IOITC 2022 TST 3

Tree interval queries

You are given a tree with N vertices numbered $1, 2, \ldots N$. First, you sort the adjacency lists of all the vertices and run depth first search from node 1. This gives you an array of dfs start times. Note that the start times are uniquely determined, as the adjacency list of each vertex is sorted. The following code represents the dfs:

```
// let st denote the start times
// let adj[x] be the adjacency list (the list of neighbors) of x
int timer = 1;
void dfs(int s, int p){
    st[s] = timer;
    timer++;
    // iterate over the neighbors in increasing order
    sort(adj[s].begin(), adj[s].end());
    for(int v: adj[s]) {
        if(v != p){
            dfs(v, s);
        }
    }
}
```

The timer is initialized with 1 in the beginning, and you call `dfs(1, 0)` to compute all the start times. Let st_v denote the start time of v.

Now, you have to process Q queries.

Each query contains an integer k and a list of disjoint intervals $[L_1, R_1], [L_2, R_2], \ldots [L_k, R_k]$, where $1 \leq L_i \leq R_i \leq N$ for all valid i and $R_i < L_{i+1}$ for all valid i. Let S be the set of vertices whose **start time** lies in one of the intervals $[L_i, R_i]$. Then, the answer for a query is the number of connected components in the graph, whose set of vertices is S, and the edges are the edges of the tree with both endpoints in S.

Input

- The first line contains N and Q, the number of vertices in the tree, and the number of queries.
- Each of the next $N - 1$ lines contains two integers, a and b denoting the endpoint of an edge.
- The following lines contain the queries. Each query contains 2 lines:
  - The first line of each query contains k, the number of intervals
  - The second line contains $2k$ space separated integers, $L_1, R_1, L_2, R_2, \ldots, L_k, R_k$.

Output

For each query, print one line containing the number of connected components in the graph described in the problem statement.

Test Data

In all inputs,

- $1 \leq N, Q \leq 5 \times 10^5$
- $1 \leq k \leq 5 \times 10^5$
- $1 \leq L_i \leq R_i \leq N$ for all valid i
- $R_i < L_{i+1}$ for all valid i.
- The sum of k over all queries doesn't exceed 5×10^5

Subtask 1 (8 Points):

- $1 \leq N \leq 2000$
- The sum of k over all queries doesn't exceed 2000

Subtask 2 (17 Points):

- $k = 1$

Subtask 3 (17 Points):

- $1 \leq N \leq 10^5$
- The sum of k over all queries doesn't exceed 10^5

Subtask 4 (18 Points):

- $1 \leq N \leq 2.5 \times 10^5$
- The sum of k over all queries doesn't exceed 2.5×10^5
- The diameter of the tree doesn't exceed 20

Subtask 5 (10 Points):

- $1 \leq N \leq 2.5 \times 10^5$
- The sum of k over all queries doesn't exceed 2.5×10^5

Subtask 6 (28 Points): No additional constraints

Sample Input

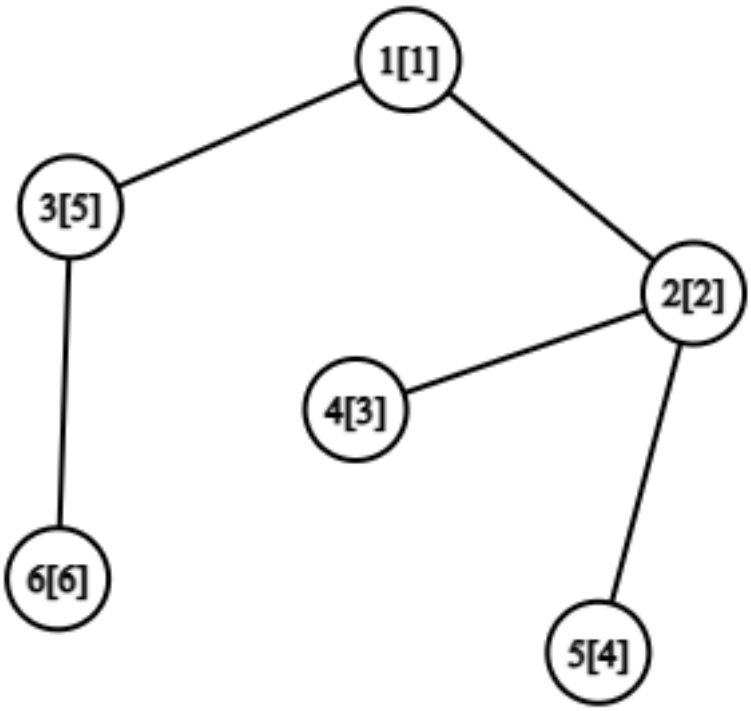
```
6 2
1 3
1 2
2 4
2 5
3 6
2
1 3
5 6
1
4 5
```

Sample Output

```
1
2
```

Explanation

The tree is the following (with start times written inside brackets):



In the first query, the valid start times are $\{1, 2, 3, 5, 6\}$ and hence the set S is $\{1, 2, 3, 4, 6\}$, and this set forms a single connected component.

In the second query, the valid start times are $\{4, 5\}$ and hence the set S is $\{3, 5\}$, and this set forms two connected components.

Limits

Time: 2 seconds

Memory: 256 MB