Labyrinth

Problem author: Michael Mirzayanov; problem developers: Sergey Melnikov, Michael Mirzayanov

Let the required two paths have the form:

- $s = u_1, u_2, \dots, u_x, t$;
- $s = v_1, v_2, \dots, v_y, t$.

Let us show that there is always a pair of required paths such that the vertices u_x and v_y (that is, the penultimate vertices of the paths) lie in different subtrees of any depth-first search tree rooted at s. This only applies when both vertices are different from s.

There is a separate corner case in this problem when $u_x = s$ or $v_y = s$. Just remember about it, it is easy to handle it in code.

Indeed, let's take t such that the distance from s to t is minimal.

Suppose this is not the case and there is a depth-first search tree such that vertices u_x and v_y are in the same DFS subtree rooted at s. But since t is the answer, there are two distinct vertex-disjoint (except vertices s and t) paths: $u_1, u_2, \dots, u_x, t$ and $v_1, v_2, \dots, v_y, t$.

Since $u_1 \neq v_1$, then at least one of these paths starts not in the subtree where u_x and v_y are located. Without loss of generality, let this path be $u_1, u_2, \dots, u_x, t$. Find the first vertex in it (minimum index j) such that u_j belongs to the path in the DFS tree from s to u_x. Thus, we have built a pair of non-intersecting paths (from s to u_j) that end at the same vertex, and this vertex is closer to s than t. We get a contradiction with the fact that the distance from s to t is minimal.

Thus, it is enough to run a depth-first search and choose such a vertex as t, such that:

- let the DFS parent of vertex t be vertex u_x,
- t has an edge from some vertex v_y, which in this DFS tree is in a different DFS subtree than t relative to the root s (or $v_y = s$ and $u_x \neq s$).

These paths in DFS tree (from s to u_x and from s to v_y) will induce the required paths.