

Задача А. Хипуй!

Имя входного файла: `heap.in`
Имя выходного файла: `heap.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайт

В этой задаче вам необходимо организовать структуру данных `Heap` для хранения целых чисел, над которой определены следующие операции:

- `Insert(X)` — добавить в `Heap` число X
- `Extract` — достать из `Heap` наибольшее число (удалив его при этом)

Формат входного файла

Во входном файле записано количество команд N ($1 \leq N \leq 100000$), потом последовательность из N команд, каждая в своей строке. Каждая команда имеет такой формат: “0 <число>” или “1”, обозначающие соответственно операции `Insert(<число>)` и `Extract`. Гарантируется, что при выполнении команды `Extract` в структуре находится по крайней мере один элемент.

Формат выходного файла

В выходной файл для каждой команды извлечения необходимо отдельно вывести число, полученное при выполнении команды `Extract`.

Пример

<code>heap.in</code>	<code>heap.out</code>
7	100
0 100	50
0 10	
1	
0 5	
0 30	
0 50	
1	

Задача В. Дорожная реформа

Имя входного файла: `roads.in`
Имя выходного файла: `roads.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Как известно, Флатландское государство состоит из n городов, которые соединены системой двусторонних железнодорожных путей и автомобильных

шоссе.

Недавно, после очередной поездки по стране, президент Флатландии остался крайне недовольным состоянием дорог и сделал вывод о необходимости радикальных реформ системы путей и сообщений.

Так как дорог в государстве за долгие годы было построено много, и почти все они нуждались в капитальном ремонте, то комиссией было решено, что необходимо закрыть часть существующих дорог, а на восстановление оставшихся выделить деньги из бюджета. При этом, так как средств в бюджете крайне не хватало, то было решено оставить как можно меньше железных и автомобильных дорог, лишь бы только по ним можно было добраться из любого города государства в любой другой (возможно через промежуточные города, используя несколько железных и/или автомобильных дорог), а именно, решено было оставить и отремонтировать ровно $n - 1$ дорогу.

После изучения результатов работы комиссии правительство решило выделить деньги на ремонт a автомобильных и b железных дорог. Но перед ним встала непростая задача: какие же конкретно дороги следует закрыть, а какие — оставить и отремонтировать, чтобы все города государства остались связанными друг с другом?

После тщетных попыток придумать план ремонта дорог, было решено обратиться к вам за помощью.

Формат входного файла

В первой строке входного файла записано четыре целых числа: n ($1 \leq n \leq 100000$) — количество городов Флатландии, m ($n - 1 \leq m \leq 200000$) — количество существующих автомобильных и железных дорог, a и b ($0 \leq a, b$; $a + b = n - 1$) — количество автомобильных и железных дорог, которые необходимо оставить. Следующие m строк описывают имеющиеся дороги. Каждая из них содержит по три целых числа: u_i , v_i ($1 \leq u_i, v_i \leq n$) — номера городов, которые соединяет i -я дорога, и t_i — тип дороги, который равен 0 для автомобильной дороги, и 1 для железной.

Гарантируется, что ни один город не связан дорогой сам с собой, и что каждая пара городов соединена максимум одной автомобильной и одной железной дорогой. Также гарантируется, что перед реформой из каждого города можно добраться до каждого.

Формат выходного файла

Если не существует плана ремонта дорог, удовлетворяющего описанным требованиям, то выведите в выходной файл единственное слово «Impossible». Иначе выведите через пробел $n - 1$ число — номера дорог, которые необходимо

отремонтировать (дороги нумеруются числами от 1 до m в порядке, в котором они заданы во входном файле). Если решений несколько, выведите любое.

Примеры

roads.in	roads.out
4 4 1 2 1 2 1 1 3 0 2 3 1 3 4 1	1 2 4
3 2 2 0 1 2 1 2 3 0	Impossible

Задача C. Range Variation Query

Имя входного файла: rvq.in
Имя выходного файла: rvq.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

В начальный момент времени последовательность a_n задана следующей формулой: $a_n = n^2 \bmod 12345 + n^3 \bmod 23456$

Требуется много раз отвечать на запросы следующего вида:

- найти разность между максимальным и минимальным значением среди элементов a_i, a_{i+1}, \dots, a_j ;
- присвоить элементу a_i значение j .

Формат входного файла

Первая строка входного файла содержит натуральное число k — количество запросов ($k \leq 100\,000$). Следующие k строк содержат запросы, по одному на строке. Запрос номер i описывается двумя целыми числами x_i, y_i .

Если $x_i > 0$, то требуется найти разность между максимальным и минимальным значением среди элементов $a_{x_i} \dots a_{y_i}$. При этом $1 \leq x_i \leq y_i \leq 100\,000$.

Если $x_i < 0$, то требуется присвоить элементу a_{-x_i} значение y_i . При этом $-100\,000 \leq x_i \leq -1$ и $|y_i| \leq 100\,000$.

Формат выходного файла

Для каждого запроса первого типа в выходной файл требуется вывести одну строку, содержащую разность между максимальным и минимальным значением на соответствующем отрезке.

Пример

rvq.in	rvq.out
7	34
1 3	68
2 4	250
-2 -100	234
1 5	1
8 9	
-3 -101	
2 3	

Задача D. Менеджер памяти

Имя входного файла: memory.in
Имя выходного файла: memory.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Пете поручили написать менеджер памяти для новой стандартной библиотеки языка C++. В распоряжении у менеджера находится массив из N последовательных ячеек памяти, пронумерованных от 1 до N . Задача менеджера — обрабатывать запросы приложений на выделение и освобождение памяти.

Запрос на выделение памяти имеет один параметр K . Такой запрос означает, что приложение просит выделить ему K последовательных ячеек памяти. Если в распоряжении менеджера есть хотя бы один свободный блок из K последовательных ячеек, то он обязан в ответ на запрос выделить такой блок. При этом непосредственно перед самой первой ячейкой памяти выделяемого блока не должно располагаться свободной ячейки памяти. После этого выделенные ячейки становятся занятыми и не могут быть использованы для выделения памяти, пока не будут освобождены. Если блока из K последовательных свободных ячеек нет, то запрос отклоняется.

Запрос на освобождение памяти имеет один параметр T . Такой запрос означает, что менеджер должен освободить память, выделенную ранее при обработке запроса с порядковым номером T . Запросы нумеруются, начиная с единицы. Гарантируется, что запрос с номером T — запрос на выделение, причем к нему еще не применялось освобождение памяти. Освобожденные ячейки могут снова быть использованы для выделения памяти. Если запрос с номером T был отклонен, то текущий запрос на освобождение памяти игнорируется.

Требуется написать менеджер памяти, удовлетворяющий приведенным

критериям.

Формат входного файла

Первая строка входного файла содержит числа N и M — количество ячеек памяти и количество запросов соответственно ($1 \leq N \leq 2^{31}-1$; $1 \leq M \leq 10^5$). Каждая из следующих M строк содержит по одному числу: $(i + 1)$ -я строка входного файла ($1 \leq i \leq M$) содержит либо положительное число K , если i -й запрос — запрос на выделение с параметром K ($1 \leq K \leq N$), либо отрицательное число T , если i -й запрос — запрос на освобождение с параметром T ($1 \leq T \leq i$).

Формат выходного файла

Для каждого запроса на выделение памяти выведите в выходной файл результат обработки этого запроса: для успешных запросов выведите номер первой ячейки памяти в выделенном блоке, для отклоненных запросов выведите число -1 . Результаты нужно выводить в порядке следования запросов во входном файле.

Пример

memory.in	memory.out
6 8	1
2	3
3	-1
-1	-1
3	1
3	-1
-5	
2	
2	

Задача Е. Горнолыжные соревнования

Имя входного файла: skier.in
Имя выходного файла: skier.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

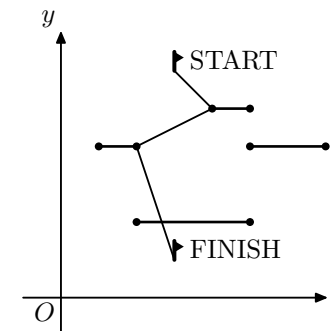
Горнолыжник, готовясь к соревнованиям, нарисовал на бумаге схему горнолыжной трассы для выбора оптимального маршрута спуска. На схеме расположенные на трассе ворота представлены горизонтальными отрезками. Никакая пара ворот не имеет общих точек.

Маршрут должен представлять собой ломаную, начинающуюся в точке старта на вершине горы и заканчивающуюся в точке финиша у ее подножия. Маршрут

выбирается таким образом, что y -координата каждой следующей вершины ломаной оказывается строго меньше y -координаты предыдущей вершины. Один из возможных маршрутов представлен на рисунке.

За каждые ворота, через которые не проходит маршрут, лыжнику начисляются штрафные очки. Общий штраф за спуск по маршруту вычисляется как сумма длины маршрута и штрафных очков за непройденные ворота.

Требуется написать программу, которая определяет, какой минимальный общий штраф горнолыжник может получить при прохождении трассы.



Формат входного файла

В первой строке входного файла задано число N — количество ворот на трассе ($0 \leq N \leq 500$), в следующих двух строках заданы S_x, S_y, F_x, F_y — координаты точек старта и финиша соответственно. В каждой из следующих N строк записаны четыре числа a_i, b_i, y_i, c_i — x -координаты левого и правого концов ворот, y -координата ворот и штраф за непрохождение данных ворот ($a_i < b_i, F_y < y_i < S_y, c_i$ — целое число, $0 \leq c_i \leq 10000$). Все координаты — целые числа, не превосходящие по модулю 10000.

Формат выходного файла

В выходной файл выведите наименьший возможный общий штраф за прохождение трассы с точностью не менее 4 знаков после десятичной точки.

Пример

*

Сборы Московской области
Третий день, 25 марта 2010, Группа В

skier.in	skier.out
4	7.8126
3 6	
3 1	
5 7 4 1	
4 5 5 10	
1 2 4 5	
2 5 2 0	