

Содержание

Обязательные задачи	2
Задача А. От матрицы смежности к списку ребер [1 sec, 256 mb]	2
Задача В. Связность [1 sec, 256 mb]	3
Задача С. Поиск пути на гриде [1 sec, 256 mb]	4
Обычные задачи	5
Задача D. TopSort. Топологическая сортировка [1 sec, 256 mb]	5
Задача Е. Поиск цикла [1 sec, 256 mb]	6
Задача F. Сумма расстояний [1 sec, 256 mb]	7
Дополнительные задачи	8
Задача G. Сумма расстояний v2.0 [4 sec, 256 mb]	8
Задача H. Autotourism [2 sec, 256 mb]	9
Задача I. Здоровье Графа [1 sec, 256 mb]	10

Вы не умеете читать/выводить данные, открывать файлы? Воспользуйтесь [примерами](#).

В некоторых задачах большой ввод и вывод. Пользуйтесь [быстрым вводом-выводом](#).

Обратите внимание, что ввод-вывод во всех задачах стандартный.

Задачи расположены в **произвольном порядке**!

Обязательные задачи

Задача А. От матрицы смежности к списку ребер [1 сек, 256 mb]

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

Формат входных данных

Входной файл содержит число N ($1 \leq N \leq 100$) — 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

stdin	stdout
3	1 2
0 1 1	2 3
1 0 1	1 3
1 1 0	

Задача В. Связность [1 sec, 256 mb]

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 ($1 \leq N \leq 100$, $0 \leq M \leq 10\,000$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leq u_i, v_i \leq N$); 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

stdin	stdout
3 2 1 2 3 2	YES
3 1 1 3	NO

Задача С. Поиск пути на гриде [1 сек, 256 mb]

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 ($1 \leq x_1, x_2 \leq W \leq 1000, 1 \leq y_1, y_2 \leq H \leq 1000$). 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

stdin	stdout
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

Обычные задачи

Задача D. TopSort. Топологическая сортировка [1 sec, 256 mb]

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входных данных

В первой строке входного файла даны 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 выходных данных

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

Пример

stdin	stdout
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 Е. Поиск цикла [1 sec, 256 mb]

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 ($1 \leq N \leq 100\,000$, $M \leq 100\,000$) — 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 выходных данных

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

Примеры

stdin	stdout
2 2 1 2 2 1	YES 1 2
2 2 1 2 1 2	NO

Задача F. Сумма расстояний [1 sec, 256 mb]

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 ($1 \leq n \leq 1000$, $0 \leq m \leq 10\,000$).

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

Гарантируется, что граф связан.

Формат выходных данных

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

Пример

stdin	stdout
5 5 1 2 2 3 3 4 5 3 1 5	16

Дополнительные задачи

Задача G. Сумма расстояний v2.0 [4 сек, 256 mb]

Дан невзвешенный ориентированный граф. Определим $f(u, v)$ как длину кратчайшего пути между вершинами u и v (если пути не существует, скажем, что $f(u, v) = 0$).

Найдите $\sum_{u=1}^n \sum_{v=1}^n f(u, v)$.

Формат входных данных

В первой строке содержится число n — размер графа ($2 \leq n \leq 2000$).

В каждой из последующих n строк задано по n чисел. j -е число в i -й строке равняется 1, если существует ориентированное ребро из вершины i в вершину j .

Формат выходных данных

Выведите одно число — сумму попарных кратчайших путей.

Примеры

stdin	stdout
3 0 1 1 1 0 1 1 1 0	6
4 0 1 1 1 0 0 1 1 0 0 0 1 0 0 0 0	6
3 0 1 1 0 0 0 0 0 0	2

Замечание

В этой задаче помогает `std::bitset`.

Задача Н. Autotourism [2 sec, 256 mb]

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 ($2 \leq n \leq 500\,000$, $1 \leq m \leq 200\,000\,000$) — 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 ($1 \leq a, b \leq n$) — 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

stdin	stdout
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 $4 \rightarrow 5 \rightarrow 7 \rightarrow 5 \rightarrow 6 \rightarrow 5 \rightarrow 2$ или по схеме $3 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 5$.

Задача I. Здоровье Графа [1 sec, 256 mb]

Этично ли удалять рёбра у связанного графа?

Граф Безциклов решил проверить своё здоровье. Он хочет проверить, что все его рёбра достаточно крепко держатся в нём. Для этого он хочет посчитать *устойчивость* некоторых из них. *Устойчивостью* ребра называется количество простых путей, проходящих через это ребро.

Формат входных данных

Все числа в файле целые.

$0 \leq N \leq 10^5$, $0 \leq M \leq 10^5$ — количество вершин и рёбер.

Затем M пар чисел $1 \leq v_i, u_i \leq N$ — i -ое ребро соединяет вершины v_i и u_i .

$0 \leq Q \leq 10^5$ — количество запросов.

Затем Q чисел $1 \leq e_i \leq M$.

Граф неориентирован. Гарантируется, что Граф ацикличен.

Формат выходных данных

Для i -ого запроса вывести устойчивость e_i -ого ребра.

Примеры

stdin	stdout
2 1 1 2 1 1	1