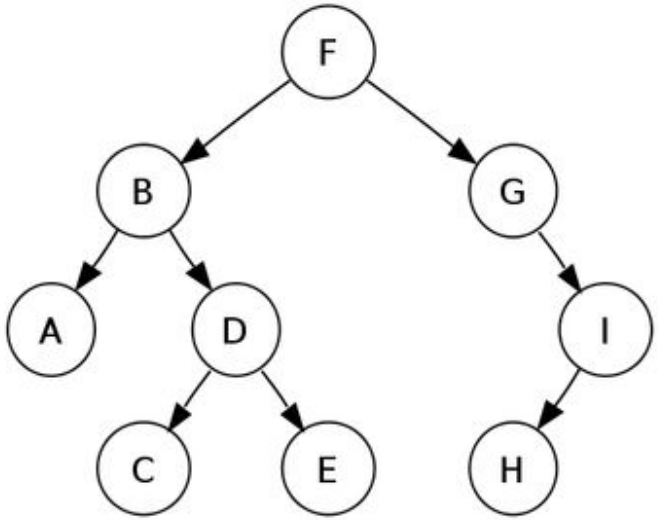
Lowest Common Ancestor

By Max, Peter, Kenneth, and ChrisT

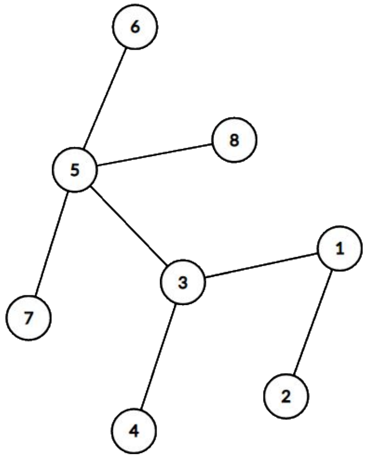
Trees

- Trees are a type of graph where there is exactly one path between any two nodes
- Trees are a common type of graph you may find on CS problems
- There are special algorithms that apply to tree graphs
- Reflect many real-life examples
  - Best way to connect nodes using the least amount of edges
- https://csacademy.com/app/graph_editor/

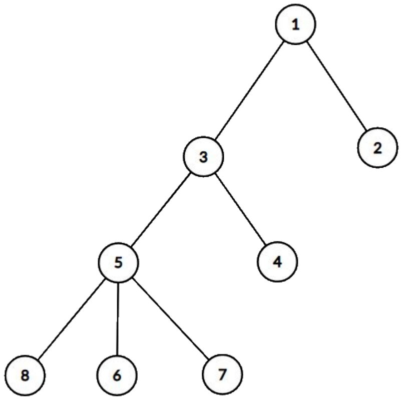


Trees

- Rooted vs Unrooted representation
- Same tree with different visualizations, useful for applying algorithms
- Rooted Tree
  - One node is designated as the root
  - Each node in the tree has a parent and a child node
  - Exceptions:
  - Leaf node has no child node
  - Root node has no parent node
  - e.g. Family Tree



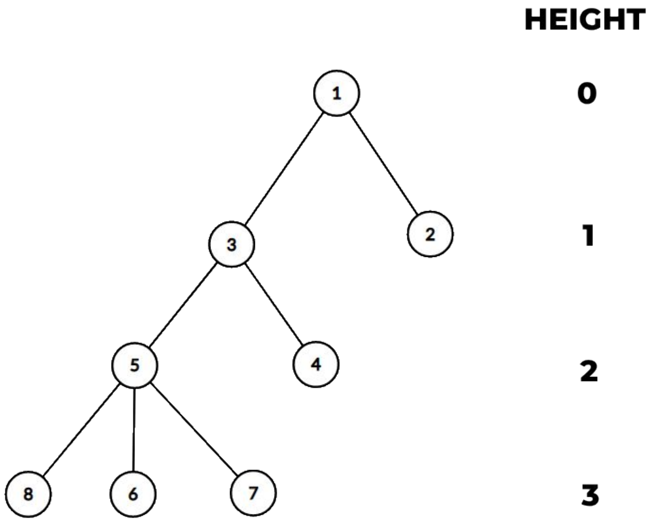
Unrooted



Rooted

LCA

- A common query that must be completed is finding the LCA (lowest common ancestor) of a rooted tree
  - The lowest node, height-wise, of the tree that contains nodes A and B



LCA

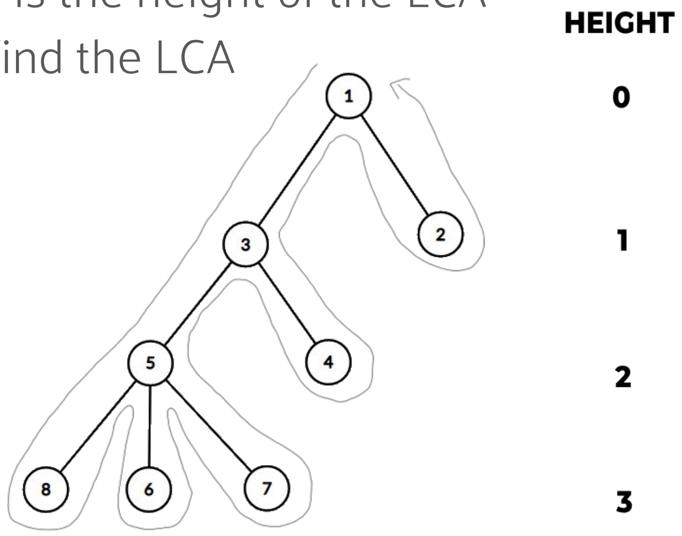
- Why is the LCA useful?
- Often, problems involving trees will ask you to perform some query on a path
- Finding the LCA enables you to split the query into more manageable parts
- Any path from nodes u -> v can be split into u -> lca(u,v) -> v
- Then we can answer queries for the individual paths u -> lca(u,v) and lca(u,v) -> v and combine them to obtain the final result
- For example, how would we find the distance between any two nodes in a tree efficiently?
- Define dis[u] to be the distance from the root (if unrooted tree, just root arbitrarily) to the node u
- Then, the distance between u and v can be found by calculating (dis[u] - dis[lca(u,v)]) + (dis[v] - dis[lca(u,v)]) = dis[u] + dis[v] - 2 * dis[lca(u,v)]

Euler Tour LCA

- The Euler Tour is similar to DFS order, used to linearize a tree
- In DFS, every time you enter or exit a node, append it to the Euler Tour
- Make an array storing the height of every node in the Euler Tour
- The RMQ between nodes a and b in that array is the height of the LCA
- Instead you can find the index of the RMQ to find the LCA

Euler Tour: 1, 3, 5, 8, 5, 6, 5, 7, 5, 3, 4, 3, 1, 2, 1

Height: 0, 1, 2, 3, 2, 3, 2, 3, 2, 1, 2, 1, 0, 1, 0



Binary Lifting

- For every node i, precompute its 2^0-th, 2^1-th, 2^2-th, ... , 2^j-th parent in an array p[i][j]
- For every node i, precompute its height from the root in an array h[i]
- To find the LCA of 2 nodes a, b:
    - Make sure a and b have the same height. If one is lower than the other, use the parent array to decrease its height to the same as b
    - If a == b, then return a
    - Do:
        - Search for the highest j such that p[a][j] != p[b][j]
        - Set a to p[a][j] and b to p[b][j]
        - while there exists a j
    - return p[a][0]

Practice Problems

- https://mcpt.ca/problem/treedistance
- https://old.yosupo.jp/problem/lca (check your lca implementation works)

LCA + Other Stuff

- https://dmoj.ca/problem/coci19c5p4 (difference array on tree)
- https://dmoj.ca/problem/acc2p3 (sparse table maintains other info)
- https://dmoj.ca/problem/bbc08b (directing edges, good editorial)
- https://dmoj.ca/problem/roadredirection (directing edges, requires ds)
- https://dmoj.ca/problem/utso15p5 (mst)
- https://dmoj.ca/problem/inaho8 (... good luck)
- https://www.acmicpc.net/problem/16074 (mst + lca)