EGOI 2025 Editorial - Laserstrike

Task Author: Luca Versari

July 18, 2025

1 Introduction

In this problem, we have two players, the first of whom knows the structure of a tree. The first player needs to choose an order in which to delete the nodes from the tree (always deleting a leaf), then the second player receives, one by one in the same order, the one edge to which those nodes belong.

The second player needs to figure out which node on each edge is the leaf and can use additional bits of information sent by the first player.

2 $N - 1$ bits

Divide all edges $\{a, b\}$ (where $a < b$) into "minor" and "major" edges.

- A "minor edge" is an edge such that, when removing the edge, a is a leaf (and b is not).
- A "major edge" is an edge such that, when removing the edge, b is a leaf (and a is not).

We can then use a single bit per edge, with the bit being 0 if the edge is a minor edge and 1 if it is a major edge.

3 $\approx 0.5N$ bits

We can save half the bits by using the *order of two edges* to encode the type of the next edge.

More precisely, for every edge in an even non-zero position $2i$, we look at the two previous edges that were received. If edge $2i - 2$ compares greater[1] than edge $2i - 1$, the second player treats edge $2i$ as a major edge and as a minor edge otherwise.

The first player might need to swap two edges for this to work. As every tree has always at least two leaves, we can order the edges such that it is always possible to do so, and then decide whether to swap edges in backwards order.

4 1 bit, star

If the tree is a star, we can solve the problem with a single bit. The first player sends a bit that encodes which of the two nodes of the first edge is the center of the star; then, the second player can easily know that the non-center node in every edge is the leaf in that edge.

[1] according to any order, i.e., lexicographic order of the unordered pair of nodes

5 1 bit, path

On a path, we can encode the edges one at a time, starting from one end of the path. For the first edge, we use one bit to encode whether it is a major or a minor edge as usual. For all other edges, we do not need any information: the second player already knows that the node that belongs to the previous edge is the leaf.

6 $2\log_2 N + 1$ bits, star with lines

Let us handle separately the case of paths of length 1 and of length greater than 1.

We first use $\log_2 N$ bits to indicate how many paths have length greater than 1; we then use $\log_2 N$ to indicate how many of those paths have a minor edge as their last edge (going from the star center).

We can then proceed by removing all but the first edge for all of the paths of length greater than 1, applying the path solution. We can tell when we are switching to a new path by the fact that, at that point, the received edge will not share any nodes with the previous edge.

After this step, we are left with a star, and we can use the remaining bit to remove all the remaining edges following the star solution.

7 7 bits, star with lines

Similarly to the previous solution, we can divide edges connected to leaves into three sets:

- edges that are the last edges of paths of length 1
- minor edges belonging to long paths
- major edges belonging to long paths

First, we use one bit per set to identify whether the set is empty.

Within each set, we can sort edges in increasing order. Then we can proceed by first removing the set of edges that contains the largest edge, communicating 2 bits to identify which of the three it is.

- If this set is one of the long-path sets, we also remove most of the path as before; this causes additional edges to be inserted in-between the sorted edges, but this case can be recognized by the fact that the additional path edges share a node with the previous edge, and the leaf edges do not.
- Otherwise, if this set is the length-1 set, we communicate 1 bit to proceed with the star solution for this subset of edges.

Then, we proceed to do the same with the second set of edges, using 1 bit to identify the set. We can notice that we are switching to a new set by the fact that the first new edge will compare *smaller* to the previous leaf-edge, unlike for edges in the same set.

Finally, we remove the remaining edges connected to the star center. Note that at this point we either know the star center (because there was at least one length-1 path), or we didn't use an additional bit during the previous phases and can use an additional bit now to communicate the star center.

8 $\approx 10\log_2 N$ bits, longest distance 10

In this case, we can proceed by iteratively removing all the current leaves; the fact that the longest distance in the tree is 10 guarantees that we will only remove leaves 5 times.

At every iteration, we use $\log_2 N$ bits to represent the number of leaves connected to minor edges, followed by $\log_2 N$ bits to represent the number of leaves connected to major edges.

9 $5(\log_2 N + 1)$ bits, longest distance 10

We can use the idea of sorting two sets of leaves from the solution to star with lines case.

In particular, at every iteration we transmit the total number of leaves at that iteration with $\log_2 N$ bits, then divide the leaves on that layer into minor and major leaves. We then use one bit to indicate whether the first set is the set of major leaves or not. Note that, as we know the sum of the sizes of the sets, we don't need to handle one of the sets being empty.

10 2 bits

Let us divide the nodes of the tree in levels. Level 0 will be the *center* of the tree (one node if the longest path is odd, two nodes if it is even). Each other node will then have a level equal to its distance from one of the level 0 nodes.

Pick a longest path in the tree. This path will have exactly two nodes on each level; we call one half of the path the *starting* path and the second half the *marker* path.

We will remove leaves level by level, starting from the deepest level. On each level, first we will remove all the leaves connected to the *starting* path. This will let us understand when we are switching levels (since on each level there is at least a leaf not connected to the starting path).

We will also remove all the leaves connected to the *marker* path together, either immediately after the starting path or at the end of the level.

For all the other leaves on this level, we proceed as in each iteration of the short-distance case, by splitting leaves in major and minor and encoding which set is the first according to the *position of the marker path leaves in the layer*.

We are left with the task of identifying the starting and marker paths, as well as removing leaves from the deepest level. The first edge we send will be used to identify the starting path, while the second edge (ignoring other edges connected to the starting path) will identify the marker path. We use two bits to encode the ordering on these two paths.

Finally, to remove the remaining leaves on the last level, we proceed as in the other levels, using the ordering of the two starting and marker edges to encode whether the first set of edges are minor or major edges.

11 1 bit

The second player applies the following rules:

1. **First edge**: Use the extra communication bit to determine which node to cut.
2. **If the edge intersects the last removed edge**: remove the node that is *not* in common between the two edges (like in the solution for the star case).
3. **If one of the nodes in the edge has been seen before**: remove the most recently seen node.
4. **Otherwise**: compare to the last edge that was handled by case 1 or 4. If this edge is greater than the previous, then it has the same type (i.e. minor or major). Otherwise, it has the opposite type.

The first player will proceed by iteratively finding all the leaves in the current tree and removing them. The order in which the leaves are removed is as follows:

- If on the current layer there are more leaves that connect to the same node as the leaf that was just removed, remove one of those. Rule 2 ensures that the second player will remove the leaf correctly.

- If we are on the first layer, select a maximum set of leaves that do not share parents, and split them into two sets according to their type. Then, use the one available bit to communicate the type of the first edge, and proceed as usual. Rules 1 and 4 for the second player ensure that they will deduce the correct kind for the edge.
- Otherwise, if this is the first leaf of the current layer, pick any leaf that is **not** connected to the last removed leaf in the previous layer. This is always possible, as every layer contains at least two leaves. This ensures that Rule 3 applies and not Rule 2, and the correct node is removed by the second player.
- Otherwise, remove any leaf. Since the leaf was seen before (as it's not the first layer), the edge doesn't share nodes with the previous edge, and the parent of the leaf was not seen before, this ensures that the correct node is removed by Rule 3.