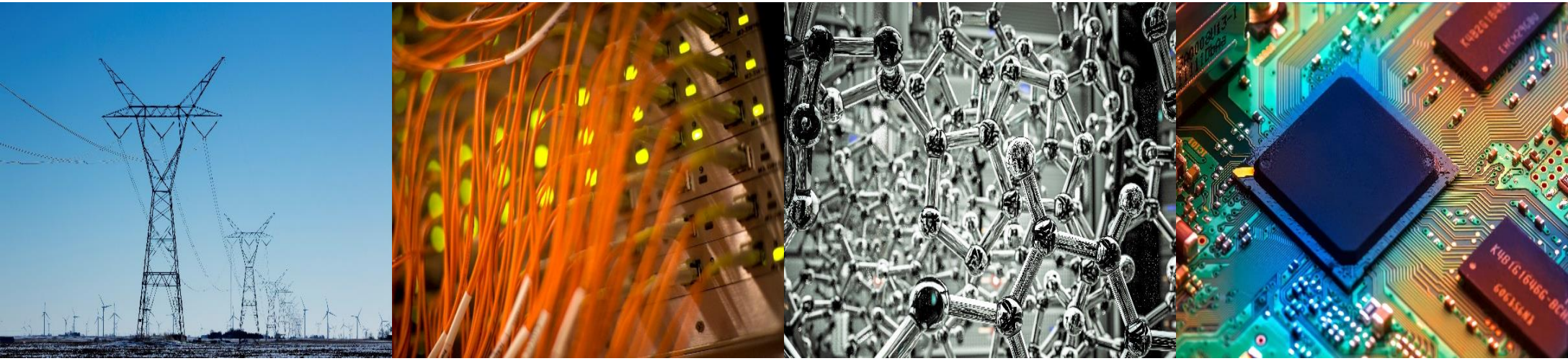
ECE 220 Computer Systems & Programming

Lecture 23 – Trees: traversal and search

November 19, 2024



- Quiz5 should be taken @ CBTF this week
- Final exam conflict request is due on Wednesday, 12/11

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

Tree Data Structure

Array, linked list (stack, queue) – linear data structures

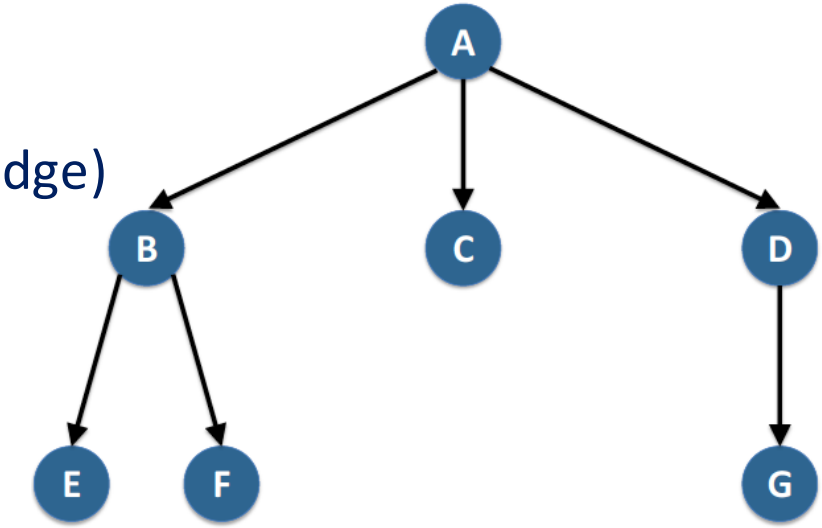
Tree: a collection of nodes connected by edges. It's a **nonlinear** data structure.

Tree Terminology:

parent, child, sibling

root, external node (leaf), internal node, branch (edge)

height, depth, level



- Elements are stored hierarchically
- A tree has at most _____ root
- Besides the root, each element has _____ parent
- Each element has _____ children

Binary Tree

- Each node has ***at most 2*** children – left child and right child

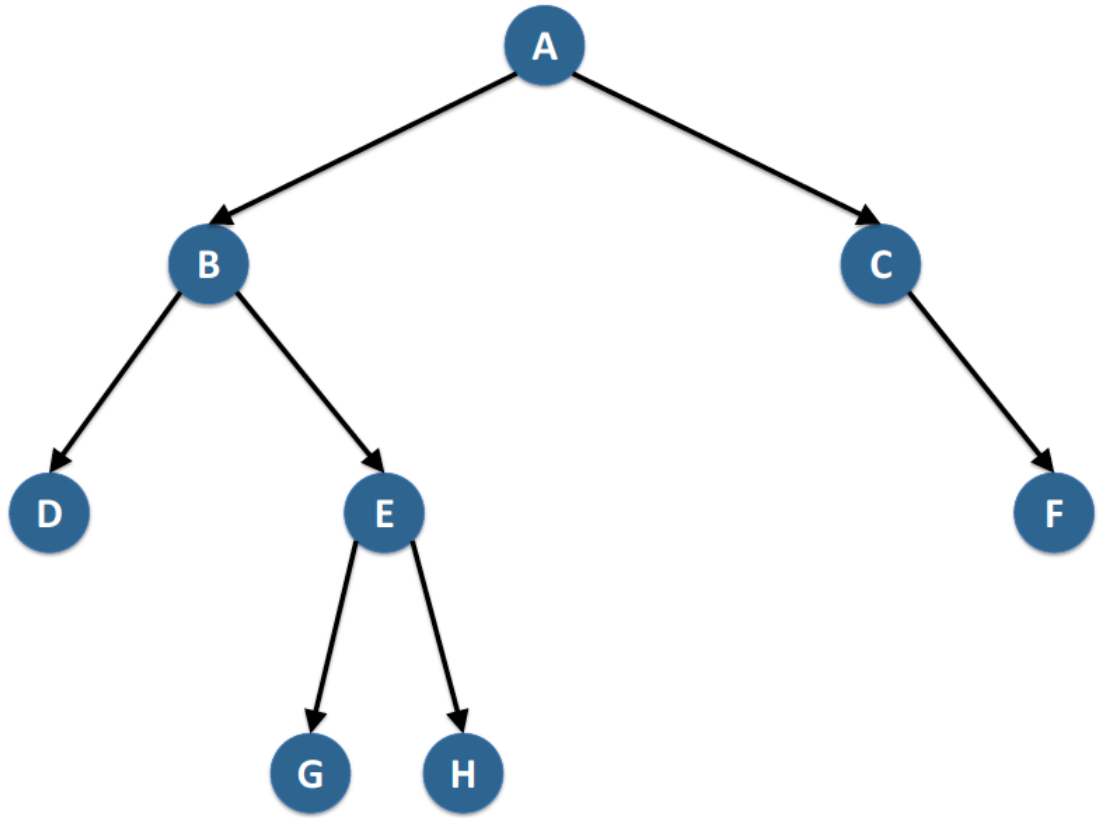
What is the height of the tree?

What is the depth of node E?

What is the height of node E?

Which nodes are internal nodes?

Which nodes are leaves?

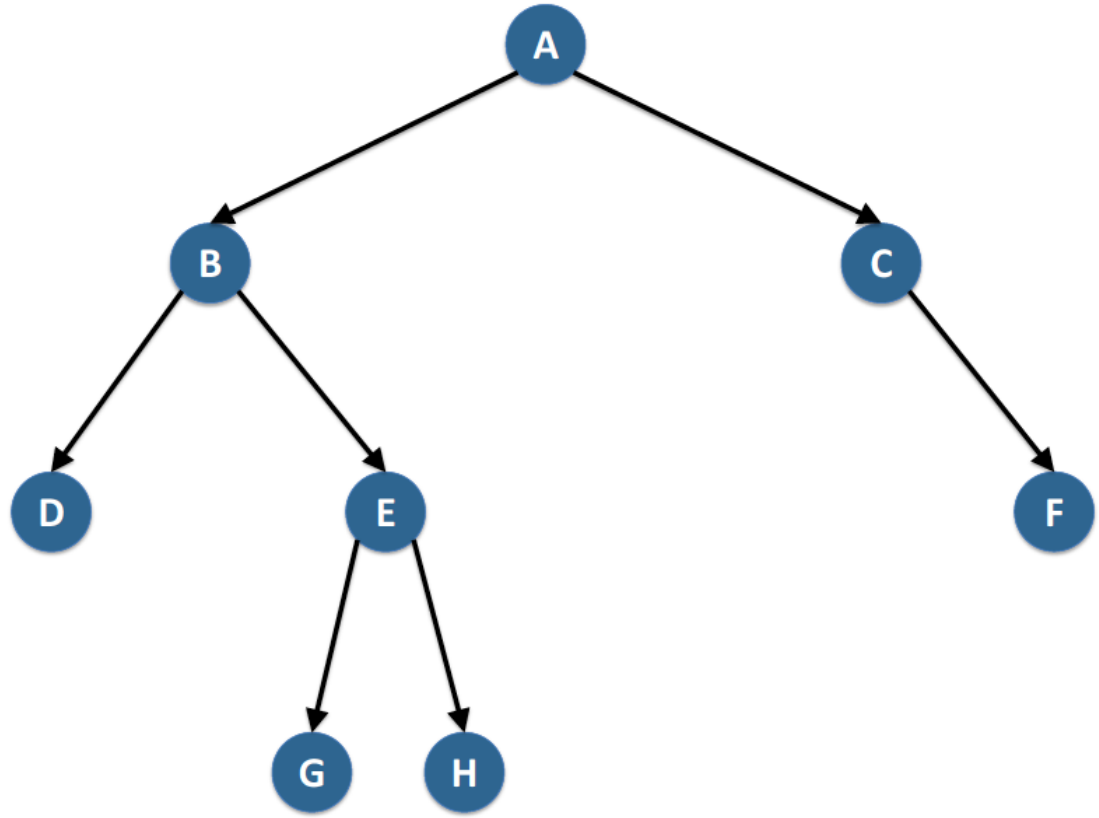


Tree Traversal – BFS & DFS

Breadth-First Search (level-order)

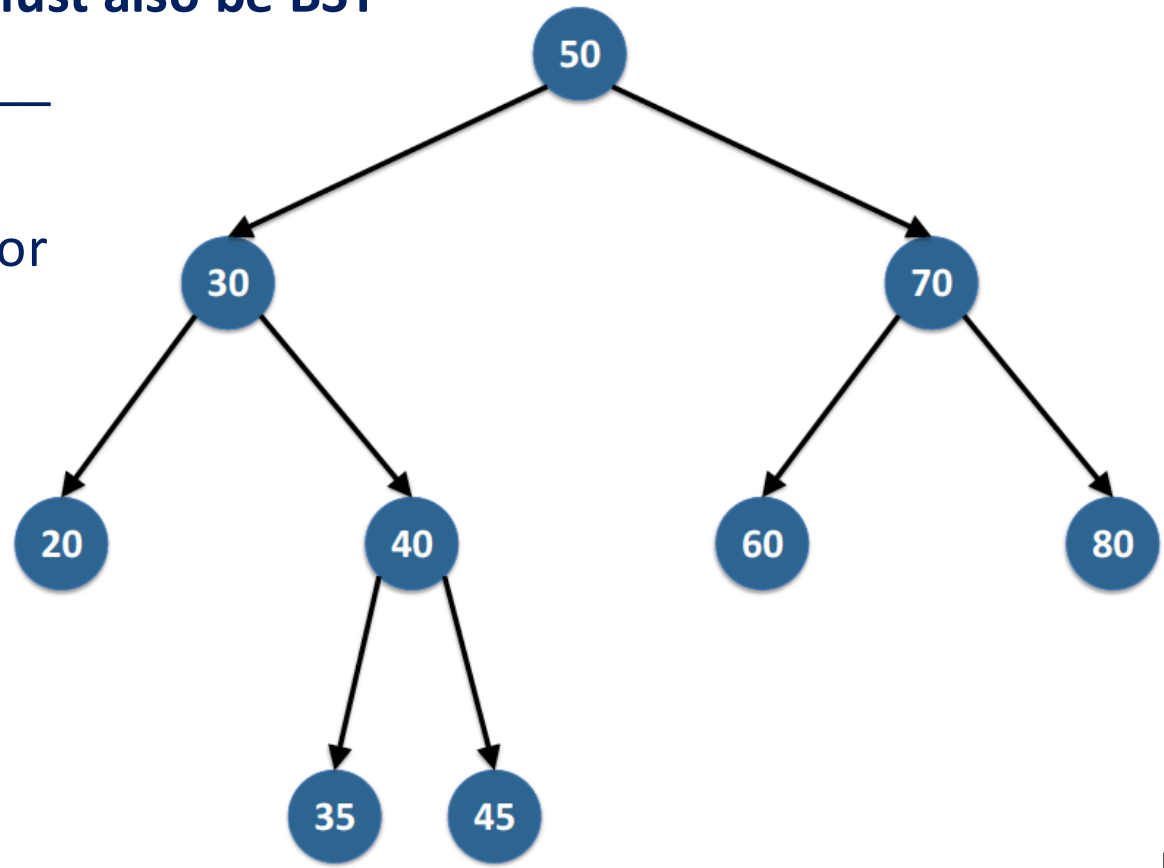
Depth-First Search

1. Pre-order: **root**, left, right
2. In-order: left, **root**, right
3. Post-order: left, right, **root**



Binary Search Tree

- Data of nodes on the **left** subtree is _____ than the data of parent node
- Data of nodes on the **right** subtree is _____ than the data of parent node
- Both left and right **subtrees must also be BST**
- Data in each node is _____



What is the sequence of access for

1. pre-order traversal?

2. in-order traversal?

3. post-order traversal?

http://visualgo.net/bst.html

Traverse a BST

```
typedef struct btNode node;
struct btNode{
    int data;
    node *left;
    node *right;
};
```

```
void preorder(node *root){



}
void inorder(node *root){



}
void postorder(node *root){



}
```

Find a Node in BST

```
node *search(node* root, int data){
   /*base cases: 1)"root" is NULL; 2) found node*/

   /*recursive cases: search left and right subtrees*/

}
```

```
node *find_min(node *root){ /*find the smallest node in a BST*/




}
node *find_max(node *root){ /*find the largest node in a BST*/




}
```

Count the Number of Leaf Nodes in a BST

```
int leaf_count(node *root){
/*base cases: 1)"root" is NULL; 2)"root" is a leaf node*/




/*recursive cases: count the leaf nodes on the left and right subtrees;
                   total leaf count = left_count + right_count*/




}
```

Calculate the Height of a BST

```
int height(node *root){
/*base cases: 1)"root" is NULL; 2)"root" is a leaf node*/

/*recursive cases: calculate the height of the left and right subtrees;
                   height = 1 + max(left_height, right_height)*/

}
```