

Задача А. Остовное дерево

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

Даны точки на плоскости, являющиеся вершинами полного графа. Вес ребра равен расстоянию между точками, соответствующими концам этого ребра. Требуется в этом графе найти остовное дерево минимального веса.

Формат входного файла

Первая строка входного файла содержит натуральное число n — количество вершин графа ($1 \leq n \leq 5000$). Каждая из следующих n строк содержит два целых числа x_i, y_i — координаты i -й вершины ($-10\,000 \leq x_i, y_i \leq 10\,000$). Никакие две точки не совпадают.

Формат выходного файла

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

Примеры

spantree.in	spantree.out
3 0 0 1 0 0 1	2

Задача В. 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 входного файла

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 выходного файла

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 С. Транспортная сеть

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

Вовочка только что был назначен министром транспорта. В стране в настоящее время отсутствуют какие-либо транспортные средства между своими городами, так что Вовочке нужно все построить с нуля. Для этого он может строить дороги, чтобы соединить пары городов, а также может строить аэропорты в городах.

Построить дорогу длиной l стоит $R \times l$. Дорогу можно строить только от одного города до другого, по прямой. Построить аэропорт в городе стоит A рублей.

Вовочка хочет, чтобы из каждого города можно было попасть в каждый. Формально, в город Y можно добраться из города X , если выполняется одно какое-либо из следующих условий:

- Существует прямая дорога между X и Y .
- В X и Y есть аэропорты.
- Существует город Z , такой, что Z достижим из A и Y достижим из Z .

По данным координатам городов и константам R и A , найдите минимальную стоимость транспортной сети, которую министерство транспорта может построить.

Формат входного файла

Первая строка входного файла содержит число n — число городов ($1 \leq n \leq 150$). Вторая строка содержит n чисел — x -координаты городов. Третья строка содержит n чисел — y -координаты городов. Координаты от 0 до 10^6 . Четвертая строка содержит вещественные числа R и A .

Формат выходного файла

Выведите минимальную стоимость, которую можно получить, с точностью до 6 знаков.

Примеры

transport.in	transport.out
4 0 0 400 400 0 100 0 100 1.0 150.0	500.0
5 0 0 400 400 2000 0 100 0 100 2000 1.0 500.0	1600.0
8 0 100 200 300 400 2000 2100 2200 0 100 200 300 400 2000 2100 2200 0.5 200.0	824.2640687119285

Задача D. Кратчайший путь в невзвешенном графе

Имя входного файла: pathbge1.in
Имя выходного файла: pathbge1.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Дан неориентированный невзвешенный граф. Найдите кратчайшее расстояние от первой вершины до всех вершин.

Формат входного файла

В первой строке входного файла два числа: n и m ($2 \leq n \leq 30000, 1 \leq m \leq 400000$), где n — количество вершин графа, а m — количество ребер.

Следующие m строк содержат описание ребер. Каждое ребро задается стартовой вершиной и конечной вершиной. Вершины нумеруются с единицы.

Формат выходного файла

Выведите n чисел — для каждой вершины кратчайшее расстояние до нее.

Пример

pathbge1.in	pathbge1.out
2 1	0 1
2 1	

Задача Е. Поиск цикла

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

Дан ориентированный невзвешенный граф. Необходимо определить есть ли в нём циклы, и если есть, то вывести любой из них.

Формат входного файла

В первой строке входного файла находятся два натуральных числа N и M ($1 \leq N \leq 100\,000, M \leq 100\,000$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходного файла

Если в графе нет цикла, то вывести «NO», иначе — «YES» и затем перечислить все вершины в порядке обхода цикла.

Пример

<code>cycle.in</code>	<code>cycle.out</code>
2 2 1 2 2 1	YES 2 1
2 2 1 2 1 2	NO

Задача F. Топологическая сортировка

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

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входного файла

В первой строке входного файла даны два натуральных числа N и M ($1 \leq N \leq 100\,000$, $0 \leq M \leq 100\,000$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходного файла

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, вывести -1.

Пример

<code>topsort.in</code>	<code>topsort.out</code>
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5
3 3 1 2 2 3 3 1	-1

Задача G. Кратчайший путь

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

Дан ориентированный взвешенный ациклический граф. Требуется найти в нем кратчайший путь из вершины s в вершину t .

Формат входного файла

Первая строка входного файла содержит четыре целых числа n , m , s и t — количество вершин, дуг графа, начальная и конечная вершина соответственно. Следующие m строк содержат описания дуг по одной на строке. Ребро номер i описывается тремя натуральными числами b_i , e_i и w_i — началом, концом и длиной дуги соответственно ($1 \leq b_i, e_i \leq n$, $|w_i| \leq 1000$).

Входной граф не содержит циклов и петель.

$1 \leq n \leq 100\,000$, $0 \leq m \leq 200\,000$.

Формат выходного файла

Первая строка выходного файла должна содержать одно целое число — длину кратчайшего пути из s в t . Если пути из s в t не существует, выведите «Unreachable».

Пример

shortpath.in	shortpath.out
2 1 1 2 1 2 -10	-10
2 1 2 1 1 2 -10	Unreachable

Задача Н. Гамильтонов путь

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

Дан ориентированный граф без циклов. Требуется проверить, существует ли в нем путь, проходящий по всем вершинам.

Формат входного файла

Первая строка входного файла содержит два целых числа n и m — количество вершин и дуг графа соответственно. Следующие m строк содержат описания дуг по одной на строке. Ребро номер i описывается двумя натуральными числами b_i и e_i — началом и концом дуги соответственно ($1 \leq b_i, e_i \leq n$).

Входной граф не содержит циклов и петель.

$1 \leq n \leq 100\,000$, $0 \leq m \leq 200\,000$.

Формат выходного файла

Если граф удовлетворяет требуемому условию, то выведите YES, иначе NO.

Пример

hamiltonian.in	hamiltonian.out
3 3 1 2 1 3 2 3	YES
3 2 1 2 1 3	NO

Задача I. Игра

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

Дан ориентированный невзвешенный ациклический граф. На одной из вершин графа стоит «фишка». Двое играют в игру. Пусть «фишка» находится в вершине u , и в графе есть ребро (u, v) . Тогда за ход разрешается перевести «фишку» из вершины u в вершину v . Проигрывает тот, кто не может сделать ход.

Формат входного файла

В первой строке входного файла находятся три натуральных числа N , M и S ($1 \leq N, S, M \leq 100\,000$) — количество вершин, рёбер и вершина, в которой находится «фишка» в начале игры соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин.

Формат выходного файла

Если выигрывает игрок, который ходит первым, выведите «First player wins», иначе — «Second player wins».

Пример

game.in	game.out
3 3 1 1 2 2 3 1 3	First player wins
3 2 1 1 2 2 3	Second player wins

Задача J. Мосты

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

Дан неориентированный граф. Требуется найти все мосты в нем.

Формат входного файла

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 выходного файла

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

bridges.in	bridges.out
6 7	1
1 2	3
2 3	
3 4	
1 3	
4 5	
4 6	
5 6	

Задача К. Точки сочленения

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

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

Формат входного файла

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 выходного файла

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

<code>points.in</code>	<code>points.out</code>
9 12 1 2 2 3 4 5 2 6 2 7 8 9 1 3 1 4 1 5 6 7 3 8 3 9	3 1 2 3

Задача L. Компоненты сильный связности

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

Дан ориентированный невзвешенный граф. Необходимо выделить в нем компоненты сильной связности и топологически их отсортировать.

Формат входного файла

В первой строке входного файла находятся два натуральных числа N и M ($1 \leq N \leq 20\,000, 1 \leq M \leq 200\,000$) — количество вершин и рёбер в графе, соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходного файла

Первая строка выходного файла должна содержать целое число k — количество компонент сильной связности в графе. Вторая строка выходного файла должна содержать n чисел — для каждой вершины выведите номер компоненты сильной связности, которой она принадлежит. Компоненты должны быть занумерованы таким образом, чтобы для каждого ребра (u, v) номер компоненты, которой принадлежит u не превосходил номер компоненты, которой принадлежит v .

Пример

cond.in	cond.out
6 7	2
1 2	1 1 1 2 2 2
2 3	
3 1	
4 5	
5 6	
6 4	
2 4	

Задача М. Конденсация графа

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

Дан ориентированный невзвешенный граф. Необходимо найти его конденсацию.

Формат входного файла

В первой строке входного файла находятся два натуральных числа N и M ($1 \leq N \leq 20\,000, 1 \leq M \leq 200\,000$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходного файла

Выведите в выходной файл два числа: число вершин и ребер в конденсации графа.

Пример

cond.in	cond.out
6 7 1 2 2 3 3 1 4 5 5 6 6 4 2 4	2 1

Задача N. Пожарные депо

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

В городе есть n перекрестков, некоторые из которых соединены односторонними дорогами. Всего в городе m дорог.

Мэр решил построить несколько пожарных депо. Для этого ему надо выбрать, на каких перекрестках их построить. При этом когда пожарная машина едет на пожар, она может ехать как по направлению движения по дороге, так и против этого направления. А вот при возвращении в депо срочности нет и ехать «по встречке» уже нельзя.

Помогите мэру построить минимальное число депо, чтобы при пожаре дома на любом перекрестке пожарная машина из одного из депо могла доехать до этого перекрестка (возможно против движения), а затем могла вернуться в свое депо (уже двигаясь только по направлению дорог).

Формат входного файла

В первой строке входного файла находятся два натуральных числа n и m ($1 \leq n \leq 20\,000, 1 \leq m \leq 200\,000$) — количество вершин и улиц. Далее в m строках перечислены улицы, каждая улица задаётся парой чисел — номерами начального и конечного перекрестка.

Формат выходного файла

Выведите одно число — минимальное число депо, которые необходимо построить.

Пример

fire.in	fire.out
6 7 1 2 2 3 3 1 1 4 1 5 4 6 6 4	2

В приведенном примере можно построить депо, например, на перекрестках 4 и 5.