G. *Диаметр дерева*

Дано дерево, содержащее N вершин.

Диаметром дерева называется длина максимального пути (количество рёбер) между двумя его вершинами.

Вычислите диаметр данного дерева.

В первой строке входных данных содержатся число вершин дерева N ($1 \leqslant N \leqslant 10^4$). В следующих $N-1$ строках записано по 2 числа, обозначающие рёбра дерева (вершины нумеруются числами от 1 до N).

Программа должна вывести одно число — диаметр данного дерева.

Input	Output
5 1 2 1 3 3 4 3 5	3

Замечание к тесту: диаметр дерева соответствует такому пути: $2 \to 1 \to 3 \to 5$

H. *Проверка обхода в глубину*

Дан неориентированный граф и некоторая последовательность номеров его вершин. Определить — может ли эта последовательность вершин быть результатом работы программы, перечисляющей *все* вершины графа в порядке обхода в глубину?

В первой строке через пробел записаны два целых числа N и M ($1 \leqslant n \leqslant 10^5; 0 \leqslant m \leqslant 10^6$) — количество вершин и ребер графа. В следующих M строчках записано по два целых числа a и b ($1 \leqslant a, b \leqslant n; a \neq b$) — номера вершин, которые соединяет описываемое ребро.

В следующей строке записано число Q — количество запросов. Наконец, в каждой из следующих Q строк записана последовательность чисел a_i ($1 \leqslant a_i \leqslant N$).

Программа должна вывести Q строк: в i-й строке следует вывести `YES`, если последовательность a_i может быть получена при выполнении обхода в глубину на данном графе начиная с какой-то вершины в каком-то порядке обхода и `NO`, если не может быть получена ни при каком порядке обхода.

Input	Output
3 2 1 2 3 2 2 1 2 3 3 1 2	YES NO
4 2 1 2 3 4 1 1 3 2 4	NO

I. *Максимальное количество рёбер*

Недавно на кружке по программированию Петя узнал об обходе в глубину. Обход в глубину используется во многих алгоритмах на графах. Петя сразу же реализовал обход в глубину на `Python` и `C++`.

```
def dfs(G, visited, v):
    print(v + 1)
    visited[v] = 1
    for u in G[v]:
        if not visited[u]:
            dfs(G, visited, u)
            print(v + 1)

V, E = map(int, input().split())
G = [[] for k in range(N)]
for k in range(E):
    a, b = map(int, input().split())
    G[a - 1].append(b - 1)
    G[b - 1].append(a - 1)
for k in range(len(G)):
    G[k].sort()
visited = [False] * N
for v in range(V):
    if not visited[v]:
        dfs(G, visited, v)
```

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void DFS(const vector <vector <int>>& G, vector<bool>& visited, int v) {
    visited[v] = true;
    cout << v + 1;
    for (int u: G[v]) {
        if (!visited[u]) {
            DFS(G, visited, u);
            cout << v + 1;
        }
    }
}

int main(){
    int N, V;
    cin >> N >> V;
    vector <vector <int>> G(N);
    vector <bool> visited(N, false);
    for (int i = 0; i < V; i++) {
        int u, v;
        cin >> u >> v;
        G[u - 1].push_back(v - 1);
        G[v - 1].push_back(u - 1);
    }
    for (int k = 0; k < N; k++) {
        sort(sort(G[k].begin(), G[k].end()));
    }
    for (int v = 0; v < N; v++) {
        if (!visited[v]) {
            DFS(G, visited, v);
        }
    }
}
```

Петина программа хранит граф с использованием списков смежности (вершины графа пронумерованы от 1 до N, в каждом списке вершины упорядочены по возрастанию). В массиве `visited` помечается, в каких вершинах обход в глубину уже побывал.

Петя запустил программу для некоторого неориентированного графа G с N вершинами и сохранил её вывод. А вот сам граф потерялся. Теперь Пете интересно, какое максимальное количество рёбер могло быть в графе G. Помогите ему выяснить это.

Первая строка входного файла содержит два целых числа: N и M — количество вершин в графе и количество чисел в выведенной последовательности ($1 \leqslant N \leqslant 300, 1 \leqslant M \leqslant 2N - 1$). Следующие l строк по одному числу вывод Петиной программы. Гарантируется, что существует хотя бы один граф, запуск программы Пети на котором приводит к приведенному во входном файле выводу. В первой строке программа должна вывести одно целое число P — максимальное возможное количество рёбер в графе.

Следующие P строк должны содержать по два целых числа: номера вершин, соединённых ребрами. В графе не должно быть петель и кратных рёбер.

Input	Output
6 10 1 2 3 2 4 2 1 5 6 5	6 1 2 1 3 1 4 2 3 2 4 5 6

J. *Построение (топологическая сортировка)*

Группа солдат-новобранцев прибыла в армейскую часть.

Прапорщик пронумеровал новобранцев числами от 1 до N. После этого он велел им построиться по росту (начиная с самого высокого). При этом прапорщик уверил себя, что знает про некоторых солдат, кто из них кого выше, и это далеко не всегда соответствует истине.

После трёх дней обучения новобранцам удалось выяснить, что знает (а точнее, думает, что знает) прапорщик. Помогите им, используя эти знания, построиться так, чтобы товарищ прапорщик остался доволен.

Сначала на вход программы поступают числа N и M ($1 < N \leqslant 100, 1 \leqslant M \leqslant 5000$) — количество солдат в роте и количество пар солдат, про которых прапорщик знает, кто из них выше. Далее идут эти пары чисел A и B по одной на строке ($1 \leqslant A, B \leqslant N$), что означает, что, по мнению прапорщика, солдат A выше, чем B. Не гарантируется, что все пары чисел во входных данных различны.

В первой строке выведите `YES` (если можно построиться так, чтобы прапорщик остался доволен) или `NO` (если нет). После ответа `YES` на следующей строке выведите N чисел, разделенных пробелами, — одно из возможных построений.

Input	Output
4 5 1 2 2 3 3 4 1 4 4 1	NO

K. *Алфавит*

Страна X использует при письме маленькие буквы латинского алфавита. Однако упорядочивают их в другом порядке.

Вам дан набор слов. Необходимо восстановить порядок букв в алфавите таким образом, чтобы данные слова оказались упорядоченными по возрастанию в лексикографическом порядке. Если порядок существует, но его нельзя восстановить однозначно — нужно вернуть любой подходящий порядок букв. Если не существует порядка букв, удовлетворящего условию задачи — вывести -1.

В первой строке входных данных указано количество слов ($N < 1000$). Затем N строк с одним словом на каждой. Суммарная длина слов не превосходит 10000.

Программа должна вывести одну строку: упорядоченную последовательность букв без разделителей, либо -1.

В последовательности букв должны встретиться обязательно все буквы из набора слов и никаких других букв.

Input	Output
3 abc bc bb	acb