Задача A. Почти полный граф

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

Граф называется *почти полным*, если для любых трёх различных вершин в нём есть хотя бы два ребра между какими-то из этих вершин. Требуется проверить, является ли данный граф почти полным. В графе нет петель и кратных рёбер.

Поскольку граф может быть очень большим, он задан не матрицей смежности и даже не списком рёбер, а следующим кодом. Пусть $a_{i,j}$ равно единице, если между вершинами i и j есть ребро, и нулю в противном случае. Выпишем подряд числа $a_{1,2}$, $a_{1,3}$, …, $a_{1,n}$, $a_{2,3}$, $a_{2,4}$, …, $a_{2,n}$, $a_{3,4}$, …, $a_{n-1,n}$ (всего у нас получится $\frac{n(n-1)}{2}$ чисел). Теперь разобьём эту последовательность на группы одинаковых чисел, идущих подряд. Закодируем её так: сначала запишем s — первое число (0 или 1), затем — размер первой группы t_1 (символы в начале последовательности, совпадающие с s), затем — размер второй группы t_2 (группа одинаковых символов после первых t_1, не совпадающих с s), затем — размер третьей группы t_3 (группа одинаковых символов после первых $t_1 + t_2$, совпадающих с s), и так далее до t_l — размера последней группы одинаковых символов.

Формат входных данных

В первой строке входных данных заданы два целых числа n и l через пробел — количество вершин графа и количество групп в последовательности, задающей граф ($2 \leqslant n \leqslant 50\,000$, $1 \leqslant l \leqslant 50\,000$). Во второй строке записаны через пробел целые числа s t_1 t_2 … t_l; s равно 0 или 1, все t_i положительны и сумма $t_1 + t_2 + \ldots + t_l$ равна $\frac{n(n-1)}{2}$.

Формат выходных данных

Выведите в первой строке слово «`YES`», если данный граф является почти полным, и «`NO`» в противном случае. Кроме того, если ответ — «`NO`», то во второй строке выведите три числа — номера трёх вершин графа, между которыми нет хотя бы двух рёбер.

Пример

almost.in	almost.out
6 4 0 1 4 5 5	NO 2 3 4

Пояснение к примеру

Матрица смежности данного графа имеет следующий вид:

0	0	1	1	1	1
0	0	0	0	0	0
1	0	0	0	1	1
1	0	0	0	1	1
1	0	1	1	0	1
1	0	1	1	1	0

Очевидно, что этот граф не является почти полным: например, между вершинами 2, 3 и 4 (считая с единицы) нет ни одного ребра, а требуется, чтобы их было не менее двух.

Задача B. Кротовья нора

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

Кротовья нора состоит из пещерок, соединённых коридорами, причём из каждой пещерки выходит ровно три коридора. Коридорами и пещерками пользуется не только крот — в одной из пещерок обосновался жук-носорог, а в другую только что свалился жук-олень. Жук-олень чует своего соперника и спешит встретиться, чтобы выяснить, кто из них будет главным в кротовьей норе!

Выясните, каким минимальным количеством коридоров потребуется пройти жуку-оленю, чтобы попасть в пещерку, где его поджидает жук-носорог.

Формат входных данных

В первой строке заданы три числа: N ($4 \leqslant N \leqslant 1000$) — количество пещерок в кротовьей норе, X ($1 \leqslant X \leqslant N$) — номер пещерки жука-носорога и Y ($1 \leqslant Y \leqslant N$) — номер пещерки жука-оленя. В последующих N строках стоит по три числа в каждой: в $(i+1)$-й строке стоят номера пещерок, в которые из i-й пещерки ведут коридоры. Каждый коридор упоминается в строках обеих вершин, которые он соединяет. Никакой коридор не соединяет пещерку с самой собой, и никакие две пещерки не соединены более чем одним коридором. Гарантируется, что $X \neq Y$. Все числа во входных данных целые.

Формат выходных данных

Выведите одно целое число — минимальное количество коридоров, которыми требуется пройти, чтобы попасть из пещерки жука-оленя в пещерку жука-носорога. Если это оказалось невозможно, выведите число -1.

Примеры

burrow.in	burrow.out
4 1 4 2 3 4 3 4 1 4 1 2 1 2 3	1
6 2 3 4 5 6 4 5 6 4 5 6 1 2 3 1 2 3 1 2 3	2
8 5 4 2 3 4 1 3 4 1 2 4 1 2 3 6 7 8 5 7 8 5 6 8 5 6 7	-1

Задача C. Любители Кошек

Имя входного файла:	`catlover.in`
Имя выходного файла:	`catlover.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мебибайт

В университетском клубе любителей кошек зарегистрировано n членов. Естественно, что некоторые из членов клуба знакомы друг с другом. Нужно сосчитать, сколькими способами можно выбрать из них троих, которые могли бы свободно общаться (то есть, любые два из которых знакомы между собой).

Формат входных данных

В первой строке ввода заданы числа n и m ($1 \leqslant n \leqslant 1000$, $1 \leqslant m \leqslant 30\,000$), где m обозначает общее число знакомств. В последующих m строках идут пары чисел a_i b_i, обозначающие, что a_i знаком с b_i. Информация об одном знакомстве может быть записана несколько раз, причём даже в разном порядке (как (x, y), так и (y, x)).

Формат выходных данных

Выведите одно число — количество способов выбрать троих попарно знакомых друг с другом людей из клуба.

Пример

catlover.in	catlover.out
3 3 1 2 2 3 3 1	1

Задача D. Связность

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

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 входных данных

В первой строке 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 выходных данных

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

Задача E. Дейкстра

Имя входного файла:	dijkstra.in
Имя выходного файла:	dijkstra.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дан ориентированный взвешенный граф. Найдите кратчайшее расстояние от одной заданной вершины до другой.

Формат входных данных

В первой строке даны три числа: N, S и F ($1 \leqslant N \leqslant 1000$, $1 \leqslant S, F \leqslant N$), где N — количество вершин графа, S — начальная вершина, а F — конечная. В следующих N строках задано по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы всегда стоят нули. Веса всех рёбер целые и не превосходят $10\,000$.

Формат выходных данных

Вывести искомое расстояние или -1, если пути не существует.

Пример

dijkstra.in	dijkstra.out
3 1 2 0 -1 2 3 0 -1 -1 4 0	6

Задача F. Флойд

Имя входного файла:	floyd.in
Имя выходного файла:	floyd.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дан ориентированный взвешенный граф. В нём вам необходимо найти пару вершин, кратчайшее расстояние от одной из которых до другой максимально среди всех пар вершин.

Формат входных данных

В первой строке запиасно целое число N — количество вершин графа ($1 \leqslant N \leqslant 100$). В каждой из следующих N строк задано по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы всегда стоят нули. Гарантируется, что хотя бы одно ребро в графе присутствует.

Формат выходных данных

Вывести искомое максимальное кратчайшее расстояние.

Пример

floyd.in	floyd.out
4 0 5 9 -1 -1 0 2 8 -1 -1 0 7 4 -1 -1 0	16

Задача G. (p, q)-лошадь

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

(p, q)-лошадь — это обобщение обычного шахматного коня. (p, q)-лошадь своим ходом перемещается на p клеток в одном направлении и на q — в другом (перпендикулярном). Например, $(3, 4)$-лошадь может переместиться с клетки $(5, 6)$ на клетки $(1, 3)$, $(2, 2)$, $(2, 10)$, $(1, 9)$, $(8, 10)$, $(9, 9)$, $(8, 2)$ и $(9, 3)$. Очевидно, что обычный шахматный конь — это $(2, 1)$-лошадь.

Ваша задача — определить минимальное число ходов, которое требуется (p, q)-лошади, чтобы добраться от одной клетки шахматной доски $M \times N$ до другой. За пределы доски выходить запрещается.

Формат входных данных

Единственная строчка входных данных содержит восемь целых чисел M, N, p, q, x_1, y_1, x_2, y_2 ($1 \leqslant x_1, x_2 \leqslant M \leqslant 100$, $1 \leqslant y_1, y_2 \leqslant N \leqslant 100$, $0 \leqslant p \leqslant 100$, $0 \leqslant q \leqslant 100$).

Формат выходных данных

В первой строке выведите целое число k — минимальное число ходов, которое требуется (p, q)-лошади, чтобы добраться из клетки (x_1, y_1) в клетку (x_2, y_2). Далее должны следовать $k + 1$ строк, в которых должны быть записаны последовательные положения (p, q)-лошади на этом пути.

Если (p, q)-лошадь не может добраться из (x_1, y_1) в (x_2, y_2), выведите единственное число -1.

Примеры

horse.in	horse.out
3 3 1 1 1 1 3 3	2 1 1 2 2 3 3
2 2 1 1 1 1 1 2	-1

Задача H. Матрица инцидентности

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

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-й 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 входных данных

В первой строке входных данных 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 заданы, начиная с единицы.

Формат выходных данных

Выведите в N строк, по M чисел в каждой; j-й элемент i-й 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 I. Друзья и враги

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

На острове Дженту живут n крокодилов. Каждый крокодил имеет ровно одного друга и ровно одного врага среди остальных крокодилов на острове. Отношения дружбы и вражды симметричны: если крокодил $\mathcal{A}$ — друг крокодила $\mathcal{B}$, то крокодил $\mathcal{B}$ — друг крокодила $\mathcal{A}$; если же крокодил $\mathcal{A}$ — враг крокодила $\mathcal{B}$, то и крокодил $\mathcal{B}$ — враг крокодила $\mathcal{A}$. Никакой крокодил не является одновременно врагом и другом другого крокодила. Кроме того, никакой крокодил не может быть ни другом, ни врагом самому себе.

Крокодилы острова Дженту весьма эмоциональны. Когда встречаются два врага, они со страшной силой колотят по земле хвостами, а затем происходит Битва Крокодилов. Когда встречаются два друга, они исполняют разрушительный Танец Дружбы Крокодилов.

Недавно танцы и битвы разбудили вулкан, находящийся под островом, и остров раскололся на две части. Вулкан успокоился, но крокодилы опасаются, что дальнейшие танцы и битвы разбудят его снова, и обе половины острова зальёт лава.

Теперь крокодилы хотят расселиться по двум половинам острова таким образом, чтобы на каждой половине оказалось *нейтральное* множество крокодилов — такое множество, что никакие два крокодила в нём не являются ни друзьями, ни врагами. Расселившись так, крокодилы не будут устраивать танцы и битвы, а значит, можно надеяться, что вулкан не будет извергаться снова.

Крокодилы острова Дженту мудры, но непрактичны. Они понимают, что при таком устройстве дружбы и вражды разделение крокодилов на два нейтральных множества всегда возможно, но не знают, как именно разделиться на такие множества.

Помогите им! Найдите такое разбиение крокодилов на два множества, что у каждого крокодила в своём множестве нет ни друзей, ни врагов.

Формат входных данных

В первой строке задано натуральное число n — количество крокодилов ($4 \leqslant n \leqslant 100$). Следующие n строк описывают крокодилов. В первой из них записаны два числа f_1 и e_1 через пробел — номер друга первого крокодила и номер его врага. Во второй записаны числа f_2 и e_2 — номера друга и врага второго крокодила, и так далее. В последней из этих строк записаны f_n и e_n — друг и враг крокодила с номером n. Крокодилы пронумерованы числами от 1 до n в том порядке, в котором они описываются во входных данных.

Все числа f_k и e_k целые и лежат в пределах от 1 до n, включительно. Отношения дружбы и вражды симметричны. Друг и враг каждого крокодила различны. Никакой крокодил не является ни другом, ни врагом самому себе.

Формат выходных данных

В первой строке выведите n чисел через пробел. Каждое из этих чисел должно быть равно либо 1, либо 2. Если i-е и j-е числа равны, это означает, что крокодилы с номерами i и j оказались на одной половине острова. Если же эти числа различны, значит, крокодилы i и j оказались на разных половинах острова.

Если правильных ответов несколько, можно вывести любой из них.

Пример

neutral.in	neutral.out
4 2 4 1 3 4 2 3 1	1 2 1 2

Задача J. Расстояние от корня

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

В заданном корневом дереве найдите вершины, максимально удалённые от корня. Расстоянием между вершинами считается количество рёбер в пути.

Формат входных данных

В первой строке задано n — количество вершин в дереве ($1 \leqslant n \leqslant 100$). В следующих $n-1$ строках заданы вершины, являющиеся предками вершин 2, 3, …, n. Вершина 1 является корнем дерева.

Формат выходных данных

В первой строке выведите максимальное расстояние от корня до остальных вершин дерева. Во второй строке выведите, сколько вершин дерева находят-

ся от корня на таком расстоянии. В третьей строке выведите номера этих вершин через пробел в порядке возрастания.

Примеры

rootdist.in	rootdist.out
3 1 1	1 2 2 3
3 1 2	2 1 3

Задача K. Пошаговый обход графа

Имя входного файла:	step.in
Имя выходного файла:	step.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

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

В первой строке 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 выходных данных

В первой строке 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 L. Подграф

Имя входного файла:	subgraph.in
Имя выходного файла:	subgraph.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дан неориентированный граф. Требуется удалить из него некоторые рёбра так, чтобы получился связный подграф, содержащий все вершины исходного графа, в котором суммарная длина всех рёбер минимальна.

Формат входных данных

В первой строке заданы два целых числа n и m через пробел — количество вершин и рёбер в исходном графе, соответственно ($1 \leqslant n \leqslant 100$, $m \geqslant 0$). Следующие m строк описывают рёбра; i-я из них содержит три целых числа u_i, v_i и w_i — номера концов ребра и его длину ($1 \leqslant u_i, v_i \leqslant n$, $u_i \neq v_i$, $|w_i| \leqslant 10\,000$).

Формат выходных данных

Если невозможно удалить рёбра так, чтобы получился связный подграф, выведите «Impossible» (без кавычек). В противном случае выведите одно число — минимальную сумму длин рёбер подграфа.

Пример

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

Задача M. Сумма расстояний

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

Дан связный граф. Требуется найти сумму расстояний между всеми парами вершин.

Формат входных данных

Первая строка содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($1 \leqslant n \leqslant 1000$, $0 \leqslant m \leqslant 10\,000$).

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

Гарантируется, что граф связен.

Формат выходных данных

Первая строка должна содержать одно натуральное число — сумму попарных расстояний между вершинами.

Пример

sumdist.in	sumdist.out
5 5 1 2 2 3 3 4 5 3 1 5	16

Задача N. Синхронизация

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

Учёные взялись за какую-то большую и сложную проблему. Для её решения написали распределённое приложение. Процессы приложения во время решения задачи должны обмениваться сообщениями. Каждый учёный запустил один или несколько процессов приложения на своём компьютере, и спустя некоторое время решение проблемы было найдено.

После того, как решение было получено, возник вопрос о том, как происходил обмен данными между компьютерами. Но вот незадача — времена на компьютерах не были синхронизированы, и осталась только информация о времени запуска приложения, времена вызовов функций пересылки и приёма данных и их продолжительность.

Ваша задача — найти, на сколько отличаются времена на компьютерах, на которых считалась задача. Но из-за того, что время отправления сигнала регистрировалось в момент окончания отправления, а время получения сигнала регистрировалось в момент начала приёма сигнала, может сложиться такая ситуация, что определение возможной разности времён на компьютерах невозможно.

Формат входных данных

В первой строке вводятся через пробел два числа N и M ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$), где N — количество компьютеров, а M — количество пересланных пакетов данных. В следующих M строках задаются параметры пакетов данных: в каждой строке по три числа, записанных через пробел, i, j и Δt ($1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant N$, $-10\,000 \leqslant \Delta t \leqslant 10\,000$). Здесь i — номер компьютера, получившего пакет данных, j — номер компьютера, пославшего пакет данных, а Δt — разница между временем начала получения i-м компьютером и временем окончания отправки j-м компьютером пакета данных. В Δt входят относительные времена, которые измерялись на i-м и j-м компьютерах соответственно. Поэтому разность реальных времён начала получения и окончания отправки данных могла быть отлична от Δt, если времена на компьютерах отличались.

Формат выходных данных

Выведите N чисел, по одному в каждой строке. В i-й строке должна содержаться возможная разность между временем i-го компьютера и сервера (сервер — компьютер с номером 1). При этом рассчитанные времена должны удовлетворять тому условию, что момент начала получения был не раньше момента окончания отправки каждого из пакетов данных. Если возможных наборов времён несколько, то нужно выдать любой из них. Если возможных наборов времён нет, то необходимо выдать строку «`NO SOLUTION`» (без кавычек).

Пример

synchro.in	synchro.out
3 6 1 2 5 2 1 5 1 3 1 3 1 1 2 3 1 3 2 1	0 2 1

Задача O. Волновой обход графа

Имя входного файла:	wave.in
Имя выходного файла:	wave.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

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 такую последовательность вершин $u_1, u_2, \ldots, u_r$, 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 входных данных

В первой строке 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 выходных данных

В первой строке 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