Assignment 2

If anything in the assignment is unclear, you have two options. You can ask for clarifications in the #assignments channel on Slack. It is also great if you can also make assumptions: real-world problems are always unclear, and as engineers we want to move on and make progress, even if we need to re-adjust later. If you do make assumptions, please try to identify them and document them as comments in your code.

Ancestors

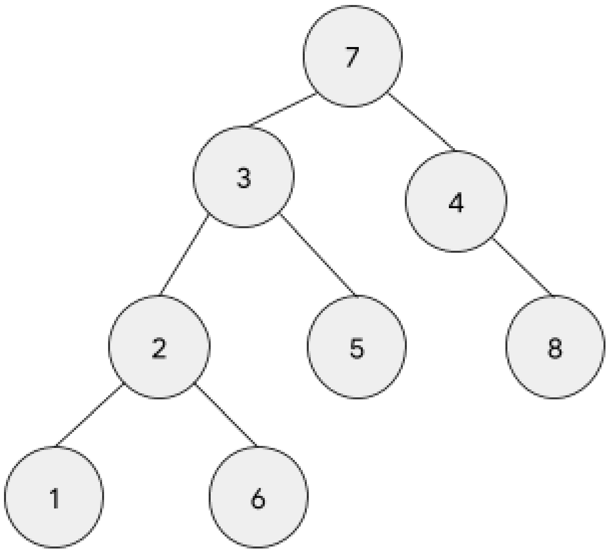
FIND A NODE
Given a binary tree and a key, write a function that returns a reference to the node with that key in the tree. Keys are unique on the tree.

Note: You should implement your own data structure to store the binary tree.

FIND THE ANCESTORS
Given a binary tree and a key, write a function that returns all the ancestors of the node with that key in the given binary tree.

For example:

- For the key 6 and the following tree we should return: [2, 3, 7].

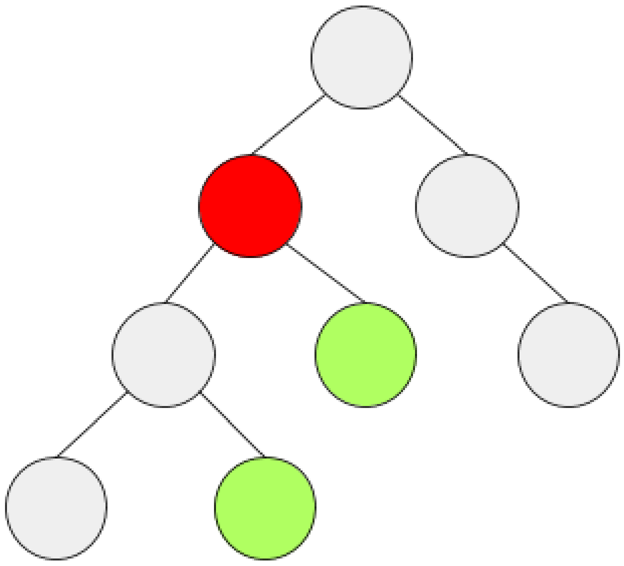


Note: the tree is NOT a binary search tree (where the keys are ordered), but an arbitrary binary tree.

COMMON ANCESTOR
Design an algorithm and write code to find the lowest common ancestor given two references to two nodes in a binary tree. Avoid copying/storing nodes in another container.

For example:

- Given the two nodes highlighted in **green**, the lowest common ancestor is the node highlighted in **red**.



Optional Challenges

If you are done with the main assignment, you can pick up any of the optional challenges below and try to solve those as well. If you decide to take one or more optional challenges, make sure these are implemented as separate files from the original challenge. You might want to refactor some of your code so that you can reuse it across the main assignment and the optional challenges. You can submit a separate pull request for the optional challenges or include them with the main assignment.

1. *OPTIMIZE FOR MEMORY*. How can you make the tree structure take as little memory as possible, while still being able to implement all questions?
2. *OPTIMIZE FOR SPEED*. If you use a little more memory than needed, can you speed-up queries for common ancestors?