

## Задача 1. Призы

Имя входного файла: prizes.in  
Имя выходного файла: prizes.out  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Петр участвует в конкурсе, в котором разыгрывается  $n$  призов. Призы пронумерованы от 1 до  $n$ .

По итогам конкурса участник может набрать от 2 до  $n$  баллов. Если участник наберет  $k$  баллов, то он получит один из призов с номером от 1 до  $k$ . Перед тем, как участник выберет свой приз, ведущий конкурса удаляет один из призов из списка. Затем участник может выбрать любой приз из оставшихся  $k - 1$ .

Список призов стал известен Петру. Он определил для каждого приза его ценность, для  $i$ -го приза она задается целым числом  $a_i$ .

Требуется написать программу, которая по заданным ценностям призов определяет для каждого  $k$  от 2 до  $n$ , приз с какой максимальной ценностью гарантированно достанется Петру, если он наберет в конкурсе  $k$  баллов.

### Формат входного файла

Первая строка входного файла содержит число  $n$  ( $2 \leq n \leq 100\,000$ ). Вторая строка этого файла содержит  $n$  целых чисел:  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^9$ ).

### Формат выходного файла

Выходной файл должен содержать одну строку, содержащую  $n - 1$  целых чисел: для каждого  $k$  от 2 до  $n$  должна быть выведена ценность приза, который достанется Петру, если он наберет  $k$  баллов.

### Пример входных и выходных файлов

prizes.in	prizes.out
5 1 3 4 2 5	1 3 3 4

### Описание подзадач и системы оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты этой подзадачи успешно пройдены.

#### Подзадача 1 (24 балла)

$$n \leq 100$$

#### Подзадача 2 (24 балла)

$$n \leq 5000$$

#### Подзадача 3 (52 балла)

$$n \leq 100\,000$$

### Получение информации о результатах окончательной проверки

По запросу сообщается результат окончательной проверки на каждом тесте.

## Задача 2. Космическое поселение

Имя входного файла: `space.in`  
Имя выходного файла: `space.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Для освоения Марса требуется построить исследовательскую базу. База должна состоять из  $n$  одинаковых модулей. Каждый модуль представляет собой жилой отсек, который в основании имеет форму прямоугольника размером  $a \times b$  метров.

Для повышения надежности модулей инженеры могут добавить вокруг каждого модуля дополнительный защитный слой. Толщина этого слоя должна составлять целое число метров, и все модули должны иметь одинаковую толщину защитного слоя. Модуль с защитным слоем, толщина которой равна  $d$  метрам, будет иметь в основании форму прямоугольника размером  $(a + 2d) \times (b + 2d)$  метров.

Все модули должны быть расположены на заранее подготовленном прямоугольном поле размером  $w \times h$  метров. При этом они должны быть организованы в виде регулярной сетки: их стороны должны быть параллельны сторонам поля, и модули должны быть ориентированы одинаково.

Требуется написать программу, которая по заданным количеству и размеру модулей, а также размеру поля для их размещения, определяет максимальную толщину дополнительного защитного слоя, который можно добавить к каждому модулю.

### Формат входного файла

Входной файл содержит пять разделенных пробелами целых чисел:  $n$ ,  $a$ ,  $b$ ,  $w$  и  $h$  ( $1 \leq n, a, b, w, h \leq 10^{18}$ ). Гарантируется, что без дополнительного защитного слоя все модули можно разместить в поселении описанным образом.

### Формат выходного файла

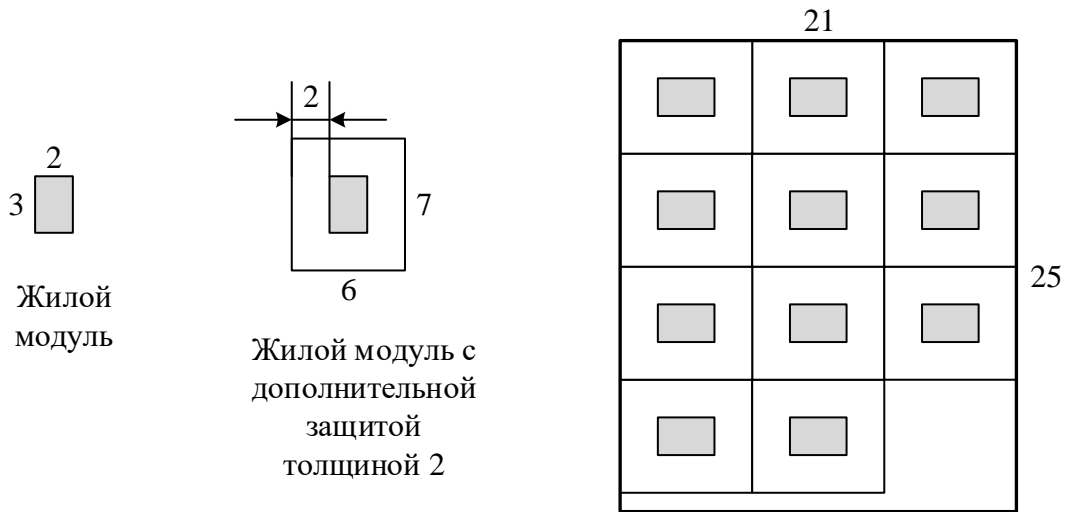
Выходной файл должен содержать одно целое число: максимальную возможную толщину дополнительного защитного слоя. Если дополнительный защитный слой установить не удастся, требуется вывести число 0.

### Примеры входных и выходных файлов

<code>space.in</code>	<code>space.out</code>
11 2 3 21 25	2
1 5 5 6 6	0

### Пояснение к примерам

В первом примере можно установить дополнительный защитный слой толщиной 2 метра и разместить модули на поле, как показано на рисунке.



Размещение модулей на поле

Во втором примере жилой отсек имеет в основании размер  $5 \times 5$  метров, а поле – размер  $6 \times 6$  метров. Добавить дополнительный защитный слой к модулю нельзя.

### Описание подзадач и системы оценивания

#### Подзадача 1 (26 баллов)

$$1 \leq n \leq 1000, 1 \leq a, b, w, h \leq 1000.$$

Баллы за подзадачу начисляются только в случае, если все тесты успешно пройдены.

#### Подзадача 2 (23 балла)

$$1 \leq n \leq 1000, 1 \leq a, b, w, h \leq 10^9.$$

Баллы за подзадачу начисляются только в случае, если все тесты успешно пройдены.

#### Подзадача 3 (до 24 баллов)

$$1 \leq n \leq 10^9, 1 \leq a, b, w, h \leq 10^{18}.$$

В этой подзадаче 8 тестов, каждый тест оценивается в 3 балла. Баллы за каждый тест начисляются независимо.

#### Подзадача 4 (до 27 баллов)

$$1 \leq n \leq 10^{18}, 1 \leq a, b, w, h \leq 10^{18}.$$

В этой подзадаче 9 тестов, каждый тест оценивается в 3 балла. Баллы за каждый тест начисляются независимо.

### Получение информации о результатах окончательной проверки

По запросу сообщается результат окончательной проверки на каждом тесте.

### Задача 3. Странные строки

Имя входного файла: `strange.in`  
Имя выходного файла: `strange.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Рассмотрим строку  $s$ , состоящую из строчных букв латинского алфавита. Примером такой строки является, например, строка «abba».

Подстрокой строки  $s$  называется строка, составленная из одного или нескольких подряд идущих символов строки  $s$ . Обозначим как  $W(s)$  множество, состоящее из всех возможных подстрок строки  $s$ . При этом каждая подстрока входит в это множество не более одного раза, даже если она встречается в строке  $s$  несколько раз.

Например,  $W(\text{«abba»}) = \{\text{«a»}, \text{«b»}, \text{«ab»}, \text{«ba»}, \text{«bb»}, \text{«abb»}, \text{«bba»}, \text{«abba»}\}$ .

Подпоследовательностью строки  $s$  называется строка, которую можно получить из  $s$  удалением произвольного числа символов. Обозначим как  $Y(s)$  множество, состоящее из всех возможных подпоследовательностей строки  $s$ . Аналогично  $W(s)$ , каждая подпоследовательность строки  $s$  включается в  $Y(s)$  ровно один раз, даже если она может быть получена несколькими способами удаления символов из строки  $s$ . Поскольку любая подстрока строки  $s$  является также ее подпоследовательностью, то множество  $Y(s)$  включает в себя  $W(s)$ , но может содержать также и другие строки.

Например,  $Y(\text{«abba»}) = W(\text{«abba»}) \cup \{\text{«aa»}, \text{«aba»}\}$ . Знак  $\cup$  обозначает объединение множеств.

Будем называть строку  $s$  *странный*, если для нее  $W(s) = Y(s)$ . Так, строка «abba» не является странной, а, например, строка «abb» является, так как для нее  $W(\text{«abb»}) = Y(\text{«abb»}) = \{\text{«a»}, \text{«b»}, \text{«ab»}, \text{«bb»}, \text{«abb»}\}$ .

Будем называть *странностью* строки число ее различных странных подстрок. При вычислении странности подстрока считается один раз, даже если она встречается в строке  $s$  в качестве подстроки несколько раз. Так, для строки «abba» ее странность равна 7, любая ее подстрока, кроме всей строки, является странной.

Требуется написать программу, которая по заданной строке  $s$  определяет ее странность.

#### Формат входного файла

Входной файл содержит строку  $s$ , состоящую из строчных букв латинского алфавита. Строка имеет длину от 1 до 200 000.

#### Формат выходного файла

Выходной файл должен содержать одно целое число: странность заданной во входном файле строки.

#### Пример входных и выходных файлов

<code>strange.in</code>	<code>strange.out</code>
abba	7

### **Описание подзадач и системы оценивания**

В этой задаче четыре подзадачи. Баллы за каждую подзадачу начисляются только в случае, если все тесты для данной подзадачи успешно пройдены.

#### **Подзадача 1 (29 баллов)**

Строка  $s$  состоит только из букв «а» и «б». Длина строки  $s$  не превышает 50.

#### **Подзадача 2 (12 баллов)**

Длина строки  $s$  не превышает 50.

#### **Подзадача 3 (25 баллов)**

Длина строки  $s$  не превышает 1000.

#### **Подзадача 4 (34 балла)**

Длина строки  $s$  не превышает 200 000.

### **Получение информации о результатах окончательной проверки**

По запросу сообщается результат окончательной проверки на каждом тесте.

## Задача 4. Поездка на каникулах

Имя входного файла: `trains.in`  
Имя выходного файла: `trains.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Железная дорога Флатландии представляет собой прямую, вдоль которой расположены  $n$  станций. Будем называть участок железной дороги от некоторой станции до следующей перегоном.

Поезд следует от станции 1 до станции  $n$ , делая остановку на каждой станции. В поезде  $k$  мест, пронумерованных от 1 до  $k$ . На поезд продаются билеты, каждый билет характеризуется тремя числами:  $s$ ,  $t$  и  $a$ . Такой билет позволяет проехать от станции  $s$  до станции  $t$  на месте  $a$ .

Иван планирует в один из дней летних каникул проехать на поезде от одной станции до другой. Он выяснил, что на поезд в этот день уже продано  $m$  билетов, и возможно уже нет мест, свободных на всех перегонах между интересующими его станциями. Билет от одной станции до другой на определенное место можно купить, только если это место свободно на всех перегонах между этими станциями.

Иван сообразил, что иногда все равно можно проехать от одной станции до другой, купив несколько билетов и пересеживаясь с одного места на другое на некоторых промежуточных станциях. Разумеется, пересеживаться с места на место неудобно, поэтому Иван хочет купить минимальное количество билетов, чтобы на каждом перегоне у него было свое место.

Иван еще не решил, от какой станции и до какой он поедет. Он записал  $q$  вариантов поездки, и для каждого из них хочет узнать, какое минимальное число билетов ему придется купить, если он выберет этот вариант.

Требуется написать программу, которая по заданному описанию уже проданных билетов и вариантов поездки Ивана определяет для каждого варианта, какое минимальное количество билетов необходимо купить, чтобы совершить такую поездку.

### Формат входного файла

Первая строка входного файла содержит числа  $n$ ,  $m$  и  $k$  ( $2 \leq n \leq 200\,000$ ,  $0 \leq m \leq 200\,000$ ,  $1 \leq k \leq 200\,000$ ) – количество станций, количество уже проданных билетов и количество мест в поезде. Последующие  $m$  строк содержат информацию о проданных билетах. Каждая строка содержит три числа:  $s_i$ ,  $t_i$  и  $a_i$  – номер станции, от которой куплен билет, номер станции, до которой куплен билет, и номер места, на которое куплен билет ( $1 \leq s_i < t_i \leq n$ ,  $1 \leq a_i \leq k$ ). Гарантируется, что все билеты куплены таким образом, что ни на каком перегоне ни на какое место нет более одного билета.

Далее идет строка, которая содержит число  $q$  ( $1 \leq q \leq 200\,000$ ). Последующие  $q$  строк содержат описания вариантов поездки. Каждая строка содержит два числа:  $f_j$ ,  $d_j$  – номер станции, от которой Иван хочет поехать в этом варианте, и номер станции, до которой он хочет поехать ( $1 \leq f_j < d_j \leq n$ ).

### Формат выходного файла

Выходной файл должен содержать  $q$  чисел: для каждого варианта поездки требуется вывести минимальное количество билетов, которое необходимо купить Ивану, чтобы совершить соответствующую поездку. Если поездку совершить невозможно, то для этого варианта требуется вывести  $-1$ .

### **Пример входных и выходных файлов**

<code>trains.in</code>	<code>trains.out</code>
5 4 3	-1
1 4 1	2
2 5 3	1
2 3 2	
4 5 2	
3	
1 5	
3 5	
4 5	

### **Пояснение к примеру**

На перегоне от 2-й до 3-й станции все места заняты, поэтому проехать от 1-й до 5-й станции невозможно. От 3-й до 5-й станции можно проехать, используя два билета: от 3-й до 4-й станции на место 2 и от 4-й до 5-й на место 1. От 4-й до 5-й станции можно проехать, используя один билет на место 1.

### **Описание подзадач и системы оценивания**

В этой задаче три подзадачи. Баллы за каждую подзадачу начисляются только в случае, если все тесты этой подзадачи успешно пройдены.

**Внимание!** Тест из примера не подходит под ограничения для подзадач 1 и 2, но решение принимается на проверку только в том случае, если оно выводит правильный ответ на тесте из примера. Решение должно выводить правильный ответ на тест даже, если оно рассчитано на решение только каких-либо из подзадач 1 и 2.

#### **Подзадача 1 (33 балла)**

$$n \leq 100, m \leq 100, k \leq 100, q = 1$$

#### **Подзадача 2 (30 баллов)**

$$n \leq 200\,000, m \leq 200\,000, k \leq 200\,000, q = 1$$

#### **Подзадача 3 (37 баллов)**

$$n \leq 200\,000, m \leq 200\,000, k \leq 200\,000, q \leq 200\,000$$

### **Получение информации о результатах окончательной проверки**

По запросу сообщаются баллы за каждую подзадачу.

## Задача 5. Три сына

Имя входного файла: `division.in`  
Имя выходного файла: `division.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Во владениях короля Флатландии находится прямая дорога длиной  $n$  километров, по одну сторону от которой расположен огромный лесной массив. Король Флатландии проникся идеями защиты природы и решил превратить свой лесной массив в заповедник. Но сыновья стали сопротивляться: ведь им хотелось получить эти земли в наследство.

У короля три сына: младший, средний и старший. Король решил, что в заповедник не войдут участки лесного массива, которые он оставит сыновьям в наследство. При составлении завещания король хочет, чтобы для участков выполнялись следующие условия:

- каждый участок должен иметь форму квадрата, длина стороны которого выражается целым положительным числом. Одна из сторон каждого квадрата должна лежать на дороге. Пусть участки имеют размеры  $a \times a$ ,  $b \times b$  и  $c \times c$ ;
- стороны квадратов должны полностью покрывать дорогу: величина  $a + b + c$  должна быть равна  $n$ ;
- участок младшего сына должен быть строго меньше участка среднего сына, а участок среднего сына должен, в свою очередь, быть строго меньше участка старшего сына, то есть должно выполняться неравенство  $a < b < c$ ;
- суммарная площадь участков  $a^2 + b^2 + c^2$  должна быть минимальна.

Требуется написать программу, которая по заданной длине дороги определяет размеры участков, которые следует выделить сыновьям короля.

### Формат входного файла

Входной файл содержит одно целое число  $n$  ( $6 \leq n \leq 10^9$ ).

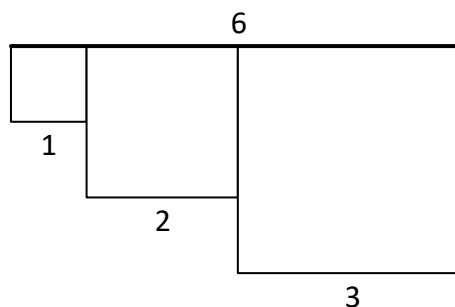
### Формат выходного файла

Выходной файл должен содержать три целых положительных числа, разделенных пробелами:  $a$ ,  $b$  и  $c$  – длины сторон участков, которые следует выделить младшему, среднему и старшему сыну, соответственно. Если оптимальных решений несколько, разрешается вывести любое.

### Пример входных и выходных файлов

<code>division.in</code>	<code>division.out</code>
6	1 2 3

### Пояснение к примеру





### **Описание подзадач и системы оценивания**

В этой задаче четыре подзадачи. Баллы за подзадачу начисляются только в случае, если все тесты для данной подзадачи пройдены.

#### **Подзадача 1 (25 баллов)**

$$n \leq 50.$$

#### **Подзадача 2 (25 баллов)**

$$n \leq 2000.$$

#### **Подзадача 3 (25 баллов)**

$$n \leq 40\,000.$$

#### **Подзадача 4 (25 баллов)**

$$n \leq 10^9.$$

### **Получение информации о результатах окончательной проверки**

По запросу сообщается результат окончательной проверки на каждом тесте.

## Задача 6. Гипершашки

Имя входного файла: `game.in`  
Имя выходного файла: `game.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Андрей работает судьей на чемпионате по гипершашкам. В каждой игре в гипершашки участвует три игрока. По ходу игры каждый из игроков набирает некоторое положительное целое число баллов. Если после окончания игры первый игрок набрал  $a$  баллов, второй —  $b$ , а третий  $c$ , то говорят, что игра закончилась со счетом  $a:b:c$ .

Андрей знает, что правила игры гипершашек устроены таким образом, что в результате игры баллы любых двух игроков различаются не более чем в  $k$  раз.

После матча Андрей показывает его результат, размещая три карточки с очками игроков на специальном табло. Для этого у него есть набор из  $n$  карточек, на которых написаны числа  $x_1, x_2, \dots, x_n$ . Чтобы выяснить, насколько он готов к чемпионату, Андрей хочет понять, сколько различных вариантов счета он сможет показать на табло, используя имеющиеся карточки.

Требуется написать программу, которая по числу  $k$  и значениям чисел на карточках, которые имеются у Андрея, определяет количество различных вариантов счета, которые Андрей может показать на табло.

### Формат входного файла

Первая строка входного файла содержит два целых числа:  $n$  и  $k$  ( $3 \leq n \leq 100\,000$ ,  $1 \leq k \leq 10^9$ ).

Вторая строка входного файла содержит  $n$  целых чисел  $x_1, x_2, \dots, x_n$  ( $1 \leq x_i \leq 10^9$ ).

### Формат выходного файла

Выходной файл должен содержать одно целое число — искомое количество различных вариантов счета.

### Пример входных и выходных файлов

<code>game.in</code>	<code>game.out</code>
5 2 1 1 2 2 3	9

### Пояснение к примеру

В приведенном примере Андрей сможет показать следующие варианты счета: 1:1:2, 1:2:1, 2:1:1, 1:2:2, 2:1:2, 2:2:1, 2:2:3, 2:3:2, 3:2:2. Другие тройки чисел, которые можно составить с использованием имеющихся карточек, не удовлетворяют заданному условию, что баллы любых двух игроков различаются не более чем в  $k = 2$  раза.

### **Описание подзадач и системы оценивания**

В этой задаче четыре подзадачи. Баллы за подзадачу начисляются только в случае, если все тесты для данной подзадачи пройдены.

**Внимание!** Тест из примера не подходит под ограничения для подзадач 1 и 3, но решение принимается на проверку только в том случае, если оно выводит правильный ответ на тесте из примера. Решение должно выводить правильный ответ на тест, даже если оно рассчитано на решение только каких-либо из подзадач 1 и 3.

#### **Подзадача 1 (15 баллов)**

$$3 \leq n \leq 100\,000, k = 1, 1 \leq x_i \leq 100\,000$$

#### **Подзадача 2 (23 балла)**

$$3 \leq n \leq 100, 1 \leq k \leq 100, 1 \leq x \leq 100$$

#### **Подзадача 3 (30 баллов)**

$$3 \leq n \leq 100\,000, 1 \leq k \leq 10^9, 1 \leq x_i \leq 10^9, \text{ все } x_i \text{ различны.}$$

#### **Подзадача 4 (32 балла)**

$$3 \leq n \leq 100\,000, 1 \leq k \leq 10^9, 1 \leq x \leq 10^9$$

### **Получение информации о результатах окончательной проверки**

По запросу сообщается результат окончательной проверки на каждом тесте.

## Задача 7. Интересные числа

Имя входного файла: numbers.in  
Имя выходного файла: numbers.out  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Софья считает число *интересным*, если его цифры идут в неубывающем порядке. Например, числа 123, 1111 или 888999 – интересные.

Софья заинтересовалась, сколько существует интересных положительных чисел, лежащих в диапазоне от  $L$  до  $R$  включительно. Это число может оказаться довольно большим для больших  $L$  и  $R$ , поэтому Софья хочет найти остаток от деления этого числа на  $10^9 + 7$ .

Требуется написать программу, которая по заданным  $L$  и  $R$  определяет количество интересных чисел, лежащих в диапазоне от  $L$  до  $R$  включительно, и выводит остаток от деления этого числа на  $10^9 + 7$ .

### Формат входного файла

Входной файл содержит две строки. Первая строка содержит число  $L$ , вторая строка содержит число  $R$  ( $1 \leq L \leq R \leq 10^{100}$ ).

### Формат выходного файла

Выходной файл должен одно целое число – остаток от деления количества интересных чисел, лежащих в диапазоне от  $L$  до  $R$  включительно, на  $10^9 + 7$ .

### Примеры входных и выходных файлов

numbers.in	numbers.out
1 100	54

### Описание подзадач и системы оценивания

#### Подзадача 1 (21 балл)

$$L = 1, R \leq 1000$$

Баллы за подзадачу начисляются только в случае, если все тесты подзадачи пройдены.

#### Подзадача 2 (до 22 баллов)

$$1 \leq L \leq R \leq 10^{18}$$

В этой подзадаче 11 тестов, каждый тест оценивается в 2 балла. Баллы за каждый тест начисляются независимо.

#### Подзадача 3 (до 24 баллов)

$$L = 1, R = 10^k \text{ для некоторого целого } k, 2 \leq k \leq 100.$$

В этой подзадаче 8 тестов, каждый тест оценивается в 3 балла. Баллы за каждый тест начисляются независимо.

#### Подзадача 4 (до 33 баллов)

$$1 \leq L \leq R \leq 10^{100}$$

В этой подзадаче 11 тестов, каждый тест оценивается в 3 балла. Баллы за каждый тест начисляются независимо.

### Получение информации о результатах окончательной проверки

По запросу сообщается результат окончательной проверки на каждом тесте.

## Задача 8. Гармоничная последовательность

Имя входного файла: `sequence.in`  
Имя выходного файла: `sequence.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Цикл лекций в университете Флатландии посвящен изучению последовательностей.

Профессор называет последовательность целых чисел  $a_1, a_2, \dots, a_n$  гармоничной, если каждое число, кроме  $a_1$  и  $a_n$ , равно сумме соседних:  $a_2 = a_1 + a_3, a_3 = a_2 + a_4, \dots, a_{n-1} = a_{n-2} + a_n$ . Например, последовательность  $[1, 2, 1, -1]$  является гармоничной, поскольку  $2 = 1 + 1$ , и  $1 = 2 + (-1)$ .

Рассмотрим последовательности равной длины:  $A = [a_1, a_2, \dots, a_n]$  и  $B = [b_1, b_2, \dots, b_n]$ . Расстоянием между этими последовательностями будем называть величину  $d(A, B) = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$ . Например,  $d([1, 2, 1, -1], [1, 2, 0, 0]) = |1 - 1| + |2 - 2| + |1 - 0| + |-1 - 0| = 0 + 0 + 1 + 1 = 2$ .

В конце лекции профессор написал на доске последовательность из  $n$  целых чисел  $B = [b_1, b_2, \dots, b_n]$  и попросил студентов в качестве домашнего задания найти гармоничную последовательность  $A = [a_1, a_2, \dots, a_n]$ , такую, что  $d(A, B)$  минимально. Чтобы облегчить себе проверку, профессор просит написать в качестве ответа только искомое минимальное расстояние  $d(A, B)$ .

Требуется написать программу, которая по заданной последовательности  $B$  определяет, на каком минимальном расстоянии от последовательности  $B$  найдется гармоничная последовательность  $A$ .

### Формат входного файла

Первая строка входного файла содержит целое число  $n$  – количество элементов в последовательности ( $3 \leq n \leq 300\,000$ ).

Вторая строка содержит  $n$  целых чисел  $b_1, b_2, \dots, b_n$  ( $-10^9 \leq b_i \leq 10^9$ ).

### Формат выходного файла

Выходной файл должна содержать одно целое число: минимальное возможное расстояние от последовательности во входном файле до гармоничной последовательности.

### Примеры входных и выходных файлов

<code>sequence.in</code>	<code>sequence.out</code>
4 1 2 0 0	2

### Пояснение к примеру

В приведенном примере оптимальной является, например, гармоничная последовательность  $[1, 2, 1, -1]$ .

### **Описание подзадач и системы оценивания**

В этой задаче пять подзадач. Баллы за подзадачу начисляются только в случае, если все тесты для данной подзадачи пройдены.

**Внимание!** Тест из примера не подходит под ограничения для подзадачи 1, но решение принимается на проверку только в том случае, если оно выводит правильный ответ на тесте из примера. Решение должно выводить правильный ответ на тест, даже если оно рассчитано на решение только подзадачи 1.

#### **Подзадача 1 (14 баллов)**

$$n = 3, -10 \leq b_i \leq 10$$

#### **Подзадача 2 (14 баллов)**

$$3 \leq n \leq 500, -100 \leq b_i \leq 100$$

#### **Подзадача 3 (16 баллов)**

$$3 \leq n \leq 100\,000, -100 \leq b_i \leq 100$$

#### **Подзадача 4 (16 баллов)**

$$3 \leq n \leq 1000, -10^9 \leq b_i \leq 10^9$$

#### **Подзадача 5 (40 баллов)**

$$3 \leq n \leq 300\,000, -10^9 \leq b_i \leq 10^9$$

### **Получение информации о результатах окончательной проверки**

По запросу сообщается баллы за каждую подзадачу.