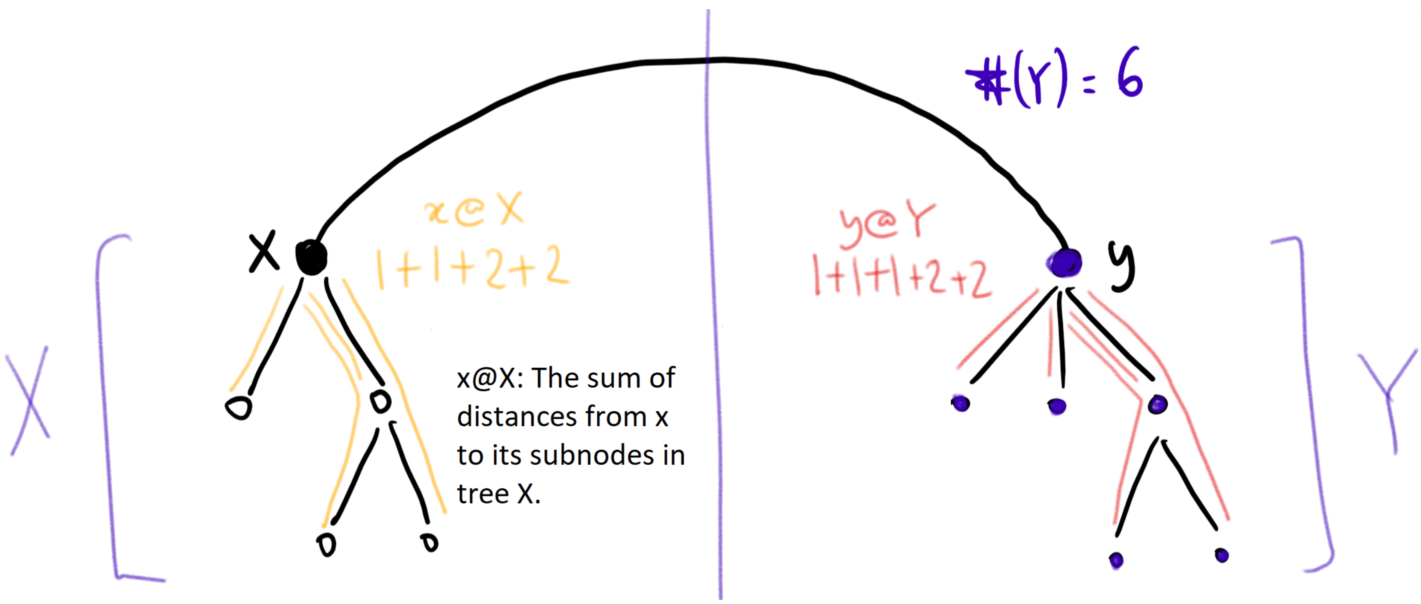
Approach #1: Subtree Sum and Count [Accepted]

Intuition

Let `ans` be the returned answer, so that in particular `ans[x]` be the answer for node `x`.

Naively, finding each `ans[x]` would take $O(N)$ time (where N is the number of nodes in the graph), which is too slow. This is the motivation to find out how `ans[x]` and `ans[y]` are related, so that we cut down on repeated work.

Let's investigate the answers of neighboring nodes x and y. In particular, say xy is an edge of the graph, that if cut would form two trees X (containing x) and Y (containing y).



ans[x] = x@X + y@Y + #(Y)
= 6 + 7 + 6

Then, as illustrated in the diagram, the answer for x in the entire tree, is the answer of x on X `"x@X"`, plus the answer of y on Y `"y@Y"`, plus the number of nodes in Y `"#(Y)"`. The last part `"#(Y)"` is specifically because for any node `z in Y`, `dist(x, z) = dist(y, z) + 1`.

By similar reasoning, the answer for y in the entire tree is `ans[y] = x@X + y@Y + #(X)`. Hence, for neighboring nodes x and y, `ans[x] - ans[y] = #(Y) - #(X)`.

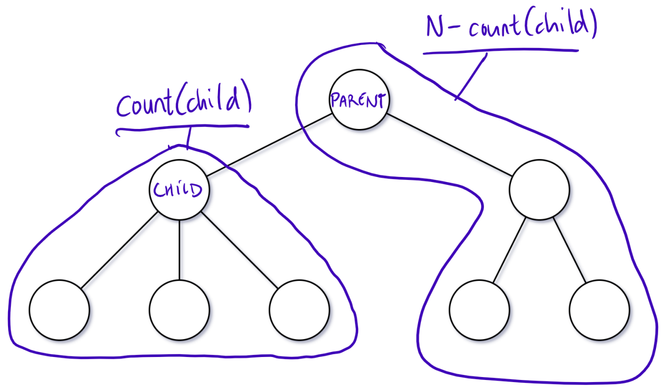
Algorithm

Root the tree. For each node, consider the subtree S_{node} of that node plus all descendants. Let `count[node]` be the number of nodes in S_{node}, and `stsum[node]` ("subtree sum") be the sum of the distances from `node` to the nodes in S_{node}.

We can calculate `count` and `stsum` using a post-order traversal, where on exiting some `node`, the `count` and `stsum` of all descendants of this node is correct, and we now calculate `count[node] += count[child]` and `stsum[node] += stsum[child] + count[child]`.

This will give us the right answer for the `root`: `ans[root] = stsum[root]`.

Now, to use the insight explained previously: if we have a node `parent` and it's child `child`, then these are neighboring nodes, and so `ans[child] = ans[parent] - count[child] + (N - count[child])`. This is because there are `count[child]` nodes that are `1` easier to get to from `child` than `parent`, and `N-count[child]` nodes that are `1` harder to get to from `child` than `parent`.



Using a second, pre-order traversal, we can update our answer in linear time for all of our nodes.

Complexity Analysis

- Time Complexity: $O(N)$, where N is the number of nodes in the graph.
- Space Complexity: $O(N)$.