# Trees

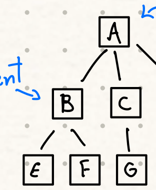
A tree is an abstract model of a hierarchical structure

root

Edge/link/arc

Parent

Node/Vertex

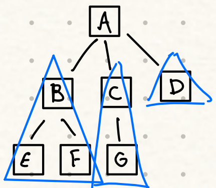


G is a descendant of A
A is an ancestor of G

Siblings
& children of B

Internal Node: with at least 1 child (A, B, C)
External Node / leaf: No children (E, F, G, D)
Degree of Node: Number of children

# Subtrees



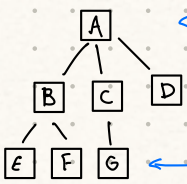
All trees, each child is the root of its own subtree, and reaches the base case

Subtree: Any node and all its descendants

## Recursive Definition:

Base Case: A tree is 1 node
Recursive step: A tree is a node whose children are roots of subtrees

# Terminology



Depth/level = 0

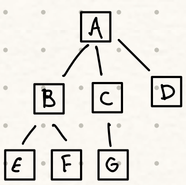
Depth/level = 1

Depth/level = 2

Height of tree: Max depth of any node

# Traversals

## Preorder Traversal:

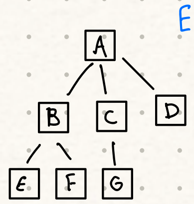


ABEFCGD

```
preorder(r)
  if node != null
    visit(r)
    Preorder(r.left)
    Preorder(r.right)
```

## Postorder:

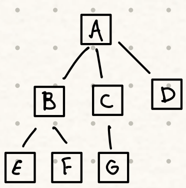
EFBGCDA



```
postorder(r)
  if node != null
    Postorder(r.left)
    Postorder(r.right)
    visit(r)
```

DFS

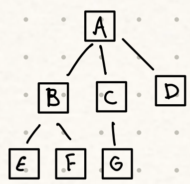
## Inorder Traversal:



BEFACGD

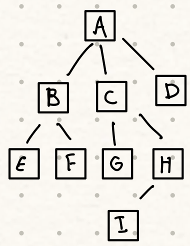
```
inorder(r)
  if node != null
    inorder(r.left)
    visit(r)
    inorder(r.right)
```

## Levelorder:



ABCDEFG

## Computing height of tree:




```
height(r)
  if r is leaf return 0      } C1
  else {
      maxheight = 0
      for each child c of r do {
   C3 {  h = height(c)
         if h > maxheight then
             maxheight = h
      }
      return maxheight + 1
  }
```

$C_2$

## Compute Complexity:

1. Compute complexity ignoring recursive calls
   Leaf : $C_1$
   Internal Node : $C_2 + C_3 \text{ degree}(r)$

2. Compute number of recursive calls
   #calls = $n$, one per node

3. Combine 1 and 2 to find total operations

$$f(n) = \sum_{\text{leaves}} C_1 + \sum_{\text{internal node}} (C_2 + C_3 \text{ degree}(r))$$

$$= C_1 \cdot \underbrace{\#\text{leaves} + C_2 \cdot \#\text{internal nodes}}_{n} + C_3 \underbrace{\sum_{\text{internal nodes}} \text{degree}(r)}_{n-1}$$

$$= \cancel{C_1} \cdot \#\text{leaves} + \cancel{C_2} \cdot \#\text{internal nodes} + \cancel{C_3}\cancel{(n-1)}$$

$\therefore O(n)$

## Binary Trees

For a tree to be a binary tree it must satisfy:

- Internal nodes have $\leq 2$ children (exactly 2 for proper)
- The children are an ordered pair

Proper Binary trees

- # leaves = # internal nodes + 1

$$\#\text{leaves} = \frac{n+1}{2}$$

$$\#\text{internal} = \frac{n-1}{2}$$