course_content

top

W9: BSTs and Heaps

W10: Graphs

W11: Greedy Algorithms

Data Structures and Algorithms

Traversals & Binary Search Trees

Week 9

Time:
10 min

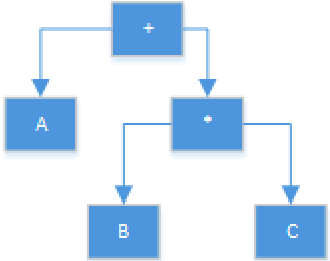
Trees - Traversals (Self-learning)

- There are many different traversal methods:
    - Depth-First Search (Mostly Binary Trees)
        - Pre-order traversal
        - In-order traversal
        - Post-order traversal
    - Breadth-First Search (All Graphs)
- What do they mean? How do they work?
- Can you write the method recursively?
- Can you write the method iteratively?

Time:
10 min

Traversal Applications

- In-order Traversals: Binary Search Trees (see next slides)
- Pre-order Traversal: duplicating a tree (have to do parents first)
- Post-order Traversal: deleting trees, expression trees (compiling calculations)
- E.g. In the expression tree, you calculate children first
- Same if you have tasks dependent on other tasks
- Breadth-first: Shortest Path Algorithms! Facebook Friends



Time:
15 min

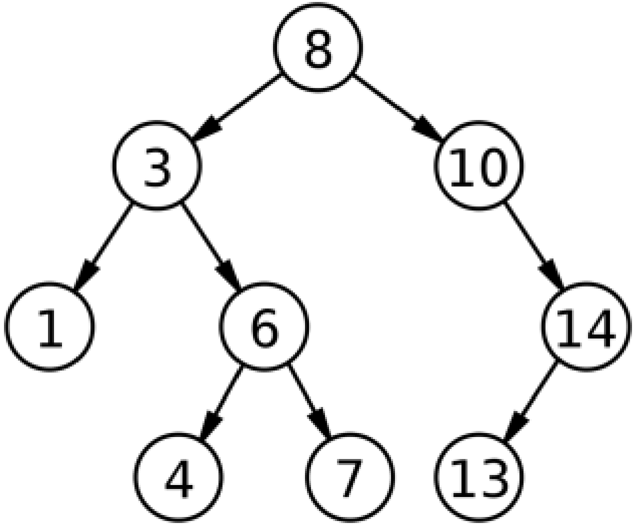
Trees - Binary Search Tree

- We previously looked at sorted arrays and found that maintaining them was hard
- A binary search tree can be a better way of doing this
- Values are inserted into their correct position. Smaller than root means left sub-tree, larger than root means right sub-tree.
- Construct a BST (binary search tree) with the following 5,10,7,8,1,3,6.
- What are the runtimes of the following in a BST? (self-learning)
  - Insert, Search, Delete, Create (with a list of given values)
- An in-order traversal of a BST produces a sorted list.

Time:
15 min

Trees - Binary Search Tree

Binary search tree conducted from list: 8,3,6,10,1,14,7,4,13



Try with the order 1,14,13,3,4,10,8,6,7. How many levels does this have? Too many!

Time:
15 min

Trees - AVL Tree

- To minimise the height of a BST, we use an ‘AVL Tree’

$$balanceFactor = height(leftSubTree) - height(rightSubTree)$$

- An AVL Tree has balanceFactor 1, 0 or -1 (for all roots)
- This forces it to be *balanced*
- Once the balance factor leaves this range, it has a procedure to fix it.
- This a fairly involved procedure that you can read about but it involves a ‘rotation’ of your tree by changing the root and one of the branches. (optional self-learning)

Time:
5 min

Using BST/AVL trees as ordered maps

- Back in Week 3, we discussed maps and how they can be implemented as associative lists or hash tables. We can also implement them as BSTs (or AVLs)
- How does it work?
  - Decide on an ordering of your keys. (small to large, alphabetical etc)
  - Store as a BST(AVL) based on the order of the keys.
  - At each node, store an extra parameter which is the value.
- Searching, Accessing, Inserting and Deleting are all O(log n) time now.
- Not as good as hash tables but no need to worry about collisions.
- Easier to extract all keys here.

Time:
15 min

Videos

- Breadth-First vs Depth-First Search using Queues & Stacks
- Pre-order, In-order, Post-order traversals (Binary Trees)
- Binary Search Tree & AVL Tree (Also great for how to represent binary tree as array and how to rotate AVL trees to make them balanced)

Time:
10 min

Quiz Time

bit.ly/DSA1920Quiz8

Deadline: Monday 30th March (midnight)

Time:
1 min

Questions?....Office Hours!