concept

for (auto e : adj[c]){

if (e == p) continue;

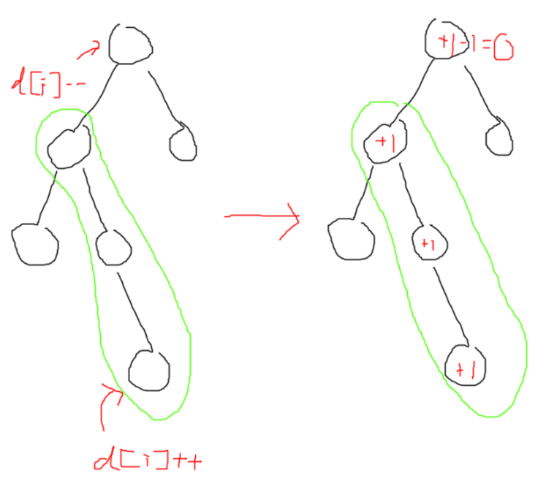
value[c] += value[e] + d[e];

}

(init value[n] to be all 0)

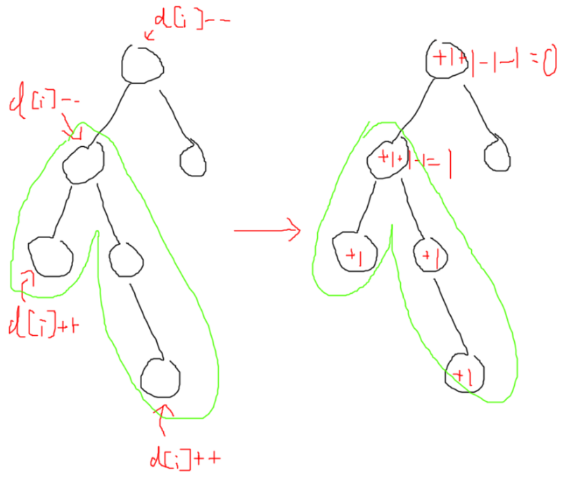
on vertices

Given a tree with vertices, and paths, find out how many paths crossed each vertex.

Let’s consider the case where the path is from i to its ancestor.

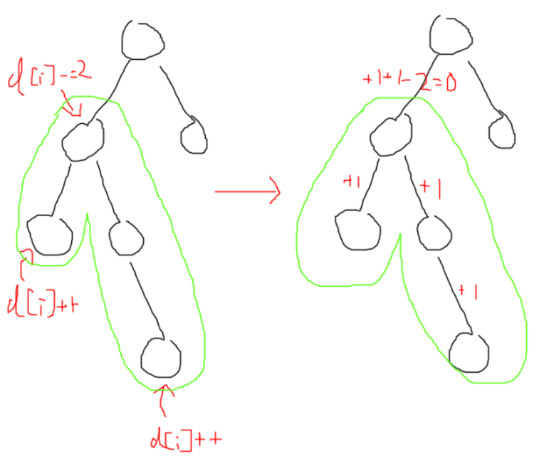
In the diagram, you want to add the green path.

What you can do is the bottom end d[i]++, and the father of the upper end d[i]--. It is like 1D difference array if you only consider that branch from the root.



Now, what if the path is not like that? We can find its LCA.

We should do the 2 ends d[i]++. Then, maybe we should do parent of lca d[i] -= 2? But, if you do it like that, the lca node will += 2 because it is the overlapping part! Therefore, you should do lca d[i]-- and parent of lca d[i]--.

On edges

now, value[i] means the value of edge from i to its parent.

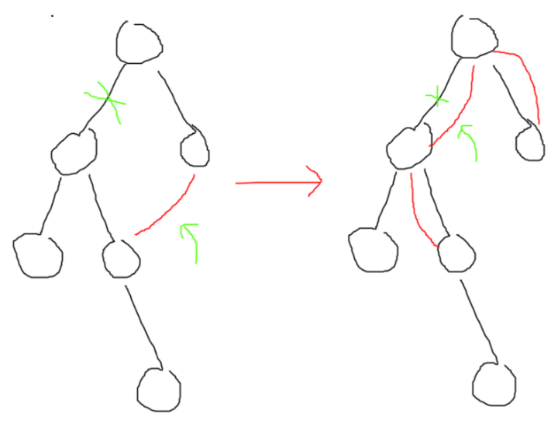
Same as above, we should consider the path’s LCA.

This time, because there is no “overlapping” edges, we can directly lca d[i] -= 2.

Problem 1

<https://www.acwing.com/problem/content/description/354/>

Given an undirected graph with vertices. There are 2 types of edges. There are edges of the first type, and it forms a tree. There are edges of the second type. How many ways can you cut the graph into 2 connected components by deleting 1 edge of each type?

We can think of the second type of edges as adding multiple edges in the tree.

For example, if you delete the tree edge, than the red type 2 edge is the only potection (against being cut apart). After the transformation, this is still the case.

Therefore, the answer can be obtained by doing difference array on the edges. If an edge has 0, then you can cut any type 2 edges, so answer += m. If an edge has 1, then you need to cut both, so answer++. If an edge has more than 1, you cannot cut the tree by cutting that edge.