A. Diffusion Model

time limit per test: 2 s.
memory limit per test: 256 MB

In unsupervised image generation with latent diffusion models, a noise latent representation sampled from a standard Gaussian distribution is gradually denoised to ultimately arrive at a latent representation point on a low-dimensional data manifold within the variational autoencoder's latent space, which is then mapped back to an image that follows the training data distribution by the corresponding decoder. This process is akin to particle diffusion modeled by a certain form of differential equations.

To simplify the problem, we consider that all possible diffusion paths of a sampled noise latent representation form a diffusion tree with that noise latent representation as the root, with N being the number of nodes on the tree. Under an unsupervised setting, each denoising step allows the latent representation to randomly move from its current node to one of its child nodes. After several denoising steps, the latent representation will eventually reach a leaf node, yielding the final image.

In this context, we will mark K different leaf nodes to represent the final nodes we want the latent representation to reach. However, the randomness inherent in the denoising process of diffusion models often leads to an inability to reach these nodes, meaning it may not generate the desired image.

To address this issue, the text-to-image latent diffusion model introduces control over the denoising process through the semantics of natural language text, building upon the original latent diffusion model. This can be simplified to the selection of M candidate words, which we can choose to use. When the i-th candidate word is selected, it acts on the node u_i of the diffusion tree, ensuring that when the latent representation is at u_i, it deterministically denoises and moves to the child node v_i. For nodes not influenced by candidate words, the latent representation will still randomly move to a child node.

For efficiency, we aim to use as few candidate words as possible, ensuring that the latent representation can reach a marked leaf node starting from the root.

Considering the practical nature of diffusion models, we ensure that all leaf nodes in the diffusion tree have consistent depth.

Specifically, when selecting candidate words, if two candidate words act on the same node and both are chosen, the earlier candidate in the input will override the effect of the later one.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

- The first line contains three integers N, M, K $(2 \le N \le 5 \times 10^5,\ 0 \le M,\ K \le N)$, indicating the number of nodes of the diffusion tree, number of candidate words, and number of leaf nodes marked.
- For the following N lines, in the i-th $(1 \le i \le N)$ line, an integer k_i will be given, following k_i different integers c_{ij} $(i < c_{ij} \le N)$, indicating the children nodes of diffusion tree node i. It is guaranteed that the graph given is a tree and rooted at node 1.
- For the following M lines, the i-th $(1 \le i \le M)$ line contains two integers u_i, v_i, indicating that the i-th candidate word takes effect on the node u_i and will deterministically denoise to node v_i. It is guaranteed that v_i is the child of u_i.
- The last line of input contains K different integers, $t_1, t_2, \ldots, t_K$, indicating the marked leaf

nodes.

It is guaranteed that the sum of N over all test cases doesn't exceed 5×10^5.

Output

Output T lines.

For i-th test case, output one line containing an integer $0 \leq L_i \leq M$, indicating the minimum number of candidate words you need to select in order to satisfy the requirement. In particular, if no solution exists, $L_i = -1$.

Examples

input

```
2
7 3 2
2 2 3
2 4 7
2 5 6
0
0
0
0
1 3
2 4
3 5
4 5
7 2 1
2 2 3
2 4 5
2 6 7
0
0
0
0
1 2
3 6
4
```

output

```
2
-1
```


- DP Knapsack - Preclass
- DP Edit Distance - Preclass
- Segment Tree - Homework
- DP Palindromes - Preclass
- Lazy Segment Tree - Homework
- LCA and Binary Lifting - Homework
- DP intro - Preclass
- Square Root Decomposition - Homework
- DP Longest Increasing Subsequence - Preclass
- Greedy - Preclass
- Fenwick Tree - Preclass
- Bit Manipulation - Homework
- Square Root Decomposition - Preclass
- Fast Exponentiation - Homework
- MST - Homework
- Lazy Segment Tree - Preclass
- LCA and Binary Lifting - Preclass
- Segment Tree - Preclass
- Bit Manipulation - Preclass
- Fast Exponentiation - Preclass
- MST - Preclass
- Graph Traversal 2 - Homework
- Graph Traversal 2 - In Class
- All Pairs Shortest Path - Homework
- All Pairs Shortest Path - In Class
- Single Source Shortest Path - Homework
- Single Source Shortest Path - In Class
- Graph Traversal 1 - Homework
- Graph Traversal 1 - In Class
- Binary Search Tree - Homework
- Binary Search Tree - In Class
- Disjoint Sets - Homework
- Disjoint Sets - In Class
- Divide and Conquer - Homework
- Divide and Conquer - In Class
- Complete Search - Homework
- Complete Search - In Class
- STL - Homework
- STL - In Class
- IO Problems - Preclass
- Test Contest