COP 3502 10/25/21

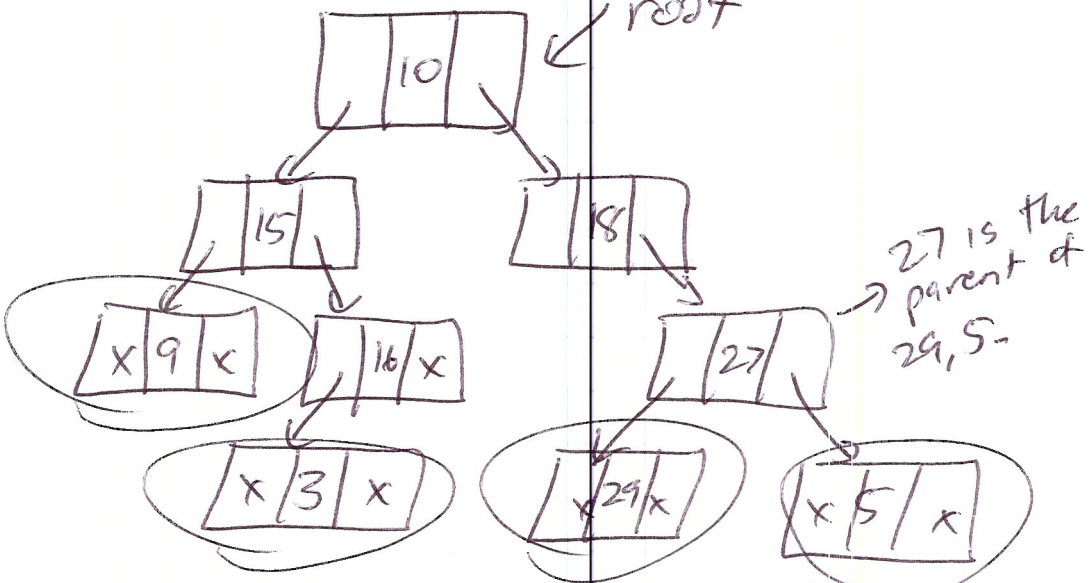
Binary Trees

Data Structures

Dyn Array
Linked List

Insert	O(1)	O(n)
Search	O(n)	↓ maybe faster
Delete	O(n)	

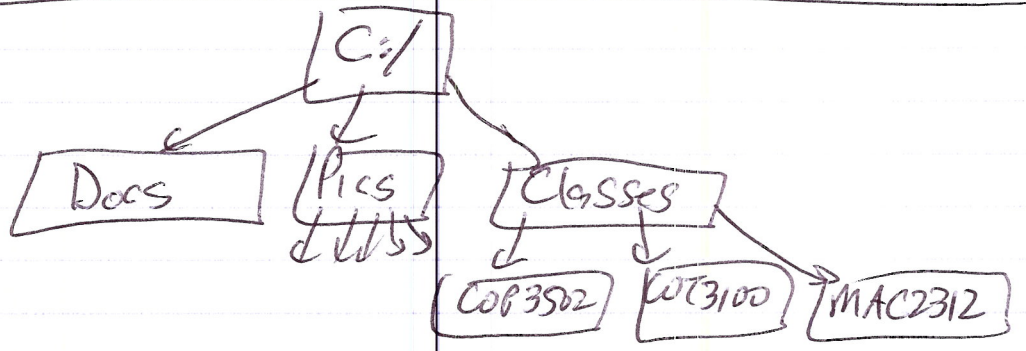
WORST CASE O(n) of the 3



binary = each node has at most 2 children

tree = no cycles

Common Example of a Tree: Directory Structure



Binary Search Tree

height = 4

BST Node Property:
everything left subtree of a node is less than what's stored in the node.

Everything right subtree of a node is greater than what's stored in the node

Equal Values

(1) Always choose same side

(2) Store 1 node with a frequency count

(3) Find a way to break ties

↳ PS

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

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

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

height = longest path from root to any node in the tree

nodes	height
0	-1
1	0
2	1

Tree Traversals

3 PIECES

1. ROOT NODE
2. LEFT SUBTREE
3. RIGHT SUBTREE

} Visit in any order

* CONVENTION - GO LEFT BEFORE RIGHT

3 ORDERINGS

1. ROOT, LEFT, RIGHT (PREORDER)
2. LEFT, ROOT, RIGHT (INORDER)
3. LEFT, RIGHT, ROOT (POSTORDER)

```
void preorder (bintreenode* root) {

    if (root == NULL) return;
    printf("%d", root->data);
    preorder (root->left);
    preorder (root->right);

}
```