Задача A. Матрица инцидентности

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

Вершина графа u называется *инцидентной* ребру e, если u является одним из концов ребра e.

Аналогично, ребро e называется *инцидентным* вершине u, если один из концов e — это вершина u.

Матрицей инцидентности графа $G = (V, E)$ называется прямоугольная таблица из $|V|$ строк и $|E|$ столбцов, в которой на пересечении i-ой строки и j-го столбца записана единица, если вершина i инцидентна ребру j, и ноль в противном случае.

Дан неориентированный граф. Выведите его матрицу инцидентности.

Формат входного файла

В первой строке входного файла заданы числа N и M через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$); каждая такая строка означает, что в графе существует ребро между вершинами u_i и v_i. Рёбра нумеруются в том порядке, в котором они даны во входном файле, начиная с единицы.

Формат выходного файла

Выведите в выходной файл N строк, по M чисел в каждой. j-ый элемент i-ой строки должен быть равен единице, если вершина i инцидентна ребру j, и нулю в противном случае. Разделяйте соседние элементы строки одним пробелом.

Примеры

incident.in	incident.out
3 2 1 2 2 3	1 0 1 1 0 1
2 2 1 1 1 2	1 1 0 1

Задача B. Кратчайшее расстояние

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

Взвешенный граф — это граф, каждому ребру которого сопоставлено некоторое число, называемое *весом* этого ребра. Вес ребра может иметь различный смысл — например, считаться длиной ребра или стоимостью прохода по нему.

Дан взвешенный неориентированный граф. Найдите кратчайшее расстояние между заданными двумя вершинами.

Формат входного файла

В первой строке входного файла заданы числа N и M через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$). Следующие M строк содержат по три числа u_i, v_i и w_i через пробел ($1 \leqslant u_i, v_i \leqslant N$, $1 \leqslant w_i \leqslant 10^6$); каждая строка соответствует ребру между вершинами u_i и v_i длиной w_i. И наконец, в следующей за ними строке записаны два числа x и y ($1 \leqslant x, y \leqslant N$) — номера вершин, между которыми необходимо найти кратчайшее расстояние. Все числа во входном файле целые.

Формат выходного файла

Выведите в выходной файл одно число — кратчайшее расстояние между вершинами x и y. Если между этими вершинами нет пути, выведите -1.

Примеры

distance.in	distance.out
3 3 1 2 1 2 3 1 1 3 10 1 3	2
2 3 1 2 1 2 1 3 1 2 2 2 1	1
2 0 1 2	-1

Задача C. Связность

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

В этой задаче требуется проверить, что граф является *связным*, то есть что из любой вершины можно по рёбрам этого графа попасть в любую другую.

Формат входного файла

В первой строке входного файла заданы числа N и M через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$); каждая такая строка означает, что в графе существует ребро между вершинами u_i и v_i.

Формат выходного файла

Выведите "`YES`", если граф является связным, и "`NO`" в противном случае.

Примеры

connect.in	connect.out
3 2 1 2 3 2	YES
3 1 1 3	NO
4 4 1 2 1 2 1 4 2 3	YES

Задача D. Волновой обход графа

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

Пусть *расстояние* от вершины u до вершины v — это минимальное количество рёбер в пути между u и v; так, расстояние между u и u — 0, а расстояние между любыми двумя различными соседними вершинами — 1.

Волновым обходом графа из вершины v назовём последовательность вершин $u_1, u_2, \ldots, u_r$ такую, что:

- $u_1 = v$,
- Каждая вершина графа, достижимая из v, встречается в ней хотя бы один раз, и
- Каждая следующая вершина последовательности удалена от вершины v не меньше, чем предыдущая.

Задан связный неориентированный граф и его вершина v. Выведите любой волновой обход этого графа.

Формат входного файла

В первой строке входного файла заданы числа N, M и v через пробел — количество вершин и рёбер в графе и начальная вершина обхода ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$, $1 \leqslant v \leqslant N$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$); каждая такая строка означает, что в графе существует ребро между вершинами u_i и v_i.

Формат выходного файла

В первой строке входного файла выведите число r — количество вершин в найденном волновом обходе ($1 \leqslant r \leqslant 10\,000$; гарантируется, что обход, удовлетворяющий этим ограничениям, существует). Во второй строке выведите сами числа $u_1, u_2, \ldots, u_r$ через пробел.

Примеры

wave.in	wave.out
3 2 1 1 2 2 3	3 1 2 3
4 4 1 1 2 2 3 3 4 4 1	4 1 2 4 3

Задача E. Пошаговый обход графа

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

Пошаговым обходом графа из вершины v назовём последовательность вершин $u_1, u_2, \ldots, u_r$ такую, что:

- $u_1 = u_r = v$,
- Каждая вершина графа, достижимая из v, встречается в ней хотя бы один раз, и
- Между любыми двумя соседними вершинами последовательности в графе существует ребро.

Задан связный неориентированный граф и его вершина v. Выведите любой пошаговый обход этого графа.

Формат входного файла

В первой строке входного файла заданы числа N, M и v через пробел — количество вершин и рёбер в графе и начальная вершина обхода ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$, $1 \leqslant v \leqslant N$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$); каждая такая строка означает, что в графе существует ребро между вершинами u_i и v_i.

Формат выходного файла

В первой строке входного файла выведите число r — количество вершин в найденном пошаговом обходе ($1 \leqslant r \leqslant 10\,000$; гарантируется, что обход, удовлетворяющий этим ограничениям, существует). Во второй строке выведите сами числа $u_1, u_2, \ldots, u_r$ через пробел.

Примеры

step.in	step.out
3 2 1 1 2 2 3	5 1 2 3 2 1
4 4 1 1 2 2 3 3 4 4 1	5 1 2 3 4 1

Задача F. Количество циклов

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

Формально, *путь* в графе — это чередующаяся последовательность вершин и рёбер $u_1, e_1, u_2, e_2, u_3, \ldots, u_k$, начинающаяся и заканчивающаяся вершиной и такая, что любые соседние вершина и ребро в ней инцидентны.

Цикл — это путь, начальная и конечная вершины которого совпадают. В цикле должно быть хотя бы одно ребро.

Простой путь отличается от обычного пути тем, что в нём не может быть повторяющихся вершин.

Простой цикл — это цикл, в котором нет повторяющихся вершин и рёбер.

Дан неориентированный граф. Посчитайте, сколько в нём различных простых циклов. Заметим, что циклы считаются одинаковыми, если они обходят одно и то же множество вершин в одном и том же порядке, возможно, начиная при этом из другой вершины, или если порядок обхода противоположный. Например, циклы с порядком обхода вершин $1, 2, 3, 1$, $2, 3, 1, 2$ и $1, 3, 2, 1$ считаются одинаковыми, а циклы $1, 2, 3, 4, 1$ и $1, 3, 4, 2, 1$ — нет, поскольку порядок обхода вершин различен.

Формат входного файла

В первой строке входного файла заданы числа N и M через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 10$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$, $u_i \neq v_i$); каждая такая

строка означает, что в графе существует ребро между вершинами u_i и v_i. В графе нет кратных рёбер.

Формат выходного файла

Выведите одно число — количество простых циклов в заданном графе.

Примеры

numcycle.in	numcycle.out
3 2 1 2 2 3	0
4 5 1 2 2 3 3 4 4 1 1 3	3

Задача G. Наибольшее расстояние

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

В некоторой стране N городов и M дорог. Каждая дорога представляет собой отрезок прямой, соединяющий два различных города, причём известно, что ни на одной дороге нет других городов, кроме тех, что задают её концы. Расстоянием между двумя городами считается минимальная суммарная длина дорог, по которым надо пройти, чтобы попасть из одного города в другой. Требуется вычислить наибольшее из таких расстояний.

Формат входного файла

В первой строке входного файла задано число N ($2 \leqslant N \leqslant 100$). В последующих N строках содержатся пары целых чисел $X_i\,Y_i$ ($-1000 \leqslant X_i, Y_i \leqslant 1000$), задающие координаты i-го города на плоскости. Координаты всех городов различны. В следующей, $N+2$-ой строке, задано число M. Далее в M строках записаны пары чисел $U_j\,V_j$, означающие, что существует дорога между городами U_j и V_j. Дороги не имеют общих точек вне городов. Длина дороги между городами A и B задаётся обычным евклидовым расстоянием между точками плоскости: $\rho(A,B) = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}$.

Формат выходного файла

Выведите одно число — максимальное расстояние между двумя городами с точностью по крайней мере 10^{-6}. Если существует пара городов, между которыми нет пути из дорог, выведите -1.

Примеры

maxdist.in	maxdist.out
2 0 0 1 1 1 1 2	1.414213
2 0 0 1 1 0	-1
5 0 0 0 2 2 0 2 2 1 1 6 1 2 2 4 4 3 3 1 2 5 4 5	4

Задача H. Стек

Имя входного файла:	stack.in
Имя выходного файла:	stack.out
Ограничение по времени:	
Ограничение по памяти:	64 мегабайта

Вам нужно промоделировать работу стека. Во входном файле написаны команды, после выполнения каждой команды вам надо выводить состояние стека.

Формат входного файла

В первой строке одно число $1 \le N \le 1000$ - количество команд. В каждой из N последующих строк по одной из команд.

- `PUSH` число - положить на вершину стека целое число и вывести состояние стека.
- `POP` - взять с вершины стека число, вывести его и вывести состояние стека.

Формат выходного файла

В каждой строке выходного файла вывести состояние стека (для `PUSH`) и и полученный элемент и состояние стека (для `POP`) как показано в примерах.

Примеры

stack.in	stack.out
4 PUSH 2 POP PUSH 1 POP	[2] 2 [] [1] 1 []
5 PUSH 3 PUSH 4 POP PUSH 2 POP	[3] [3, 4] 4 [3] [3, 2] 2 [3]
7 PUSH 1 PUSH 7 PUSH 3 POP POP PUSH 2 PUSH 9	[1] [1, 7] [1, 7, 3] 3 [1, 7] 7 [1] [1, 2] [1, 2, 9]

Задача I. Очередь

Имя входного файла: `queue.in`
Имя выходного файла: `queue.out`
Ограничение по времени:
Ограничение по памяти: 64 мегабайта

Вам нужно промоделировать работу очереди. Во входном файле написаны команды, после выполнения каждой команды вам надо выводить состояние очереди.

Формат входного файла

В первой строке одно число $1 \le N \le 1000$ - количество команд. В каждой из N последующих строк по одной из команд.

- `PUT` X - положить в начало очереди целое число и вывести состояние очереди. $0 \le X \le 10^9$.
- `GET` - взять из конца очереди элемент, вывести его и вывести состояние очереди.

Формат выходного файла

В каждой строке выходного файла вывести состояние очереди (для `PUT`) и и полученный элемент и состояние стека (для `GET`) как показано в примерах.

Примеры

queue.in	queue.out
5 PUT 3 PUT 4 PUT 3 GET PUT 6	[3] [4, 3] [3, 4, 3] 3 [3, 4] [6, 3, 4]
6 PUT 7 PUT 9 PUT 7 PUT 9 GET GET	[7] [9, 7] [7, 9, 7] [9, 7, 9, 7] 7 [9, 7, 9] 9 [9, 7]