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.

Вам кажется, что задача **много**?
Не бойтесь, на самом деле их две: первая и «написать `dfs`».

Написав `dfs`, вы сможете сдать его в 6 простых задач.

Задачи здорового человека

Задача 4A. Любители Кошек [0.2 sec, 256 mb]

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 входного файла заданы числа n и m ($1 \leqslant n \leqslant 1\,000$, $1 \leqslant m \leqslant 30\,000$), 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, причем даже в разном порядке (как (x, y), так и (y, x)).

Формат выходных данных

В выходной файл необходимо вывести количество способов выбрать троих попарно знакомых друг с другом людей из клуба.

Пример

stdin	stdout
3 3 1 2 2 3 3 1	1

Замечание

Предполагается решение за $\mathcal{O}(nm)$.

Если каждый треугольник сразу считать ровно один раз, получится быстрее.

Задача разобрана на практике.

Задача 4B. 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

stdin	stdout
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 4C. 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

stdin	stdout
4 2 1 2 3 4	2 1 1 2 2

Замечание

Приличные люди хранят граф списками смежности.

Задача 4D. 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.

stdin	stdout
3 1 2 1	1 1 3

Подсказка по решению

Как связать граф, если бы он был пустым?

Причём тут компоненты связности?

Задача 4E. Глубинный путь [0.2 sec, 256 mb]

Дан ориентированный (по рёбрам можно ходить только в одну сторону) граф из n вершин и m рёбер. Найдите любой путь из вершины s в вершину t.

Формат входных данных

На первой строке числа n, m, s, t ($2 \leqslant n \leqslant 50\,000$, $0 \leqslant m \leqslant 100\,000$, $1 \leqslant s, t, n$, $s \neq t$). Следующие m строк содержат пары целых чисел a_i b_i, описывающие ребро из a_i в b_i. Граф не содержит петель, но может содержать кратные рёбра.

Формат выходных данных

Если пути нет, выведите -1. Иначе выведите вершины пути в порядке от s до t.

Если путей из s в t несколько, выведите любой.

Путь должен быть простым (вершины пути не повторяются).

Примеры

stdin	stdout
4 5 1 3 1 2 2 4 2 1 4 3 4 1	1 2 4 3
4 5 3 1 1 2 2 4 2 1 4 3 4 1	-1

Подсказка по решению

Просто dfs. Просто восстановить путь.

Учились это делать «на обратном ходу рекурсии».

Задача 4F. 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

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

Замечание

Всё ещё dfs. На лекции умеем им решать и такую задачу.

Для искателей острых ощущений

Задача 4G. 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

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

Замечание

Всё ещё dfs. На лекции научились делать им и такое.

Задача 4H. 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

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

Замечание

К сожалению, графы бывают и такие тоже.

`dfs` умеет восстанавливать путь на обратном ходу рекурсии.

Задача 4I. 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

stdin	stdout
4 4 2 1 3 2 2 3 4 3	2

Замечание

Обсудили на лекциях.