

## Задача А. Разрезание графа

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

Дан неориентированный граф. Над ним в заданном порядке производят операции следующих двух типов:

- `cut` — разрезать граф, то есть удалить из него ребро;
- `ask` — проверить, лежат ли две вершины графа в одной компоненте связности.

Известно, что после выполнения всех операций типа `cut` рёбер в графе не осталось. Найдите результат выполнения каждой из операций типа `ask`.

### Формат входных данных

Первая строка входного файла содержит три целых числа, разделённые пробелами — количество вершин графа  $n$ , количество рёбер  $m$  и количество операций  $k$  ( $1 \leq n \leq 50\,000$ ,  $0 \leq m \leq 100\,000$ ,  $m \leq k \leq 150\,000$ ).

Следующие  $m$  строк задают рёбра графа;  $i$ -я из этих строк содержит два числа  $u_i$  и  $v_i$  ( $1 \leq u_i, v_i \leq n$ ), разделённые пробелами — номера концов  $i$ -го ребра. Вершины нумеруются с единицы; граф не содержит петель и кратных рёбер.

Далее следуют  $k$  строк, описывающих операции. Операция типа `cut` задаётся строкой «`cut u v`» ( $1 \leq u, v \leq n$ ), которая означает, что из графа удаляют ребро между вершинами  $u$  и  $v$ . Операция типа `ask` задаётся строкой «`ask u v`» ( $1 \leq u, v \leq n$ ), которая означает, что необходимо узнать, лежат ли в данный момент вершины  $u$  и  $v$  в одной компоненте связности. Гарантируется, что каждое ребро графа встретится в операциях типа `cut` ровно один раз.

### Формат выходных данных

Для каждой операции `ask` во входном файле выведите на отдельной строке слово «YES», если две указанные вершины лежат в одной компоненте связности, и «NO» в противном случае. Порядок ответов должен соответствовать порядку операций `ask` во входном файле.

### Пример

<code>cutting.in</code>	<code>cutting.out</code>
3 3 7	YES
1 2	YES
2 3	NO
3 1	NO
ask 3 3	
cut 1 2	
ask 1 2	
cut 1 3	
ask 2 1	
cut 2 3	
ask 3 1	

## Задача В. Минимальное остовное дерево

Имя входного файла: `mst.in`  
Имя выходного файла: `mst.out`  
Ограничение по времени: 0.5 секунда  
Ограничение по памяти: 256 мегабайт

Дан связный неориентированный взвешенный граф. Необходимо выбрать в этом графе некоторое подмножество рёбер минимального суммарного веса таким образом, чтобы между любыми двумя вершинами графа существовал путь из выбранных рёбер.

Очевидно, что выбранное подмножество рёбер должно быть деревом (для минимальности суммарного веса ребер), и такое подмножество называется *минимальным остовным деревом* или *минимальным каркасом*.

### Формат входных данных

Первая строка входного файла содержит два натуральных числа  $n$  и  $m$  — количество вершин и ребер графа соответственно ( $1 \leq n \leq 20\,000$ ,  $0 \leq m \leq 100\,000$ ). Следующие  $m$  строк содержат описание рёбер по одному на строке. Ребро номер  $i$  описывается тремя натуральными числами  $b_i$ ,  $e_i$  и  $w_i$  — номера концов ребра и его вес соответственно ( $1 \leq b_i, e_i \leq n$ ,  $0 \leq w_i \leq 100\,000$ ).

Гарантируется, что данный граф является связным.

### Формат выходных данных

Программа должна вывести одно целое число — вес минимального остовного дерева.

### Примеры

<code>mst.in</code>	<code>mst.out</code>
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

## Задача С. День Объединения

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

В Бйатландии есть целых  $n$  городов, но нет ни одной дороги. Король страны, Вальдемар де Беар, решил исправить эту ситуацию и соединить некоторые города дорогами так, чтобы по этим дорогам можно было добраться от любого города до любого другого. Когда строительство будет завершено, король планирует отпраздновать День Объединения. К сожалению, казна Бйатландии почти пуста, поэтому король требует сэкономить деньги, минимизировав суммарную длину всех построенных дорог.

### Формат входных данных

Первая строка входного файла содержит натуральное число  $n$  ( $1 \leq n \leq 5000$ ) — количество городов в Бйатландии. Каждая из следующих  $n$  строк содержит по два целых числа  $x_i, y_i$  — координаты  $i$ -го города ( $-10000 \leq x_i, y_i \leq 10000$ ). Никакие два города не расположены в одной точке.

### Формат выходных данных

Первая строка выходного файла должна содержать минимальную суммарную длину дорог. Выведите число с точностью не менее  $10^{-3}$ .

### Пример

<code>unionday.in</code>	<code>unionday.out</code>
6	9.6568542495
1 1	
7 1	
2 2	
6 2	
1 3	
7 3	

## Задача D. Oil stations

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В столице Байтландии  $n + m$  улиц,  $n$  улиц проходят с севера на юг,  $m$  — с запада на восток. Улицы, проходящие с запада на восток, занумерованы от 1 до  $m$  (чем больше номер, тем южнее), улицы, проходящие с севера на юг, занумерованы от 1 до  $n$  (чем больше номер, тем восточнее). Перекрёстки в этом городе обозначаются как  $(x, y)$ , где  $x$  — номер улицы, проходящей с запада на восток, а  $y$  — номер улицы, проходящей с севера на юг. Вы находитесь на перекрёстке  $(1, 1)$  в автомобиле, бензобак которого имеет объём  $f$  литров и заправлен полностью.

На преодоление одного квартала (расстояния между соседними перекрёстками) автомобиль тратит 1 литр бензина. На  $k$  перекрёстках в городе имеются бензозаправочные станции, при этом цена бензина на каждой станции своя. На станции вы можете дозаправить автомобиль любым объёмом бензина, при условии, что суммарный объём купленного и оставшегося бензина не превосходит объёма бензобака.

Ваша задача — доехать до перекрёстка  $(m, n)$ , затратив наименьшее количество денег на покупку дополнительного топлива.

### Формат входных данных

В первой строке входного файла задано целое число  $T$  — количество тестовых примеров ( $1 \leq T \leq 5$ ).

В первой строке каждого тестового примера заданы четыре целых числа  $m$ ,  $n$ ,  $f$  и  $k$  — соответственно количество улиц, проходящих с запада на восток, количество улиц, проходящих с севера на юг, объём бензобака в литрах и количество заправочных станций ( $1 \leq m, n \leq 100$ ,  $1 \leq f \leq 10^5$ ,  $1 \leq k \leq 1000$ ).

В последующих  $k$  строках описаны заправочные станции. Каждая станция описывается своими координатами — двумя целыми числами  $p_i$  и  $q_i$  ( $1 \leq p_i \leq m$ ,  $1 \leq q_i \leq n$ ) и ценой на бензин  $o_i$  (число не более, чем с двумя знаками после десятичной точки) ( $0 < o_i \leq 100$ );

### Формат выходных данных

Для каждого тестового примера в случае, если доехать от  $(1, 1)$  до  $(m, n)$  возможно, выведите минимальную сумму, которую при этом придётся потратить на топливо, с точностью до  $10^{-2}$ . Если же топлива не хватит в любом случае, выведите  $-1$ .

### Примеры

стандартный ввод	стандартный вывод
2	1.20
6 6 7 2	-1
3 3 0.6	
4 2 0.4	
12 13 6 2	
1 2 2	
8 12 4.8	

## Задача Е. Уничтожение массива

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Вам дан массив, состоящий из  $n$  неотрицательных целых чисел  $a_1, a_2, \dots, a_n$ .

В этом массиве один за другим зачёркиваются числа. Вам задана перестановка чисел от 1 до  $n$  — порядок, в котором это происходит.

После зачёркивания очередного числа вам необходимо найти в этом массиве подотрезок с максимальной суммой, не содержащий ни одного зачёркнутого числа. Сумму чисел в пустом подотрезке считайте равной 0.

### Формат входных данных

В первой строке входных данных записано число  $n$  ( $1 \leq n \leq 100\,000$ ) — длина массива.

В второй строке записаны  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i \leq 10^9$ ).

В третьей строке входных данных записана перестановка чисел от 1 до  $n$  — порядок, в котором зачеркиваются числа.

### Формат выходных данных

В выходной файл выведите  $n$  строк, каждая из которых должна содержать одно число — максимальную сумму на подотрезке заданного массива, не содержащем зачёркнутых чисел, после выполнения очередного действия.

### Примеры

стандартный ввод	стандартный вывод
4 1 3 2 5 3 4 1 2	5 4 3 0
5 1 2 3 4 5 4 2 3 5 1	6 5 5 1 0
8 5 5 4 4 6 6 5 5 5 2 8 7 1 3 4 6	18 16 11 8 8 6 6 0

### Замечание

В первом тестовом примере происходит следующее:

1. Зачеркивается третий элемент, массив принимает вид  $1\ 3\ *\ 5$ . Отрезок с максимальной суммой 5 состоит из одного числа 5.
2. Зачеркивается четвертый элемент, массив принимает вид  $1\ 3\ *\ *$ . Отрезок с максимальной суммой 4 состоит из двух чисел 1 3.
3. Зачеркивается первый элемент, массив принимает вид  $*\ 3\ *\ *$ . Отрезок с максимальной суммой 3 состоит из одного числа 3.

4. Зачеркивается оставшийся второй элемент, в этот момент непустых допустимых подотрезков не остается, поэтому здесь ответ равен нулю.

## Задача F. Yet another set merging

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 4 секунды  
Ограничение по памяти: 256 мегабайт

Вам задано  $n$  различных чисел от 1 до  $n$ . Изначально каждое число лежит в своём множестве. Обозначим множество, которому принадлежит элемент  $x$  как  $set(x)$ .

Ваша задача поддерживать три вида операций:

1. Объединить множества  $set(x)$  и  $set(y)$ , где  $1 \leq x, y \leq n$ . Если  $set(x)$  совпадает с  $set(y)$ , ничего делать не требуется.
2. Объединить множества  $set(x)$ ,  $set(x + 1)$ ,  $\dots$ ,  $set(y)$ , где  $1 \leq x \leq y \leq n$ .
3. Ответить на вопрос, правда ли, что числа  $x$  и  $y$  лежат в одном множестве ( $1 \leq x, y \leq n$ ).

### Формат входных данных

Первая строка входных данных содержит два целых числа  $n$  и  $m$  ( $1 \leq n \leq 200000$ ,  $1 \leq m \leq 500000$ ) — количество чисел и количество запросов.

В последних  $q$  строках находятся запросы. Каждый запрос имеет вид  $type\ x\ y$ , где  $type \in \{1, 2, 3\}$ . Обратите внимание, что  $x$  может равняться  $y$  в запросе любого типа.

### Формат выходных данных

На каждый запрос типа 3 выведите «YES» или «NO» (без кавычек), в зависимости от того, находятся ли числа в одном множестве.

### Примеры

стандартный ввод	стандартный вывод
5 5	YES
2 2 3	NO
1 3 4	
2 2 3	
3 1 1	
3 4 5	
6 7	NO
1 1 5	NO
3 4 3	YES
3 4 1	NO
3 6 6	NO
3 1 6	YES
3 3 2	
3 5 5	

## Задача G. Всем чмоки в этом чатике!

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

Сегодня Мэри, как программисту социальной сети «Телеграфчик», предстоит реализовать сложную систему управления чатами.

Задача Мэри усложняется тем, что в социальную сеть «Телеграфчик» внедрена продвинутая система шифрования «ZergRus», простая, как всё гениальное. Суть её в том, что в системе хранится одна переменная  $zerg$ , которая принимает значения от 0 (включительно) до  $p = 10^6 + 3$  (исключая  $p$ ) и меняется в зависимости от событий в системе.

В социальной сети всего  $n$  пользователей ( $1 \leq n \leq 10^5$ ). В начале дня каждый пользователь оказывается в своём собственном чате, в котором больше никого нет. Переменная  $zerg$  в начале дня устанавливается равной 0.

В течение дня происходят события типов:

1. Участник с номером  $(i + zerg) \bmod n$  посылает сообщение всем участникам, сидящим с ним в чате (в том числе и себе самому), при этом переменная  $zerg$  заменяется на  $(30 \cdot zerg + 239) \bmod p$ .
2. Происходит слияние чатов, в которых сидят участники с номерами  $(i + zerg) \bmod n$  и  $(j + zerg) \bmod n$ . Если участники и так сидели в одном чате, то ничего не происходит. Если в разных, то чаты объединяются, а переменной  $zerg$  присваивается значение  $(13 \cdot zerg + 11) \bmod p$ .
3. Участник с номером  $(i + zerg) \bmod n$  хочет узнать, сколько сообщений он не прочитал, и прочитать их. Если участник прочитал  $q$  новых сообщений, то переменной  $zerg$  присваивается значение  $(100\,500 \cdot zerg + q) \bmod p$ .

Вы поможете Мэри реализовать систему, обрабатывающую эти события?

### Формат входных данных

В первой строке входного файла записаны натуральные числа  $n$  ( $1 \leq n \leq 10^5$ ) — число пользователей социальной сети. и  $m$  ( $1 \leq m \leq 3 \cdot 10^5$ ) — число событий, произошедших за день. В следующих  $m$  строках содержится описание событий. Первое целое число в строке означает тип события  $t$  ( $1 \leq t \leq 3$ ). Если  $t = 1$ , далее следует число  $i$  ( $0 \leq i < n$ ), по которому можно вычислить, какой участник послал сообщение. Если  $t = 2$ , далее следуют числа  $i$  и  $j$  ( $0 \leq i, j < n$ ), по которым можно вычислить номера участников, чаты с которыми должны объединиться. Если  $t = 3$ , далее следует число  $i$  ( $0 \leq i < n$ ), по которому можно вычислить номер участника, желающего узнать, сколько у него сообщений, и прочитать их.

### Формат выходных данных

Для каждого события типа 3 нужно вывести число непрочитанных сообщений у участника.

### Примеры

chat.in	chat.out	Пояснение
4 10	1	4 10
1 0	1	1 0
1 2	2	1 1
1 1		1 2
1 2		1 3
3 1		3 0
2 1 2		2 0 1
1 3		1 1
3 3		3 0
2 3 2		2 2 1
3 2		3 1

## Задача Н. Ещё одна задача про деревья

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

Есть граф из  $N$  вершин. Изначально он пустой. Нужно обработать  $M$  запросов:

1. добавить ребро из вершины  $v_1$  в вершину  $v_2$ , при этом вершины  $v_1$  и  $v_2$  находятся в разных деревьях и вершина  $v_2$  является корнем какого-то дерева.
2. по двум вершинам  $a$  и  $b$  определить, лежат ли они в одном дереве.

Решение задачи: реализуем СНМ с эвристикой сжатия путей:

```
int n, m, l[NMAX];

int calc_leader (int v) {
    if (l[v] != v)
        l[v] = calc_leader (l[v]);
    return l[v];
}

int main() {
    scanf ("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        l[i] = i;
    for (int i = 1; i <= m; i++) {
        int x, y, z;
        scanf ("%d%d%d", &z, &x, &y);
        if (z == 1)
            l[y] = x;
        else if (calc_leader (x) == calc_leader (y))
            printf ("YES\n");
        else
            printf ("NO\n");
    }
}
```

Вам же предстоит сделать тест, на котором это решение будет работать долго. Более точно, нужно сделать тест, на котором количество вызовов функции `calc_leader` будет не меньше, чем  $\frac{1}{4}M \log_2 M$ .

### Формат входных данных

Входной файл содержит два целых числа  $N$  и  $M$ . В тестах, отличных от примера  $N = 1000000$ ,  $M = 100000$ .

### Формат выходных данных

Выведите  $M$  строк.  $i$ -ая строка должна иметь вид  $1\ x\ y$ , если  $i$ -ый запрос заключается в добавлении ребра из вершины  $x$  в вершину  $y$ , или  $0\ x\ y$ , если спрашивается, лежат ли вершины  $x$  и  $y$  в одном дереве.

### Примеры

<code>trees.in</code>	<code>trees.out</code>
2 2	1 2 1 0 1 1

## **Замечание**

На тест из примера будет зачтен любой ответ, который удовлетворяет формату вывода.