, то Петя выбирает в качестве большего первое из исходных чисел. Напишите программу, которая находила бы максимум из набора чисел по петиним правилам сравнения. При поиске максимума в наборе каждое последующее число сравнивается как второе с максимумом из предыдущих чисел.

**Формат входа:** В первой строке задано целое число  $n$  — количество чисел ( $1 \leq n \leq 1000$ ). В следующих  $n$  строках по одному на строке записаны числа, каждое из которых состоит не менее чем из одной и не более чем из 255 цифр.

**Формат выхода:** Выведите единственную строку, содержащую максимальное (в петиним смысле) число из данного набора.

**Пример**

|              |               |
|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> |
| 4            | 597631        |
| 31595        |               |
| 1997053      |               |
| 597631       |               |
| 96135        |               |

**11.4. «Прохождение игрушки».** Во время, свободное от учёбы и решения задач по информатике, Петя Торопыжкин иногда играет. В том числе и в стрельялки. Однажды он решил аккуратно проанализировать свой любимый шутер. В нём имеется  $n$  уровней. Для прохождения  $k$ -го уровня требуется  $b_k$  патронов. В начале игры игрок обладает  $s$  патронами. Для восстановления запаса в конце  $k$ -го уровня можно открыть секретную комнату и дополнительно получить  $a_k$  патронов, которые можно использовать только на следующих уровнях. Однако это

требует времени, и Петя хочет сократить посещение секретных комнат, насколько это возможно. Соответственно, игрушка считается успешно пройденной, если перед началом каждого из уровней запас патронов у игрока не менее чем требуемое количество. Помогите Пете, напишите программу, которая по имеющимся данным об игре укажет какое минимальное количество секретных комнат надо будет открыть и на каких именно уровнях.

**Формат входа:** В первой строке через пробел указаны два целых числа  $s$ , начальный запас патронов, и  $n$ , количество уровней в игре ( $0 \leq s \leq 10^4$ ,  $1 \leq n \leq 10^5$ ). В следующих  $n$  строках приведена информация о каждом из уровней игры — пара чисел  $b_k$  и  $a_k$ , расход патронов на  $k$ -м уровне и дополнительный запас патронов на  $k$ -м уровне ( $0 \leq b_k, a_k \leq 10^4$ ).

**Формат выхода:** В первой строке выведите количество уровней, на которых необходимо открывать секретную комнату для пополнения запаса патронов. Если количество таких уровней больше нуля, во второй строке через пробел выведите номера этих уровней в любом порядке. Если решений несколько, выведите любое из них. Если прохождение игры невозможно, выведите единственное число -1.

| Пример 1     |               | Пример 2     |               | Пример 3     |               |
|--------------|---------------|--------------|---------------|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> | <u>Вход:</u> | <u>Выход:</u> | <u>Вход:</u> | <u>Выход:</u> |
| 100 2        | 0             | 10 3         | 1             | 10 3         | -1            |
| 10 10        |               | 10 20        | 1             | 10 10        |               |
| 10 10        |               | 10 10        |               | 20 10        |               |
|              |               | 10 10        |               | 10 10        |               |

**11.5. «Организация связи».** Кроме стрелялок Петя Торопыжкин иногда играет и в стратегические игры. В одной из них важным является организация связи между опорными пунктами. При очередном запуске этой игры Пете понадобилось обеспечить двустороннюю связь между пунктами  $A$  и  $B$ , расположенными на прямой. Для этого он может устанавливать ретрансляционные станции в некоторых из оговоренных точек  $C_i$ , расположенных на той же прямой. На каждой станции он должен предусмотреть два узконаправленных передатчика, транслирующих сигнал в направлении от  $A$  к  $B$  и от  $B$  к  $A$ . Кроме, конечно, станций, расположенных в самих пунктах  $A$  и  $B$ : там нужно установить только по одному передатчику, вещающему в соответствующую сторону. Всего имеется  $m$  видов передатчиков; передатчик  $j$ -го вида имеет дальность вещания  $d_j$ , а его установка стоит  $r_j$  денежных единиц. Естественно, Петя желает установить связь, потратив наименьшее количество денежных ресурсов. Сможет ли он организовать связь в указанных условиях и, если да, то в какую сумму ему это обойдётся?

**Формат входа:** В первой строке заданы два целых числа  $x_A$  и  $x_B$  — координаты пунктов  $A$  и  $B$ . Во второй строке задано число  $n$  — количество промежуточных точек  $C_i$  ( $0 \leq n \leq 3000$ ). В третьей строке в каком-то порядке через пробел заданы координаты  $x_i$  точек  $C_i$ . В четвёртой строке задано число  $m$  — количество

типов передатчиков ( $1 \leq m \leq 3000$ ). В следующих  $m$  строках описываются параметры каждого вида передатчиков — пары целых чисел  $d_j$  и  $r_j$  ( $1 \leq d_j \leq 10^6$ ,  $0 \leq r_j \leq 10^4$ ). Все координаты являются целыми числами, по модулю не превосходящими  $10^5$ . Гарантируется, что никакие две точки  $C_{i'}$  и  $C_{i''}$  не совпадают друг с другом и не совпадают с пунктами  $A$  и  $B$ .

**Формат выхода:** Выведите единственное целое число — наименьшую стоимость установленных передатчиков, обеспечивающих двустороннюю связь между пунктами  $A$  и  $B$ . Если связь установить невозможно, выведите -1.

| Пример 1     |               | Пример 2     |               |
|--------------|---------------|--------------|---------------|
| <u>Вход:</u> | <u>Выход:</u> | <u>Вход:</u> | <u>Выход:</u> |
| 0 10         | 80            | 0 10         | -1            |
| 3            |               | 3            |               |
| 2 6 9        |               | 2 6 9        |               |
| 3            |               | 3            |               |
| 2 10         |               | 2 10         |               |
| 5 15         |               | 3 15         |               |
| 1 8          |               | 1 8          |               |

**Муниципальный этап Всероссийской олимпиады  
школьников по информатике  
в 2014–2015 учебном году  
8 класс. Разбор решений и идеи тестов**

Далее в разборах используются обозначения типов: short int — 2-байтовый целый тип (integer в TurboPascal/BorlandPascal и int в BorlandC++), int — 4-байтовый целый тип (longint в TurboPascal/BorlandPascal/FreePascal, long int в BorlandC++, integer в Delphi/PascalABC и int в VisualC++/C++ Builder), int64 — 8-байтный целый (отсутствует в BorlandPascal и BorlandC++, но присутствует в системах программирования под Windows).

Задачи оцениваются по 100 баллов за каждую; стало быть, полная стоимость пакета — 500 баллов. В рамках одной задачи все тесты считать равноценными.

**8.1. «Носим кирпичи».** Семья Торопыжских строит на даче новый сарай, для чего привезено  $n$  кирпичей. Петя Торопыжский за один раз переносит не более  $m$  кирпичей, а его брат — не более  $k$  кирпичей. Переноской кирпичей будет заниматься кто-то один из них. Какое минимальное количество ходок нужно будет сделать для переноски всех имеющихся кирпичей?

Программный комитет оценивает эту задачу как весьма простую. Вычисляем количество ходок, которые надо сделать Пете, количество ходок, которые нужно сделать брату, и выбираем меньшее из этих чисел. Некоторая тонкость состоит в вычислении количества ходок; если выражать математически, то это будут величины  $\lceil n/m \rceil$  и  $\lceil n/k \rceil$ , где  $\lceil \cdot \rceil$  — операция округления вверх. В большинстве языков эта операция реализована, однако лучше не прибегать к вещественным вычислениям и написать код примерно такой:

```
res := n div m;  
if n mod m <> 0 then res := res+1;
```

**Идеи тестов:**

1. Минимальный тест  $n = m = k = 1$ ;
2. Минимальный тест  $n = 10, m = k = 1$ ;
3. Минимальный тест  $n = m = 1, k = 10$ ;
4. Минимальный тест  $n = 1, m = 10, k = 1$ ;
5. Максимальный тест  $n = m = k = 10^4$ ;
6. Максимальный тест  $n = 10, m = k = 10^4$ ;
7. Максимальный тест  $n = m = 10^4, k = 10$ ;
8. Максимальный тест  $n = 10^4, m = 10, k = 10^4$ ;
- 9–20. Случайные тесты.

**8.2. «Решаем задачи».** В течение октября, готовясь к районному туру олимпиады по информатике, Петя Торопыжский решает задачи. В  $n$ -й день октября

( $1 \leq n \leq 31$ ) он прорешивает  $k(n) = an^2 + bn + c$  задач. Если коэффициенты таковы, что в какой-то день значение  $k(n)$  отрицательно, то это означает, что в этот день он не решил ни одной задачи. По заданным целым коэффициентам  $a, b, c$  найдите, сколько задач прорешал Петя за тридцать один день октября.

Данная задача представляется простой: для её решения необходимо аккуратно организовать цикл суммирования величин, вычисляемых по указанной формуле, с проверкой их неотрицательности.

**Идеи тестов:**

1. Зависимость — константа, для всех  $n$  значения отрицательны:  $a = b = 0, c = -1$ ;
2. Зависимость возрастающая линейная, для всех  $n$  значения отрицательны:  $a = 0, b = 1, c = -100$ ;
3. Зависимость убывающая линейная, для всех  $n$  значения отрицательны:  $a = 0, b = -1, c = -100$ ;
4. Зависимость квадратичная, «рога вверх», для всех  $n$  значения отрицательны:  $a = 1, b = -33, c = -70$ ;
5. Зависимость квадратичная, «рога вниз», для всех  $n$  значения отрицательны:  $a = -1, b = -33, c = -70$ ;
6. Зависимость — константа, для всех  $n$  значения нулевые:  $a = b = c = 0$ ;
7. Зависимость возрастающая линейная, для некоторых  $n$  значения отрицательны:  $a = 0, b = 1, c = -10$ ;
8. Зависимость убывающая линейная, для некоторых  $n$  значения отрицательны:  $a = 0, b = -1, c = 10$ ;
9. Зависимость возрастающая линейная, для всех  $n$  значения положительные:  $a = 0, b = 1, c = 10$ ;
10. Зависимость убывающая линейная, для всех  $n$  значения положительные:  $a = 0, b = -1, c = 100$ ;
11. Зависимость квадратичная, «рога вверх», промежуток отрицательных значений в начале:  $a = 2, b = -24, c = -90$ ;
12. Зависимость квадратичная, «рога вверх», промежуток отрицательных значений в середине:  $a = 2, b = -66, c = 400$ ;
13. Зависимость квадратичная, «рога вверх», промежуток отрицательных значений в конце:  $a = 3, b = -121, c = 560$ ;
14. Зависимость квадратичная, «рога вверх», все значения положительные:  $a = 1, b = 3, c = 1$ ;
15. Зависимость квадратичная, «рога вниз», промежуток отрицательных значений в начале:  $a = -3, b = 121, c = -560$ ;
16. Зависимость квадратичная, «рога вниз», промежуток отрицательных значений в середине:  $a = -2, b = 66, c = -400$ ;
17. Зависимость квадратичная, «рога вниз», промежуток отрицательных значений в конце:  $a = -2, b = 24, c = 90$ ;

18. Зависимость квадратичная, «рога вниз», все значения положительны:  
 $a = -1$ ,  $b = 33$ ,  $c = 70$ ;  
 19. Максимальный тест,  $a = b = c = -1000$ ;  
 20. Максимальный тест,  $a = b = c = 1000$ ;  
 21–25. Случайные большие тесты.

**Примечание:** Ответы к тестам с 1 по 18 входят в short int. Прочие тесты подразумевают ответ, входящий в int.

**8.3. «Книги в макулатуру».** В результате сбора макулатуры, проведённого в школе у Пети Торопыжкина, кроме всей прочей бумажной продукции было собрано  $n$  книг. Таская их в машину, Петя обнаружил, что их нечётное количество, и задумался о том, сколько всего страниц в сумме в самой тонкой книге, в самой толстой и в самой «средней» (то есть в книге, оказавшейся бы в середине ряда, упорядоченного по толщине). Конечно, таская книги, он узнал, сколько страниц в каждой из них.

Идейно данная задача является лёгкой идейно, однако для её решения требуется привлечь процедуру сортировки, что делает её для восьмиклассников средней по сложности.

Идея решения крайне проста: отсортируем (по возрастанию или по убыванию) полученный массив чисел и найдём сумму первого, последнего и  $(n + 1)/2$ -го элементов (при нумерации элементов массива от единицы; при нумерации от нуля надо брать элемент с индексом  $(n - 1)/2$ ).

Следует также отметить, что при размерах сортируемого массива более 14000 – 15000 не будут проходить по времени «наивные» алгоритмы сортировки, имеющие квадратичную сложность: сортировка пузырьком, сортировка вставкой и т.п. Полный балл можно получить, используя «умные» алгоритмы, например, быструю сортировку (сортировку Хоара), или же используя библиотечные функции сортировки (в которых, собственно, как раз реализована быстрая сортировка). Те, кто будут использовать неоптимальные алгоритмы, получают около 65% полного балла.

Альтернативный путь решения, более оптимальный по времени работы и более простой по реализации, хотя и идейно более сложный, состоит в использовании идеи линейной сортировки. Поскольку количество различных объектов (возможных количеств страниц в книге) не так велико, то создав массив  $a[\cdot]$  из 1500 элементов, можно заполнить его ячейки за время, пропорциональное количеству книг (то есть  $n$ ). В  $i$ -й ячейке этого массива хранится число книг с количеством страниц  $i$ . Поэтому заполнение этого массива состоит в том, чтобы после прочтения количества страниц в очередной книжке увеличить на 1 соответствующую ячейку массива (вначале массив  $a[\cdot]$  заполнен нулями).

После заполнения массива имеем, что количество страниц в самой тонкой книге — это номер первой ненулевой ячейки массива, а количество страниц в самой

толстой — номер последней ненулевой ячейки. Некоторое затруднение вызывает нахождение количества страниц у «средней» книги, поскольку в отличие от предыдущего пути решения у нас нет явной упорядоченной последовательности книг.

Для нахождения этой величины надо найти ячейку с номером  $i^*$  таким, что

$$\sum_{k=1}^{i^*-1} a[k] < (n + 1)/2 \quad \text{и} \quad \sum_{k=1}^{i^*} a[k] \geq (n + 1)/2,$$

то есть ячейку, содержащую «среднюю» книгу. Этот поиск можно провести, суммируя последовательно элементы массива  $a[\cdot]$  и остановив перебор элементов, когда выполнится второе неравенство.

#### Идеи тестов:

1. Минимальный тест,  $n = 3$ , все элементы одинаковы;
2. Минимальный тест,  $n = 3$ , все элементы различны;
3.  $n = 11$ , все элементы одинаковы;
4.  $n = 11$ , все элементы различны;
5.  $n = 11$ , есть одинаковые элементы;
- 6–13. Случайные тесты, позволяющие использовать «наивные» алгоритмы сортировки;
14. Максимальный тест,  $n = 99999$ , все элементы одинаковы;
15. Максимальный тест,  $n = 99999$ , не все элементы одинаковы;
- 16–20. Случайные тесты, требующие использования оптимальных сортировок.

**8.4. «Прогулка по магазинам».** Родители поручили Пете Торопыжкину купить к празднику колбасу и торт, выдав на руки некоторую достаточно большую сумму денег и сказав, что сдачу Петя может забрать себе. Поэтому интерес Пети состоит в том, чтобы купить нужные продукты подешевле (конечно же, не в ущерб качеству!). Около его дома расположено  $n$  магазинов, цены на колбасу и торты в которых Пете известны. Ему нужно узнать длину кольцевого маршрута, по которому ему придётся пройти от дома до магазина (или двух магазинов, если дешёвая колбаса продаётся не там же, где дешёвые торты) и обратно до дома. Петя не ограничен во времени, так что ему нет нужды находить самый короткий маршрут.

Следует отметить, что поскольку Петя живёт в городе, то он вынужден перемещаться между двумя точками не напрямик, а по прямоугольной сетке улиц. Поэтому для вычисления расстояния между двумя точками  $A(x_a, y_a)$  и  $B(x_b, y_b)$  применяется формула так называемого манхэттенского расстояния:  $d_1(A, B) = |x_a - x_b| + |y_a - y_b|$ . Действительно: сначала надо по улице, параллельной оси абсцисс, пройти расстояние  $|x_a - x_b|$ , а потом по улице, параллельной оси ординат, — расстояние  $|y_a - y_b|$ . Сочетание «манхэттенское расстояние»

происходит от того, что такой сеткой улиц обладает Манхэттен, район города Нью-Йорка.

Программный комитет считает, что эта задача имеет сложность несколько выше средней.

Центральной идеей здесь является применение алгоритма поиска минимума в массиве. Считываем данные, которые можно хранить или в четырёх параллельных массивах, или в массиве структур. Затем за один проход ищем индекс  $i$  элемента, хранящего минимальное значение цены на колбасу, и индекс  $j$  элемента, хранящего минимальную цену на торты. Затем с использованием указанной формулы ищем значение

$$d = (|x_0 - x_i| + |y_0 - y_i|) + (|x_0 - x_j| + |y_0 - y_j|) + (|x_j - x_i| + |y_j - y_i|)$$

— длину в пути в требуемой метрике. Затем выводим величины  $i, j, d$  (корректируя значения индексов, если в используемом языке нумерация элементов идёт от нуля).

#### Идеи тестов:

1. Минимальный тест — один магазин,  $n = 1$ ;
2.  $n = 2$ , минимальные цены на колбасу и торты в одном магазине;
3.  $n = 2$ , минимальные цены на колбасу и торты в разных магазинах;
4.  $n = 10$ , все цены разные, минимальные цены в одном магазине;
5.  $n = 10$ , все цены разные, минимальные цены в разных магазинах;
6.  $n = 10$ , есть несколько магазинов с минимальными ценами;
- 7–12. то же, что в тестах 1–6, но расстояние не входит в short int;
- 13–17. случайные тесты, расстояние входит в short int;
- 18–20. случайные тесты, расстояние не входит в short int.

**8.5. «Степень строки».**  $n$ -й степенью строки в математике называют строку, получаемую  $n$ -кратным записыванием данной строки. Например, строка  $ABCD$  во второй степени — это  $ABCDABCD$ , а  $XY$  в четвёртой — это  $XYXYXYXY$ . Заметим, что строку  $XYXYXYXY$  также можно рассматривать как квадрат строки  $XYXY$ . Для данной строки определите, является ли она какой-либо степенью какой-либо более короткой строки.

По мнению программного комитета, данная задача является сложной для восьмиклассников, так как требует работы со строками и реализации нестандартного алгоритма.

В целом, подразумевается «лобовое» решение, в принципе, доступное для реализации восьмиклассниками. Идея состоит в следующем: перебираем возможные длины  $d$  базовой строки от 1 до  $\lfloor l/2 \rfloor$ , где  $l$  — длина проверяемой строки  $\mathbf{str}$ , а  $\lfloor \cdot \rfloor$  — округление вниз. Для всякой тестируемой длины  $d$  нужно проверить, что

длина  $l$  тестируемой строки кратна  $d$ , и кроме того, что для всех индексов  $i$  от 1 до  $d$  выполняется совпадение символов

$$\mathbf{str}[i] = \mathbf{str}[i + d] = \mathbf{str}[i + 2d] = \dots = \mathbf{str}[i + kd],$$

где  $k = l/d - 1$ . (В случае отсчёта символов строки от нуля следует скорректировать эту формулу.) Если это равенство верно для всех  $i$ , то проверяемая строка является  $(k+1)$ -й степенью строки, состоящей из первых  $d$  символов. Если ни для какого  $d$  из указанного диапазона такое совпадение не верно хотя бы для одного  $i$ , то проверяемая строка степенью не является, о чём и следует сообщить.

При заданных ограничениях на входные данные «лобовой» алгоритм укладывается в ограничения по времени.

#### Идеи тестов:

1. минимальный тест — односимвольная строка;
2. минимальный тест — двухсимвольная строка из одинаковых символов;
3. минимальный тест — двухсимвольная строка из разных символов;
4. шестисимвольная строка, являющаяся квадратом;
5. шестисимвольная строка, являющаяся кубом;
6. шестисимвольная строка, отличающаяся от куба в 3-м символе;
7. шестисимвольная строка, отличающаяся от куба в 4-м символе;
8. 64-символьная строка, являющаяся 8-й степенью;
9. 64-символьная строка, являющаяся квадратом;
10. максимальный тест, 255-символьная строка, не являющаяся степенью;
11. максимальный тест, 255-символьная строка, являющаяся 3-й степенью;
12. максимальный тест, 255-символьная строка, являющаяся 5-й степенью;
13. максимальный тест, 255-символьная строка, являющаяся 15-й степенью;
14. максимальный тест, 255-символьная строка, являющаяся 17-й степенью;
15. максимальный тест, 255-символьная строка, являющаяся 51-й степенью;
16. максимальный тест, 255-символьная строка, являющаяся 85-й степенью;
17. максимальный тест, 255-символьная строка, являющаяся 255-й степенью;
- 18–22. случайные тесты, являющиеся степенью;
- 23–25. случайные тесты, не являющиеся степенью.

**Муниципальный этап Всероссийской олимпиады  
школьников по информатике  
в 2014–2015 учебном году  
9 класс. Разбор решений и идеи тестов**

Далее в разборах используются обозначения типов: short int — 2-байтовый целый тип (integer в TurboPascal/BorlandPascal и int в BorlandC++), int — 4-байтовый целый тип (longint в TurboPascal/BorlandPascal/FreePascal, long int в BorlandC++, integer в Delphi/PascalABC и int в VisualC++/C++ Builder), int64 — 8-байтный целый (отсутствует в BorlandPascal и BorlandC++, но присутствует в системах программирования под Windows).

Задачи оцениваются по 100 баллов за каждую; стало быть, полная стоимость пакета — 500 баллов. В рамках одной задачи все тесты считать равноценными.

**9.1. «Собираем урожай».** *В огороде на даче у Пети Торопыжкина выросло  $n$  кочанов капусты, которые нужно перетаскать в дом. Петя за одну ходку относит не более  $k_1$  кочанов, что занимает у него  $t_1$  минут. Его брат может унести не более  $k_2$  кочанов, причём у него ходка занимает  $t_2$  минут. Уборкой капусты займётся один из братьев (другой будет убирать морковь). За какое наименьшее время можно убрать всю капусту?*

Программный комитет оценивает эту задачу как весьма простую. Вычисляем количество ходок, которые надо сделать Пете, количество ходок, которые нужно сделать брату, затем умножаем их на соответствующие времена и выбираем меньшее из результатов умножения. Некоторая тонкость состоит в вычислении количества ходок; если выражать математически, то это будут величины  $\lceil n/k_1 \rceil$  и  $\lceil n/k_2 \rceil$ , где  $\lceil \cdot \rceil$  — операция округления вверх. В большинстве языков эта операция реализована, однако лучше не прибегать к вещественным вычислениям и написать код примерно такой:

```
res := n div k1;  
if n mod k1 <> 0 then res := res+1;
```

**Идеи тестов:**

1. Минимальный тест  $n = k_1 = k_2 = 1, t_1 = t_2 = 1$ ;
2. Минимальный тест  $n = 10, k_1 = k_2 = 1, t_1 = t_2 = 1$ ;
3. Минимальный тест  $n = k_1 = 1, k_2 = 10, t_1 = t_2 = 1$ ;
4. Минимальный тест  $n = 1, k_1 = 10, k_2 = 1, t_1 = t_2 = 1$ ;
5. Максимальный тест  $n = k_1 = k_2 = 10^4, t_1 = t_2 = 10$ ;
6. Максимальный тест  $n = 10, k_1 = k_2 = 10^4, t_1 = t_2 = 10$ ;
7. Максимальный тест  $n = k_1 = 10^4, k_2 = 10^4, t_1 = t_2 = 10$ ;
8. Максимальный тест  $n = 10^4, k_1 = 10, k_2 = 10^4, t_1 = t_2 = 10$ ;
9. Максимальный тест  $n = k_1 = k_2 = 10^4, t_1 = t_2 = 10$ ;

10. Максимальный тест  $n = 10, k_1 = k_2 = 10^4, t_1 = t_2 = 10^4$ ;
11. Максимальный тест  $n = k_1 = 10^4, k_2 = 10, t_1 = t_2 = 10^4$ ;
12. Максимальный тест  $n = 10^4, k_1 = 10, k_2 = 10^4, t_1 = t_2 = 10^4$ ;
- 13–20. Случайные тесты.

**9.2. «Нужная сумма».** *Иногда Пете Торопыжкину приходится решать задачи, которые имеют формализованную постановку. Вот одна из них: напишите программу, которая из заданного набора натуральных чисел выделяет два элемента, кратных 17, так, чтобы их сумма была кратна 3.*

Сложность этой задачи программному комитету представляется ниже среднего, поскольку её решение достаточно технично.

Вначале читаем числа, игнорируя не кратные 17. Далее можно применить «наивный» алгоритм, перебирающий все возможные пары чисел и проверяющий их сумму на кратность 3. Однако если чисел, кратных 17, достаточно много (больше 14000–15000), то «наивный» алгоритм не уложится в ограничения по времени. Более разумный алгоритм использует следующую идею: чтобы сумма двух чисел была кратна 3, либо они оба должны быть кратны 3, либо одно из них даёт остаток 1 при делении на 3, а другое — остаток 2.

Реализовать эту идею можно следующим образом: заведём массив из трёх элементов с индексами 0, 1 и 2. Вначале заполним все ячейки признаком отсутствия чисел, дающих соответствующий остаток при делении на 3, например, значениями -1. При появлении очередного числа, кратного 17, проверяем его остаток при делении на 3. Если он равен 0, то при отсутствии ранее чисел, кратных 3, записываем его в 0-ю ячейку массива и продолжаем поиск. Если же число, кратное 3, уже встречалось, то заканчиваем поиск, выводим то число и последнее прочитанное.

Если остаток при делении на 3 нового числа равен 1 (соответственно, 2), то проверяем, было ли ранее найдено число с остатком 2 (соответственно, 1). Если было найдено, то прекращаем поиск и выводим эту пару чисел. Если же не было найдено, то запоминаем найденное число в соответствующей ячейке массива и продолжаем поиск.

Если мы прочитали весь набор чисел до конца и не прервали поиск, то подходящей пары чисел нет, о чём надо сообщить.

**Идеи тестов:**

1. минимальный тест,  $n = 2$ , оба числа не кратны 17;
2. минимальный тест,  $n = 2$ , только одно число кратно 17;
3. минимальный тест,  $n = 2$ , оба числа кратны 17, одно кратно 3, другое — нет;
4. минимальный тест,  $n = 2$ , оба числа кратны 17, оба кратны 3;
5. минимальный тест,  $n = 2$ , оба числа кратны 17, оба имеют остаток 1 при делении на 3;

6. минимальный тест,  $n = 2$ , оба числа кратны 17, оба имеют остаток 2 при делении на 3;
7. минимальный тест,  $n = 2$ , оба числа кратны 17, они имеют разные остатки при делении на 3;
- 8–14. то же, что в тестах 1–7, только  $n = 20$  и все числа, кроме описываемых, не кратны 17;
15.  $n = 1000$ , много чисел кратных 17, при этом все они имеют остаток от деления на 3, равный 0;
16.  $n = 1000$ , много чисел кратных 17, при этом все они имеют остаток от деления на 3, равный 1;
17.  $n = 1000$ , много чисел кратных 17, при этом все они имеют остаток от деления на 3, равный 2;
18.  $n = 1000$ , много чисел кратных 17, при этом они имеют остатки от деления на 3, равные 1 и 2;
- 19–21. случайные тесты такие, что чисел, кратных 17, не слишком много (работает «наивный» алгоритм);
- 22–25. случайные тесты такие, что чисел, кратных 17, много («наивный» алгоритм не работает).

**9.3. «Обеспечение продовольствием».** *Во время, свободное от учёбы и решения задач по информатике, Петя Торопыжский иногда играет. В том числе и в стратегии. Однажды он решил аккуратно проанализировать свой любимый экономический симулятор. В нём имеется  $n$  складов и  $m$  магазинов, в которых можно продавать продовольствие, хранящееся на складах. На  $i$ -м складе имеется запас в  $s_i$  единиц продовольствия. В  $j$ -м магазине потребность в продовольствии составляет  $d_j$  единиц, которые могут быть проданы по  $p_j$  денежных единиц каждая. Какую максимальную прибыль может получить Петя в этой игре? В один магазин продовольствие может доставляться с любого количества складов и с одного склада в любое количество магазинов.*

Программный комитет считает эту задачу средней по сложности, поскольку идейно она несложна, но для своего решения требует реализации сортировки.

В целом, достаточно очевидно, что данную задачу решает «жадный» алгоритм: сначала продукты надо везти в магазин с самыми высокими ценами. Поэтому алгоритм может выглядеть следующим образом: упорядочиваем магазины по убыванию цены, обнуляем накопленное значение выручки, затем начинаем параллельно обрабатывать массивы магазинов (отсортированный) и складов (неотсортированный). Если оставшаяся потребность очередного магазина больше или равна оставшегося предложения очередного склада, то всё продовольствие с этого склада везём в этот магазин: уменьшаем потребность магазина, увеличиваем выручку на произведение предложения склада на цену, переходим к следующему складу. Если оставшаяся потребность очередного магазина меньше оставшегося

предложения очередного склада, то со склада в этот магазин везём объём продовольствия, равный оставшейся потребности, уменьшаем предложение склада, увеличиваем выручку на произведение потребности магазина и цены, переходим к следующему магазину. Останавливаем цикл, когда исчерпан либо массив складов, либо массив магазинов. Накопленное значение выручки является ответом.

Некоторая тонкость заключается в том, что при указанных ограничениях может получиться так, что «наивные» алгоритмы сортировки (пузырьковая сортировка, сортировка вставкой и т.п.) могут не укладываться в ограничения по времени. Для получения полного балла необходимо применять «умные» алгоритмы, например, быструю сортировку (сортировку Хоара), или же использовать библиотечные функции сортировки (в которых, собственно, как раз реализована быстрая сортировка). Те, кто будут использовать неоптимальные алгоритмы, получат около 65% полного балла.

Ещё один момент, потенциально требующий внимания, — ответ может не входить в тип `short int`, хотя обязательно войдёт в тип `int`.

#### Идеи тестов:

1. минимальный тест:  $n = m = 0$ ;
2. минимальный тест:  $n = 0$ ;
3. минимальный тест:  $m = 0$ ;
4. минимальный тест:  $n \neq 0$ , но все предложения равны нулю;
5. минимальный тест:  $m \neq 0$ , но все потребности равны нулю;
6. минимальный тест: все цены равны нулю;
7. максимальный тест:  $n = 10^5$ , все  $s_i = 10$ ,  $m$  не слишком большое;
8. максимальный тест:  $n = 10^5$ , сто значений  $s_i = 10^4$ , остальные  $s_i = 0$ ,  $m$  не слишком большое;
9. максимальный тест:  $m = 10^5$ ,  $n$  не слишком большое;
10. максимальный тест:  $n = 10^5$ , все  $s_i = 10$ ,  $m = 10^5$ ;
11. максимальный тест:  $n = 10^5$ , сто значений  $s_i = 10^4$ , остальные  $s_i = 0$ ,  $m = 10^5$ ;
- 12–17. случайные тесты с небольшим  $m$ ;
- 18–20. случайные тесты с большим  $m$ .

**9.4. «Переносы в расписании».** *Расписание дел Пети Торопыжского очень плотное. И все дела важные. Но тут внезапно возникло ОЧЕНЬ ВАЖНОЕ дело, которое (возможно) пересекается с частью уже запланированных мероприятий. По имеющемуся петинскому расписанию на день и срокам начала и конца очень важного дела определите, с каким количеством дел пересекается по времени очень важное дело? (Если очень важное дело примыкает спереди или с конца к другому делу, считается, что они не пересекаются.)*

Программный комитет считает, что сложность этой задачи выше средней, поскольку, несмотря на относительную идейную простоту, для её решения привле-

кается не вполне тривиальный для 9-классников алгоритм пересечения отрезков на прямой.

Для начала следует отметить, что нужно организовать аккуратный ввод данных. Для тех, кто работает на языках C/C++, Java или Python, тут особой проблемы не будет — встроенные средства достаточно мощные, чтобы разобрать числа из указанного представления или чтобы разобрать входную строку на отдельные подстроки, представляющие числа. Однако участникам, использующим BASIC или Паскаль, тут надо будет немного потрудиться.

Идея решения оказывается достаточно простой, если сформулировать задачу на геометрическом языке: на прямой задан набор непересекающихся отрезков, нужно посчитать, сколько из них пересекаются с заданным отрезком. Видно, что центральным является алгоритм пересечения отрезков на прямой. Он формулируется следующим образом: пусть даны отрезки  $[a, b]$  и  $[c, d]$  ( $a \leq b, c \leq d$ ). Найдём величины  $\alpha = \max\{a, c\}$ ,  $\beta = \min\{b, d\}$ . Если  $\alpha \leq \beta$ , то исходные отрезки пересекаются и их пересечение есть отрезок  $[\alpha, \beta]$ . Если же  $\alpha > \beta$ , то исходные отрезки не пересекаются.

Соответственно, решение заключается в том, что поочерёдно пытаемся пересекать отрезок очень важного дела с каждым из отрезков дел из расписания, подсчитывая количество пересекающихся пар. Для удобства границы отрезков можно из формата **hh:mm** «часы–минуты» переводить в минуты  $M = 60 \cdot \text{hh} + \text{mm}$ .

#### Идеи тестов:

1. минимальный тест,  $n = 0$ ;
2. максимальный тест, важное дело начинается в 00:00 и заканчивается в 23:59,  $n$  невелико;
3. максимальный тест,  $n = 1440$ ;
4. максимальный тест, важное дело начинается в 00:00 и заканчивается в 23:59,  $n = 1440$ ;
5. очень важное дело не пересекается ни с одним делом из расписания,  $n = 100$ ;
6. очень важное дело не пересекается ни с одним делом из расписания,  $n = 1439$ ;
- 7–13. случайные тесты,  $n < 500$ ;
- 14–20. случайные тесты,  $n > 600$ .

**9.5. «Делимость на 11».** На парте в кабинете математики Петя обнаружил несколько длинных неотрицательных целых чисел. Он решил исследовать каждое число на делимость на 11: делится ли оно на 11 и, если нет, можно ли переставить в нём одну пару соседних цифр так, чтобы оно стало кратным 11. Помогите ему, напишите соответствующую программу. Напомним, что ведущие нули в записи числа не допускаются.

Программный комитет считает эту задачу сложной, так как здесь используется обработка строк вместе с алгоритмами из теории чисел (признак делимости на 11).

Напомним признак делимости на 11: считаем сумму цифр, стоящих на чётных позициях в десятичной записи числа; считаем сумму цифр, стоящих на нечётных позициях. Если разность этих сумм делится на 11, то и само число делится на 11.

Алгоритм обработки одного числа предлагается следующим: считываем строку, переводим для удобства все символы в соответствующие им числа. Первым проходом считаем суммы  $O$  и  $E$  цифр, стоящих на нечётных и чётных позициях соответственно. Если разность  $O$  и  $E$  делится на 11 (проверяется непосредственно с использованием операции **mod**, поскольку  $O$  и  $E$  не превосходят величины  $128 \cdot 9 = 1152$ ), то исходное число делится на 11, о чём сообщаем и прекращаем обработку числа.

В противном случае надо попытаться найти пару цифр, перестановка которых превратит число в делящееся на 11. Пусть мы пытаемся переставить цифры  $d_1$  и  $d_2$ , первая из которых стоит на нечётном месте, а вторая — на чётном. Тогда новая разность будет равна

$$O' - E' = (O - d_1 + d_2) - (E - d_2 + d_1) = (O - E) + 2(d_2 - d_1), \quad (*)$$

то есть при перестановке соседних цифр разность  $O - E$  может измениться только на чётную величину. Соответственно, если разность  $O - E$  уже чётная, то никакая перестановка пары соседних цифр не превратит число в кратное 11; обработку можно остановить и сообщить о невозможности достижения цели. (Впрочем, эту проверку можно и опустить, сразу перейдя к непосредственному поиску. В этом случае он гарантированно завершится неуспехом.)

Наконец, если потенциально нужная пара цифр может существовать, то попытаемся найти её прямым поиском: проходим по массиву цифр и, аккуратно учитывая, какая цифра из очередной пары стоит на чётном месте, а какая на нечётном, по формуле (\*) вычисляем разность сумм после перестановки и проверяем её на делимость на 11. Если перестановка очередной пары цифр даёт делимость на 11, выводим индекс первой цифры из пары. Если мы прошли весь массив, но не нашли подходящей пары, сообщаем о невозможности превращения числа в кратное 11. При анализе перестановки нужно не забыть проверить, что она не ставит ноль на первое место в числе.

Заметим, что школьники, знающие языки, в библиотеку которых встроены функции длинной арифметики (например, Java), если и получают выгоду при решении данной задачи, то не слишком большую, поскольку операция перестановки двух цифр в числе не вполне тривиальна; её сложность на уровне указанной процедуры линейной обработки массива цифр.

Также отметим, что в силу наличия достаточно больших чисел неразумно пытаться использовать для их хранения стандартные числовые типы языков программирования. Участники, пренебрегшие этим соображением, получают около 70% полного балла.

## Идеи тестов:

1. минимальный тест, все числа однозначные,  $n$  невелико;
2. минимальный тест, все числа однозначные,  $n = 1000$ ;
3. все числа двузначные, есть делящиеся на 11, есть не делящиеся;  $n$  невелико;
4. все числа двузначные, есть делящиеся на 11, есть не делящиеся;  $n = 1000$ ;
5. все числа трёхзначные, есть делящиеся на 11, есть те, которые можно превратить в кратные 11, есть те, которые нельзя превратить;  $n$  невелико;
6. все числа трёхзначные, есть делящиеся на 11, есть те, которые можно превратить в кратные 11, есть те, которые нельзя превратить;  $n = 1000$ ;
7. максимальный тест, все числа 255-значные, есть делящиеся на 11, есть те, которые можно превратить в кратные 11, есть те, которые нельзя превратить;  $n$  невелико;
8. максимальный тест, все числа 255-значные, есть делящиеся на 11, есть те, которые можно превратить в кратные 11, есть те, которые нельзя превратить;  $n = 1000$ ;
- 9–11. есть числа, в которых подходящая перестановка ставит на первое место ноль; есть другие подходящие перестановки;
- 12–14. есть числа, в которых подходящая перестановка ставит на первое место ноль; других подходящих перестановок нет;
- 15–20. случайные тесты, все числа не более чем 9-значные;
- 21–25. случайные тесты, есть числа, не входящие в `int64`.

## Муниципальный этап Всероссийской олимпиады школьников по информатике в 2014–2015 учебном году 10 класс. Разбор решений и идеи тестов

Далее в разборах используются обозначения типов: `short int` — 2-байтовый целый тип (`integer` в TurboPascal/BorlandPascal и `int` в BorlandC++), `int` — 4-байтовый целый тип (`longint` в TurboPascal/BorlandPascal/FreePascal, `long int` в BorlandC++, `integer` в Delphi/PascalABC и `int` в VisualC++/C++ Builder), `int64` — 8-байтный целый (отсутствует в BorlandPascal и BorlandC++, но присутствует в системах программирования под Windows).

Задачи оцениваются по 100 баллов за каждую; стало быть, полная стоимость пакета — 500 баллов. В рамках одной задачи все тесты считать равноценными.

**10.1. «Приятная покупка».** *На Новый Год родители пообещали Пете Торопыжжину купить не более двух предметов из трёх: (1) новый телевизор; (2) новый ноутбук; (3) новую игровую приставку. Петя оценил удовольствие, которое он получит от обладания каждым предметом в отдельности, а также от обладания каждой парой предметов (эти значения не учитывают удовольствие от каждой вещи в отдельности). Основываясь на этих знаниях, он высказал свои пожелания по покупке. Напишите программу, которая автоматизирует этот выбор.*

Данная задача позиционируется программным комитетом как простая: в основе лежит аккуратный поиск максимума из шести чисел. Единственная тонкость: при подсчёте удовольствия от обладания двумя предметами нужно не забыть учесть удовольствие от обладания каждым предметом в отдельности. Разумнее всего реализовывать проверку большим ветвлением.

В описании тестов символами  $P(x)$  и  $P(y, z)$  описывается величина удовольствия от обладания предметами, указанными в скобках.

### Идеи тестов:

1.  $P(1) > 0$  и больше всех остальных величин;
2.  $P(2) > 0$  и больше всех остальных величин;
3.  $P(3) > 0$  и больше всех остальных величин;
4.  $P(1, 2) > 0$  и больше всех остальных величин;
5.  $P(2, 3) > 0$  и больше всех остальных величин;
6.  $P(1, 3) > 0$  и больше всех остальных величин;
7.  $P(1) < 0$  и больше всех остальных величин;
8.  $P(2) < 0$  и больше всех остальных величин;
9.  $P(3) < 0$  и больше всех остальных величин;
10.  $P(1, 2) < 0$  и больше всех остальных величин;
11.  $P(2, 3) < 0$  и больше всех остальных величин;

12.  $P(1, 3) < 0$  и больше всех остальных величин;
13.  $P(1) = P(2)$  и больше всех остальных величин;
14.  $P(1) = P(3)$  и больше всех остальных величин;
15.  $P(2) = P(3)$  и больше всех остальных величин;
16.  $P(1) = P(1, 2)$  и больше всех остальных величин;
17.  $P(1) = P(2, 3)$  и больше всех остальных величин;
18.  $P(1) = P(1, 3)$  и больше всех остальных величин;
19.  $P(2) = P(1, 2)$  и больше всех остальных величин;
20.  $P(2) = P(2, 3)$  и больше всех остальных величин;
21.  $P(2) = P(1, 3)$  и больше всех остальных величин;
22.  $P(3) = P(1, 2)$  и больше всех остальных величин;
23.  $P(3) = P(2, 3)$  и больше всех остальных величин;
24.  $P(3) = P(1, 3)$  и больше всех остальных величин;
25. все величины равны.

**10.2. «Наибольшее подчисло».** Назовём натуральное число  $k$  подчислом натурального числа  $n$ , если десятичная запись числа  $k$  входит в десятичную запись числа  $n$ . Например, 123 является подчислом 671235, а 27 не является подчислом 529725. Напишите программу, которая по заданному натуральному числу находит его наибольшее двузначное подчисло. Напомним, что ведущие нули в записи числа не допускаются.

Программный комитет считает эту задачу достаточно лёгкой идеологически и не слишком тяжёлой с технической точки зрения.

Понятно, что основной здесь является процедура извлечения двузначного числа из десятичной записи натурального числа. Если число хранится в виде строки, то извлечение просто тривиально:  $(\text{str}[k] - \text{ord}('0')) \cdot 10 + (\text{str}[k+1] - \text{ord}('0'))$ , где  $\text{ord}$  — функция получения кода символа, а индекс  $k$  меняется от 1 до длины строки  $\text{str}$ , уменьшенной на 1 (при отсчёте индексов в строке от 1).

Если же число хранится в целой переменной, то эта процедура становится чуть более сложной. Во-первых, надо вычислить количество цифр в числе, например, так:

```
len := 0;
while m > 0 do begin
  len := len + 1;
  m := m div 10;
end;
```

Единственно, само число при этом «портится», так что надо в переменную  $m$  скопировать значение анализируемого числа. Теперь  $k = 1$  означает позицию единиц числа, а  $k = \text{len}$  означает старший разряд числа. Выделить двузначное число, состоящее из цифр в разрядах  $k$  и  $k - 1$  числа  $m$ , можно посредством вычисления  $(m \text{ div } 10^k) \bmod 100$ . Заметим, что поскольку нам нужно перебирать все значения

индекса  $k$ , то степень  $10^k$  можно не вычислять каждый раз, а накапливать, на каждой итерации цикла умножая на 10 соответствующую переменную.

### Идеи тестов:

- 1–3. различные двузначные числа без нулей в десятичной записи;
- 4–6. различные трёхзначные числа без нулей в десятичной записи;
7. число 1111111;
8. число 9999999;
9. число 102030409;
10. минимальный тест 10;
11. максимальный тест 10000000000;
- 12–20. случайные тесты.

**10.3. «Подготовка к олимпиаде».** *Готовясь к областной олимпиаде по информатике, Петя Торпыжский решает задачи. Пока что он взялся за задачи не слишком сложные, так что решение каждой задачи занимает ровно  $t$  минут. Но помимо решения задач у него есть и другие дела, расписание которых известно. К сожалению, решение задачи должно идти непрерывно, так что если промежуток между двумя делами равен 40 минутам, а решение одной задачи занимает 30 минут, то в этот промежуток удастся решить только одну задачу, а ещё 10 минут потратить на другие дела. По имеющемуся расписанию дел Пети найдите, сколько задач в день он сможет решить. Гарантируется, что в начале суток и в конце их Петя занят — он спит (в расписании обязательно присутствует дело, начинающееся в 00:00, и присутствует дело, оканчивающееся в 23:59).*

Программный комитет считает данную задачу имеющей среднюю сложность: идея решения в целом понятна, но для её реализации требуется определённая аккуратность.

Для начала следует отметить, что нужно организовать аккуратный ввод данных. Для тех, кто работает на языках C/C++, Java или Python, тут особой проблемы не будет — встроенные средства достаточно мощные, чтобы разобрать числа из указанного представления или чтобы разобрать входную строку на отдельные подстроки, представляющие числа. Однако участникам, использующих BASIC или Паскаль, тут надо будет немного потрудиться.

Геометрическая интерпретация этой задачи достаточно прямолинейна: имеется набор непересекающихся отрезков на прямой. Вопрос: сколько непересекающихся отрезков заданной длины  $t$  можно вставить между ними? Для соседних отрезков  $[a, b]$  и  $[c, d]$  ( $b \leq c$ ) ответ очевиден: в промежуток между ними можно вставить  $\lfloor (c - b)/t \rfloor$  непересекающихся отрезков длины  $t$ . Здесь  $\lfloor \cdot \rfloor$  — операция округления вниз.

Таким образом, сначала нужно расположить отрезки дел из расписания в порядке слева направо, то есть отсортировать их по возрастанию, например, левых концов. При этом для удобства границы отрезков можно из формата **hh:mm** «часы–минуты» переводить в минуты  $M = 60 \cdot \text{hh} + \text{mm}$ . Затем, аккуратно вычисляя длины промежутков между делами (в силу предложенного представления дел из разности начала следующего дела и конца предыдущего надо ещё вычесть 1 для получения количества минут в интервале между этими делами) и находя, как указано выше, количество отрезков длины  $t$ , уместяющихся в очередном интервале, суммируем эти количества и получаем ответ.

Определённой тонкостью здесь является сортировка сложных объектов (пар целых чисел) по какой-то их составляющей (по моменту начала или конца).

#### Идеи тестов:

1. минимальный тест,  $n = 1$ ;
2. минимальный тест,  $n = 2$ ; промежуток между делами существенно маленький;
3. минимальный тест,  $n = 2$ ; промежуток между делами равен  $t - 1$ ;
4. минимальный тест,  $n = 2$ ; промежуток между делами равен  $t$ ;
5. минимальный тест,  $n = 2$ ; промежуток между делами равен примерно  $1.5t$ ;
6. минимальный тест,  $n = 2$ ; промежуток между делами равен  $2t - 1$ ;
7. минимальный тест,  $n = 2$ ; промежуток между делами равен  $2t$ ;
8. минимальный тест,  $n = 2$ ; промежуток между делами достаточно большой;
9.  $n = 3$ , суммарная длина промежутков меньше  $t$ ;
10.  $n = 3$ , суммарная длина промежутков равна  $t$ ;
11.  $n = 3$ , суммарная длина промежутков меньше  $2t$ ; один промежуток короткий;
12.  $n = 3$ , один промежуток имеет длину  $t$ , другой  $2t$ ;
13.  $n = 3$ , оба промежутка имеют длину между  $t$  и  $2t$ ;
14. максимальный тест,  $n > 500$ , нет промежутков длины  $t$  или более;
15. максимальный тест,  $n > 500$ , есть промежутки длины  $t$  или более;
16.  $t = 1440$ ;
17. два одномоментных дела в начале и конце суток,  $t = 1439$ ;
- 18–25. случайные тесты.

**10.4. «Разбор предложения».** Петя Торопыжский наполовину выполнил домашнее задание по английскому языку: набрал (естественно, латиницей) и распечатал предложения, которые нужно разобрать по составу. Набор осуществлялся в одну строку моноширинным шрифтом, то есть все символы имеют одинаковую ширину, включающую в себя свободное пространство вокруг символа и равную в данном случае 6 мм. Слова в предложении разделены в точности

одним пробелом; в начале и конце текста пробелов нет. Каждое предложение завершается точкой.

Дальше нужно было сделать разбор этих предложений по составу, что не являлось слишком сложной задачей, поскольку все предложения имели стандартную структуру: в начале предложения шли определения, за ними — подлежащие, потом — сказуемые, а остальные слова до конца предложения были дополнениями. При этом однородные члены предложения разделялись запятой, то есть если в конце предыдущего слова стояла запятая, то следующее слово являлось тем же членом предложения, что и предыдущее. Пробелы между предыдущим словом и запятой не ставятся, а после запятой ставится в точности один пробел.

Тут Петю заинтересовал вопрос: а какова суммарная длина линий, которые ему нужно будет нарисовать?

Напомним, что подлежащее подчёркивается одной прямой линией, сказуемое — двумя, дополнение — штриховой линией так, что штрихи длиной в полсимвола центрированы относительно символов. Определение подчёркивается волнистой линией, но Петя вместо неё рисует «прямоугольную волну»: в начале слова рисует вертикально вниз отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вверх отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вниз отрезок на 1 мм, и т.д. Прямоугольная волна всегда заканчивается вертикальным отрезком. Знаки препинания не подчёркиваются. Например:

Beautiful flowers smell tasty, delicately.

(Везде между словами стоит ровно по одному пробелу.)

По мнению программного комитета, данная задача имеет сложность чуть выше средней. Для её решения требуется аккуратная работа со строками. По сути, наиболее разумный алгоритм, решающий задачу, представляет собой конечный автомат.

Заметим, что если слово имеет длину в  $k$  символов (без учёта знака препинания), то при его подчёркивании будет проведено линий общей длиной в  $6k$  мм, если это подлежащее,  $12k$  мм, если это сказуемое,  $3k$  мм, если это дополнение, и  $6k + (k + 1) = 7k + 1$  мм, если это определение. (В последней формуле  $6k$  — это длина всех горизонтальных отрезков, а  $k + 1$  — длина всех вертикальных отрезков.)

При обработке строки используется переменная, например, **state**, отвечающая за ожидаемый тип следующего слова, которая может принимать значение «определение», «подлежащее», «сказуемое», «дополнение». Например, этим величины можно сопоставить в соответствие целые числа 0, 1, 2, 3. Тогда взятие следующего состояния осуществляется вычислением **state** + 1 (кроме случая «следующий после дополнения», который обрабатывается отдельно).

В начале обработки строки выставляем текущее состояние в «определение».

Суммарную длину линий выставляем в нуль. Цикл:

1. если необработанная часть строки пуста, выходим из цикла;
2. считываем очередное слово;
3. если в конце есть знак препинания (запятая или точка), выкидываем его;
4. по длине полученной строки и типу члена предложения вычисляем длину линий, которые будут нарисованы при подчёркивании этого слова, и прибавляем её к суммарной длине;
5. если в конце слова не было знака препинания, переставляем ожидаемый тип члена предложения в следующее значение; переход на п.1;
6. если в конце слова была точка, переставляем ожидаемый тип члена предложения в «определение» (предложение закончилось, далее идёт новое предложение, которое начинается с определения); переход на п.1;
7. иначе (в конце слова была запятая) просто переходим к п.1 (идут однородные члены предложения, тип члена предложения не меняется).

Разделение строки по словам можно осуществить, пройдя по строке и запомнив в массиве позиции пробелов. Или же использовать функцию поиска символа (пробела) в строке, начиная с позиции, следующей за предыдущим пробелом. В обоих случаях нужна аккуратная обработка последнего слова, чтобы не выйти за границу массива, содержащего строку.

#### Идеи тестов:

1. минимальный тест, пустая строка;
2. одно предложение, одно слово;
3. одно предложение, два слова-определения;
4. одно предложение, два слова — определение и подлежащее;
5. одно предложение, три слова — определение и два подлежащих;
6. одно предложение, три слова — определение, подлежащее и сказуемое;
7. одно предложение, четыре слова — определение, подлежащее и два сказуемых;
8. одно предложение, четыре слова — все члены предложения по одному;
9. одно предложение, восемь слов — все члены предложения по два;
- 10–17. то же, что в тестах 1–9, только в строке имеются два предложения, одинаковых по составу;
18. одно предложение, длина строки — 255 символов;
19. два предложения, длина строки — 255 символов;
- 20–25. случайные тесты.

**10.5. «Делимость на 4».** У Пети Торопыжкина есть набор из  $n$  целых чисел. Он выбирает из них несколько так, чтобы их сумма была кратной 4. Каково максимальное значение такой суммы?

Данная задача позиционируется программным комитетом как сложная. Для создания полного её решения требуется привлекать идеи метода динамического программирования.

Понятно, что «наивный» подход, связанный с перебором все возможных наборов и поиском среди них того, который даёт максимальную сумму, кратную 4, не работает из-за слишком большого времени вычислений на достаточно больших входных массивах (если быть точным, то на массивах из 31–32 элементов и более длинных такой алгоритм, вообще говоря, не уложится в ограничения по времени).

Предлагается следующее решение. Заведём двумерный массив **info** с  $(n + 1)$ -м элементом по первому измерению (с индексами от 0 до  $n$ ) и 4-мя элементами по второму (с индексами от 0 до 3). Смысл ячейки **info**[ $i, j$ ] — максимальное значение суммы, составленной из членов исходного массива из его начальной части до  $i$ -го элемента включительно и дающей в остатке от деления на 4 остаток  $j$ . В начале работы алгоритма занесём во все ячейки массива **info** значение, означающее  $-\infty$ . Например, это может быть  $-10^9$ .

Далее проводим цикл по всем элементам входного массива **arr**. Обработка  $i$ -го элемента заключается в следующем. К этому моменту мы уже имеем информацию о суммах, выбранных из первых  $(i - 1)$  элементов. Мы пытаемся прибавить  $i$ -й элемент ко всем четырём суммам (со всеми возможными остатками от деления на 4) и смотрим, увеличится ли значение суммы с соответствующим остатком в результате такого прибавления. Отдельно нужно обработать случай, когда сумма с соответствующим остатком ещё не начала копиться (соответствующее значение равно  $-\infty$ ). Если прибавление даёт улучшение, то в соответствующую ячейку **info**[ $i, \cdot$ ] записываем результат этого прибавления; если не даёт, то переносим значение из соответствующей ячейки **info**[ $i - 1, \cdot$ ]. Формально эту процедуру можно записать следующим образом:

1. цикл по  $i$  от 1 до  $n$
2.   цикл по  $j$  от 0 до 3
3.   если **info**[ $i - 1, j$ ]  $\neq -\infty$  (если сумма с остатком  $j$  уже накоплена), то
4.      $j' := (j + \text{arr}[i]) \bmod 4$ ;  
      (вычисляем остаток от деления на 4 накопленной суммы  
      с остатком  $j$  и элемента **arr**[ $i$ ])
5.   если **info**[ $i - 1, j'$ ]  $\neq -\infty$   
      (если сумма с остатком  $j'$  уже накоплена), то (пытаемся улучшить)
6.     если **info**[ $i - 1, j'$ ] < **info**[ $i - 1, j$ ] + **arr**[ $i$ ], то (можно улучшить)
7.       **info**[ $i, j'$ ] := **info**[ $i - 1, j$ ] + **arr**[ $i$ ];
8.     иначе (улучшить не получилось, переносим старый результат)
9.       **info**[ $i, j'$ ] := **info**[ $i - 1, j'$ ];
10.   конец если
11.   иначе  
      (сумма с остатком  $j'$  ещё не накоплена, просто записываем результат)
12.     **info**[ $i, j'$ ] := **info**[ $i - 1, j$ ] + **arr**[ $i$ ];

13.       конец если
14.       конец если (сумма с остатком  $j$  ещё не накоплена, улучшать нечего)
15.       конец цикла по  $j$
16.        $j' := \text{arr}[i] \bmod 4$ ; (пытаемся начать новую сумму)
17.        $\text{info}[i, j'] := \max\{\text{info}[i, j'], \text{arr}[i]\}$ ;
18.       конец цикла по  $i$

Ответом служит величина в ячейке  $\text{info}[n, 0]$ .

Смысл идеи решения — классическое понимание принципа динамического программирования: если сумма

$$\text{arr}[k_1] + \text{arr}[k_2] + \dots + \text{arr}[k_\sigma] + \text{arr}[i]$$

даёт наибольшее значение (со своим остатком от деления на 4) по всем наборам суммируемых элементов среди начала массива  $\text{arr}$  до  $i$ -го элемента, то сумма

$$\text{arr}[k_1] + \text{arr}[k_2] + \dots + \text{arr}[k_\sigma]$$

должна давать наибольшее значение (со своим остатком от деления на 4) по всем наборам суммируемых элементов среди начала массива  $\text{arr}$  до  $(i-1)$ -го элемента.

Заметим, что если у нас есть набор из 4 натуральных чисел, то в нём обязательно найдётся подмножество, сумма элементов которого кратна 4. Действительно, возможны следующие варианты, описывающие все возможности:

1. есть чётное число, кратное 4;
2. есть два чётных числа, каждое из которых не кратно 4;
3. есть одно чётное число, не кратное 4, остальные числа нечётные, хотя бы два из них дают при делении на 4 остаток 1;
4. есть одно чётное, число не кратное 4, остальные числа нечётные, хотя бы два из них дают при делении на 4 остаток 3;
5. все числа нечётные, хотя бы одно даёт при делении на 4 остаток 1 и хотя бы одно даёт остаток 3;
6. все числа нечётные, все четыре из них дают при делении на 4 остаток 1;
7. все числа нечётные, все четыре из них дают при делении на 4 остаток 3.

В каждом описаны те наборы, которые дают сумму, кратную 4.

#### Идеи тестов:

- 1–7. минимальный тесты,  $n = 4$ , тесты соответствуют случаям 1–7 из вышеперечисленного;
- 8–14. максимальный тест  $n = 10^4$ , тесты соответствуют случаям 1–7 из вышеперечисленного;
- 15–20. случайные тесты,  $n < 30$ ;
- 21–25. случайные тесты,  $n > 40$ .

Тесты таковы, что «наивный» алгоритм, связанный с полным перебором, наберёт чуть больше 50% полного балла. В каждой группе присутствуют как тесты только с положительными числами, так и тесты с числами обоих знаков.

## Муниципальный этап Всероссийской олимпиады школьников по информатике в 2014–2015 учебном году 11 класс. Разбор решений и идеи тестов

Далее в разборах используются обозначения типов: short int — 2-байтовый целый тип (integer в TurboPascal/BorlandPascal и int в BorlandC++), int — 4-байтовый целый тип (longint в TurboPascal/BorlandPascal/FreePascal, long int в BorlandC++, integer в Delphi/PascalABC и int в VisualC++/C++ Builder), int64 — 8-байтный целый (отсутствует в BorlandPascal и BorlandC++, но присутствует в системах программирования под Windows).

Задачи оцениваются по 100 баллов за каждую; стало быть, полная стоимость пакета — 500 баллов. В рамках одной задачи все тесты считать равноценными.

**11.1. «Модель Эйфелевой башни».** *Петя Торопыжский решил построить модель Эйфелевой башни, для чего ему нужно купить  $r$  металлических прутков, стоимостью  $r$  рублей каждый. В  $n$ -й день от начала покупок ( $n \geq 1$ ) родители выдают ему на карманные расходы  $k(n) = an^2 + bn + c$  рублей, из которых он максимальное количество тратит на покупку заготовок, а оставшуюся неистраченную сумму помещает в копилку. Если коэффициенты таковы, что в какой-то день значение  $k(n)$  отрицательно, то это означает, что в этот день ему карманных денег не дали, и он ничего не покупает и ничего в копилку не кладёт. Сколько денег окажется в копилке по окончанию покупки нужного количества заготовок?*

С точки зрения программного комитета, данная задача является лёгкой, поскольку подразумевает прямое вычисление

#### Идеи тестов:

1. вся закупка осуществляется в первый день, лишних денег нет;
2. вся закупка осуществляется в первый день, лишние деньги — только сдача;
3. вся закупка осуществляется в первый день, лишних денег много;
4. вся закупка осуществляется к концу второго дня, лишних денег нет ни в первый день, ни во второй;
5. вся закупка осуществляется к концу второго дня, лишних денег нет в первый день, но есть во второй (сдача);
6. вся закупка осуществляется к концу второго дня, лишних денег нет в первый день, но есть во второй (много);
7. вся закупка осуществляется к концу второго дня, лишние деньги есть в первый день, но их нет во второй;
8. вся закупка осуществляется к концу второго дня, лишние деньги есть и в первый день (сдача), и во второй (сдача);

9. вся закупка осуществляется к концу второго дня, лишние деньги есть и в первый день (сдача), и во второй (много);
- 10–18. то же, что в тестах 1–9, только в начале идёт период, когда Пете денег не дают;
- 19–25. тесты, когда в середине есть промежутки, когда Пете денег не дают.

**11.2. «Неубывающая последовательность».** Имеется упорядоченный набор из  $n$  целых чисел, не превосходящих по модулю 1000. Петя Торопыжкин может прибавлять к любому числу любую неотрицательную величину. Он хочет достичь того, чтобы этот набор представлял собой неубывающую последовательность (каждый последующий член должен быть не меньше предыдущего). Каково наименьшее значение суммы величин, которые необходимо прибавить к элементам набора для достижения этой цели?

Сложность данной задачи, по мнению программного комитета, ниже средней: над идеей решения надо поразмышлять, реализация является весьма простой.

Решение строится на следующем наблюдении: если максимум среди элементов массива с 1-го по  $i$ -й равен  $m_i$ , а  $(i + 1)$ -й элемент  $a_{i+1}$  меньше  $m_i$ , то к нему надо добавить величину  $m_i - a_{i+1}$  для сохранения неубывания. Соответственно, реализация состоит в том, чтобы, перебирая элементы массива один за одним, вычислять максимум перебранной части классическим алгоритмом поиска максимума и суммировать величины, прибавляемые для сохранения неубывания:

```
curMax := a[1]; /* начальное значение максимума равно a[1];
                  к первому элементу ничего добавлять не надо */
toAdd := 0; /* начальное значение добавляемых величин */
for i := 2 to n do begin
  if a[i] < curMax
  then toAdd := toAdd + curMax - a[i]
  else curMax := a[i];
end;
```

#### Идеи тестов:

1. минимальный тест,  $n = 1$ ;
2.  $n = 10$ , последовательность постоянная;
3.  $n = 10$ , последовательность возрастающая;
4.  $n = 10$ , последовательность неубывающая;
5.  $n = 10$ , последовательность невозрастающая;
6.  $n = 10$ , последовательность убывающая;
- 7–9.  $n = 10$ , случайные тесты;
- 10–17. то же, что и в тестах 2–9,  $n = 100$ ;
- 18–25. то же, что и в тестах 2–9,  $n = 1000$  — максимальные тесты.

**11.3. «Странный максимум».** Пете Торопыжкину разонравились нечётные цифры (то есть цифры 1, 3, 5, 7 и 9). И теперь он сравнивает неотрицательные целые числа следующим образом: сначала из чисел вычёркиваются все нечётные цифры, затем вычёркиваются ведущие нули (если они появились), наконец, если от числа ничего не осталось, то оно полагается равным нулю. После такого «прореживания» числа сравниваются по обычным правилам. Если два числа получились равными, то Петя выбирает в качестве большего первое из исходных чисел. Напишите программу, которая находила бы максимум из набора чисел по петиным правилам сравнения. При поиске максимума в наборе каждое последующее число сравнивается как второе с максимумом из предыдущих чисел.

Данная задача представляется имеющей среднюю сложность, поскольку подразумевает активную работу со строками.

Алгоритм решения может быть следующим. Вначале читаем числа со входа числа в строчном виде, обрабатывая их: запоминаем исходную строку числа, строку–результат его прореживания по петиному алгоритму и длину строки–результата. Затем ищем максимум (по строке–результату) в наборе по классическому алгоритму, единственно, для этого понадобится процедура сравнения чисел, заданных строками. Впрочем, она не слишком сложна: сначала сравниваем длины (которые запомнены после первичной обработки входных данных). Если длины не равны, то большее число найдено: это то, которое имеет большую длину. Если же длины равны, то строчные записи чисел можно сравнивать как строки. Если и строчные представления чисел совпадают, то по условию задачи в качестве максимума выдаём первый объект.

#### Идеи тестов:

1. минимальный тест,  $n = 1$ , число состоит только из нечётных цифр;
2. минимальный тест,  $n = 1$ , после выкидывания нечётных цифр, образуются ведущие нули, но имеются и значащие чётные цифры;
3. минимальный тест,  $n = 1$ , после выкидывания нечётных цифр, остаются только нули;
- 4–6. то же, что и в тестах 1–3, только  $n = 2$ , в рамках одного теста все числа одинакового типа;
- 7–9. то же, что и в тестах 1–3, только  $n = 10$ , в рамках одного теста все числа одинакового типа;
- 10–12. то же, что и в тестах 1–3, только  $n = 1000$ , в рамках одного теста все числа одинакового типа;
- 13–20. случайные тесты, длины чисел меньше 255;
- 21–25. случайные тесты, есть числа с длиной 255.

**11.4. «Прохождение игрушки».** Во время, свободное от учёбы и решения задач по информатике, Петя Торопыжкин иногда играет. В том числе и в стрелялки. Однажды он решил аккуратно проанализировать свой любимый шутер.

В нём имеется  $n$  уровней. Для прохождения  $k$ -го уровня требуется  $b_k$  патронов. В начале игры игрок обладает  $s$  патронами. Для восстановления запаса в конце  $k$ -го уровня можно открыть секретную комнату и дополнительно получить  $a_k$  патронов, которые можно использовать только на следующих уровнях. Однако это требует времени, и Петя хочет сократить посещение секретных комнат, насколько это возможно. Соответственно, игрушка считается успешно пройденной, если перед началом каждого из уровней запас патронов у игрока не менее чем требуемое количество. Помогите Пете, напишите программу, которая по имеющимся данным об игре укажет какое минимальное количество секретных комнат надо будет открыть и на каких именно уровнях.

Программный комитет считает, что эта задача имеет сложность выше средней. Идея решения достаточно прямолинейна, однако аккуратная реализация, претендующая на полный балл, требует дополнительных знаний в области структур данных.

Алгоритм решения состоит в последовательном переборе уровней игры с учётом текущего количества оставшихся патронов. Если перед очередным уровнем патронов на его прохождение недостаточно, это значит, что нужно (было) открыть секретную комнату на одном из предыдущих уровней. Если уж мы вскрываем секретную комнату, то следует вскрыть ту, в которой больше всего патронов; находим среди предыдущих уровней такой, на котором секретная комната ещё не открыта и содержит наибольшее количество патронов. Открываем её, делая соответствующую пометку, добавляем запас патронов из неё к нашему запасу и запоминаем номер этого уровня. Если патронов по-прежнему недостаточно для прохождения очередного уровня, повторяем поиск секретной комнаты. Если нам нужны дополнительные патроны, а неоткрытых комнат на предыдущих уровнях не осталось, то пройти игру невозможно, останавливаем цикл по уровням и сообщаем о неудаче. Если цикл по уровням успешно завершился, то прохождение возможно, о чём следует сообщить и выдать накопленный набор уровней, на которых вскрывались секретные комнаты.

Видно, что центральным местом алгоритма является процедура поиска уровня из предыдущих, на котором секретная комната ещё не открыта и содержит наибольшее количество патронов. «Наивный» поиск, связанный с честным перебором всех предыдущих уровней для поиска максимума, при большом общем количестве уровней может не укладываться в ограничения по времени. То есть нам нужна структура, которая позволяет организовать быстрый поиск максимума по своим элементам. Такая структура известна и называется *упорядоченное множество* (в англоязычной литературе — *sorted set* или просто *set*). (Не следует путать этот тип данных с типом **set** из языка Паскаль!) Упорядоченное множество хранит (как следует из названия) упорядоченный набор попарно различных элементов. В нашем случае это может быть пара (*количество патронов в секретной комна-*

*те, номер уровня*). Порядок элементов — лексикографический, то есть сначала происходит сравнение первых элементов по количеству патронов; при равенстве первых элементов пар сравниваются вторые (которые заведомо различны в нашем случае). Такая структура данных позволяет быстрые (за время  $O(\log k)$ , где  $k$  — количество элементов в хранилище) операции вставки, удаления, поиска элемента, поиска минимального и максимального элементов (в смысле используемого порядка). Стало быть, сложность алгоритма, опирающегося на упорядоченное множество, составляет  $O(n \log n)$ : суммарное количество вставок, так же, как и суммарное количество удалений, не превосходит  $n$ , а каждая из операций имеет сложность  $O(\log n)$ .

Таким образом, алгоритм имеет следующий вид:

1. подготовка основного цикла:  
`set := ∅` — информация о секретных комнатах на предыдущих уровнях;  
`levels := ∅` — информация об уровнях, на которых открывали комнаты;  
`ammo := s` — текущий запас патронов равен начальному запасу;
2. цикл по уровням,  $k$  меняется от 1 до  $n$
3. готовимся пойти на следующий уровень:  
`while (ammo <  $b_k$ ) && (set ≠ ∅)`  
(собираем патроны, пока надо и пока есть, откуда их брать)  
`(maxAmmo, maxLevel) ←max set;`  
(открываем самую богатую комнату и удаляем соответствующую информацию из `set`)  
`levels ← maxLevel;` (запоминаем уровень, где открыли комнату)  
`end while`
4. если `ammo <  $b_k$` , то не смогли набрать патронов на следующий уровень, выходим, сообщаем о невозможности пройти игру
5. `ammo := ammo -  $b_k$` ; (тратим патроны на прохождение уровня)
6. `set ← ( $a_k, k$ )`; (запоминаем информацию о комнате на текущем уровне)
7. конец цикла по уровням
8. игру прошли, сообщаем об уровнях, на которых открывали комнаты (используем `levels`)

Тесты подобраны таким образом, что участники, построившие своё решение на «наивном» алгоритме поиска секретных комнат, получают 60% от максимального балла.

### Идеи тестов:

1. минимальный тест,  $n = 1$ , начального запаса патронов хватит, чтобы пройти единственный уровень;
2. минимальный тест,  $n = 1$ , начального запаса патронов не хватит, чтобы пройти единственный уровень;
3.  $n = 2$ , начального запаса патронов хватит, чтобы пройти оба уровня;

4.  $n = 2$ , начального запаса патронов не хватит, чтобы пройти оба уровня; секретная комната даст достаточное количество патронов, чтобы пройти второй уровень;
5.  $n = 2$ , начального запаса патронов не хватит, чтобы пройти оба уровня; секретная комната не даст достаточного количества патронов, чтобы пройти второй уровень;
6.  $n = 10$ , начального запаса хватит, чтобы пройти всю игру;
7.  $n = 10$ , начального запаса не хватит, чтобы пройти всю игру; секретные комнаты дадут достаточное количество;
8.  $n = 10$ , начального запаса не хватит, чтобы пройти всю игру; секретные комнаты не дадут достаточного количества;
- 9–14. случайные тесты,  $n < 10000$ ;
- 15–25. случайные тесты,  $n > 20000$ .

**11.5. «Организация связи».** *Кроме стрелялок Петя Торопыжский иногда играет и в стратегические игры. В одной из них важным является организация связи между опорными пунктами. При очередном запуске этой игры Пете понадобилось обеспечить двустороннюю связь между пунктами  $A$  и  $B$ , расположенными на прямой. Для этого он может устанавливать ретрансляционные станции в некоторых из оговоренных точек  $C_i$ , расположенных на той же прямой. На каждой станции он должен предусмотреть два узконаправленных передатчика, транслирующих сигнал в направлении от  $A$  к  $B$  и от  $B$  к  $A$ . Кроме, конечно, станций, расположенных в самих пунктах  $A$  и  $B$ : там нужно установить только по одному передатчику, вещающему в соответствующую сторону. Всего имеется  $m$  видов передатчиков; передатчик  $j$ -го вида имеет дальность вещания  $d_j$ , а его установка стоит  $r_j$  денежных единиц. Естественно, Петя желает установить связь, потратив наименьшее количество денежных ресурсов. Сможет ли он организовать связь в указанных условиях и, если да, то в какую сумму ему это обойдётся?*

Данная задача, по мнению программного комитета, является сложной, поскольку не вполне тривиальна идейно, а кроме того, привлекает для своего решения принцип динамического программирования. Однако подготовленным 11-классникам данная задача по силам в полном объёме, а для прочих участников допускает частичные решения, создаваемые из общих соображений.

Отметим, что задача содержит некоторую хитрость: возможные точки установки станций могут находиться не только между точками  $A$  и  $B$ , но и вне отрезка между этими точками. Несложно понять, что бессмысленно ставить станции в точках, не принадлежащих отрезку  $AB$ , поэтому при чтении такие точки нужно отсеивать. В дальнейших рассуждениях считаем, что у нас остались только точки из отрезка  $AB$ .

Для начала сделаем наблюдение, что для самой дешёвой организации двусто-

ронней связи достаточно найти самый дешёвый вариант организации односторонней связи из  $A$  в  $B$ , а потом в обратную сторону устанавливать те же самые ретрансляторы. То есть если при организации связи из  $A$  в  $B$  какая-то станция из точки  $C_i$  вещает при помощи передатчика типа  $j$  на станцию в точке  $C_{i'}$ , то для организации обратной связи надо установить на станцию в точке  $C_{i'}$  передатчик того же типа  $j$  (очевидно, его дальности хватит, чтобы покрыть станцию в точке  $C_i$ ). Действительно, пусть мы как-то организовали связь из  $A$  в  $B$ , а обратную связь каким-то другим образом удалось установить дешевле. Тогда, применив ту же идею с повторением передатчиков, мы можем удешевить организацию связи из  $A$  в  $B$ . Стало быть, достаточно решить задачу самого дешёвого установления связи из  $A$  в  $B$  и для получения ответа для двусторонней связи умножить полученный результат на 2.

Основным объектом будет одномерный массив `price` с индексами от 0 до  $(n + 1)$ . Элемент с индексом  $i$  содержит наименьшую стоимость организации односторонней связи из  $A$  до  $C_i$ . Точку  $C_0$  отождествим с  $A$ , а точку  $C_{n+1}$  — с  $B$ . Элемент `price[0]` изначально положим нулём (из  $A$  в  $A$  связь устанавливается бесплатно), а остальные элементы зашлём  $+\infty$  (значением, заведомо большим возможной стоимости связи, например,  $10^9$ ). Обработка точек  $C_i$  идёт для индексов  $i$  от 0 до  $n$ . Шаг принципа динамического программирования заключается в следующем.

Считаем, что передатчики перенумерованы так, что  $d_j \leq d_{j+1}$  — передатчик с большим номером имеет не меньший радиус вещания, чем передатчик с меньшим номером. Считаем, что точки  $C_i$  перенумерованы так, что  $x_i < x_{i+1}$  — в порядке расположения на прямой. Этих перенумераций можно достичь, проведя соответствующие сортировки передатчиков и точек. Кроме того, в этом месте можно сделать небольшую оптимизацию: после сортировки типов передатчиков из каждой группы с одинаковым расстоянием вещания оставить только один тип с наименьшей стоимостью установки. Хотя следует отметить, что в стандартной библиотеке языков BASIC, Паскаль, C/C++ нет алгоритма, выделяющего в отсортированном массиве группы равных элементов, а при ручной его реализации следует соблюдать определённую аккуратность.

Для точки  $C_i$  выбираем индекс  $j^*$  такой, что  $d_{j^*} \geq x_{i+1} - x_i$ , то есть передатчик с номером  $j^*$  и передатчики с большими номерами могут достать хотя бы следующую станцию  $C_{i+1}$  со станции  $C_i$ . В силу упорядоченности набора передатчиков можно применить алгоритм двоичного поиска, который даст результат за время  $O(\log m)$ . Если такого индекса не нашлось, то есть не нашлось передатчика, вещающего достаточно далеко, то связь установить невозможно, цикл останавливаем и сообщаем о неудаче. (Хотя этот этап можно и пропустить, устроив далее перебор по всему множеству типов передатчиков. Соответственно, в этом переборе нужно предусмотреть завершение алгоритма с неудачей.)

Здесь потребуется ещё одно наблюдение. А именно, если передатчик какого-то типа  $j$  может вещать со станции  $C_i$  на какую-то станцию  $C_{i'}$ , то нет нужды

рассматривать ситуации, когда этот передатчик вещает на другие станции, расположенные между  $C_i$  и  $C_{i'}$ . Действительно, если оптимальная цепочка вещания выглядит как  $C_i \xrightarrow{j} C_{i_1} \rightarrow C_{i_2} \rightarrow \dots \rightarrow C_{i_\sigma} \rightarrow C_{i'}$  (то есть со станции  $C_i$  на какую-то промежуточную станцию  $C_{i_1}$  вещание идёт посредством передатчика типа  $j$  и далее до  $C_{i'}$ ), то стоимость такой цепочки станций будет не меньше, чем если вещать непосредственно с  $C_i$  на  $C_{i'}$  при помощи передатчика типа  $j$ . Если же оптимальная цепочка выглядит как  $C_i \xrightarrow{j} C_{i_1} \rightarrow C_{i_2} \rightarrow \dots \rightarrow C_{i_\sigma} \xrightarrow{j'} C_{i''}$ , причём  $x_{i_\sigma} < x_{i'} < x_{i''}$ , то такую цепочку можно преобразовать в цепочку  $C_i \xrightarrow{j} C_{i'} \xrightarrow{j'} C_{i''}$ , которая также имеет не бóльшую стоимость.

Таким образом, выбрав какой-то передатчик, нужно вещать с него на самую далёкую станцию. Поэтому затем для всех типов передатчика  $j$  с номерами от  $j^*$  до  $m$  находим номер  $i'$  станции, самой далёкой, на которую он может вещать со станции  $C_i$ . Соответствующая станция находится опять двоичным поиском по условиям  $x_{i'} - x_i \leq d_j$ ,  $x_{i'+1} - x_i > d_j$ . При этом пересчёт массива **price** осуществляется как

$$\text{price}[i'] = \min\{\text{price}[i'], \text{price}[i] + r_j\},$$

то есть если установка передатчика  $d_j$  на станцию  $C_i$  удешевляет связь на станции  $C_{i'}$ , то запомним эту новую, более низкую цену.

Если мы успешно прошли весь цикл, то можно установить одностороннюю связь из  $A$  в  $B$  (а значит, можно установить и двустороннюю связь), наименьшая стоимость которой будет равна  $\text{price}[n+1]$  (а стоимость самой дешёвой двусторонней связи будет вдвое больше).

Сложность предложенного алгоритма есть

$$O(n \log n + m \log m + n \cdot (\log m + m \cdot \log n)) = O(m \log m + mn \log n).$$

В исходной формуле оценки сложности член  $n \log n$  отвечает за сортировку станций на оси, член  $m \log m$  отвечает за сортировку передатчиков по дальности вещания, член  $n$  отвечает за цикл по станциям,  $\log m$  — за поиск диапазона достаточно дальнобойных передатчиков,  $m \log n$  — за перебор передатчиков и поиск самой дальней станции для каждого передатчика.

Некоторое замечание: в блоке поиска самой дальней станции передатчика независимые двоичные поиски для каждого типа передатчика можно заменить общим линейным поиском, пользуясь тем, что упорядочены и станции, и передатчики — самая дальняя станция для типа передатчика с бóльшим номером (и, соответственно, с бóльшей дальностью вещания) лежит к  $C_i$  не ближе, чем самая дальняя станция для предыдущего типа. Соответственно, линейный поиск самой дальней станции для следующего типа передатчика можно начинать не с номера  $i$  текущей станции, а с номера, найденного для предыдущего передатчика. Такая оптимизация приведёт к сложности  $O(m+n)$  блока перебора передатчиков и поиска для каждого самой дальней станции — мы один раз проходим по передатчикам от номера  $j^*$  до номера  $m$  (слагаемое  $m$ ) и один раз по станциям от номера  $i$  до номера

$n$  в худшем случае (слагаемое  $n$ ). Общая сложность алгоритма при таком подходе будет  $O(n(m+n)) = O(n^2 + nm)$ .

### Идеи тестов:

1. минимальный тест,  $n = 0$ ,  $m = 1$ , этот передатчик достаёт от  $A$  до  $B$ ;
2. минимальный тест,  $n = 0$ ,  $m = 1$ , этот передатчик не достаёт от  $A$  до  $B$ ;
3. минимальный тест,  $n = 0$ ,  $m > 1$ , есть один передатчик, который достанет от  $A$  до  $B$ ;
4. минимальный тест,  $n = 0$ ,  $m > 1$ , есть несколько передатчиков, которые достанут от  $A$  до  $B$ ;
5. минимальный тест,  $n = 0$ ,  $m > 1$ , нет передатчиков, которые достанут от  $A$  до  $B$ ;
6.  $n = 1$ , единственная точка  $C_1$  посередине между  $A$  и  $B$ ,  $m > 1$ , есть один передатчик, который достанет от  $A$  до  $C_1$ ;
7.  $n = 1$ , единственная точка  $C_1$  посередине между  $A$  и  $B$ ,  $m > 1$ , есть несколько передатчиков, которые достанут от  $A$  до  $C_1$ ;
8.  $n = 1$ , единственная точка  $C_1$  посередине между  $A$  и  $B$ ,  $m > 1$ , нет передатчиков, которые достанут от  $A$  до  $C_1$ ;
9.  $n = 10$ , точки  $C_i$  расположены равномерно между  $A$  и  $B$ ,  $m > 1$ , есть один передатчик, который достанет от  $A$  до  $C_1$ ;
10.  $n = 10$ , точки  $C_i$  расположены равномерно между  $A$  и  $B$ ,  $m > 1$ , есть несколько передатчиков, которые достанут от  $A$  до  $C_1$ ;
11.  $n = 10$ , точки  $C_i$  расположены равномерно между  $A$  и  $B$ ,  $m > 1$ , нет передатчиков, которые достанут от  $A$  до  $C_1$ ;
12. максимальный тест,  $n = 3000$ ,  $m = 3000$ , точки  $C_i$  расположены равномерно между  $A$  и  $B$ ,  $m > 1$ , есть один передатчик, который достанет от  $A$  до  $C_1$ ;
13. максимальный тест,  $n = 3000$ ,  $m = 3000$ , точки  $C_i$  расположены равномерно между  $A$  и  $B$ ,  $m > 1$ , есть несколько передатчиков, которые достанут от  $A$  до  $C_1$ ;
14. максимальный тест,  $n = 3000$ ,  $m = 3000$ , точки  $C_i$  расположены равномерно между  $A$  и  $B$ ,  $m > 1$ , нет передатчиков, которые достанут от  $A$  до  $C_1$ ;
- 15–18. случайные тесты,  $n < 100$ ,  $m < 100$ , все точки на отрезке  $AB$ ;
- 19–20. случайные тесты,  $n < 100$ ,  $m < 100$ , есть точки вне отрезка  $AB$ ;
- 21–23. случайные тесты,  $n > 100$ ,  $m > 100$ , все точки на отрезке  $AB$ ;
- 24–25. случайные тесты,  $n > 100$ ,  $m > 100$ , есть точки вне отрезка  $AB$ .