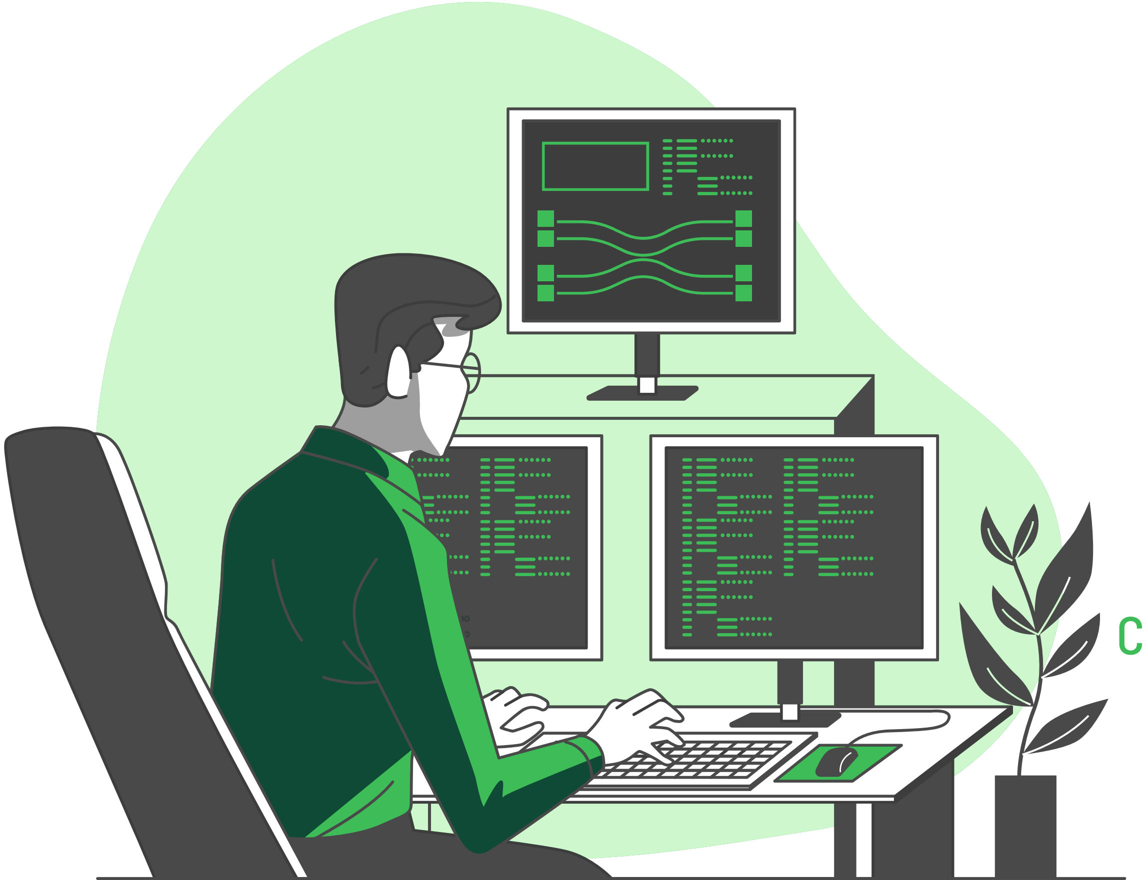
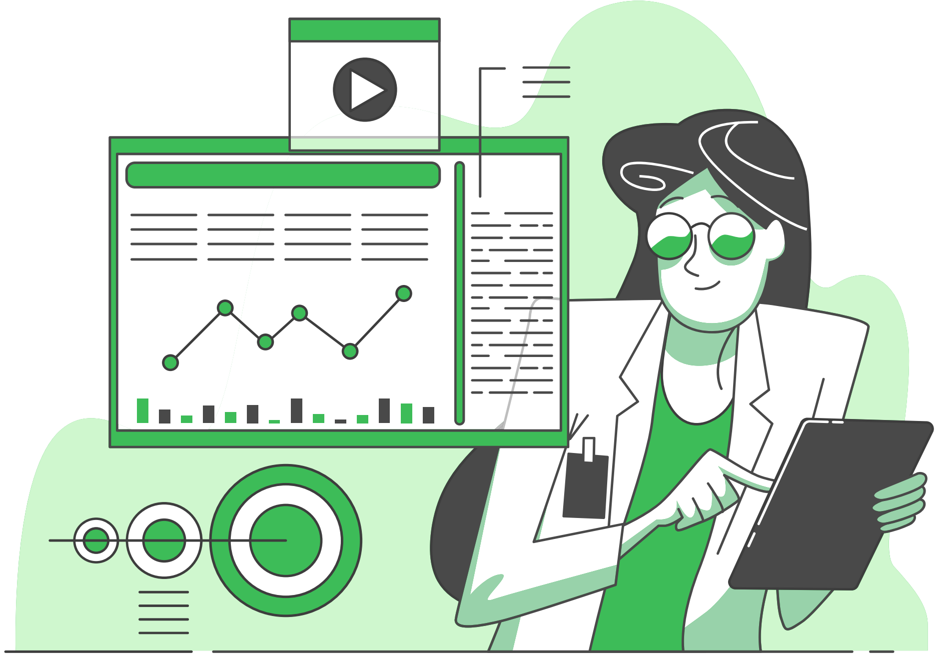


Tree

CS 251 - Data Structures and Algorithms

Note:
Slides complement the discussion in class

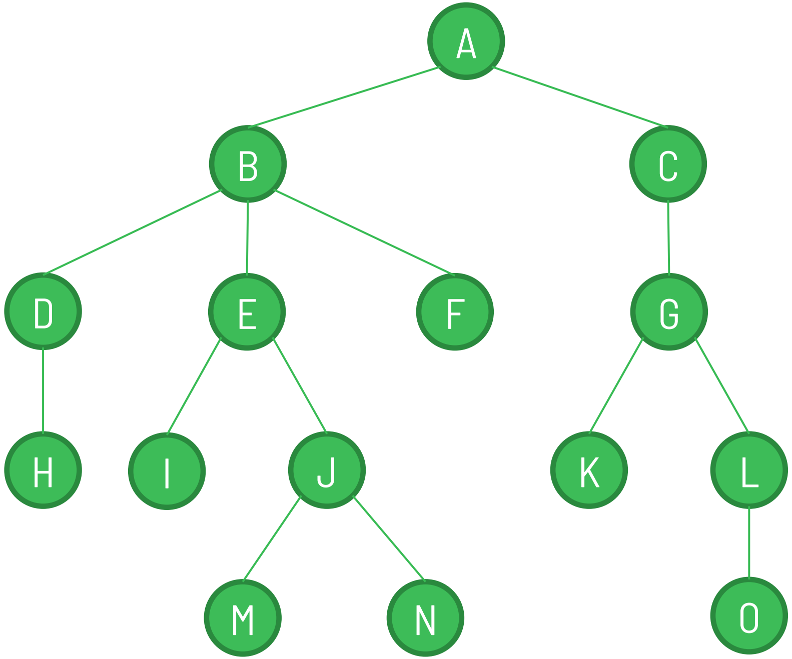
Table of Contents



01
Tree

Dynamic non-linear data structure

Tree



Dynamic Non-Linear Data Structure

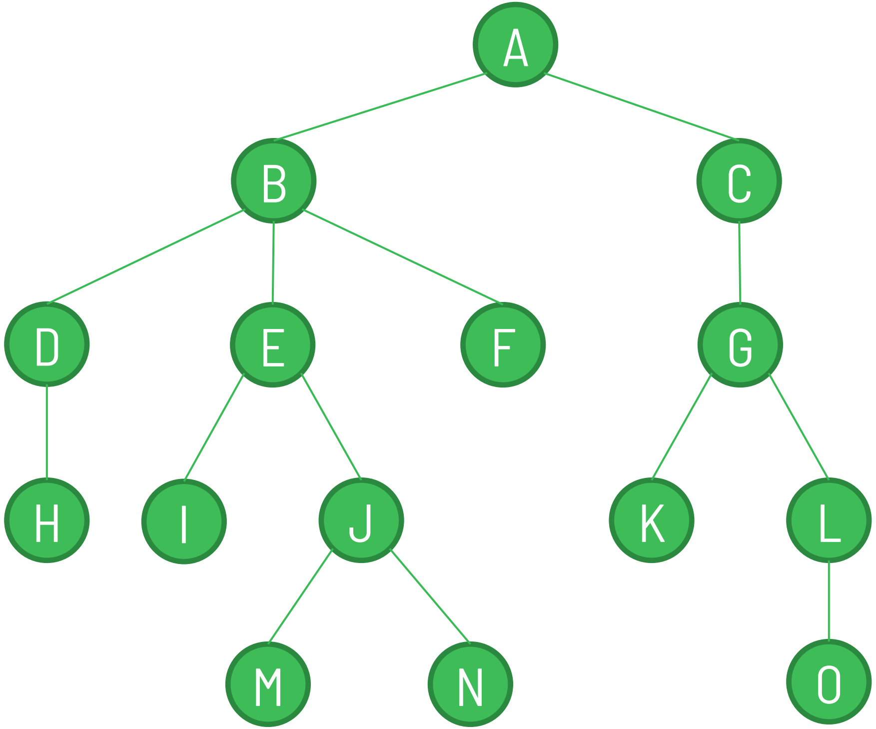
Access point through the root (i.e., pointer to the top-most node of the tree)

Each node may have none to many children.

Recursive: A tree is made of subtrees.

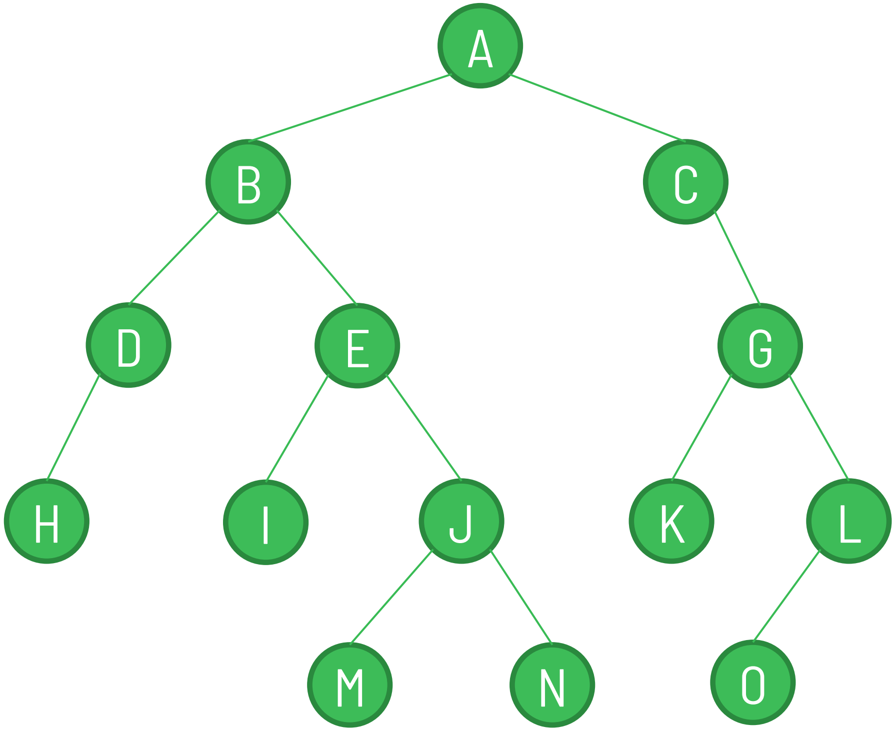
Search? $O(\log(n))$ under reasonable assumptions. Otherwise $O(n)$.

Concepts & Operations



- size() = 15
- isEmpty() = false
- root() = A
- parent(D) = B
- grandparent(O) = G
- sibling(D) = {E, F}
- children(E) = {I, J}
- isInternal(G) = true
- isExternal(M) = true
- isRoot(L) = false
- isLeaf(K) = true
- height() = 4
- height(G) = 2
- depth(B) = 1
- depth(A) = 0

Binary Tree



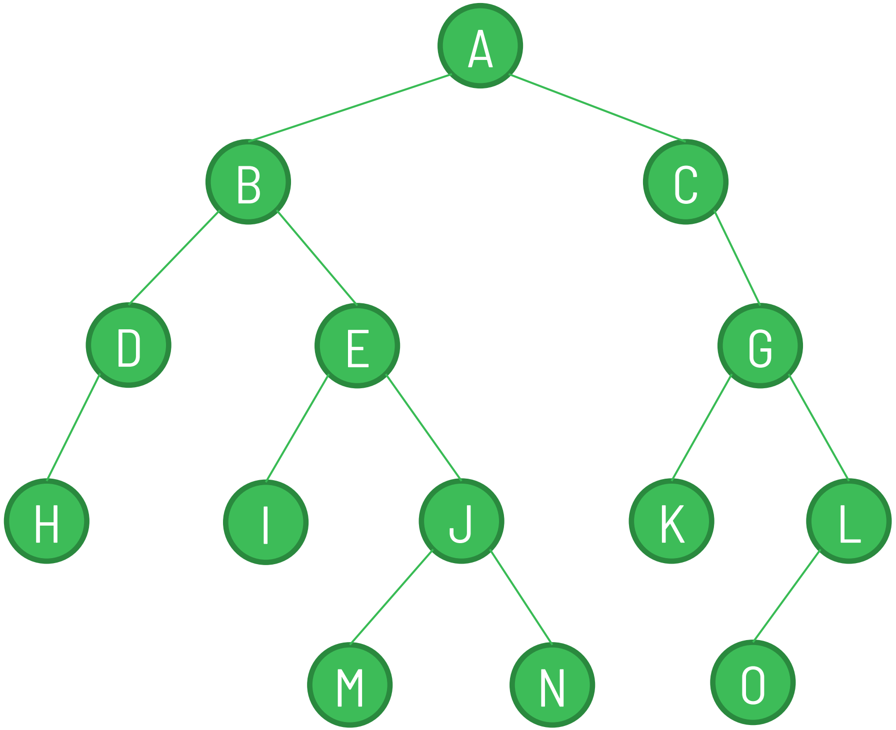
Every node has at most two children: left and right.

Example: G.left: K   and   G.right: L

Full binary tree: every node other than the leaves has two children.

Complete binary tree: every level, except possibly the last one, is completely filled, and all nodes are as far left as possible.

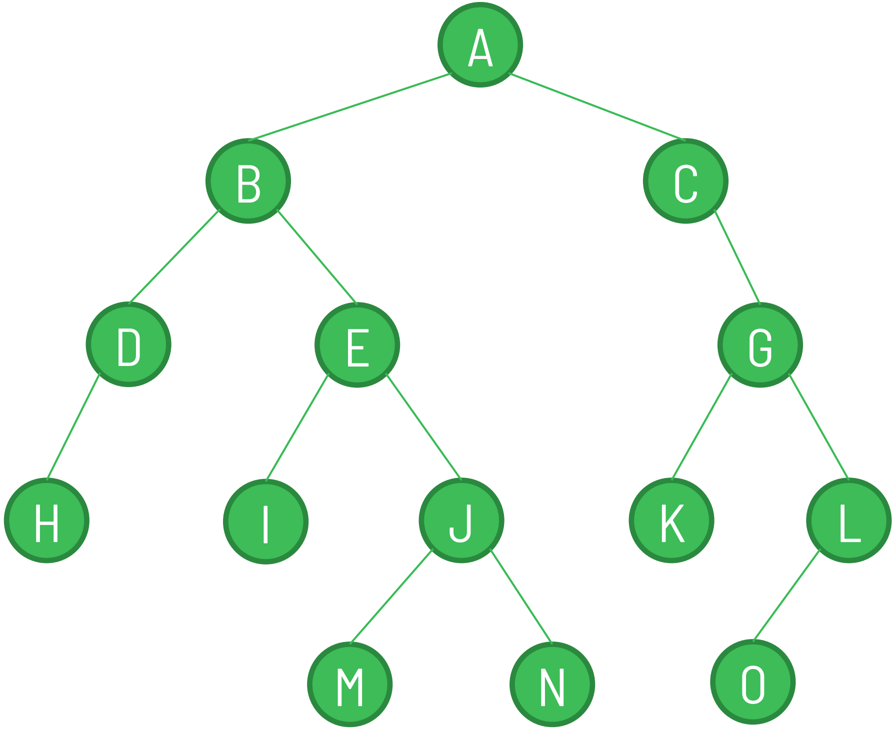
Height of a Binary Tree




```
algorithm height(x:node) → ℤ
   if x is null then
      return -1
   end if
   lh ← height(x.left)
   rh ← height(x.right)
   return 1 + max(lh, rh)
end algorithm
```

height(A) = 4

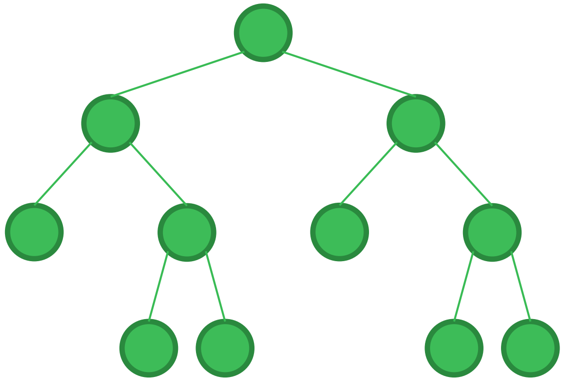
Size of a Binary Tree



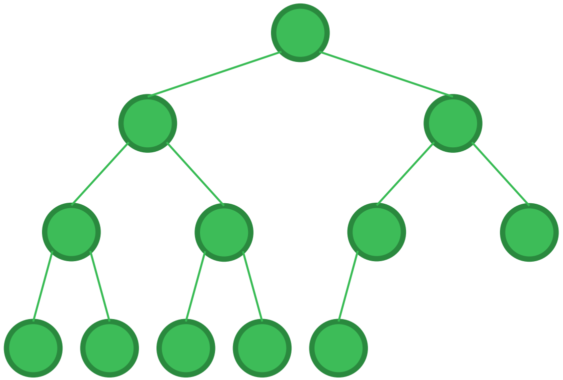

```
algorithm size(x:node) → ℤ≥0
    if x is null then
        return 0
    end if
    ls ← size(x.left)
    rs ← size(x.right)
    return 1 + ls + rs
end algorithm
```

size(A) = 14

Binary Tree Types

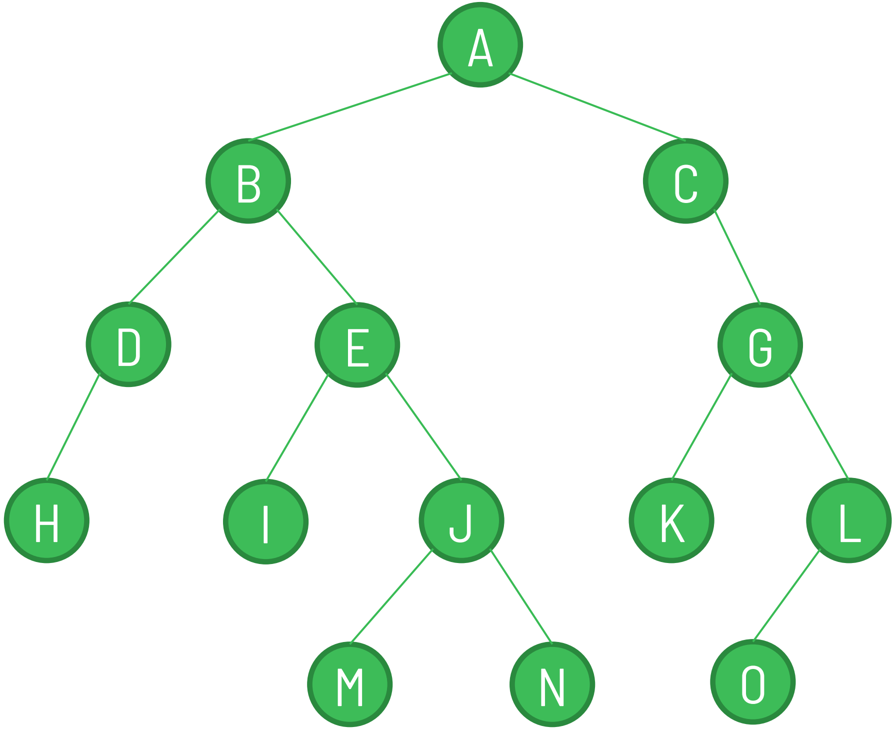


Full binary tree: Each node is either a leaf or has exactly two children.



Complete binary tree: All levels except possible the last are completely full, and the last one has all its nodes to the left side.

Max #nodes in a Binary Tree



Q: Max number of nodes at level l of a binary tree?
A: 2^l

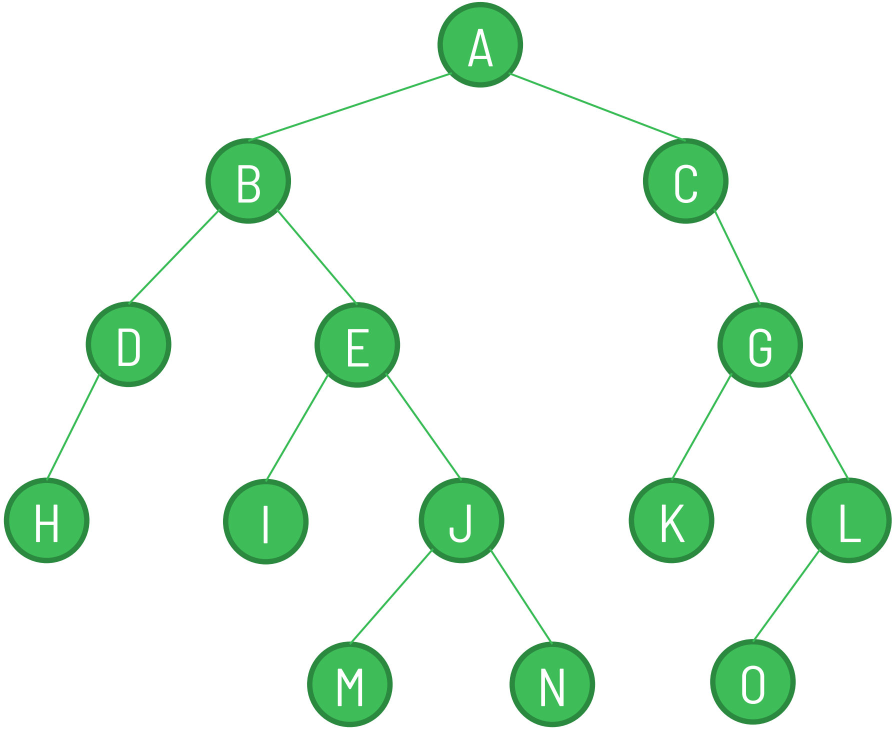
Proof: $P(l)$: The max number of nodes at level l of a binary tree is 2^l.

Show $P(0)$: The root is at level 0. Max number of nodes is $2^0 = 1$.

Assume $P(l)$ (i.e., the max number of nodes at level l is 2^l).

Show $P(l+1)$: Since in a binary tree every node has at most 2 children, the next level would have at most twice the number of nodes. That is, $2 \cdot 2^l = 2^{l+1}$

Max #nodes in a Binary Tree



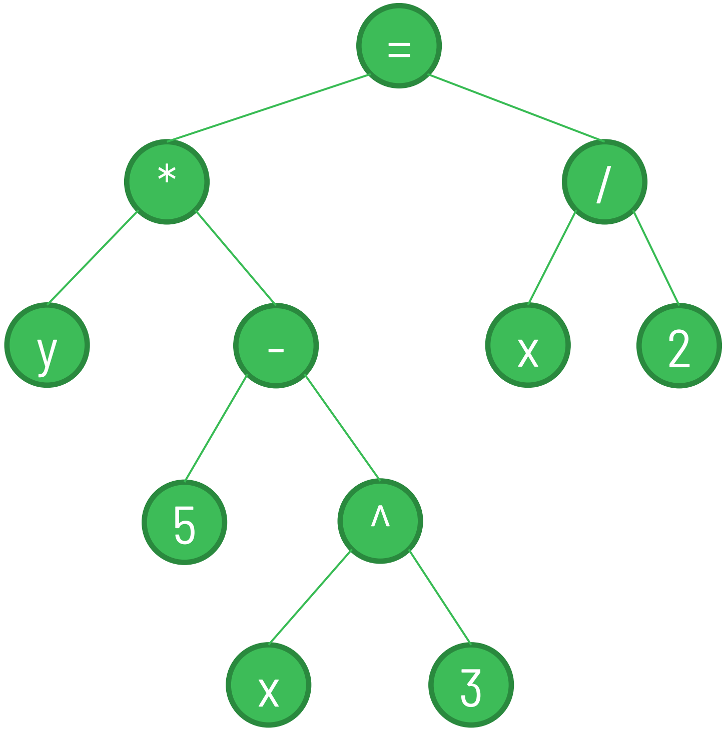
Q: Max number of nodes at level l of a binary tree?
A: 2^l

Q: Max number of leaves in a binary tree?
A: 2^h, where h is the height of the tree.

Q: Max number of nodes in a binary tree?
A:

$$\sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

Expression Tree



We can represent arithmetic expressions using binary trees:

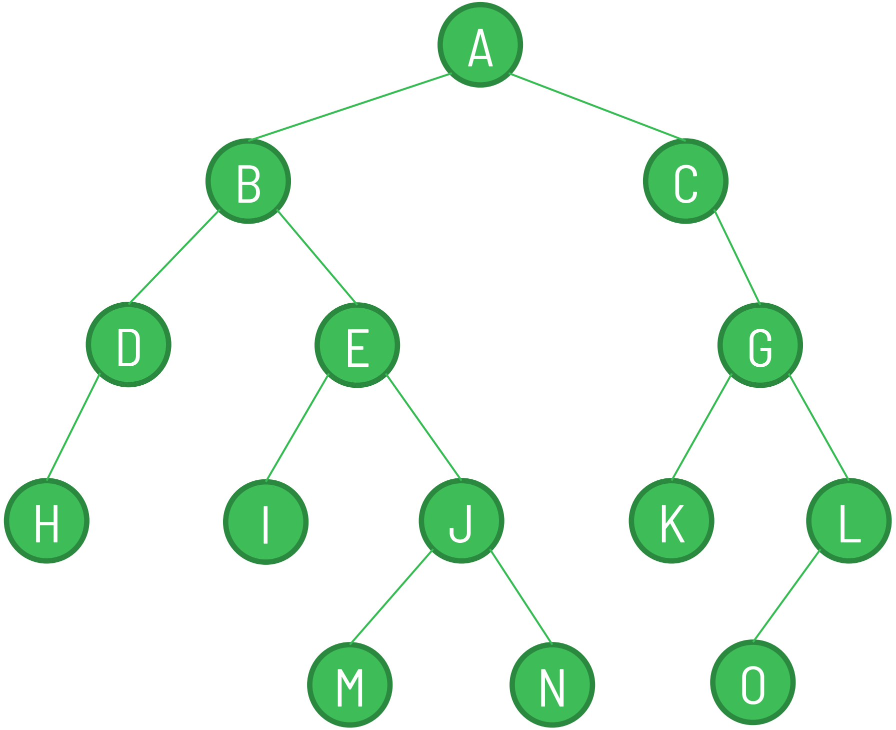
Example: Left tree represents:

$$y(5 - x^3) = \frac{x}{2}$$

Questions:

- How do we know the expression it represents?
- How do we build one?

Preorder Traversal



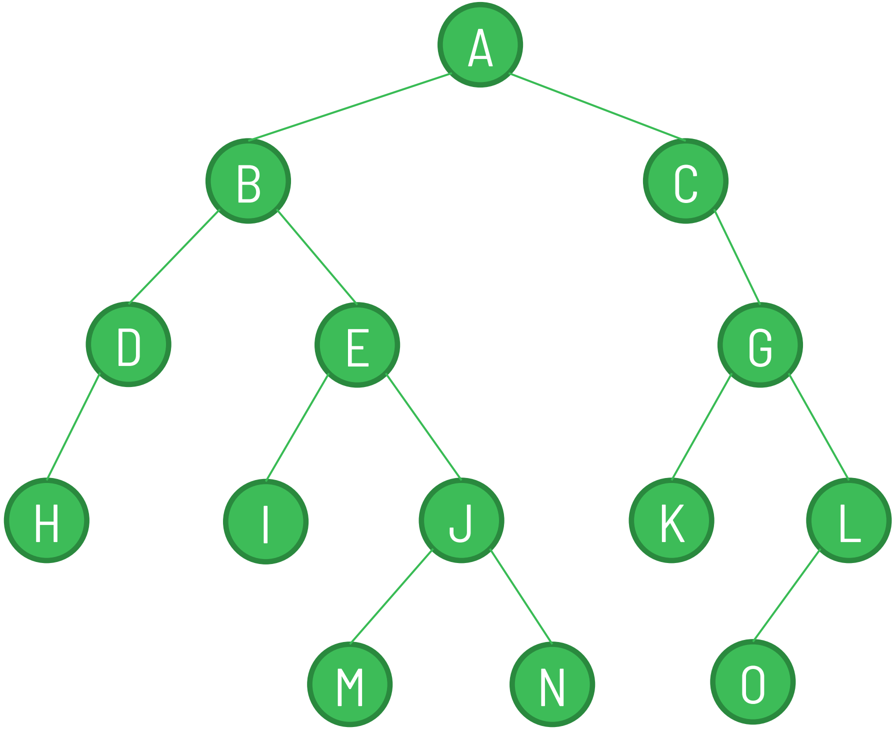
Visit a node before its descendants (NLR).

```
algorithm preorder(x:node)
   if x is null then
      return
   end if
   visit(x)
   preorder(x.left)
   preorder(x.right)
end algorithm
```

preorder(root) = A, B, D, H, E, I, J, M, N, C, G, K, L, O

Reverse Preorder (NRL) also exists.

Inorder Traversal



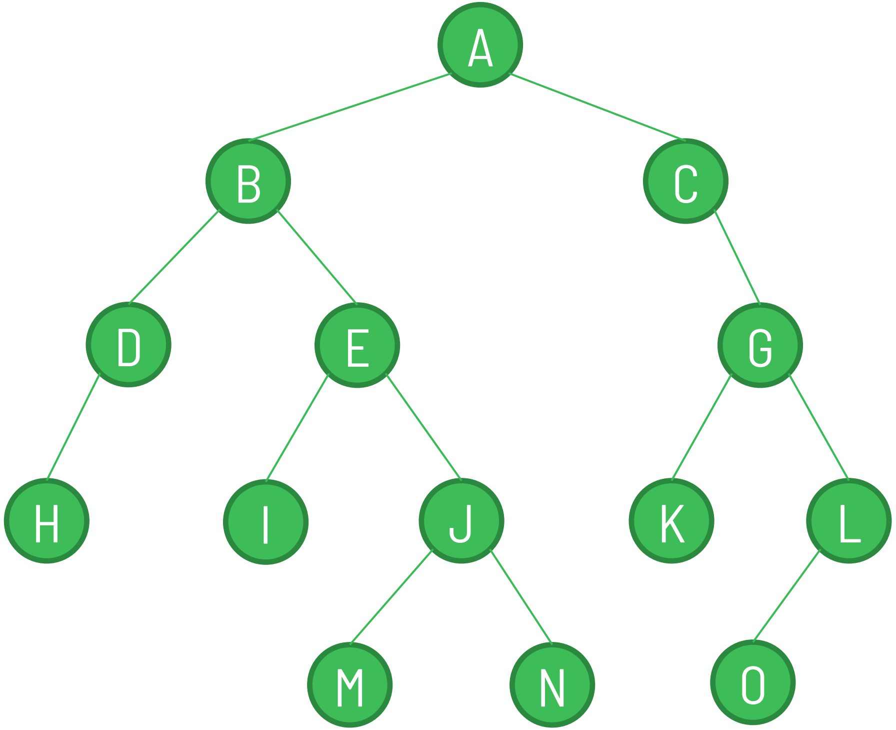
Visit a node after its left descendants and before its right descendants (LNR).

```
algorithm inorder(x:node)
   if x is null then
      return
   end if
   inorder(x.left)
   visit(x)
   inorder(x.right)
end algorithm
```

inorder(root) = H, D, B, I, E, M, J, N, A, C, K, G, O, L

Reverse Inorder (RNL) also exists.

Postorder Traversal



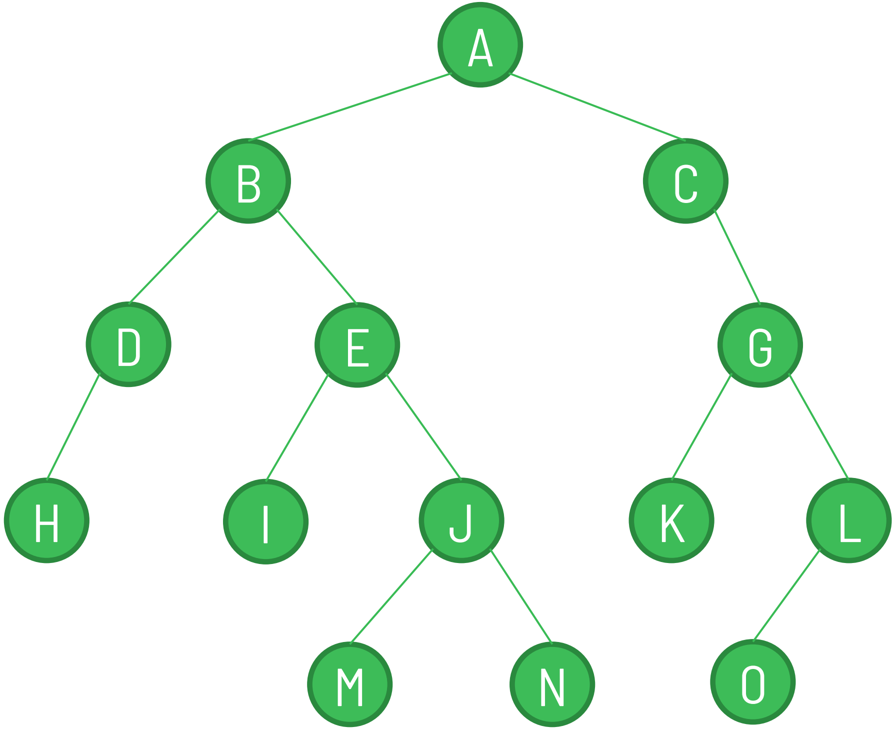
Visit a node after its descendants (LRN).

```
algorithm postorder(x:node)
   if x is null then
      return
   end if
   postorder(x.left)
   postorder(x.right)
   visit(x)
end algorithm
```

postorder(root) = H, D, I, M, N, J, E, B, K, O, L, G, C, A

Reverse Postorder (RLN) also exists.

Levels Traversal



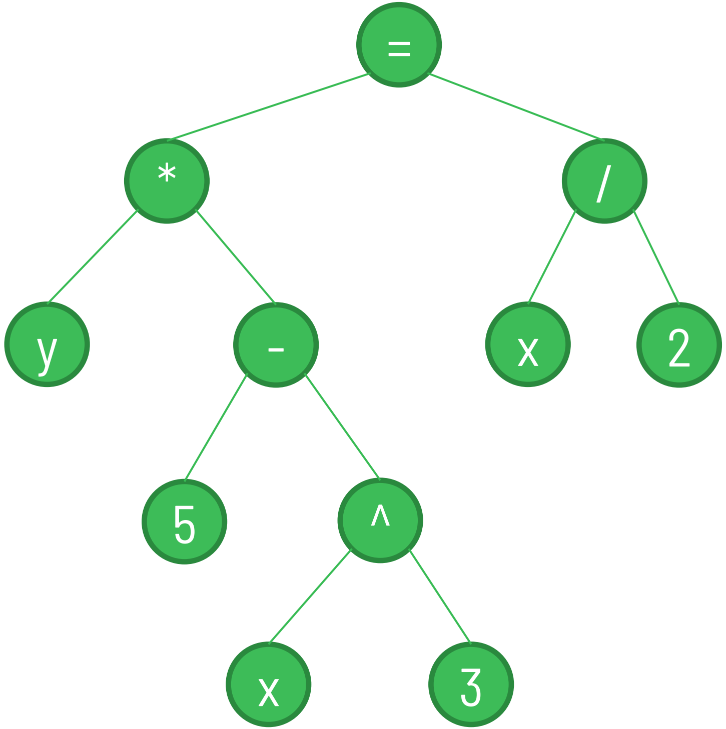
Visit the nodes by their height in the tree.

```
algorithm levels(x:node)
   if x is null then
      return
   end if
   let Q be an empty queue
   Q.enqueue(x)
   while Q is not empty do
      n ← Q.dequeue()
      visit(n)
      if n.left is not null then
         Q.enqueue(n.left)
      end if
      if n.right is not null then
         Q.enqueue(n.right)
      end if
   end while
end algorithm
```

levels(root) = A, B, C, D, E, G, H, I, J, K, L, M, N, O

This traversal is also known as **Breadth-First Search**.

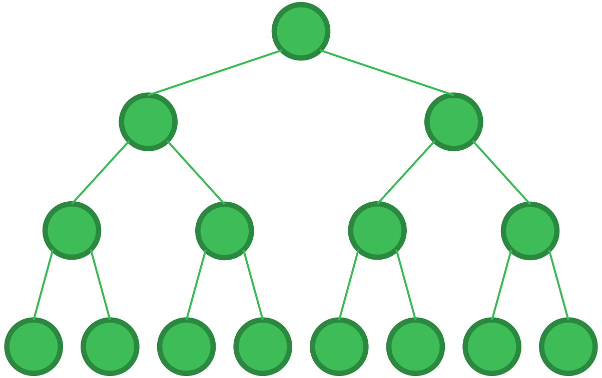
Expression Tree (Again)



Q: How do we know the expression it represents?
A: Use preorder (prefix notation) or inorder (infix notation, don't forget the parenthesis).

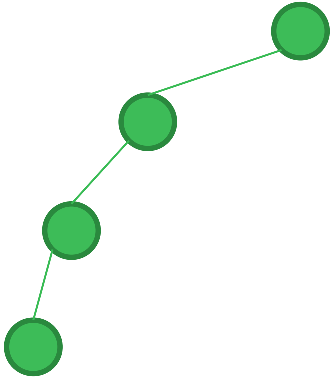
Q: How do we build one?
A: From prefix notation, adapt Dijkstra's algorithm for evaluating expressions.
tl;dr: Use two stacks (operands and operators). Create nodes and put them together accordingly.

Balancing Matters!



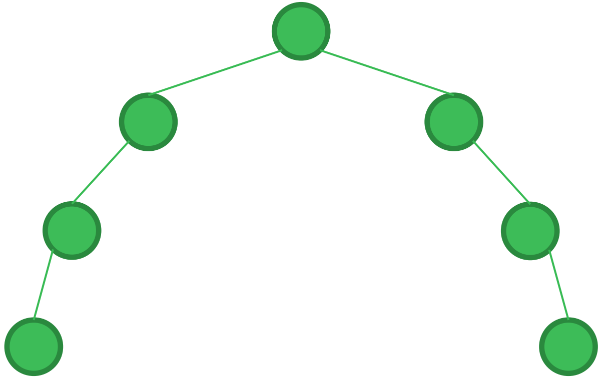
Balanced

$h \in \Theta(\log_2(n))$



Unbalanced

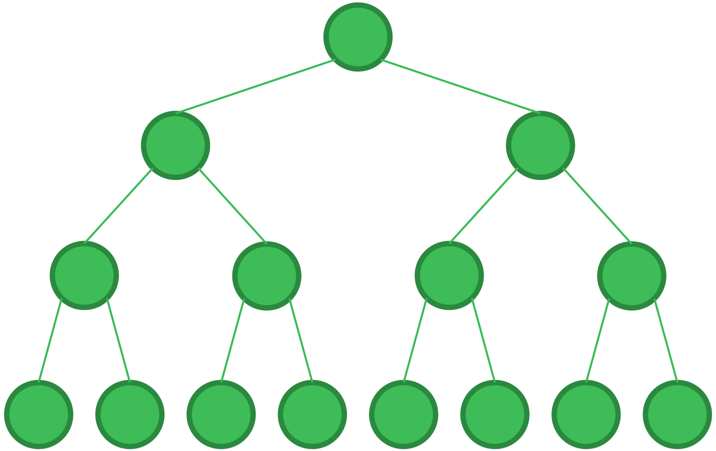
$h \in \Theta(n)$



Unbalanced

$h \in \Theta(n)$

Full Binary Tree Facts



Let T be a nonempty, full binary tree.

- If T has I internal nodes, the number of leaves is $I + 1$
- If T has I internal nodes, the total number of nodes is $2I + 1$
- If T has a total of N nodes, the number of internal nodes is $(N - 1)/2$
- If T has a total of N nodes, the number of leaves is $(N + 1)/2$
- If T has L leaves, the total number of nodes is $2L - 1$
- If T has L leaves, the number of internal nodes is $L - 1$

Proofs? Use induction on full binary trees.

Attention Span Not Found

Do you have any questions?

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik and illustrations by Stories