H. PANDA CHESS

PROBLEM DESCRIPTION

Given a list of inequality relations, find out the correct ranking.
You are also given a (potentially) incorrect ranking, figure out the minimum number of operations needed to correct it.

SOLUTION TECHINQUES

Topological Sort / Longest Common Subsequence / Longest Increasing Sequence.

SOLUTION SKETCHES

This is a tedious problem to deal with.

First, do a topological sort to determine the correct ranking of the players.

Then we need to do a LCS with the correct ranking and the given (incorrect) ranking.
Since all IC numbers are distinct, we can transform / mapping reduce it to a LIS problem and solve it in O($N \log N$) time.

THE ANSWER IS: (N(NUMBER OF PLAYERS) – LCS_LENGTH) * 2

TIME COMPLEXITY

O($M + N \log N$)

SOLUTION PROGRAM FOR REFERENCE

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <string>
#include <map>
#include <vector>

using namespace std;

const int N = 100002;

vector<int> g[N];
map<string, int> id; int it = 0;
string nm[N];
int l[N], t = 0;
char vst[N];
int cpos[N];
vector<int> proc;
vector<int> lis;

inline int get_id(string &s)
{
    auto iter = id.find(s);
    if (iter != id.end())
        return iter->second;
    else
    {
        nm[it] = s;
        id[s] = it++;
        return it - 1;
    }
}

void dfs(int vi)
{
    vst[vi] = 1;
    for (auto &vj: g[vi])
        if (!vst[vj])
            dfs(vj);
    l[++t] = vi;
}

int main()
{
    cin.tie(0);
    ios::sync_with_stdio(0);
    int i, n, m, d;
    string a, b; int ai, bi;
    cin >> n >> m >> d;
    for (i = 0; i < m; i++)
    {
        cin >> a >> b;
        ai = get_id(a);
        bi = get_id(b);
        g[ai].push_back(bi);
    }
    fill(vst, vst + n, 0);
```

```
    for (i = 0; i < n; i++)
        if (!vst[i])
            dfs(i);
    for (i = t; i >= 1; i--)
        cpos[l[i]] = t - i;
    for (i = 0; i < n; i++)
    {
        cin >> a;
        auto it = id.find(a);
        if (it == id.end()) continue;
        proc.push_back(cpos[it->second]);
    }
    for (auto &pi: proc)
    {
        if (lis.empty() || pi > lis.back())
        {
            lis.push_back(pi);
            continue;
        }
        *lower_bound(lis.begin(), lis.end(), pi) = pi;
    }
    printf("%d\n", (n - lis.size()) * 2);
    return 0;
}
```