

Задача А. Редакционное расстояние

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

В теории графов *деревом* называется связный неориентированный граф, не содержащий циклов.

Подвешенное дерево — это ориентированный граф без циклов, в котором в каждую вершину, кроме одной, называемой корнем ориентированного дерева, входит одно ребро. В корень ориентированного дерева не входит ни одного ребра (входящая степень равна 0). Вершины, в которые направлены дуги из вершины v , будем называть сыновьями или потомками вершины v .

(по материалам Wikipedia)

Упорядоченным деревом называют дерево, для которого задан порядок следования потомков для каждой вершины.

Помеченным деревом называют дерево, каждой вершине которого сопоставлена некоторая пометка.

Вам даны два упорядоченных помеченных подвешенных дерева. Вам разрешается выполнять над ними две операции:

- изменить пометку вершины на любую другую;
- удалить некоторую вершину, отличную от корня.

При удалении некоторой вершины её непосредственные потомки становятся потомками её родителя. Так как мы рассматриваем упорядоченные деревья, то важен порядок, в котором будут следовать сыновья удаляемой вершины и сыновья её родителя. Рассмотрим некоторую вершину v и её непосредственных потомков: $v_1, v_2, \dots, v_i, \dots, v_p$. Удалим вершину v_i . Тогда если вершина v_i имеет потомков w_1, w_2, \dots, w_q , то после её удаления потомки вершины v будут идти в следующем порядке $v_1, v_2, \dots, v_{i-1}, w_1, \dots, w_q, v_{i+1}, \dots, v_p$. Процесс удаления вершины показан на рисунке.

Удалять корни деревьев нельзя.

Вам даны стоимости описанных операций, ваша задача — найти минимальную стоимость операций, необходимых для того, чтобы сделать деревья одинаковыми.

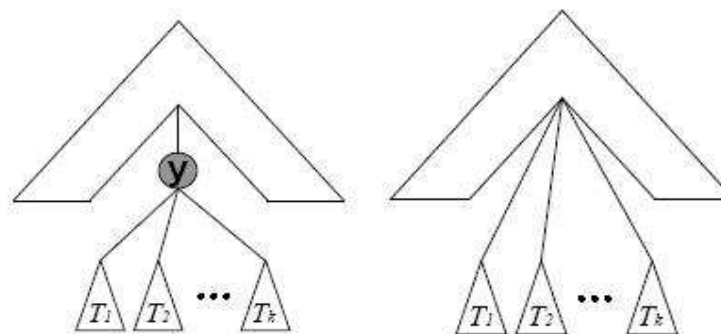


Рис. 1: дерево до и после удаления вершины u

Формат входного файла

В первой строке заданы целые числа d ($0 \leq d \leq 10^7$) и r ($0 \leq r \leq 10^7$) — стоимости операций удаления и переименования. Далее идут описание двух деревьев. Каждое описание начинается с числа N ($1 \leq N \leq 100$) — количества вершин в дереве. Вершины пронумерованы с 1 по N . Далее следует N строк, i -ая из которых соответствует вершине номер i . Каждая строка содержит метку вершины, количество её сыновей и номера сыновей в порядке их следования в упорядоченном дереве. Метка вершины — непустая строка, состоящая из не более чем 30 букв английского алфавита.

Формат выходного файла

Выведите единственное число — минимальную стоимость операций, необходимых для того, чтобы сделать деревья одинаковыми.

Примеры

ted.in	ted.out
1 1 3 A 2 2 3 B 0 C 0 4 A 1 2 D 2 3 4 B 0 C 0	1
1 1 3 A 2 2 3 B 0 C 0 4 D 1 2 A 2 3 4 B 0 C 0	2
100 1 3 A 2 2 3 C 0 B 0 3 A 2 2 3 B 0 C 0	2

Задача В. Связность лесом

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

Вам необходимо реализовать структуру данных, которая хранит неориентированный лес и позволяет выполнять следующие операции:

1. Добавить ребро между вершинами, которые лежат в различных компонентах связности.
2. Удалить ребро леса.
3. Для вершин u и v определить, лежат ли они в одной компоненте связности.

Изначально лес состоит из n изолированных вершин, которые нумеруются натуральными числами от 1 до n .

Формат входного файла

В первой строке записаны два числа: n и q ($1 \leq n \leq 50\,000$, $1 \leq q \leq 100\,000$). Здесь q — количество операций, которые необходимо выполнить. В каждой из следующих q строк записана одна из следующих операций:

1. L x y — необходимо добавить ребро между вершинами x и y ($1 \leq x, y \leq n$). Гарантируется, что эти вершины лежат в различных компонентах связности.
2. C x y — необходимо удалить ребро между вершинами x и y ($1 \leq x, y \leq n$). Гарантируется, что лес содержит это ребро.
3. Q x y — необходимо определить, лежат ли вершины x и y в одной компоненте связности ($1 \leq x, y \leq n$).

Формат выходного файла

Для каждого запроса выведите “JES” или “NE” в зависимости от ответа на него.

Пример

dyn.in	dyn.out
4 10	JES
L 1 2	JES
L 1 3	NE
L 1 4	JES
Q 2 3	
Q 1 4	
C 1 3	
Q 3 4	
L 3 4	
C 4 1	
Q 3 4	

Задача С. Наименьший общий предок

Имя входного файла:	нет
Имя выходного файла:	нет
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Подвешенное дерево — это ориентированный граф без циклов, в котором в каждую вершину, кроме одной, называемой *корнем* ориентированного дерева, входит одно ребро. В корень ориентированного дерева не входит ни одного ребра. *Отцом* вершины называется вершина, ребро из которой входит в данную.

(по материалам Wikipedia)

Дан набор подвешенных деревьев. Требуется выполнять следующие операции:

- Для двух заданных вершин u и v выяснить, лежат ли они в одном дереве. Если это так, вернуть вершину, являющуюся их наименьшим общим предком.
- Для корня r одного из деревьев и произвольной вершины v другого дерева добавить ребро (v, r) . В результате эти два дерева соединятся в одно.

Работа с модулем

Ваша программа должна взаимодействовать с модулем. Для этого она должна подключить модуль `treeunit` на Pascal или заголовочный файл `treeunit.h` на C/C++. Необходимые материалы вы можете найти на сайте сборов.

В модуле доступны следующие функции:

```
function getN: longInt;
```

```
int getN( void );
```

Функция возвращает n — количество вершин в рассматриваемом тесте. Она должна быть вызвана первой, до вызова других функций модуля. $1 \leq n \leq 50\,000$.

```
function getP( v: longInt ): longInt;
```

```
int getP( int v );
```

Параметр v должен быть целым числом от 1 до n , задающим номер вершины. Функция вернет 0, если эта вершина является корнем некоторого дерева. Иначе она вернет число от 1 до n , задающее номер её отца.

```
function nextAction( var u, v: longInt ): longInt;
```

```
int nextAction( int *u, int *v );
```

Возвращает идентификатор операции, которую следует выполнить: 0 для поиска наименьшего общего предка, 1 для добавления ребра и 2 для завершения программы. В первом случае следующей вызванной функцией модуля должна быть функция `answer`. Во втором случае u будет задавать корень некоторого дерева, а v — вершину другого дерева. В третьем случае следует корректно завершить программу, не вызывая других функций модуля.

```
procedure answer( v: longInt );
```

```
void answer( int v );
```

Эту функцию следует вызывать для ответа на запрос о поиске наименьшего общего предка. В качестве параметра v передается ответ. Между соответствующим вызовом `nextAction` и `answer` не должно быть других вызовов функций модуля.

Ответ на отдельном тесте будет признан правильным, если не выполнено ни одной недопустимой операции, решение укладывается в ограничения по времени и памяти и все `answer` вызваны с корректными ответами.

Количество вызовов `nextAction`, не возвращающих 2, не будет превышать 100 000.

Для отладки ваших решений на сайте сборов можно будет скачать демо-версии модулей. Эти версии читают данные из файла `lca.in` и записывают результат в файл `lca.out`. Их интерфейс взаимодействия с вашей программой будет идентичен используемому при тестировании.

При компиляции программ на C++ используйте следующую команду: `g++ -o a a.cpp treeunit.o`

При компиляции программ на C используйте следующую команду: `gcc -o a a.c treeunit.o`

При компиляции программ на Pascal используйте следующую команду: `fpc a.pas`

Формат входного файла

На первой строке входного файла находится число n — количество вершин в рассматриваемых деревьях. На следующей строке расположено n чисел — предок каждой вершины в начальной конфигурации, или 0, если соответствующая вершина является корнем. Затем следует число k — количество запросов к вашей программе. Каждая из следующих строк содержит по три целых числа: вид запроса (0, 1 или 2) и две вершины.

Последний запрос должен иметь тип 2.

Формат выходного файла

В случае, если программа не совершала недопустимых операций с модулем, выходной файл будет состоять из строк по одному числу, каждое из которых является ответом, выданным вашей программой с помощью функции `answer`. Иначе он будет состоять из одной или нескольких строк, содержащих сообщение о произошедшей ошибке.

Пример

<code>lca.in</code>	<code>lca.out</code>
5	0
0 0 0 0 0	3
12	0
1 5 3	1
0 1 2	3
0 3 5	1
1 4 1	3
0 4 3	
1 3 1	
0 5 4	
0 5 3	
1 2 3	
0 2 4	
0 5 2	
2 0 0	