

Содержание

Must have	2
Задача 10А. Остовное дерево 2 [0.25 (2.5), 256]	2
Задача 10В. Ближе к предкам [0.25 (2.5), 256]	3
Задачи здорового человека	4
Задача 10С. Разрезание графа [0.4 (3.5), 256]	4
Задача 10D. Больные вершины [0.3 (3), 256]	5
Для искателей острых ощущений	6
Задача 10Е. Ребра добавляются, граф растет [0.7 (6), 256]	6
Задача 10F. Электросеть [0.3 (3.5), 256]	7

У вас не получается читать/выводить данные?

Воспользуйтесь примерами (c++) (python).

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Must have

Задача 10А. Остовное дерево 2 [0.25 (2.5), 256]

Требуется найти в связном графе остовное дерево минимального веса.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно. Следующие m строк содержат описание ребер 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 ($1 \leq b_i, e_i \leq n$, $0 \leq w_i \leq 100\,000$). $n \leq 20\,000$, $m \leq 100\,000$.

Граф является связным.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — вес минимального остовного дерева.

Примеры

stdin	stdout
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

Подсказка по решению

Здесь подойдёт любой алгоритм MST. Краскал. Прим.
Важно, чтобы работал за $\mathcal{O}(m \log n)$.

Задача 10B. Ближе к предкам [0.25 (2.5), 256]

Страна Древляндия изначально состояла из одного древнего города 0. Время от времени города некоторого города p_i , недовольные условиями жизни уезжали из p_i и создавали новый город. Каждый следующий город получал минимальный не использованный номер, то есть, i . Жители города i никогда не забывают, что их предки пришли именно из p_i . Последнее время города по программе «слияние с предками» стали объединяться в регионы. Время от времени город i объявляет «не должно быть больше никаких границ между нами и предками из города p_i , давайте объединим наши городские агломерации в одну агломерацию».

Вам, как министру экономики Древляндии интересно в каждый момент времени знать размер самой большой городской агломерации, количество городов, в неё входящее.

Формат входных данных

Входные данные состоят из одного или нескольких тестов. На первой строке число тестов t , далее t однотипных тестов, каждый из которых задан следующим образом.

На 1-й строке n ($2 \leq n \leq 10^5$). На 2-й строке числа p_1, p_2, \dots, p_{n-1} : $1 \leq p_i < i - \forall$ города номер города предков. На 3-й строке перестановка из $n-1$ числа от 1 до $n-1$ – порядок операций слияния, число x обозначает, что регионы городов x и p_x сливаются в один.

Сумма n по всем тестам также не превосходит 10^5 .

Формат выходных данных

Для каждого теста выведите одну строку из $n-1$ числа, числа в этой строке – размеры максимальных регионов после каждой из операций слияния в тесте.

Пример

stdin	stdout
3	2
2	2 3 4
0	2 2 4
1	
4	
0 1 2	
3 2 1	
4	
0 1 2	
1 3 2	

Подсказка по решению

Просто DSU и хранить размеры.

Задачи здорового человека

Задача 10С. Разрезание графа [0.4 (3.5), 256]

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 -ая 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

stdin	stdout
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	

Подсказка по решению

Разрезать сложно. Просто соединять. Что же делать?

Задача 10D. Больные вершины [0.3 (3), 256]

Случилось страшное, в древнем великом дереве вершины начали заболеть. Вы пока не понимаете причину болезни, пытаетесь разобраться, для этого нужно уметь быстро узнавать ближайшую к i в направлении корня больную вершину.

Формат входных данных

Входные данные состоят из одного или нескольких тестов. На первой строке число тестов t , далее t однотипных тестов, каждый из которых задан следующим образом.

На 1-й строке число вершин в дереве n ($2 \leq n \leq 10^5$) и число запросов q ($1 \leq q \leq 10^5$). Корень дерева – вершина 1. На 2-й строке числа p_2, p_3, \dots, p_n : $1 \leq p_i < i$. p_v – отец вершины v в дереве. Изначально все вершины здоровы. Следующие q строк содержат запросы вида «? i » – найти ближайшую больную от i в направлении корня и «- i » – вершина i заболела.

Сумма n и q по всем тестам также не превосходит 10^5 .

Формат выходных данных

Для каждого теста выведите одну строку, содержащую ответы на запросы вида «?».

Если больных в направлении корня нет, ответом будет -1 .

Пример

stdin	stdout
3	-1 1 2
2 5	-1 -1
1	-1 2 2
? 2	
- 1	
? 2	
- 2	
? 2	
3 4	
1 1	
- 2	
? 3	
- 3	
? 1	
6 4	
1 2 3 4 5	
- 2	
? 1	
? 2	
? 6	

Подсказка по решению

DSU. Опять идея «смотреть с конца». Опять идея сжатия путей.

Кстати, вы знали, что на практике, если из эвристик «ранговая» и «сжатие путей» реализовать только вторую, то работает быстрее?

Для искателей острых ощущений

Задача 10Е. Ребра добавляются, граф растет [0.7 (6), 256]

В неориентированный граф последовательно добавляются новые ребра. Изначально граф пустой. После каждого добавления нужно говорить, является ли текущий граф двудольным.

Формат входных данных

На первой строке n — количество вершин, m — количество операций «добавить ребро». Следующие m строк содержат пары чисел от 1 до n — описание добавляемых ребер.

Формат выходных данных

Выведите в строчку m нулей и единиц. i -й символ должен быть равен единице, если граф, состоящий из первых i ребер, является двудольным.

$$1 \leq n, m \leq 300\,000$$

Примеры

stdin	stdout
3 3 1 2 2 3 3 1	110

Задача 10F. Электросеть [0.3 (3.5), 256]

Дан граф. Вершины – точки на плоскости. Между каждой парой городов i, j можно построить дорогу. Стоимость дороги $(|x_i - x_j| + |y_i - y_j|) \cdot (k_i + k_j)$. А ещё в каждом городе можно построить электростанцию за стоимость c_i . Изначально дорог нет. Нужно за минимальную суммарную стоимость сделать так, чтобы в каждой компоненте связности была минимум одна электростанция.

Формат входных данных

В первой строке записано одно целое число n ($1 \leq n \leq 2000$) – количество городов. Затем следует n строк. В i -й строке записаны два целых числа x_i ($1 \leq x_i \leq 10^6$) и y_i ($1 \leq y_i \leq 10^6$) – координаты i -го города. В следующей строке записаны n целых чисел c_1, c_2, \dots, c_n ($1 \leq c_i \leq 10^9$) – цена установки электростанции в i -м городе. В последней строке записаны n целых чисел k_1, k_2, \dots, k_n ($1 \leq k_i \leq 10^9$).

Формат выходных данных

В первой строке выведите суммарную стоимость.

Далее информацию где строить электростанции – число городов, сами города.

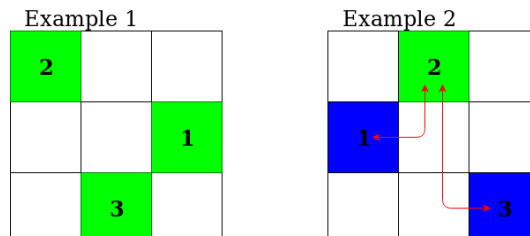
Далее информацию, какие города соединять – число дорог, какие города соединять.

Если существует несколько способов так выбрать города и соединения, чтобы получить конструкцию минимальной цены, то выведите любую из них.

Примеры

stdin	stdout
3 2 3 1 1 3 2 3 2 3 3 2 3	8 3 1 2 3 0
3 2 1 1 2 3 3 23 2 23 3 2 3	27 1 2 2 1 2 2 3

Замечание



Города с электростанциями раскрашены зеленым, остальные – синим, дороги красным.

В первом примере цена строительства электростанций во всех городах равна $3 + 2 + 3 = 8$. Можно показать, что больше никакая конфигурация не стоит меньше 8 иен.

Во втором примере цена строительства электростанции в городе 2 равна 2. Стоимость соединения городов 1 и 2 равна $2 \cdot (3 + 2)$, соединения городов 2 и 3 равна $3 \cdot (2 + 3)$. Итого 27.