COP 3502 - 10/23/23

⓪ Junior Knights - restart Spr 2024

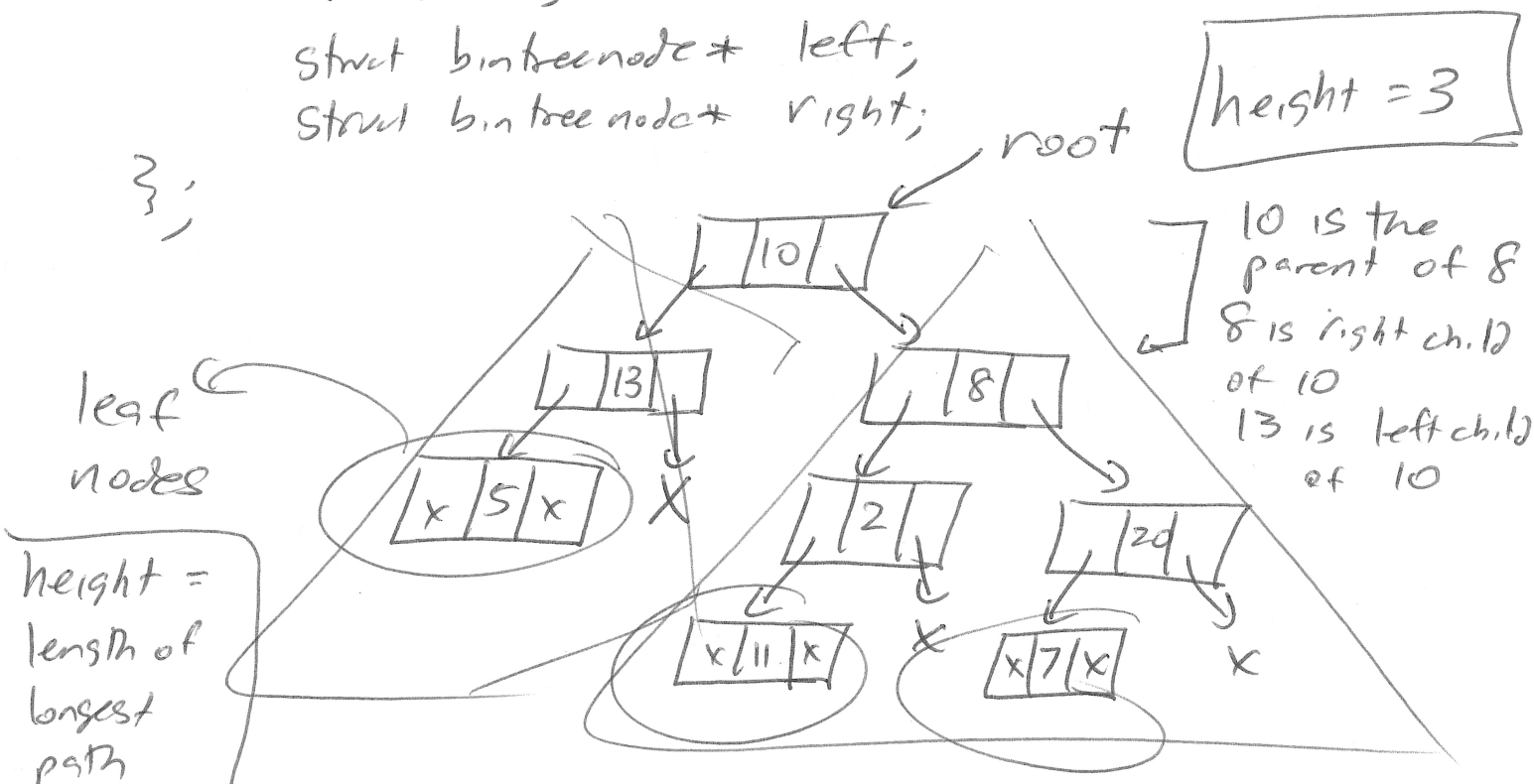
① Binary Trees / Binary Search Trees

Linked List
Dyn Sized Array

Insert data
Delete data
Search data

O(n) operation when there are n items in the structure.

```
struct bintreenode {
    int data;
    struct bintreenode* left;
    struct bintreenode* right;
};
```



Tree: a set of vertices (nodes) connected with edges (pointer links) with no cycles. There's only 1 unique simple path (no repeated vertices) btw any 2 vertices.

height of tree w/0 nodes = -1

= = = = 1 node = 0

= = = w 2 nodes = 1

Structure isn't linear. How do I explore each node?

① Preorder Traversal ①,②,③

② Inorder Traversal ②,①,③

③ Postorder Traversal ~~①,②,③~~ ②,③,①

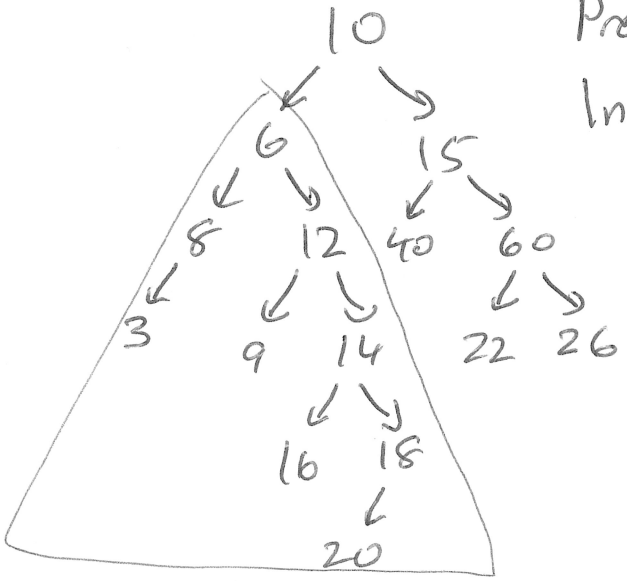
Struct

① data

② left

③ ~~left~~ right

```
void preorder(bintreenode* root) {
    if (root==NULL) return;
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}
```



pre left pre right

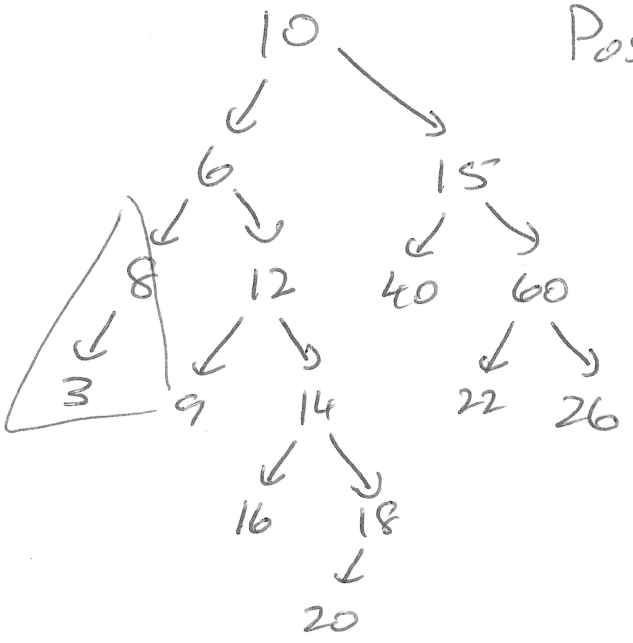
Pre: 10, 6,8,3,12,9,14,16,18,20, 15,40,60,22,26

In: 3,8,6,9,12,16,14,20,18, 10, 40,15,22,60,26

in left in right

Inorder

```
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
```



Post: 3, 8, 9, 16, 20, 18, 14, 12, 6, 40, 22, 26, 60, 15, 10

post left — post right

POST ORDER

```
postorder (root->left);
postorder (root->right);
printf("%d ", root->data);
```

Given In, Pre $\Longrightarrow$ Post

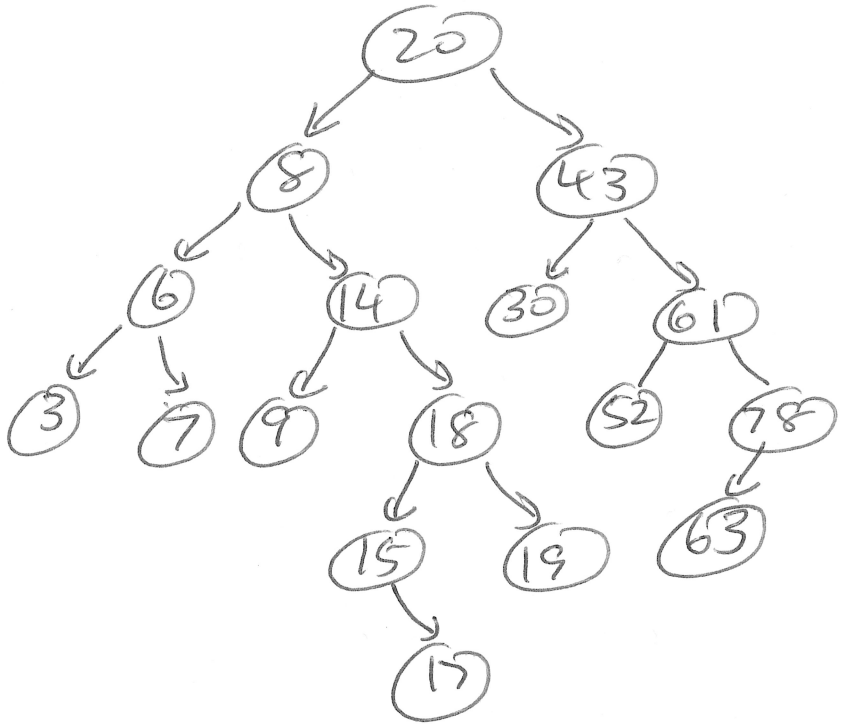
Traversal Run-Times $= O(n)$,
n nodes in the tree

In a Binary Search Tree, we add these requirements:

(a) all element ~~or~~ in left subtree are $\leq$ root

(b) all elements in right subtree are $>$ root.

Sample Binary Search Tree




```
int search(bintreenode* root, int value) {
    if (root == NULL) return 0;
    if (value < root->data)
        return search(root->left, value);
    else if (value > root->data)
        return search(root->right, value);
    else
        return 1;
}
```

$O(h)$, h = height of tree