2 Navigation

2.1 Solution

In the task analysis, we refer to the network as a graph (which is a tree), each node being a vertex, and each cable being an edge. We call the power stations special nodes and others plain nodes, and the cable types as colors of the edges.

From now on, we only consider deterministic strategy, which means the robot should be able to correctly determine the answer for every plain nodes and for any permutation.

For each node u, we refer to the subtree of each edge as the connected component not containing u when the edge is removed.

2.2 Basic Observations

From the problem, there are several properties and simple observations we can make:

- The number of special nodes with distance decreased after following an edge, is equivalent to the number of special nodes in that subtree.
- If two edges adjacent to a node have the same color, the answer of the edge must be the same, because we cannot distinguish the edges.
- We **don't** need to find the answer for special nodes. This property is actually not a derived one, but stated in the statement twice (three times if you count the one in the examples), but it remains relevant for developing many of the ideas.

2.3 Subtask 1: $N = 7$

With the observation above, it should be very clear that this subtask is a "sanity check" subtask.

When $K = 6$ and $N = 7$, it is certain that the only possible query is the remaining node. Thus, it is possible to directly encode the answer; the naive method uses type $0, 1, ..., 6$, giving $S = 7$.

Since at most one number is greater than 3, we can decrease the largest number to 3. For any query such that the color indices does not sum to $K = 6$, we increase the largest entry until it satisfies the condition, and report that as the answer. This method gives $S = 4$.

2.4 Subtask 2: the network forms a chain

No queries is made at the endpoints and the special nodes, so we can partition the chain into disjoint paths by the special nodes. For each path, the answer multiset is either $\{0, 6\}, \{1, 5\}, \{2, 4\}$, or $\{3, 3\}$.

When the answer is $\{3, 3\}$, we can always assign *any* same color to both edges. This is because all other paths should never have two same color edges incident to a node, as that makes the two edges indistinguishable.

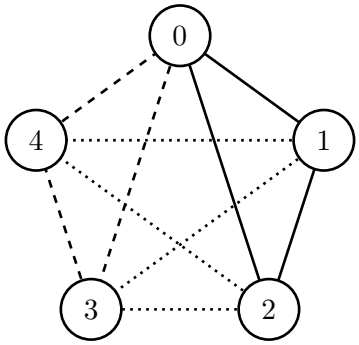
For the remaining paths, for any node that is adjacent to edges with color c_1 and c_2, respectively, we add a directed edge from $c_1 \to c_2$ if the answer to edge with color c_1 is bigger, and $c_1 \leftarrow c_2$ otherwise.

The resulting graph must have these properties:

- No self loop exists.
- For any pair of distinct vertices u, v, at most one of $u \to v$ and $v \to u$ exists.

Suppose that path is colored by $c_1, c_2, ..., c_k$, because k can be made almost arbitrarily large, it follows that the edges added must form a cycle. Combining the observation above, in order to answer $\{0, 6\}, \{1, 5\}, \text{and } \{2, 4\}$, it remains to solve the following problem: what is the minimum S such that the complete graph K_S can be partitioned into 3 edge-disjoint simple cycles?

The answer is $S = 5$, as follows:



Surprisingly, this construction is unique up to permuting the colors, and it is rather straightforward when the required observation are known.

2.5 Partial Approaches to Subtask 3, 4, and 5

Subtask 3, 4, and 5 only differs by the constraint on the degree of the internal nodes. From now on, we will instead describe different coloring schemes and their related observations or idea.

These solutions are all partial solutions, but their various observations will hint and even give some intuition of how an optimal encoding might look like.

2.5.1 $S = 64$ for subtask 3 and 4

If the coloring is separate for two directions of each edge, the task becomes very easy since we basically can encode the answer. However, the main challenge is we don't know the direction. Thus, one natural way to solve it is by first rooting the tree, encode the rooted information, then encode something for the node to find its parent.

When the minimum degree is at least 3, it is possible to encode the parent information with two colors. Specifically, we color each edge by their distance to the root modulo 2. Thus, the parent edge will always be different from other edges, and it is unambiguous when each internal node has degree of at least 3.

Based on this, we can recover the answer with $S = 2^6 = 64$: we simply root at every special node and use a bit to encode the parent information. To recover, we find each parent and add 1 to the edge for each of the 6 bits.

2.5.2 $S = 12$ for subtask 3 and 4

For subtask 3 and 4, we can extend the rooting idea to root for just one special node and encode the remaining 5 special nodes separately. Since the root information gives us orientation, we can encode the number of special nodes in each subtree on the edge. Thus, when recovering each node, we first identify the parent edge, and invert the number of nodes in the parent edge by $x \mapsto 6 - x$. This solution works because a special node is never queried, so we won't have the off-by-one issue.

This approach may yield $S = 14$ if the participant did not realize we only need 0 to 5 instead of 0 to 6.

2.5.3 $S = 18$ for the general case

Extending the idea above, the issue for the previous approach is because we cannot identify which is the parent edge for internal node with degree 2. However, inspired from subtask 2, it is possible to use 3 colors to find the parent edge.

We root at one special node similarly, and instead of modulo 2, we color each edge by their distance to the root modulo 3. When recovering, if we know the missing color is c, then the parent edge is the one with color $+1 (\mathrm{mod}\, 3)$.

Similarly, this approach may yield $S = 21$ if the participant did not realize we only need 0 to 5 instead of 0 to 6.

2.5.4 $S = 8$ for subtask 3 and 4

We root at a special node in the previous approach, but we actually did not benefit from any of those since identify the root is still possible.

Thus, we can instead root at a *weighted centroid* of the tree. That is, we pick some node such that every subtree has at most 3 special nodes in each subtree. Finding one such node is similar to finding a ordinary centroid of the tree, and the proof of such node always existing is similar to the original one.

Combining with the previous approach, we achieves $S = 8$ because the number of special nodes in each subtree can only have $0, 1, 2$, and 3.

2.5.5 $S = 7$ for subtask 3 and 4

It follows that we don't need to identify the parent if the answer is 3, because $3 = 6 - 3$. Thus, we don't perform the two-coloring for this case, saving one color.

2.5.6 $S = 10$ for the general case

The previous solution naturally generalizes to the general case by replacing the two-coloring with three-coloring. Similarly, naively this approach takes $S = 12$, but the case for answer equal to 3 is similar, saving two colors.

2.6 Full Solution to Subtask 3 and 4

After these partial attempts, we have some good tricks to perform and we should have a sense of why are we using so many colors. In this subsection, we will introduce two approaches.

2.6.1 Approach 1

The first approach is by trying to relax the rooting condition. For each edge, we encode the smaller number of special nodes in each subtree. When recovering, if the neighboring edge sums to 6, then we know that each edge is at the correct orientation; however, if it is not, the answer multiset must be one of $\{4, 1, 0, ...\}$, $\{5, 2, 0, ...\}$, and $\{6, 0, ...\}$. In addition, there is one more related case: $\{4, 1, 1, 0, ...\}$. We will refer to the cases by 4-2, 4-1-1, 5-1, and 6-0.

These cases do form a really good structure because of the tree property: the 4-2 and the 5-1 cases forms a chain, connected by 4-1-1, and all other branches are the case 6-0. Naturally, we would want to apply the solution for subtask 2.

But how can we try to obtain one such encoding? There are two major constraints that give us a really good idea:

1. These special encoding should never produce a case where the edge colors sum to 6.
2. Because finding two 3-cycles in a graph requires 5 nodes, with $S = 4$, we must use the two extra degree of each internal node to also represent some information.

It is natural to write a bruteforce program to find a set that satisfies the constraints, but because the restrictions are tight, it is actually feasible to found a solution by hand.

The following section describes one such scheme. We use k-subtree to refer to all edges whose subtree has answer equal to k.

- Case 4-2: Always color the 0-subtree with color 3.
  - If the parent edge has color 2, color the 2-subtree with 1.
  - If the parent edge has color 1, color the 2-subtree with 0.
  - If the parent edge has color 0, color the 2-subtree with 2.
- Case 4-1-1: Always color the 0-subtree with color 3.
  - If the parent edge has color 2, color the 1-subtrees with 0.
  - If the parent edge has color 1, color the 1-subtrees with 0.
  - If the parent edge has color 0, color the 1-subtrees with 1.
- Case 5-2: Always color the 0-subtree with color 2.
  - If the parent edge has color 1, color the 1-subtree with 0.
  - If the parent edge has color 0, color the 1-subtree with 3.
  - If the parent edge has color 3, color the 1-subtree with 1.
- Case 6-0:
  - If the parent edge has color 3, color the remaining edges with color 2.
  - Otherwise, color the remaining edges with color 3.

One extra note of handling these cases: since it is possible to encounter special nodes in these constructions, one way to avoid the cases is to treat the special node belonging to the subtree of the parent edge. By giving default values for each case (for example, deliberately write `else` instead of `else if` for the final case as default fallback), this case would be easily circumvented.

For partial scores, for example $S = 5$ or 6, it is possible to use some extra color to mark these cases, and thus it is very constructible once the relevant observations have been made.

2.6.2 Approach 2

The second approach is very different from the previous approaches. The idea is to use a similar idea to the orienting idea. Since there are at most 4 different answers (0, 1, 2, 3), we try to use $S = 5$ and use the missing color to find the correct cyclic shift.

The encoding scheme is as follows. For each possible integer partition, we assign a canonical encoding, which will be a prefix of the colors. Thus, we can always recover the canonical encoding with the missing colors.

- If the answer is $\{6, 0, ..., 0\}$, the canonical encoding is $\{1, 0, ..., 0\}$.
- If the answer is $\{5, 1, 0, ..., 0\}$, the canonical encoding is $\{2, 1, 0, ..., 0\}$.
- If the answer is $\{4, 2, 0, ..., 0\}$, the canonical encoding is $\{2, 0, 1, ..., 1\}$.

- If the answer is $\{4, 1, 1, 0, ..., 0\}$, the canonical encoding is $\{2, 1, 1, 0, ..., 0\}$.
- If the answer is $\{3, 3, 0, ..., 0\}$, the canonical encoding is $\{1, 1, 0, ..., 0\}$.
- If the answer is $\{3, 2, 1, 0, ..., 0\}$, the canonical encoding is $\{3, 2, 1, ..., 0\}$.
- If the answer is $\{3, 1, 1, 1, 0, ..., 0\}$, the canonical encoding is $\{2, 1, 1, 1, 0, ..., 0\}$.
- If the answer is $\{2, 2, 2, 0, ..., 0\}$, the canonical encoding is $\{1, 1, 1, 0, ..., 0\}$.
- If the answer is $\{2, 2, 1, 1, 0, ..., 0\}$, the canonical encoding is $\{2, 2, 1, 1, 0, ..., 0\}$.
- If the answer is $\{2, 1, 1, 1, 1, 0, ..., 0\}$, the canonical encoding is $\{2, 1, 1, 1, 1, 0, ..., 0\}$.
- If the answer is $\{1, 1, 1, 1, 1, 1, 0, ..., 0\}$, the canonical encoding is $\{1, 1, 1, 1, 1, 1, 0, ..., 0\}$.

Note that the second case and the third case requires 4 edges to distinguish, which fits within this constraint.

This gives uses $S = 5$. To reduce one color, observe that the only case requiring all 4 colors and 1 extra is this specific case of canonical encoding of $\{3, 2, 1, 0, ..., 0\}$ being $\{3, 2, 1, ..., 0\}$. If only one such node exists, we can take it as the root. Otherwise, there must be at most two. Since the path connecting them will all have the $\{3, 3, 0, ..., 0\}$ case, it is always possible to encode them with the exact canonical encoding.

2.7 Full Solution to Subtask 5

The complete solution is very similar to that of the Approach 1 of the previous subtask. Because of subtask 2, it is clear that the scheme should exactly follow the unique cycle partition of K_5.

If the participant realizes this for both solutions, writing a bruteforce program is very beneficial and should run sufficiently fast due to the known constraints.

Nevertheless, we show one possible scheme. This scheme is constructed by hand and thus, practically, participants are not required to write a solver for this.

- Case 4-2:
  - If the parent edge has color 2, color the 2-subtree with 1 and the 0-subtrees with color 0.
  - If the parent edge has color 1, color the 2-subtree with 0 and the 0-subtrees with color 4.
  - If the parent edge has color 0, color the 2-subtree with 2 and the 0-subtrees with color 4.
- Case 4-1-1: Always color the 0-subtree with color 4.
  - If the parent edge has color 2, color the 1-subtrees with 1.
  - If the parent edge has color 1, color the 1-subtrees with 3.
  - If the parent edge has color 0, color the 1-subtrees with 1.
- Case 5-2:
  - If the parent edge has color 1, color the 1-subtree with 3 and the 0-subtrees with color 0.
  - If the parent edge has color 3, color the 1-subtree with 4 and the 0-subtrees with color 2.
  - If the parent edge has color 4, color the 1-subtree with 1 and the 0-subtrees with color 2.
- Case 6-0:
  - If the parent edge has color 0, color the remaining edges with color 4.
  - If the parent edge has color 4, color the remaining edges with color 2.
  - If the parent edge has color 2, color the remaining edges with color 3.
  - If the parent edge has color 3, color the remaining edges with color 0.

  This solves the general case with $S = 5$.