Содержание

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

В некоторых задачах большой ввод и вывод. Пользуйтесь быстрым вводом-выводом.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу.

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`, они позволяют пользоваться дополнительной библиотекой. Под ними можно сдать вот это.

Must have

Задача 1A. Матрица инцидентности [0.1 sec, 256 mb]

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 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. Рёбра нумеруются в том порядке, в котором они даны во входном файле, начиная с единицы.

Формат выходных данных

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

стандартный ввод	стандартный вывод
3 2 1 2 2 3	1 0 1 1 0 1
2 2 1 1 1 2	1 1 0 1

Замечание

Разминаемся.

Пожалуйста, **не забывайте**, быстрый ввод-вывод. `#include "optimization.h"`

Задача 1B. Дерево [0.1 sec, 256 mb]

Дан неориентированный граф. Проверьте, является ли он деревом.

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и m — количество вершин и рёбер в графе, соответственно ($1 \leqslant n \leqslant 100$). В следующих m строках заданы рёбра; i-я из этих строк содержит два целых числа u_i и v_i через пробел — номера концов i-го ребра ($1 \leqslant u_i, v_i \leqslant n$). Граф не содержит петель и кратных рёбер.

Формат выходных данных

В первой строке выходного файла выведите "`YES`", если граф является деревом, и "`NO`" в противном случае.

Примеры

стандартный ввод	стандартный вывод
3 2 1 2 1 3	YES
3 3 1 2 2 3 3 1	NO

Замечание

`dfs` дарует власть и силу, используй его.

Задача 1C. Компоненты связности [0.2 sec, 256 mb]

Вам задан неориентированный граф с N вершинами и M ребрами ($1 \leqslant N \leqslant 20\,000$, $1 \leqslant M \leqslant 200\,000$). В графе отсутствуют петли и кратные ребра.

Определите компоненты связности заданного графа.

Формат входных данных

Граф задан во входном файле следующим образом: первая строка содержит числа N и M. Каждая из следующих M строк содержит описание ребра — два целых числа из диапазона от 1 до N — номера концов ребра.

Формат выходных данных

На первой строке выходного файла выведите число L — количество компонент связности заданного графа. На следующей строке выведите N чисел из диапазона от 1 до L — номера компонент связности, которым принадлежат соответствующие вершины. Компоненты связности следует занумеровать от 1 до L произвольным образом.

Пример

стандартный ввод	стандартный вывод
4 2 1 2 3 4	2 1 1 2 2

Замечание

Приличные люди хранят граф списками смежности.

Обязательные задачи

Задача 1D. Связанность графа [0.1 sec, 256 mb]

Дан граф, содержащий N вершин и M ребёр ($1 \leqslant N \leqslant 1000, 1 \leqslant M \leqslant 7000$). Требуется найти наименьшее число рёбер и эти рёбра, которые нужно добавить, чтобы граф стал связным.

Формат входных данных

Во входном файле записаны сначала числа N и M, затем идёт описание рёбер графа — M пар чисел, где каждая пара описывает начало и конец ребра.

Формат выходных данных

В первую строку вывести единственное число K — минимальное количество рёбер, которое нужно добавить. В следующих K строках выведите по 2 числа — начало и конец нового ребра.

стандартный ввод	стандартный вывод
3 1 2 1	1 1 3

Замечание

Идейная задача.

Задача 1E. TopSort. Топологическая сортировка [0.2 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 ($1 \leqslant N \leqslant 100\,000, M \leqslant 100\,000$) — 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

стандартный ввод	стандартный вывод
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

Замечание

Пора вспомнить чуть-чуть теории с прошлого модулю.

Задача 1F. Поиск пути на гриде [1.5 sec, 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 ($1 \leqslant x_1, x_2 \leqslant W \leqslant 1000$, $1 \leqslant y_1, y_2 \leqslant H \leqslant 1000$). 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

стандартный ввод	стандартный вывод
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

Замечание

К сожалению, графы бывают и такие тоже.

`dfs` умеет восстанавливать путь на обратном ходу рекурсии.

Задача 1G. Поиск цикла [0.3 sec, 256 mb]

Дан ориентированный невзвешенный граф без петель и кратных рёбер. 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

стандартный ввод	стандартный вывод
2 2 1 2 2 1	YES 1 2
3 3 1 2 2 3 1 3	NO

Замечание

Продолжаем вспоминать теорию.

Задача 1H. Unique Topsort [0.2 sec, 256 mb]

Дан ориентированный ацикличный граф G. Проверить, что существует единственный топологический порядок вершин графа.

Формат входных данных

Первая строка входных данных содержит число вершин графа n ($1 \leqslant n \leqslant 100\,000$) и число ребер графа m ($0 \leqslant m \leqslant 100\,000$). Следующие m строк содержат пары чисел от 1 до n, задающие начало и конец соответствующего ребра. Гарантируется, что граф не содержит циклов.

Формат выходных данных

Если топологический порядок единственный, выведите на первой строке `YES`, а на второй номера вершин в топологическом порядке, иначе выведите `NO`.

Примеры

стандартный ввод	стандартный вывод
1 0	YES 1
2 1 2 1	YES 2 1
4 2 1 2 4 3	NO

Замечание

Что-то такое было на практике.

Задача 1I. Condense 2. Конденсация графа [0.2 sec, 256 mb]

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

стандартный ввод	стандартный вывод
4 4 2 1 3 2 2 3 4 3	2

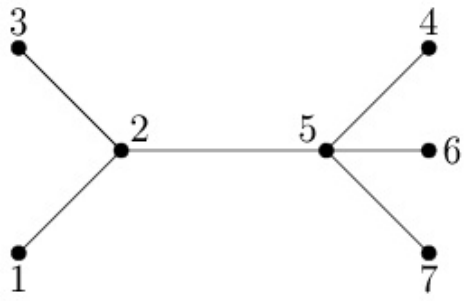
Замечание

Стандартный алгоритм. Пройдём в пятницу на лекции.

Дополнительные задачи

Задача 1J. Autotourism [1.5 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 ($2 \le n \le 500\,000$, $1 \le m \le 200\,000\,000$) — 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

стандартный ввод	стандартный вывод
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 $3 \to 2 \to 1 \to 2 \to 5 \to 6 \to 5$.

Задача 1K. Дорожные работы [3 sec, 256 mb]

В республике Икс издавна действует двухпартийная система. Каждый год граждане, имеющие избирательные права, голосуют, какой партии они больше доверяют — партии Мошенников или партии Грабителей, и в течение этого года вся реальная власть сосредоточена в руках избранной партии.

В последние M лет между партиями разразилась нешуточная война по перестройке дорожной сети республики «под себя». Партия Мошенников стремится построить как можно больше государственных дорог, чтобы прикарманить побольше бюджетных денег на их «обслуживание», а партия Грабителей стремится сделать платными как можно большее число дорог. Движение на всех дорогах республики Икс двустороннее.

Известно, что в течение одного года правления партии Мошенников удавалось построить ровно одну новую дорогу (которая поначалу является бесплатной), а партии Грабителей — ввести плату за проезд по одной из бесплатных на текущий момент дорог (при этом деньги на содержание этой дороги выделяются уже не из бюджета, а из средств, вырученных за проезд).

Президент республики, в настоящее время не имеющий реального политического влияния, решил привлечь внимание общественности к проблеме дорог. Он назвал дорожную сеть *удобной* (для простых граждан), если из любого города можно доехать до любого, используя только бесплатные дороги, но при этом количество бесплатных дорог (а, соответственно, и бюджетные средства на их содержание, полученные сбором налогов с граждан республики) — минимально возможное.

Вам поручено написать программу, которая определяет, была ли дорожная сеть удобной по завершении i-го года «дорожной войны».

Формат входных данных

В первой строке ввода заданы два числа — N ($1 \leqslant N \leqslant 1000$), число городов в республике Икс, и M ($1 \leqslant M \leqslant 100\,000$), продолжительность порядком затянувшейся «дорожной войны». Далее следуют M строк, первый символ каждой из которых — это `F`, если в данный год у власти была партия Мошенников, и `R` — если партия Грабителей, а далее в строке следуют два числа — номера городов u_i и v_i — пара городов, дорога между которыми стала объектом пристального внимания соответствующей партии (была построена новая дорога, если у власти была партия Мошенников, и одна из существующих дорог была сделана платной, если у власти была партия Грабителей). Вполне возможна ситуация, когда между двумя городами окажется более одной дороги, или будет построена дорога из города в себя — мало ли, что там удумает партия Мошенников.

Гарантируется, что входные данные корректны, то есть, все числа u_i и v_i лежат в пределах от 1 до N, и если известно, что в какой-то год дорога между двумя городами была сделана платной, то это значит, что перед началом года была хотя бы одна бесплатная дорога между этими городами.

Формат выходных данных

Для каждого года выведите в отдельной строке `YES`, если дорожная сеть по завершении соответствующего года была удобной, и `NO` в противном случае.

Пример

стандартный ввод	стандартный вывод
4 8 F 1 2 F 1 3 R 1 3 F 2 3 F 3 4 F 1 3 R 1 3 F 1 1	NO NO NO NO YES NO YES NO

Замечание

Простая задача. Очень. Не перемудрите.

Задача 1L. Редукция дерева [0.4 sec, 256 mb]

Задано неориентированное дерево, содержащее n вершин. Можно выбрать некоторое ребро и удалить его, при этом инцидентные ему вершины не удаляются. Таким образом можно удалить из дерева некоторый набор рёбер. В результате дерево распадается на некоторое количество меньших деревьев. Требуется, удалив наименьшее количество рёбер, получить в качестве хотя бы одной из компонент связности дерево, содержащее ровно p вершин.

Формат входных данных

Первая строка входного файла содержит пару натуральных чисел n и p ($1 \leqslant p \leqslant n \leqslant 1000$). Далее в $n-1$ строке содержатся описания рёбер дерева. Каждое описание состоит из пары натуральных чисел a_i, b_i ($1 \leqslant a_i, b_i \leqslant n$) — номеров соединяемых ребром вершин.

Формат выходных данных

В первую строку выведите наименьшее количество рёбер q в искомом наборе. Во вторую строку выведите номера удаляемых рёбер. Номера рёбер определяются порядком их задания по входном файле. Рёбра нумеруются с единицы. Если оптимальных решений несколько, разрешается выводить любое.

Пример

стандартный ввод	стандартный вывод
11 6 1 2 1 3 1 4 2 6 2 7 1 5 2 8 4 9 4 10 4 11	2 3 6