IOITC 2016 TST Day 1

BFS Tree

Sumeet recently started teaching Lalit graph algorithms, and he started with BFS. He now wants to test him on it. He has given Lalit a graph $G = (V, E)$, which is undirected, unweighted and simple: no self-loops and no multiple edges. The nodes are numbered $1, 2, \ldots, n$, where n is the number of vertices. It is also connected: that is, from every vertex, there is a path to every other vertex. He also gives him a root $r \in V$. He has asked him to perform a BFS from r, and write down the BFS tree. But being a novice, Lalit might have made some mistakes and so, Sumeet has to check whether that tree is a valid BFS tree. It is guaranteed that Lalit writes down some spanning tree.

The algorithm that he taught Lalit is this:

Breadth-First-Search

```
1:  procedure BREADTH-FIRST-SEARCH(G, r)
2:      for each node v in G do
3:          v.visited = False
4:          v.parent = NULL
5:      create empty queue Q
6:      r.visited = True
7:      Q.enqueue(r)
8:      while Q is not empty do
9:          current = Q.dequeue()
10:         for each node v that is a neighbour of current do
11:             if v.visited == False then
12:                 v.visited = True
13:                 v.parent = current
14:                 Q.enqueue(v)
```

By BFS tree, we refer to the rooted tree which has r as the root and the parent of vertex v is v.parent.

Notice that the only place in this algorithm which is not fixed, is Line 10. The **for** loop which starts at Line 10 looks at every neighbour of a vertex v. But this order is not specified and so can be done in any order. In particular, if a vertex v has k neighbors, there are $k!$ ways in which this can be ordered. At each vertex, we can choose any of the orderings. Various orderings might result in different parents for various vertices, and you might get many different BFS trees, as a result. We call any rooted tree that you get by properly following this algorithm, a valid BFS tree. Your task is to help Sumeet in finding out if the rooted spanning tree which Lalit has written down is a valid BFS tree.

Input

The first line contains a single integer t, which denotes the number of test cases. Each test case is as follows:
The first line contains two integers, n and m, which denote the number of vertices in G, and the number of edges in G, respectively.
m lines follow, with two space separated integers each: u and v, which denote that the edge (u, v) is in G.
Then one line follows, with $n - 1$ integers: $p_2, p_3, \ldots, p_n$. This describes the rooted tree written down by Lalit. The root r is always the vertex 1. p_i is the parent of vertex i.

Output

Output one line per test case, which should be "YES" if the inputted rooted tree is a valid BFS tree from vertex 1, of the inputted graph G. It should be "NO" otherwise.

Test Data

It is guaranteed that the inputted G is a connected graph, and the inputted tree (Lalit's tree) is a spanning tree of G. So, all the edges in Lalit's tree will be edges from the graph G.

- **Subtask 1 (20 Points)**:
  - ⋆ The sum of number of vertices over all the test cases will be $\leq 10^6$.
  - ⋆ The sum of number of edges in G over all the test cases will be $\leq 2 * 10^6$.

  And you are guaranteed that the depth of Lalit's tree is atmost two. That is, every vertex can be reached by atmost two edges from vertex 1, using the edges from the inputted tree.

- **Subtask 2 (20 Points)**:
  - ⋆ $1 \leq n \leq 500$
  - ⋆ $1 \leq m \leq 500$

- **Subtask 2 (60 Points)**:
  - ⋆ The sum of number of vertices over all the test cases will be $\leq 10^6$.
  - ⋆ The sum of number of edges in G over all the test cases will be $\leq 2 * 10^6$.

Sample Input1

```
1
5 6
1 4
3 5
3 2
4 5
3 1
2 4
4 1 1 4
```

Sample Output1

```
YES
```

Sample Input2

```
1
5 6
1 4
3 5
3 2
4 5
3 1
2 4
4 1 1 3
```

Sample Output2

```
NO
```

Explanation

In both the Sample Inputs, the graph G is the same. For this graph, there are only two valid BFS trees with 1 as root, which are:

1. Tree with edges: (1,4),(1,3),(4,2),(4,5). This corresponds to Sample Input1.
2. Tree with edges: (1,4),(1,3),(3,2),(3,5)

Limits

Time: 3 seconds

Memory: 256 MB

Note:

Large I/O. Use *scanf/printf* instead of *cin/cout*.