

Содержание

Обязательные задачи	2
Задача A. Bridges. Мосты [2 sec, 256 mb]	2
Задача B. Points. Точки сочленения [2 sec, 256 mb]	3
Задача C. Condense 2. Конденсация графа [2 sec, 256 mb]	4
Задача D. Остовное дерево 2 [2 sec, 256 mb]	5
Обычные задачи	6
Задача E. Раскраска в три цвета [2 sec, 256 mb]	6
Задача F. Unionday. День Объединения [2 sec, 256 mb]	7
Задача G. Разрезание графа [2 sec, 256 mb]	8
Дополнительные задачи	9
Задача H. MST случайных точек [4 sec, 256 mb]	9
Задача I. Возьми себе за правило — летай всегда GraphAero! [3 sec, 256 mb]	10

Вы не умеете читать/выводить данные, открывать файлы? Воспользуйтесь [примерами](#).

В некоторых задачах большой ввод и вывод. Пользуйтесь [быстрым вводом-выводом](#).

Обратите внимание, что ввод-вывод во всех задачах стандартный.

Задачи расположены в **произвольном порядке**!

Обязательные задачи

Задача А. Bridges. Мосты [2 sec, 256 mb]

Дан неориентированный граф. Требуется найти все мосты в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leq 20\,000$, $m \leq 200\,000$).

Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i , e_i — номерами концов ребра ($1 \leq b_i, e_i \leq n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество мостов в заданном графе. На следующей строке выведите b целых чисел — номера ребер, которые являются мостами, в возрастающем порядке. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Пример

stdin	stdout
6 7	1
1 2	3
2 3	
3 4	
1 3	
4 5	
4 6	
5 6	

Задача В. Points. Точки сочленения [2 sec, 256 mb]

Дан неориентированный граф. Требуется найти все точки сочленения в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leq 20\,000$, $m \leq 200\,000$).

Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i , e_i — номерами концов ребра ($1 \leq b_i, e_i \leq n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество точек сочленения в заданном графе. На следующей строке выведите b целых чисел — номера вершин, которые являются точками сочленения, в возрастающем порядке.

Пример

stdin	stdout
9 12	3
1 2	1
2 3	2
4 5	3
2 6	
2 7	
8 9	
1 3	
1 4	
1 5	
6 7	
3 8	
3 9	

Задача C. Condense 2. Конденсация графа [2 сек, 256 mb]

Требуется найти количество ребер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных ребер.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leq 10\,000$, $m \leq 100\,000$). Следующие m строк содержат описание ребер, по одному на строке. Ребро номер i описывается двумя натуральными числами b_i , e_i — началом и концом ребра соответственно ($1 \leq b_i, e_i \leq n$). В графе могут присутствовать кратные ребра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество ребер в конденсации графа.

Пример

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

Задача D. Остовное дерево 2 [2 сек, 256 mb]

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

Обычные задачи

Задача Е. Раскраска в три цвета [2 сек, 256 mb]

Петя нарисовал на бумаге n кружков и соединил некоторые пары кружков линиями. После этого он раскрасил каждый кружок в один из трех цветов — красный, синий или зеленый.

Теперь Петя хочет изменить их раскраску. А именно — он хочет перекрасить каждый кружок в некоторый другой цвет так, чтобы никакие два кружка одного цвета не были соединены линией. При этом он хочет обязательно перекрасить каждый кружок, а перекрашивать кружок в тот же цвет, в который он был раскрашен исходно, не разрешается.

Помогите Пете решить, в какие цвета следует перекрасить кружки, чтобы выполнялось указанное условие.

Формат входных данных

Первая строка содержит два целых числа n и m — количество кружков и количество линий, которые нарисовал Петя, соответственно ($1 \leq n \leq 1\,000$, $0 \leq m \leq 20\,000$).

Следующая строка содержит n символов из множества $\{\text{'R'}, \text{'G'}, \text{'B'}\}$ — i -й из этих символов означает цвет, в который раскрашен i -й кружок ('R' — красный, 'G' — зеленый, 'B' — синий).

Следующие m строк содержат по два целых числа — пары кружков, соединенных отрезками.

Формат выходных данных

Выведите в выходной файл одну строку, состоящую из n символов из множества $\{\text{'R'}, \text{'G'}, \text{'B'}\}$ — цвета кружков после перекраски. Если решений несколько, выведите любое.

Если решения не существует, выведите в выходной файл слово “Impossible”.

Примеры

stdin	stdout
4 5 RRRG 1 3 1 4 3 4 2 4 2 3	BBGR
4 5 RGRR 1 3 1 4 3 4 2 4 2 3	Impossible

Задача F. Unionday. День Объединения [2 сек, 256 mb]

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

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leq n \leq 5\,000$) — количество городов в Байтеландии. Каждая из следующих n строк содержит два целых числа x_i, y_i — координаты i -го города ($-10\,000 \leq x_i, y_i \leq 10\,000$). 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} .

Примеры

stdin	stdout
6 1 1 7 1 2 2 6 2 1 3 7 3	9.65685

Задача G. Разрезание графа [2 сек, 256 mb]

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	

Дополнительные задачи

Задача Н. MST случайных точек [4 sec, 256 mb]

Даны n различных точек на плоскости. Координаты точек — целые числа от 0 до 30 000 включительно. Точки выбраны *случайно* в следующем смысле: рассмотрим все возможные наборы из n различных точек на плоскости с заданными ограничениями на координаты и выберем из них случайно и равновероятно один набор.

Вы можете провести отрезок между любыми двумя заданными точками. Длина отрезка между точками с координатами (x_1, y_1) и (x_2, y_2) равна $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Будем говорить, что точки a и b *связаны*, если они соединены отрезком, или же существует точка d , которая связана и с a , и с b . Ваша задача — провести отрезки минимальной суммарной длины так, чтобы все точки были связаны.

Формат входных данных

В первой строке ввода задано целое число n ($2 \leq n \leq 50\,000$). Следующие n строк содержат координаты точек. Гарантируется, что все точки различны. Кроме того, во всех тестах, кроме примера, гарантируется, что точки выбраны случайно, как описано в условии.

Формат выходных данных

В первой строке выведите вещественное число w — суммарную длину отрезков. В следующих $(n - 1)$ строках выведите отрезки, по одному на строке. Каждый отрезок следует выводить как два числа от 1 до n , обозначающие номера точек, являющихся концами этого отрезка.

Пусть на самом деле суммарная длина выведенных вами отрезков равна w^* , а суммарная длина отрезков в оптимальном ответе равна w_{opt} . Тогда ваш ответ будет считаться верным, если

$$\max \left(\left| \frac{w}{w^*} - 1 \right|, \left| \frac{w^*}{w_{\text{opt}}} - 1 \right| \right) < 10^{-12}.$$

Пример

stdin	stdout
4	22.02362358924615
0 10	1 2
5 6	2 3
10 0	4 2
0 0	

Задача I. Возьми себе за правило — летай всегда GraphAero! [3 sec, 256 mb]

Наконец авиаперевозки стали доступны всем и каждому! Однако, из-за жёсткой конкуренции в сфере пассажироперевозок осталось только две авиакомпании: «GraphAero Airlines» и «Aerofloat».

Авиакомпания «GraphAero Airlines» активно развивается. Ведь для получения большей прибыли... простите, для удобства пассажиров каждый месяц компания добавляет один новый рейс. Компании «Aerofloat» остаётся довольствоваться тем, что остаётся. А именно, единственная возможность удержаться на плаву — добавлять рейсы, дублирующие самые загруженные рейсы компании «GraphAero Airlines». Рейс является самым загруженным, если существует такая пара городов, что можно долететь (возможно, с пересадками) из одного города в другой, используя рейсы авиакомпании, но если этот рейс отменить — то долететь будет невозможно. Аналитикам компании «Aerofloat» необходимо постоянно контролировать ситуацию — сколько в данный момент существует самых загруженных рейсов.

Поскольку вы уже давно мечтаете летать по льготным ценам (скидка $10^{-5}\%$), вы решили оказать посильную помощь. Помните: самолёты летают по всему миру! Между двумя крупными городами может быть более одного рейса, а города бывают настолько большими, что самолёты могут летать в пределах одного города. Рейсами можно пользоваться как в одну, так и в другую сторону.

Формат входных данных

Первая строка входного файла содержит целое число N ($1 \leq N \leq 100\,000$) — количество городов и M ($0 \leq M \leq 100\,000$) — изначальное число рейсов компании «GraphAero Airlines». Далее следует M строк, в каждой содержится описание очередного рейса — номера двух городов, между которыми осуществляется рейс. В следующей строке содержится число K ($1 \leq K \leq 100\,000$) — количество добавленных рейсов. Далее содержится описание добавленных рейсов в таком же формате.

Формат выходных данных

После каждого добавления нового рейса выведите на отдельной строке одно число — количество самых загруженных рейсов.

Примеры

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