Задача A. Сумма двух

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

Формат входных данных

В первой строке входного файла расположены два целых числа A и B, не превосходящих 1 000 по модулю.

Формат выходных данных

Ваша программа должна выдавать в выходной файл одно число — сумму чисел A и B.

Пример

sum.in	sum.out
2 3	5
17 -18	-1

Задача B. От матрицы смежности к списку ребер

Имя входного файла:	`m2e.in`
Имя выходного файла:	`m2e.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Простой неориентированный граф задан матрицей смежности, выведите его представление в виде списка ребер.

Формат входных данных

Входной файл содержит число N ($1 \leqslant N \leqslant 100$) — число вершин в графе, и затем N строк по N чисел, каждое из которых равно 0 или 1 — его матрицу смежности.

Формат выходных данных

Выведите в выходной файл список ребер заданного графа. Ребра можно выводить в произвольном порядке.

Пример

m2e.in	m2e.out
3 0 1 1 1 0 1 1 1 0	1 2 2 3 1 3

Задача C. 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 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 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 D. TopSort. Топологическая сортировка

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

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входных данных

В первой строке входного файла даны два натуральных числа N и M ($1 \leqslant N \leqslant 100\,000, M \leqslant 100\,000$) — 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 выходных данных

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

Пример

topsort.in	topsort.out
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 E. Longpath. Длиннейший путь

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

Дан ориентированный граф без циклов. Требуется найти в нем длиннейший путь.

Формат входных данных

Первая строка входного файла содержит два натуральных 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 ($1 \leqslant b_i, e_i \leqslant n$).

Входной граф не содержит циклов и петель.

$n \leqslant 10\,000$, $m \leqslant 100\,000$.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — количество дуг в длиннейшем пути.

Пример

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

Задача F. Поиск цикла

Имя входного файла:	`cycle.in`
Имя выходного файла:	`cycle.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

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

Формат входных данных

В первой строке входного файла находятся два натуральных числа N и M ($1 \leqslant N \leqslant 100\,000$, $M \leqslant 100\,000$) — 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 выходных данных

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

Примеры

cycle.in	cycle.out
2 2 1 2 2 1	YES 1 2
2 2 1 2 1 2	NO

Задача G. Condense 2. Конденсация графа

Имя входного файла:	`condense2.in`
Имя выходного файла:	`condense2.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

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 ($n \leqslant 10\,000,\ m \leqslant 100\,000$). 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 ($1 \leqslant b_i,\ e_i \leqslant n$). 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

condense2.in	condense2.out
4 4 2 1 3 2 2 3 4 3	2

Задача H. 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 входного файла 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 ребер 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 выходного файла 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 I. Поиск пути на гриде

Имя входного файла:	`dfsongrid.in`
Имя выходного файла:	`dfsongrid.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Дано прямоугольное поле $W \times H$. Некоторые клетки проходимы, через некоторые ходить нельзя. Из клетки можно ходить в соседние по ребру (слева, справа, сверху, снизу).

Нужно из клетки (x_1, y_1) найти любой (не обязательно кратчайший, даже не обязательно простой) путь в клетку (x_2, y_2).

Формат входных данных

На первой строке W, H, x_1, y_1, x_2, y_2 ($1 \leqslant x_1, x_2 \leqslant W \leqslant 1000$, $1 \leqslant y_1, y_2 \leqslant H \leqslant 1000$). Далее H строк, в каждой из которых по W символов. Символ "." означает, что клетка проходима, а символ "*" означает, что по ней ходить нельзя.

Клетки (x_1, y_1) и (x_2, y_2) не совпадают и обе проходимы.

Формат выходных данных

Если пути не существует, выведите `NO`.

Иначе выведите `YES` и последовательность клеток (x_i, y_i), в которой первая совпадает с клеткой (x_1, y_1), а последняя с клеткой (x_2, y_2).

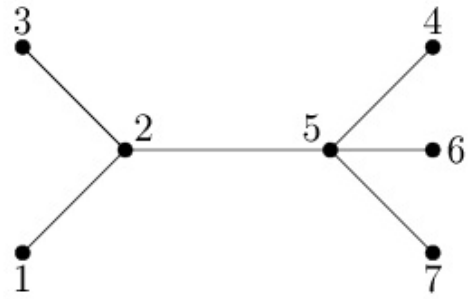
Пример

dfsongrid.in	dfsongrid.out
4 2 1 1 4 2	YES 1 1 2 1 3 1 4 1 3 1 3 2 4 2
4 2 1 1 4 2 ..*. .*..	NO
4 2 1 1 4 2 ..*. *...	YES 1 1 2 1 2 2 3 2 4 2

Задача J. Autotourism

Имя входного файла:	`autotourism.in`
Имя выходного файла:	`autotourism.out`
Ограничение по времени:	2 с
Ограничение по памяти:	256 Мб

В Байтландии существуют n городов, соединённых n - 1 дорогами с двусторонним движением таким образом, что из каждого города можно проехать в любой другой по сети дорог. Длина каждой дороги равна 1 километру.



Бензобак автомобиля позволяет проехать без заправки m километров. Требуется выбрать маршрут, позволяющий посетить наибольшее количество различных городов без дозаправки. При этом начинать и заканчивать маршрут можно в произвольных городах.

Формат входных данных

В первой строке входного файла заданы два целых числа n и m ($2 \leqslant n \leqslant 500\,000$, $1 \leqslant m \leqslant 200\,000\,000$) — количество городов в стране и количество километров, которое автомобиль может проехать без дозаправки. В последующих n - 1 строках описаны дороги. Каждая дорога задаётся двумя целыми числами a и b ($1 \leqslant a, b \leqslant n$) — номерами городов, которые она соединяет. Длина каждой дороги равна 1 км.

Формат выходных данных

Выведите одно число — максимальное количество городов, которое можно посетить без дозаправки.

Пример

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

Пояснение к примеру

5 городов можно посетить, например, по схеме $4 \to 5 \to 7 \to 5 \to 6 \to 5 \to 2$ или по схеме $3 \to 2 \to 1 \to 2 \to 5 \to 6 \to 5$.