Задача Brides. В поисках невест

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

Однажды король Флатландии решил отправить k своих сыновей на поиски невест. Всем известно, что во Флатландии n городов, некоторые из которых соединены дорогами. Король живет в столице, которая имеет номер 1, а город с номером n знаменит своими невестами.

Итак, король повелел, чтобы каждый из его сыновей добрался по дорогам из города 1 в город n. Поскольку, несмотря на обилие невест в городе n, красивых среди них не так много, сыновья опасаются друг друга. Поэтому они хотят добраться до цели таким образом, чтобы никакие два сына не проходили по одной и той же дороге (даже в разное время). Так как король любит своих сыновей, он хочет, чтобы среднее время сына в пути до города назначения было минимально.

Формат входного файла

В первой строке входного файла находятся числа n, m и k — количество городов и дорог во Флатландии и сыновей короля, соответственно ($2 \leqslant n \leqslant 200$, $1 \leqslant m \leqslant 2000$, $1 \leqslant k \leqslant 100$). Следующие m строк содержат по три целых положительных числа каждая — города, которые соединяет соответствующая дорога и время, которое требуется для ее прохождения (время не превышает 10^6). По дороге можно перемещаться в любом из двух направлений, два города могут быть соединены несколькими дорогами.

Формат выходного файла

Если выполнить повеление короля невозможно, выведите на первой строке число -1. В противном случае выведите на первой строке минимальное возможное среднее время (с точностью 5 знаков после десятичной точки), которое требуется сыновьям, чтобы добраться до города назначения, не менее чем с пятью знаками после десятичной точки. В следующих k строках выведите пути сыновей, сначала число дорог в пути, и затем номера дорог в пути в том порядке, в котором их следует проходить. Дороги нумеруются, начиная с единицы, в том порядке, в котором они заданы во входном файле.

Пример

brides.in	brides.out
5 8 2 1 2 1 1 3 1 1 4 3 2 5 5 2 3 1 3 5 1 3 4 1 5 4 1	3.00000 3 1 5 6 3 2 7 8

Задача Caves. Пещеры и туннели

Имя входного файла:	`caves.in`
Имя выходного файла:	`caves.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	64 мебибайт

После посадки на Марс учёные нашли странную систему пещер, соединённых туннелями. И учёные начали исследовать эту систему, используя управляемых роботов. Было обнаружено, что существует ровно один путь между каждой парой пещер. Но потом учёные обнаружили специфическую проблему. Иногда в пещерах происходят небольшие взрывы. Они вызывают выброс радиоактивных изотопов и увеличивают уровень радиации в пещере. К сожалению, роботы плохо выдерживают радиацию. Но для исследования они должны переместиться из одной пещеры в другую. Учёные поместили в каждую пещеру сенсор для мониторинга уровня радиации. Теперь они каждый раз при движении робота хотят знать максимальный уровень радиации, с которым придётся столкнуться роботу во время его перемещения. Как вы уже догадались, программу, которая это делает, будете писать вы.

Формат входного файла

Первая строка входного файла содержит одно целое число N ($1 \leqslant N \leqslant 100\,000$) — количество пещер. Следующие $N-1$ строк описывают туннели. Каждая из этих строк содержит два целых числа — a_i и b_i ($1 \leqslant a_i, b_i \leqslant N$), описывающие туннель из пещеры с номером a_i в пещеру с номером b_i. Следующая строка содержит целое число Q ($1 \leqslant Q \leqslant 100\,000$), означающее количество запросов. Далее идут Q запросов, по одному на строку. Каждый запрос имеет вид «C U V», где C — символ «`I`» либо «`G`», означающие тип запроса (кавычки только для ясности). В случае запроса «`I`» уровень радиации в U-й пещере ($1 \leqslant U \leqslant N$) увеличивается на V ($0 \leqslant V \leqslant 10\,000$). В случае запроса «`G`» ваша программа должна вывести максимальный уровень радиации на пути между пещерами с номерами U и V ($1 \leqslant U, V \leqslant N$) после всех увеличений радиации (запросов «`I`»), указанных ранее. Предполагается, что изначальный уровень радиации равен 0 во всех пещерах, и он никогда не уменьшается со временем (потому что период полураспада изотопов много больше времени наблюдения).

Формат выходного файла

Для каждого запроса «`G`» выведите одну строку, содержащую максимальный уровень радиации.

Пример

caves.in	caves.out
4 1 2 2 3 2 4 6 I 1 1 G 1 1 G 3 4 I 2 3 G 1 1 G 3 4	1 0 1 3

Задача Caves-Easy. Пещеры и туннели (простая)

Задача «Пещеры и туннели» с ограничением 100 на количество пещер.

Задача Evacuate. План эвакуации

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

В городе есть муниципальные здания и бомбоубежища, которые были специально построены для эвакуации служащих в случае ядерной войны. Каждое бомбоубежище имеет ограниченную вместительность по количеству людей, которые могут в нем находиться. В идеале все работники из одного муниципального здания должны были бы бежать к ближайшему бомбоубежищу. Однако, в таком случае, некоторые бомбоубежища могли бы переполниться, в то время как остальные остались бы наполовину пустыми.

Чтобы резрешить эту проблему Городской Совет разработал специальный план эвакуации. Вместо того, чтобы каждому служащему индивидуально приписать, в какое бомбоубежище он должен бежать, для каждого муниципального здания определили, сколько служащих из него в какое бомобоубежище должны бежать. Задача индивидального распределения была переложена на внутреннее управление муниципальных зданий.

План эвакуации учитывает количество служащих в каждом здании — каждый служащий должен быть учтен в плане и в каждое бомбоубежище может быть направлено количество служащих, не превосходящее вместимости бомбоубежища.

Городской Совет заявляет, что их план эвакуации оптиматен в том смысле, что суммарное время эвакуации всех служащих города минимально.

Мэр города, находящийся в постоянной конфронтации с Городским Советом, не слишком то верит этому заявлению. Поэтому он нанял Вас в качестве независимого эксперта для проверки плана эвакуации. Ваша задача состоит в том, чтобы либо убедиться в оптимальности плана Городского Совета, либо доказать обратное, представив в качестве доказательства другой план эвакуации с меньшим суммарным временем для эвакуации всех служащих.

Карта города может быть представлена в виде квадратной сетки. Расположение муниципальных зданий и бомбоубежищ задается парой целых чисел, а время эвакуации из муниципального здания с координатами (X_i, Y_i) в бомбоубежище с координатами (P_j, Q_j) составляет $D_{ij} = |X_i - P_j| + |Y_i - Q_j| + 1$ минут.

Формат входного файла

Входной файл содержит описание карты города и плана эвакуации, предложенного Городским Советом. Первая строка входного файла содержит два целых числа N ($1 \leqslant N \leqslant 100$) и M ($1 \leqslant M \leqslant 100$), разделенных пробелом. N — число муниципальных зданий в городе (все они занумерованы числами от 1 до N), M — число бомбоубежищ (все они занумерованы числами от 1 до M).

Последующие N строк содержат описания муниципальных зданий. Каждая строка содержит целые числа X_i, Y_i и B_i, разделенные пробелами, где X_i, Y_i ($-1000 \leqslant X_i, Y_i \leqslant 1000$) — координаты здания, а B_i ($1 \leqslant B_i \leqslant 1000$) — число служащих в здании.

Описание бомбоубежищ содержится в последующих M строках. Каждая строка содержит целые числа P_j, Q_j и C_j, разделенные пробелами, где P_j, Q_j ($-1000 \leqslant P_j, Q_j \leqslant 1000$) — координаты бомбоубежища, а C_j ($1 \leqslant C_j \leqslant 1000$) — вместимость бомбоубежища.

В последующихся N строках содержится описание плана эвакуации. Каждая строка представляет собой описание плана эвакуации для отдельного здания. План эвакуации из i-го здания состоит из M целых чисел E_{ij}, разделенных пробелами. E_{ij} ($0 \leqslant E_{ij} \leqslant 10000$) — количество служащих, которые должны эвакуироваться из i-го здания в j-е бомбоубежище.

Гарантируется, что план, заданный во входном файле, корректен.

Формат выходного файла

Если план эвакуации Городского Совета оптимален, то выведите одно слово `OPTIMAL`. В противном случае выведите на первой строке слово `SUBOPTIMAL`, а в последующих N строках выведите Ваш план эвакуации (более оптимальный) в том же формате, что и во входном файле. Ваш план не обязан быть оптимальным, но должен быть лучше плана Городского Совета.

Пример

evacuate.in	evacuate.out
3 4 -3 3 5 -2 2 6 2 2 5 -1 1 3 1 1 4 -2 -2 7 0 -1 3 3 1 1 0 0 0 6 0 3 0 2	SUBOPTIMAL 3 0 1 1 0 0 6 0 0 4 0 1
3 4 -3 3 5 -2 2 6 2 2 5 -1 1 3 1 1 4 -2 -2 7 0 -1 3 3 0 1 1 0 0 6 0 0 4 0 1	OPTIMAL

Задача RBTrees. Красно-черные деревья

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

Широкое распространение в стандартных библиотеках многих языков программирования получила реализация сбалансированных деревьев на основе так называемых *красно-черных деревьев*. В данной задаче вам предлагается посчитать количество красно-черных деревьев заданной формы.

Напомним, что двоичным деревом называется набор *вершин*, организованных в виде *дерева*. Каждая вершина имеет не более двух *детей*, один из которых называется *левым*, а другой — *правым*. Как левый, так и правый ребенок, а также оба могут отсутствовать.

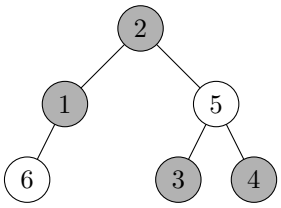
Если вершина Y — ребенок вершины X, то говорят, что вершина X является *родителем* вершины Y. У каждой вершины дерева, кроме одной, есть ровно один родитель. Единственная вершина, не имеющая родителя, называется *корнем* дерева.

Соединим каждую вершину кроме корня с ее родителем. Заметим, что для каждой вершины существует ровно один путь, ведущий в нее от корня.

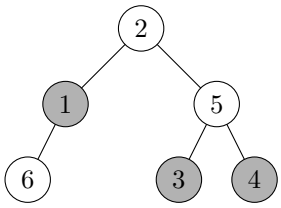
Двоичное дерево называется *красно-черным*, если каждая его вершина раскрашена в красный либо в черный цвет, причем выполняются следующие условия:

1. если вершина красная, то ее родитель — черный;
2. количество черных вершин на пути от корня до любой вершины, у которой отсутствует хотя бы один ребенок, одно и то же.

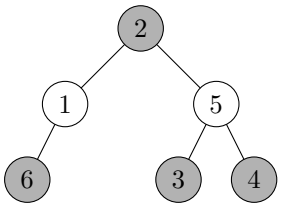
Примеры двоичного дерева, вершины которого раскрашены в два цвета, приведены на следующем рисунке.



(а)



(б)



(в)

Если считать закрашенные вершины черными, а незакрашенные — красными, то дерево на рисунке (а) является красно-черным деревом, а деревья на рисунках (б) и (в) — нет. Для дерева на рисунке (б) нарушается первое свойство — у красной вершины 5 родитель 2 также красный, а в дереве на рисунке (в) нарушается второе свойство — на пути от корня до вершины 1 одна черная вершина, а, например, на пути от корня до вершины 3 — две.

Для заданного двоичного дерева подсчитайте число способов раскрасить его вершины в черный и красный цвет так, чтобы оно стало красно-черным деревом.

Формат входного файла

Первая строка входного файла содержит число n — количество вершин в дереве ($1 \leqslant n \leqslant 1000$).

Пусть вершины дерева пронумерованы числами от 1 до n. Следующие n строк содержат по два числа — для каждой вершины заданы номера ее левого и правого ребенка. Если один из детей отсутствует, то вместо его номера записан ноль. Гарантируется, что входные данные корректны, то есть набор чисел во входном файле действительно задает двоичное дерево.

Формат выходного файла

Выведите в выходной файл одно число — количество способов раскрасить вершины заданного во входном файле двоичного дерева в красный и черный цвета так, чтобы оно стало красно-черным деревом.

Примеры

rbtrees.in	rbtrees.out
6 6 0 1 5 0 0 0 0 3 4 0 0	3
4 2 0 3 0 4 0 0 0	0

Все допустимые способы раскрасить вершины дерева из первого примера приведены на следующем рисунке.