

Задача А. Соединение точек

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

Даны N точек на плоскости. Требуется провести отрезки между некоторыми парами точек таким образом, чтобы, во-первых, из любой данной точки в любую можно было пройти по этим отрезкам, а во-вторых, суммарная длина проведённых отрезков была минимальна.

Формат входных данных

В первой строке входного файла задано число N — количество точек ($1 \leq N \leq 200$). Следующие N строк содержат по два числа X_i Y_i каждая через пробел — координаты i -ой точки ($-1000 \leq X_i, Y_i \leq 1000$). Никакие две данные точки не совпадают, никакие три не лежат на одной прямой. Все числа во входном файле целые.

Формат выходных данных

В первой строке выходного файла выведите L — суммарную длину проведённых отрезков с точностью не менее шести десятичных знаков после запятой. Во второй строке выведите K — их количество. В следующих K строках выведите по два числа A_j B_j через пробел в каждой — номера точек, соединённых j -ым отрезком ($1 \leq A_j, B_j \leq N, A_j \neq B_j$). Точки нумеруются с единицы в том порядке, в котором они даны во входном файле. Если ответов с минимальным L несколько, разрешается вывести любой из них.

Примеры

connect.in	connect.out
4	3
0 0	3
0 1	1 2
1 0	2 4
1 1	4 3
5	7.064495
0 0	4
0 2	3 1
1 1	3 2
3 0	3 4
3 2	4 5

Задача В. Unionday. День Объединения

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

В Байтландии 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 Байтландии 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 Байтландии. 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

unionday.in	unionday.out
6	9.65685
1 1	
7 1	
2 2	
6 2	
1 3	
7 3	

Задача С. Остовное дерево 2

Имя входного файла: spantree2.in
Имя выходного файла: spantree2.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

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 опи-

сание 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

spantree2.in	spantree2.out
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

Задача D. Разрезание графа

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

cutting.in	cutting.out
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	