Data Structures Final Review (Workshop)

Topics Covered:

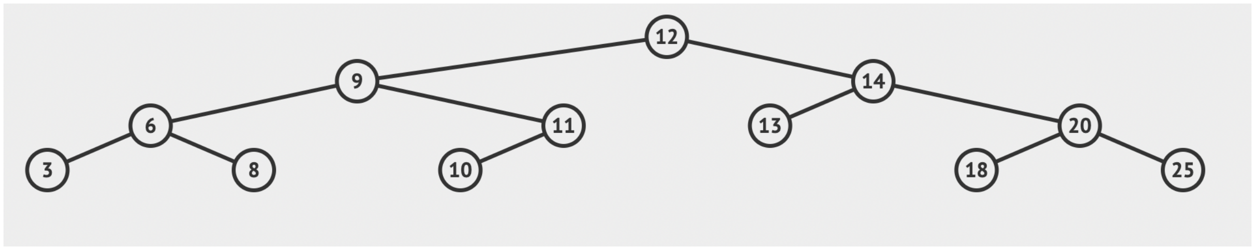
- Binary Search Trees
- AVL Trees
- Graphs
- B-Trees

Binary Search Trees

1. Explain why a Binary Search Tree's average time complexities for operations are O(log(n)), what attributes allow this?

2. Implement an **insertToBST(node* root, int value)** function. Can you do this iteratively and recursively?

3. Given the following BST, perform: **Preoder**, **Inorder**, and **Postorder** Traversal



   Can you implement the code for all of the following traversals?

4. Implement a function that returns the **inorderSuccessor(node* root, node* givenNode)** of a given node.

5. Implement a **search(node *root, int value)** function that checks whether a value is present in a BST or not.

6. Implement a function that **printsLeaves(node* root)** of a given BST.

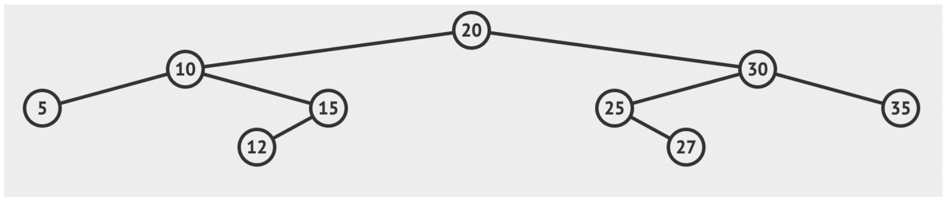
7. Implement a function that checks if a given BST **isBST(node * root)**.

8. Implement a function that adds the values in a BST into an array in descending order. **sortedValues(node* root, vector<int> &arr)**.

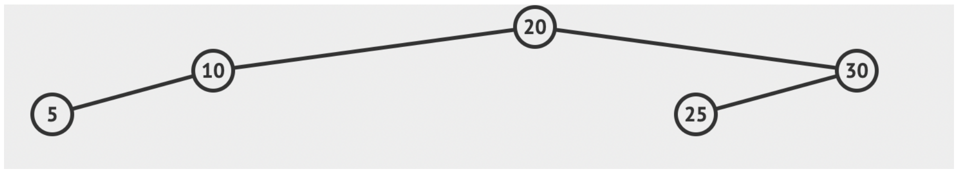
9. Implement a function that checks if a BST's structure **isMirror(node* root)** of itself (check if the root's left subtree structure is a mirror of its right subtree's structure)

10. Implement a function that returns the depth of a given node (distance from the root node) starting at 0. Return -1 if the node doesn't exist in the tree → **getDepth(node* root, int targetNode, int depth)**

isMirror() = TRUE



isMirror() = FALSE



Leetcode Problems w/ "Binary Search Tree" tags (easy/mediums) are good practice for the exam.

AVL Trees

1. Insert the following values into an AVL tree, show the tree after every insertion:

   [21, 17, 15, 19, 18, 10, 8, 13]

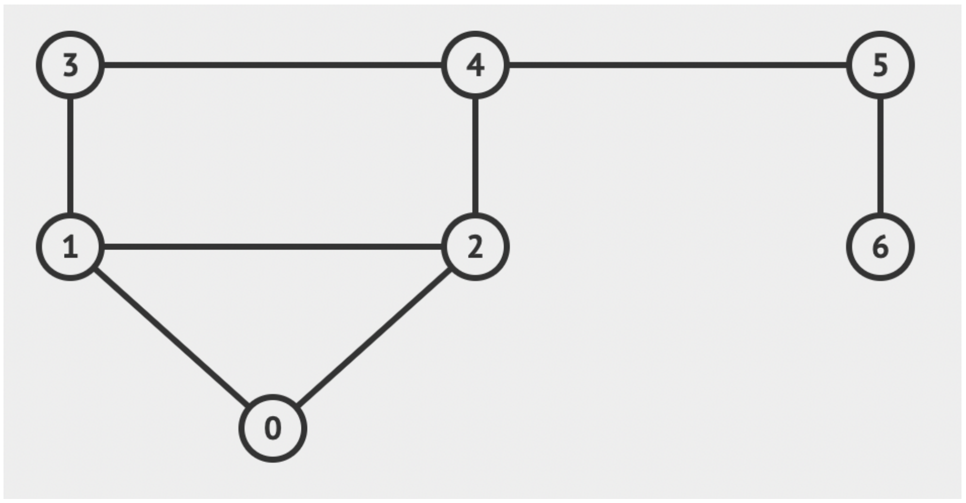
2. Name the different sorts of rotations you use when performing the self-balancing in an AVL tree. Explain how they work and when you perform them.

3. Implement all the AVL rotation methods.

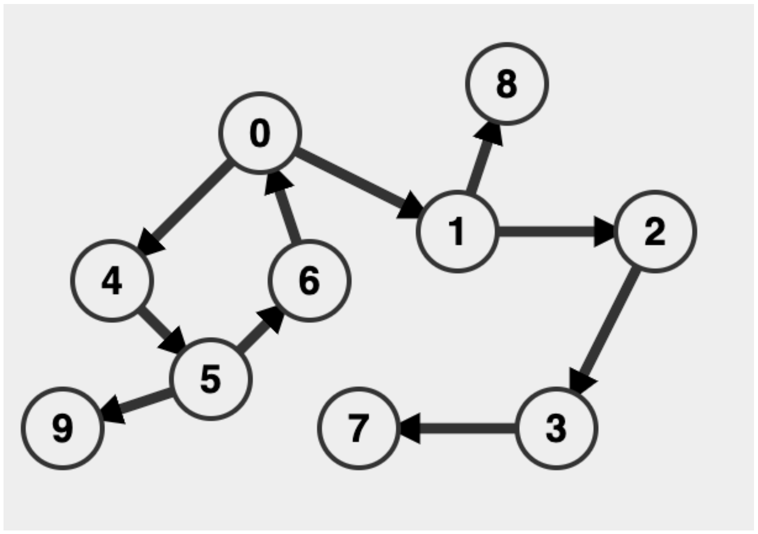
4. Implement a function that returns the **balanceFactor(node *givenNode)** of a given node.

Graphs

1. Given the graph, write out its representation as an adjacency matrix and adjacency list.

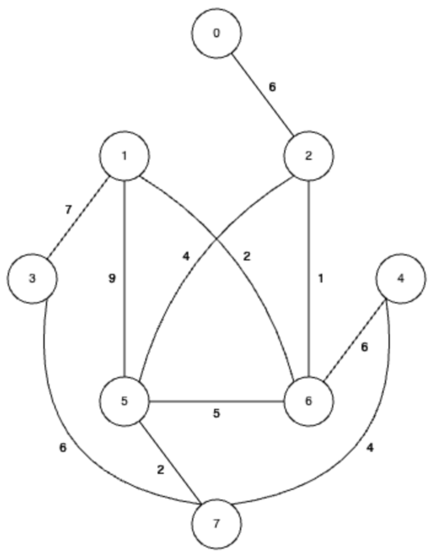


2. Perform the DFS algorithm on the following graph and show the order in which the nodes were visited (assuming 0 is the source node)

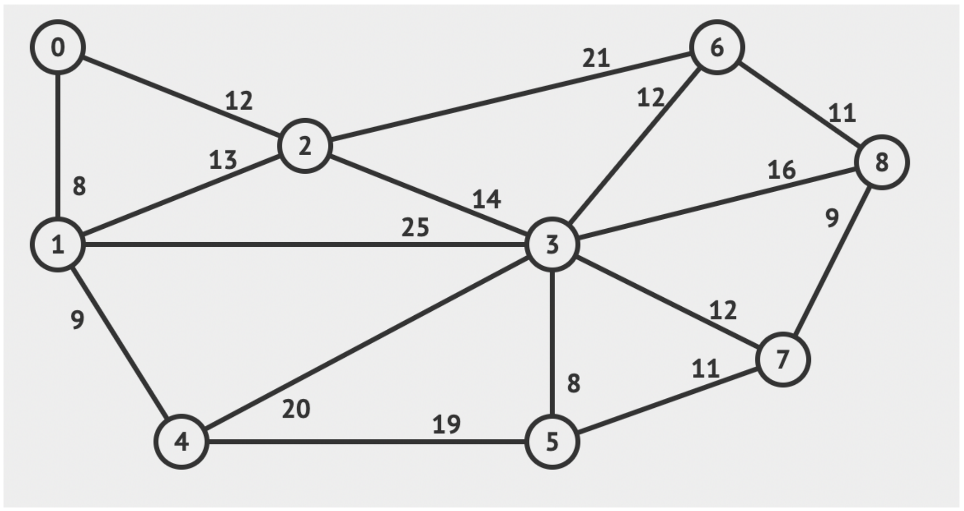


3. Implement the **BFS(int graph[][], int sourceNode, int numVertices)** algorithm represented as an adjacency matrix. It should as well print the order in which the vertices are visited.

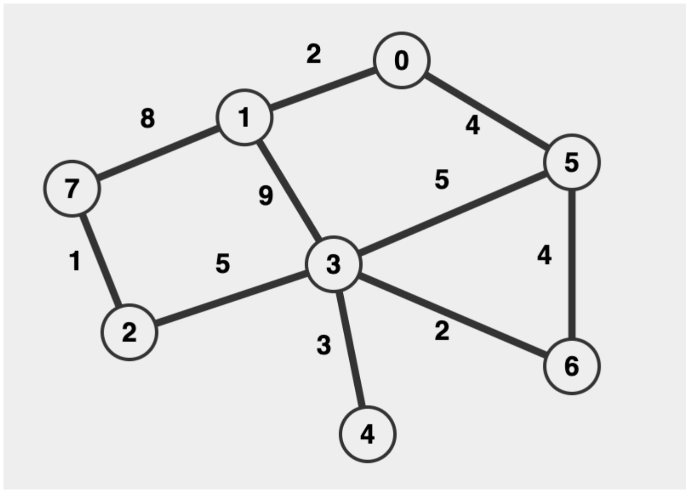
4. Perform Kruskal's Algorithm on the following graph and draw the minimum spanning tree.



5. Perform Prim's Algorithm on the following graph and draw the minimum spanning tree. Start from Vertex 5.



6. Given the graph, draw the shortest path tree using Dijkstra's Algorithm starting from Vertex 1.



0	1	2	3	4	5	6	7

B Trees

1. Go over past insertion into B-Tree problems covered in past Workshop sessions.