Code Jam 2022 - Qualification Round

Analysis: Chain Reactions

Test Set 1

In Test Set 1 there are very few modules. This means that we can try all possible orders for the manual initiators, simulate the rules as explained in the statement to get the overall fun yielded by each order, and keep the maximum result from among those. Notice that modules that are pointed at the abyss and not pointed at by any other module contribute their own fun to the total no matter the order. Therefore, we can assume they all trigger in any specific order at the beginning. This brings down the number of orders to try from $\mathbf{N}!$ to at most $(\mathbf{N}/2)!$, which is a lot less.

Test Set 2

We can start by modeling the problem. We can see the input as a rooted forest where the parenting relationship models the pointed at relationship. Root nodes are modules that are pointed at the abyss.

As it is often the case for problems on trees, we can solve this one efficiently with a divide and conquer approach, and the aid of memoization/dynamic programming to keep the running time small.

Notice we can solve each tree (connected component) in the forest independently. Then, we can notice that the root of the tree is triggered by the first manual initiator. So, we can try all possibilities for the first manual initiator and eliminate the path between them and the root. That leaves a lot of separated subtrees that we can solve recursively.

Formally, let $F(i)$ be the fun of node i and $Ftree(t)$ be the fun value of a given subtree rooted at t. To compute $Ftree(t)$, if t is a single node we simply return its fun $F(t)$. Otherwise, we take the maximum (over all possible x) of $\text{fun}(x,t) + \sum_s Ftree(s)$ where $\text{fun}(x,t)$ is the maximum fun of all nodes in the path from x to t's root and x is a leaf of this subtree and the summation is over all subtrees s whose root parent in the original tree is in the mentioned path.

The domain of F is equal to the number of subtrees, which is equal to the number of nodes of the tree. If we memoize F, the overall running time of computing it for all subtrees of a tree of size k is the domain size (k) times the time it takes to compute it for a single item in the domain, disregarding the cost of recursive calls. This is $O(k)$ because of the summation, which means an overall time complexity of $O(k^2)$. The worst case is when the entire input forest is a single tree, which means the time complexity of the algorithm overall is $O(\mathbf{N}^2)$.

Test Set 3

To solve Test Set 3 we continue on our modelling from Test Set 2. We observe that the answer for $Ftree(t)$ can be broken down to $\max(\text{fun}(x,a), F(t)) + \sum_s Ftree(s)$ where a is one of t's children and x is a leaf node. We also observe that to maximize this, we need to choose a leaf x which minimizes $\text{fun}(x,a)$. This way we guarantee that $\text{fun}(x,t)$ benefits from having $F(t)$ on the path, and that all other subtrees s have a maximum possible value. Otherwise if we pick a path other than the minimum, this means the minimum path will be considered as one of the other subtrees which reduces $\sum_s Ftree(s)$ reducing the final answer we can get.

Identifiying x can be done using a DFS traversal solving each tree of size k in $O(k)$. Leading to an overall complexity of $O(N)$.