C, C++, DSA in depth

Binary Search Tree

Saurabh Shukla (MySirG)

Agenda

1. Binary Search Tree
2. Implementation of BST

Binary Search Tree

A binary search tree is the most important data structure, that enables one to search for and find an element with an average running time

$f(n) = O(\log_2 n)$

data

level 0 — 1
level 1 — 1
level 2 — 1
level 3 — 1

$n = 15$

$\log_2 15 = 3.88$

≈ 4

```
if (ptr -> item == data)
else if (ptr -> item > data)
        ptr = ptr -> left;
else    ptr = ptr -> right;
```

$$2^3 = 8$$

$$\log_2 8 = 3$$

$$\log_a n = x$$

$$a^x = n$$

$$\log_2 15 \approx 3.88$$

$$2^4 = 16$$

Binary Search Tree is a binary tree with the value at node N is greater than every value in the left subtree of N and is less than every value in the right subtree of N.

Unless, explicitly said, BST doesn't allow duplicate values.

Implementation

1. node
2. Insertion
3. Traversing
4. Search
5. Deletion

node

```
struct node
{
    node *left;
    int item;
    node *right;
};
```

Insertion

```
if (n->item != ptr->item)
    if (n->item < ptr->item)
     if(ptr->left != NULL)
       ptr = ptr->left;
     else
       ptr->left = n; break
    else
       ptr = ptr->right;
```

90

parptr

30 50 60 70 90 100

30 50 60 70 100

Deletion

delete 60 (no child)

delete 10 (one child)

delete 70 (two children)