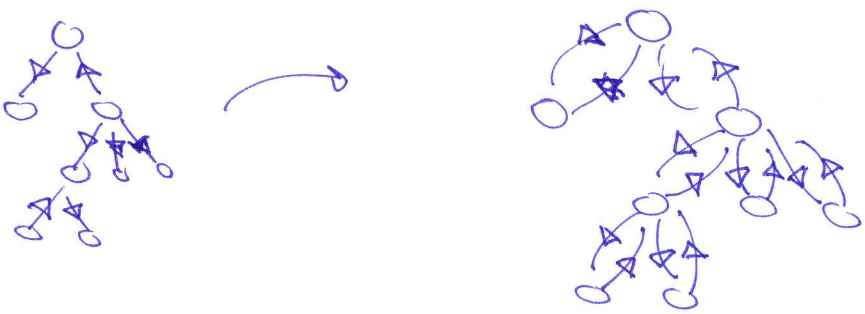
Euler Tour On Trees

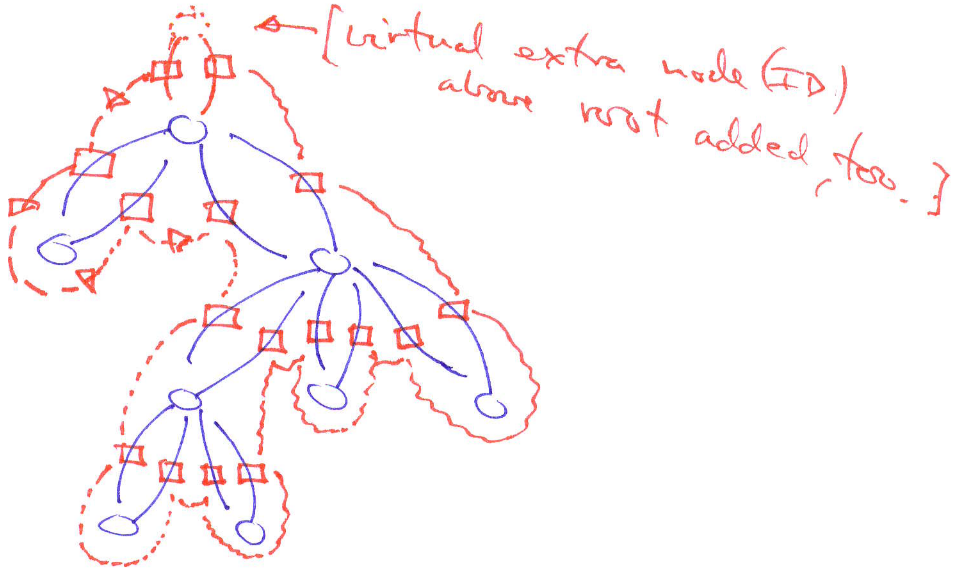
Input: Edges of a tree (any orientation) and an ID of the root node

Step 1: Duplicate all edges so each exists in both orientations:

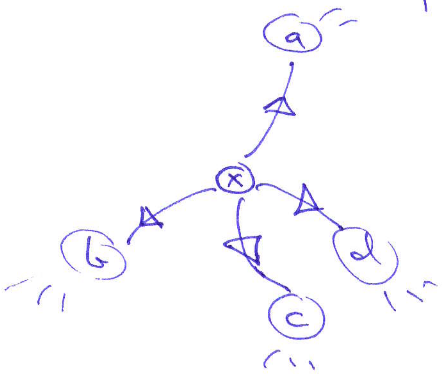


[Done in a scan step].

Step 2: Main idea: think of edges as nodes and make a path of these:

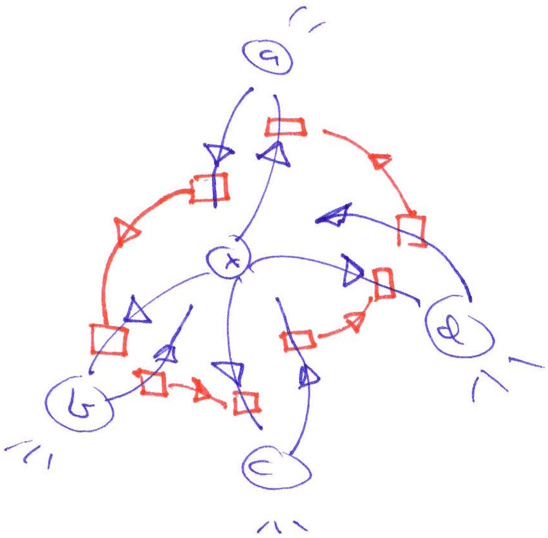


<u>Main point</u>: all the edges between the new type of nodes can be generated from the neighbor list of the original nodes. Namely, from neighbor list of x:



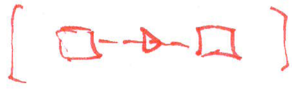
$$\begin{bmatrix} \vdots \\ (x,a) \\ (x,b) \\ (x,c) \\ (x,d) \\ \vdots \end{bmatrix}$$

we can generate



$$\begin{array}{c} \vdots \\ \big((a,x),\ (x,b)\big) \\ \big((b,x),\ (x,c)\big) \\ \big((c,x),\ (x,d)\big) \\ \big((d,x),\ (x,a)\big) \\ \vdots \end{array}$$

If we do this for <u>all</u> original nodes [and for the root remember to add the new virtual node as a neighbor], <u>we get exactly all the edges of the new type</u>.



By a sorting step on the original edges [lexicographic ordering], we can generate a file which is a concatenation of all the neighbor lists (of the original nodes).

By a scan step (traversal of this file), we can perform the action above for all original nodes. (creating all the new type edges).

__Step 3__ : The newly created edges (of type $\square \to \square$) constitute a path (a list).

Performing LIST RANKING on this path allows us to traverse this path in scan time.

__Total Cost__ is $O(\mathrm{Sort}(V) + \mathrm{Scan}(V))$

$= \underline{O(\mathrm{Sort}(V))}$ (Note: $V = E$ for trees).

__Note__ : By tranfering info (node IDs, node annotations, edge annotations) from original tree to the new edges, all calculations which can be done during a Euler Tour (DFS search) on original tree can be done during the traversal of the final path