Part - 4

Hi guys, This week we will be releasing the problems along with the studying part so that you can do it together. this week, you will be introduced to a important data structure called Binary trees, and a very important concept of graphs, usually only tougher questions are based on this part. So make yourself ready to solve them.

BST - Binary Search Trees:

Binary Search Trees (BST) - Learn about the following:

- Definition and properties of BST
- Tree traversal in BST (Pre-order, In-order and Post-order)
- Insertion and deletion in BST
- Time and space complexity of BST operations

You can refer the book and also this link

Extra on BST

1. Red-Black Trees

You can refer to the book and also this link to learn about Red-Black Trees. This is a self-balancing binary search tree which is used to maintain the balance of the tree and ensure that the operations run in O(logn) time complexity. Many STL data structures such as Set, Map, multimap are implemented using red-black trees , i.e., the inner implementation uses logic of red-black trees.

2. AVL Trees

AVL Trees are another type of self-balancing binary search tree which is used to maintain the balance of the tree and ensure that the operations run in O(logn) time complexity. You can refer to the book and also this link to learn about AVL Trees. This are even more balanced than Red-black trees, but requires additional rotations during insertion and deletion.

Graphs:

go through the chapter 11, 12, parts of 14th chapter in the book. This would give you a proper interface to the graphs. This week we would like you to learn about:

1. Basic terminology in graphs:
    a. Nodes or vertices
    b. Edges
    c. different types of graphs:
        i. directed graphs
        ii. undirected graphs
    d. connected components in undirected graphs and singly connected components in directed graphs
    e. degree of a vertex
    f. degree sequence in undirected graphs
    g. types of graphs based on its properties :
        i. bipartite graphs
        ii. trees
        iii. cycles
        iv. complete graphs, etc
2. ways of representing a graph in a code
    a. adjacency list
    b. adjacency matrix
    c. edge list
3. Graph traversals:
    a. DFS - Depth first search
        i. Learn about the different types of edges in DFS tree (Learn what is DFS tree)

        ii. Note that DFS in directed graphs is quite different from what it is in undirected graphs.

    b. BFS - Breadth first search

    c. their uses:

        i. How to find if the graphs is connected

        ii. how to find Strongly connected component in a directed graph

        iii. How to find if the graph is 2-Edge connected

        iv. How to find if the graph is 2-vertex connected

        v. Bipartite checking ,etc.

So, that is what that has to be studied in this week for graphs, I know it seems too much, but believe me it is not. So to study, you could study this from the book - programmer's handbook or from NPTEL course on DSA by naveen garg: https://www.youtube.com/watch?v=zWg7U0OEAoE&list=PLBF3763AF2E1C572F.

Note: If you are going to see the lectures, you can do it 2x speed(ig !), but it is sure takes longer time than reading these stuff from the book . You can also use other resources, make sure that you had gone through all of the above topics using it.

Note

It is highly recommended to make a well organised class for BST having all the important functions. It can be quite handy for several problems. Same applies for graphs. Although for graphs, it is not as useful as BST as the questions for graphs are of a wide variety.