

15CSE374
INTRODUCTION TO DATA STRUCTURES
AND ALGORITHMS

Sarath tv

Last Lecture

Hashing


Hash Table.

Implementation in python.

BST-Recap

- A binary search tree is a binary tree T such that
- Keys stored at nodes in the left subtree of v are less than or equal to k .
- Keys stored at nodes in the right subtree of v are greater than or equal to k .
- $O(\log N)$
- If the keys are added in a random order, the height of the tree is going to be around $\log_2 n$ where n is the number of nodes in the tree.
- Best case.
- A perfectly balanced tree has the same number of nodes in the left subtree as the right subtree.

- What happens when you insert elements in straightforward sequences(say ascending order).
- Eg insert 2,4,6,8,10,12 into an empty BST

- 
- Problem
 - Lack of “Balance”
 - Unbalanced Tree.

Balanced Binary Search Trees

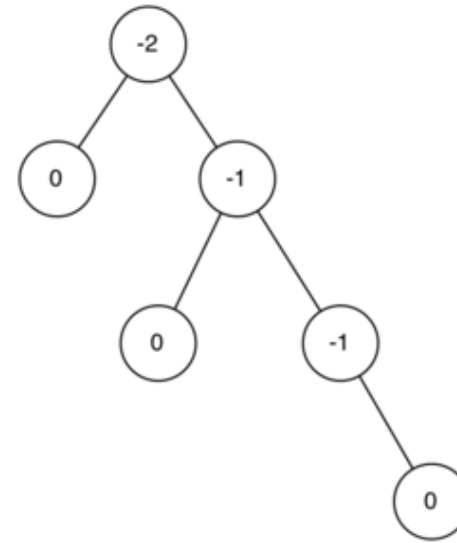
- The performance of the binary search tree can degrade to $O(n)$ when the tree becomes unbalanced
- Special kind of binary search tree that automatically makes sure that the tree remains balanced at all times.
- AVL tree
- Adelson-Velskii and E.M. Landis.
- To implement our AVL tree we need to keep track of a balance factor for each node in the tree. We do this by looking at the heights of the left and right subtrees for each node.

Balance factor

- The balance factor for a node as the difference between the height of the left subtree and the height of the right subtree

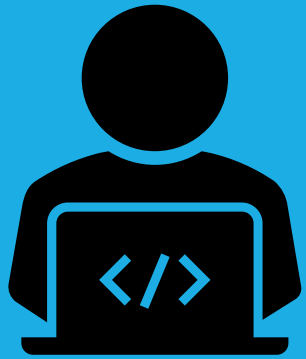
$$\text{balanceFactor} = \text{height}(\text{leftSubTree}) - \text{height}(\text{rightSubTree})$$

- Balance factor
 - Left heavy
 - Right heavy
 - Balanced



- An unbalanced, right-heavy tree and the balance factors of each node.

- Once the balance factor of a node in a tree is outside this range we will need to have a procedure to bring the tree back into balance.



THANK YOU!!!!!!