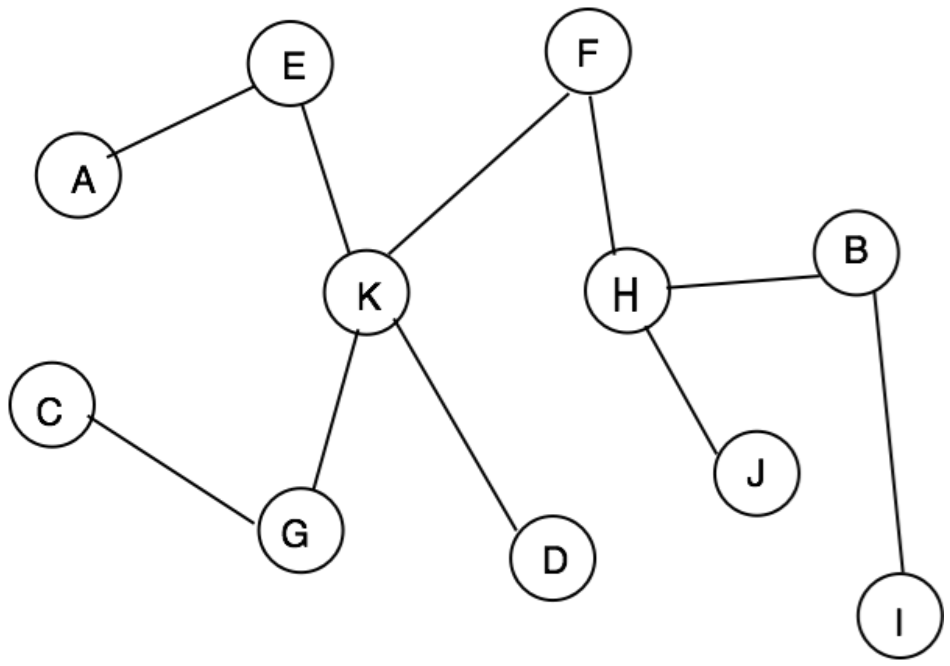
Tree

A **tree** consists of a set of nodes and a set of edges connecting pairs of nodes. A tree has the property that there is exactly one path (no more, no less) between any pair of nodes.

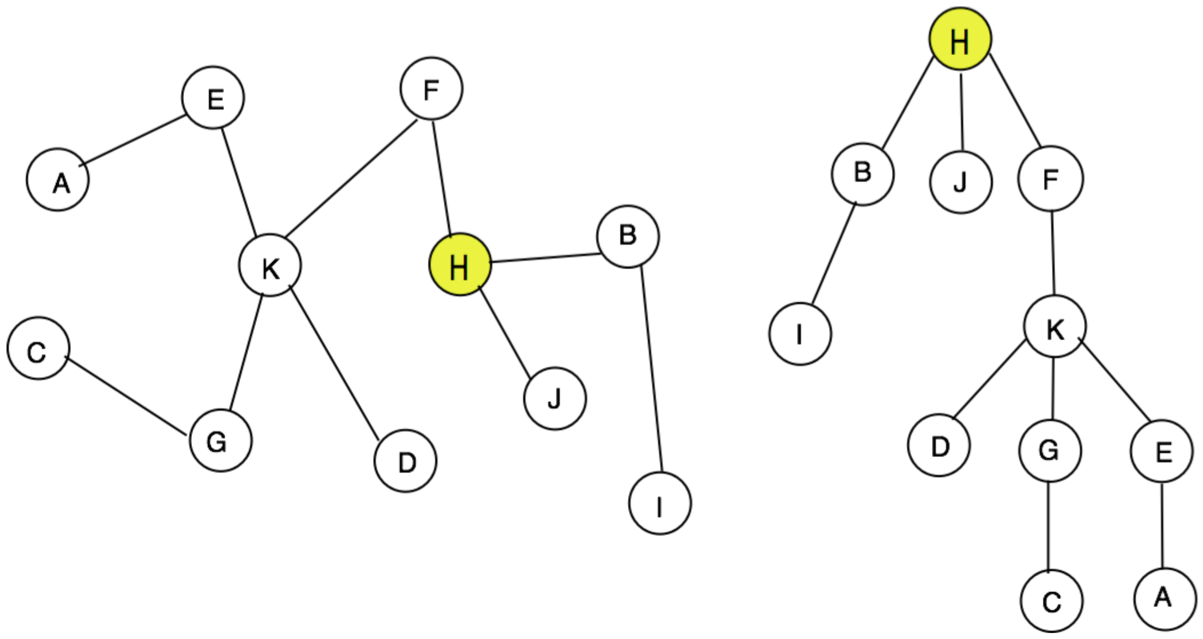


A **path** is a connected sequence of edges.

A tree topology is acyclic – there is no loop.

Rooted Tree

In a **rooted tree,** one distinguished node is called the root.



In the above figure, we chose H to be the root. Every node c, except the root, has exactly one parent node p, which is the **first node** after c on the path from c to the root. Node c is node p child.

So, K is the parent of D, and K is a child of F.

The root has no parent. A node can have any number of children.

Other definitions

A **leaf** is a node with no children.

An **internal node** is a non-leaf node

Siblings are nodes with the same parent.

The **ancestors** of a node d are the nodes on the path from d to the root. This includes node d.

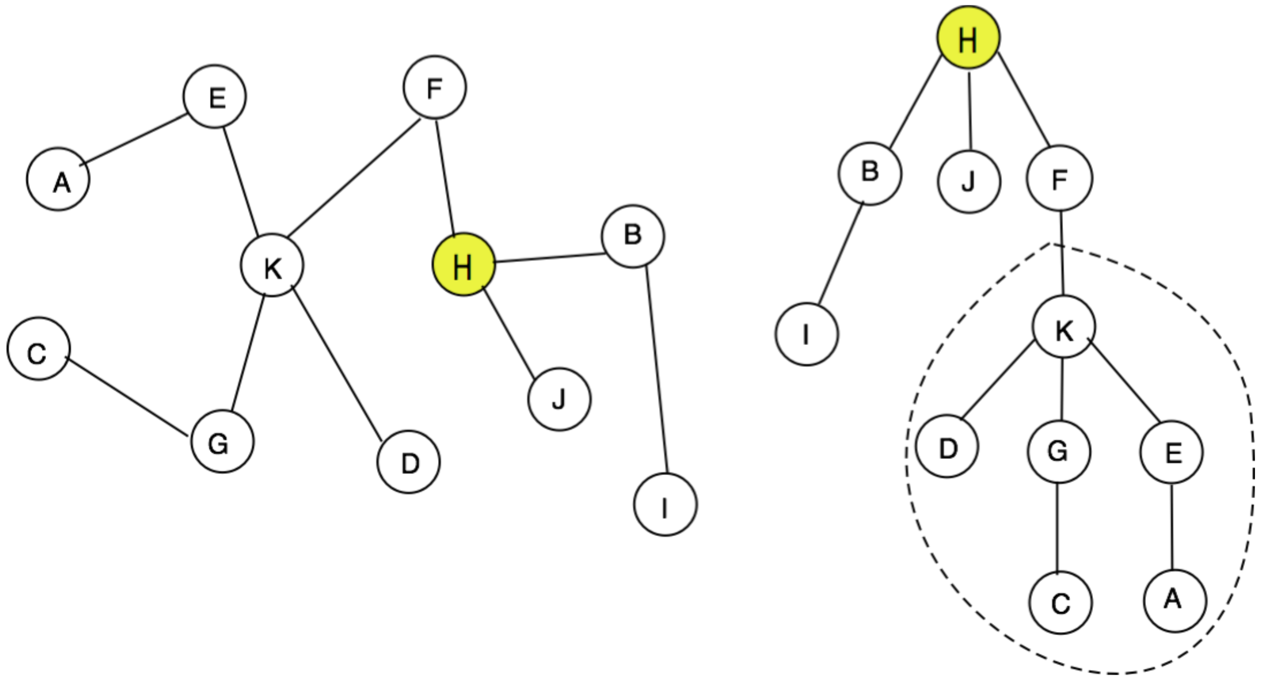
If **node a** is an ancestor of node d, then d is a **descendant** of a.

The **length** of a path is the number of edges in the path.

The **depth** of a node n is the length of the path from n to the root. (The depth of the root is zero.)

The height of a node n is the length of the path from n to its deepest descendant. (The height of a leaf node is zero.)

The height of a tree is the depth of its deepest node = height of the root.



What is the depth of node D?

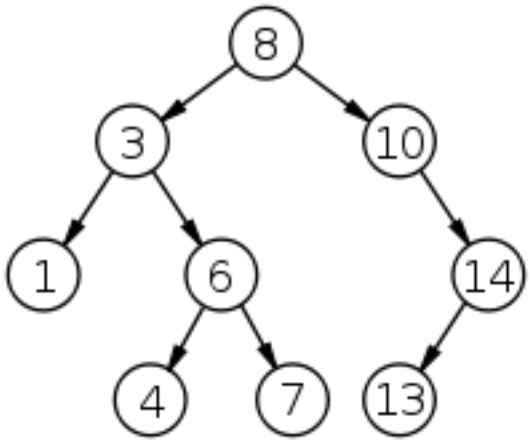
What is the height of K?

What is the height of the rooted tree?

The subtree rooted at node n is the tree formed by n and its descendants. What is the subtree rooted at node K?

Binary tree

A **binary tree** is a tree in which no node has more than two children, and every child is either a left child or a right child even if it is the only child its parent has.



(Source: Wikipedia)

A full binary tree is one in which every internal node has two children.

Is the above tree a full binary tree?

Tree data structure

A tree node has three references:

1. The item that it stores
2. Its parent node
3. The list of children

```
public static class Node<T> {
         private T data;
         private Node<T> parent;
         private List<Node<T>> children;
}
```

Another option is that the parent only stores information about the first child. Each child also links with the next sibling. Thus the siblings are directly linked

```
     Public class Node {
          Object item
          Node parent;
          Node firstchild;
          Node NextSibling;
     }
```

Sample methods

Depth of a node p

```
public int depth(node p) {
      if (isRoot(p))
            return 0;
      else
            return 1+depth(parent(p))
}
```

Height of a subtree

```
public int height(node p) {
      int h = 0
      for (node c: children(p))
            h = Math.max(h, 1+ height(c);
}
```

The Tree ADT

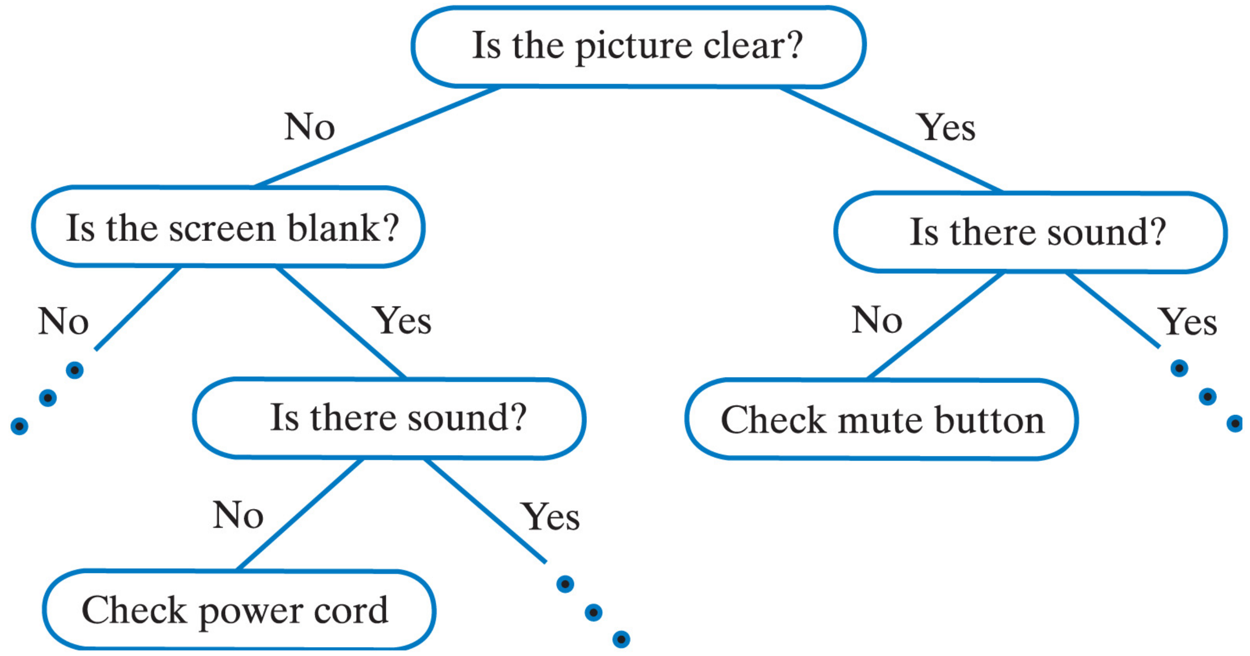
A tree ADT can have the following methods.

getElement (node p)	Return the object stored at node p
root ( )	Returns the root node
children (node p)	Returns the children of node p
parent (node p)	Returns the parent of node p
isLeaf (node p)	Returns true if node p is a leaf
isRoot (node p)	Returns true if node p is the root
size (node p)	Returns the number of nodes

and many more …

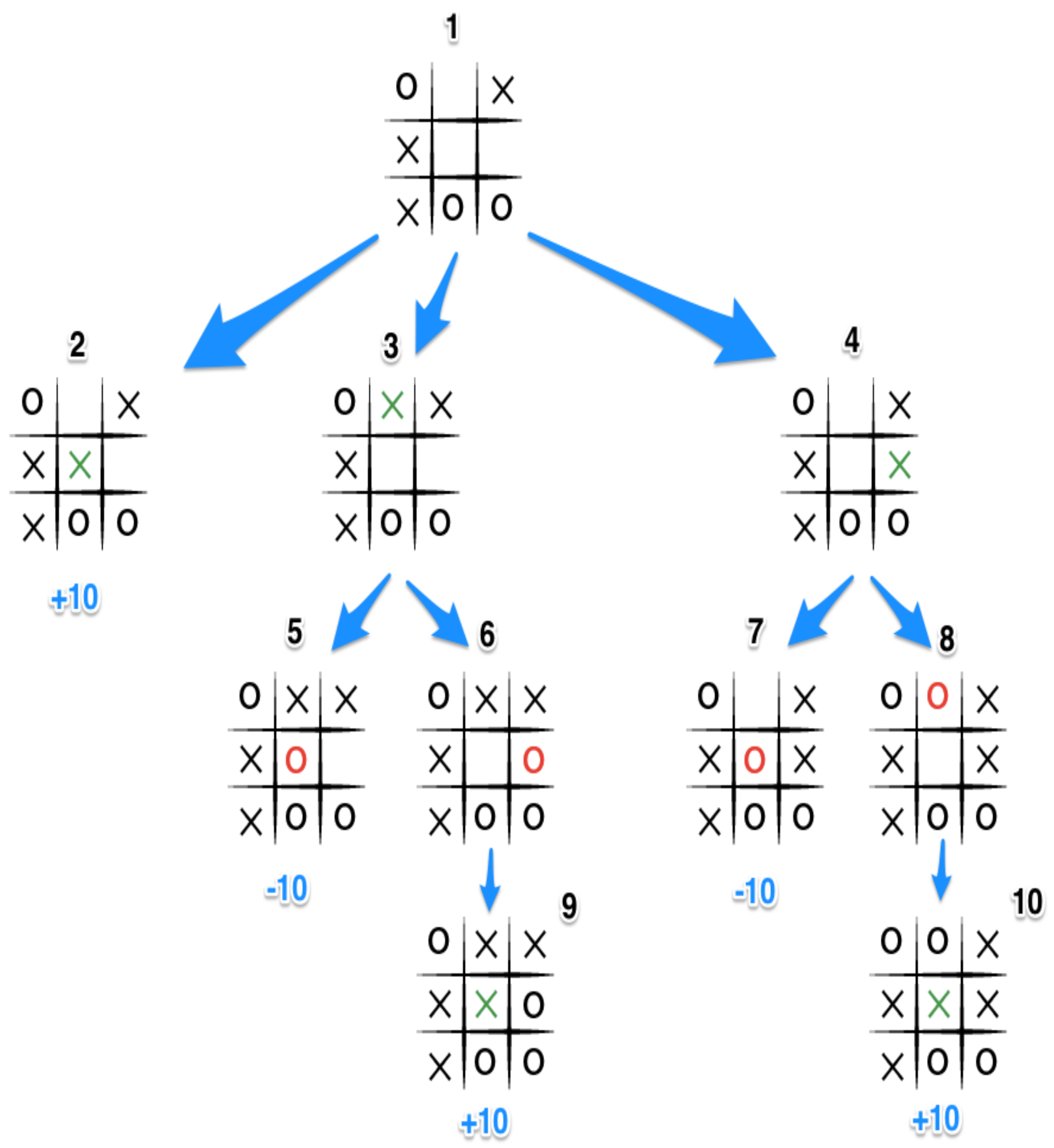
Binary tree as Decision Tree

`Troubleshooting of a TV set.`



A decision tree is a tree in which each internal node contains a question or an option that has a finite number of responses. When there are two possible responses, the tree is a binary decision tree. Nodes that are conclusions are leaf nodes.

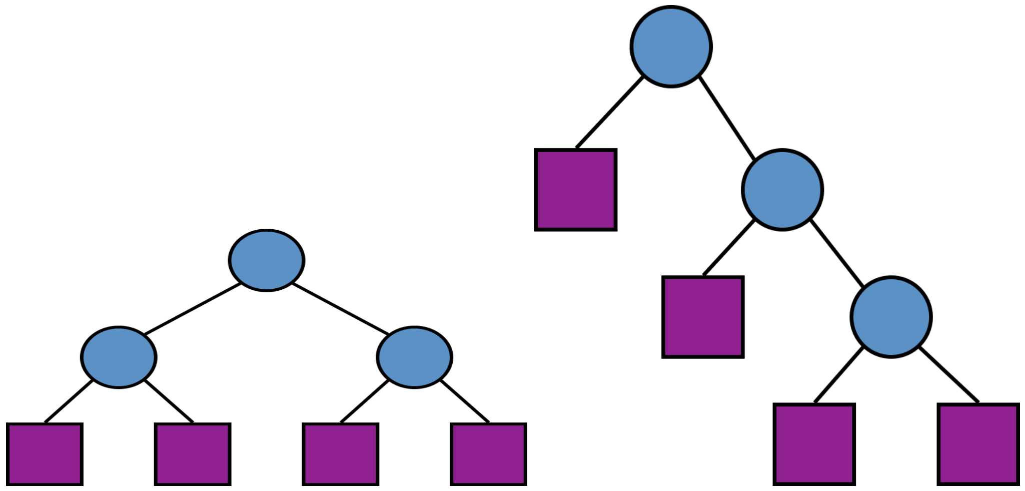
A game tree (not a binary tree)



Properties of a full binary tree

Parameters

$\boldsymbol{n}$	number of nodes
$\boldsymbol{e}$	number of leaf nodes
$\boldsymbol{i}$	number of internal nodes
$\boldsymbol{h}$	height



Properties (How will you show these are true?)

$e = i + 1$

$n = 2e - 1$

$h \leq i$

$h \leq (n - 1)/2$

$e \leq 2^h$

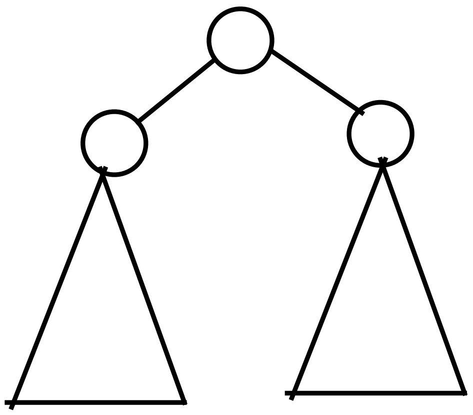
$h \geq \log_2 e$

$h \geq \log_2 (n + 1) - 1$

A Recursive Definition of a Binary Tree

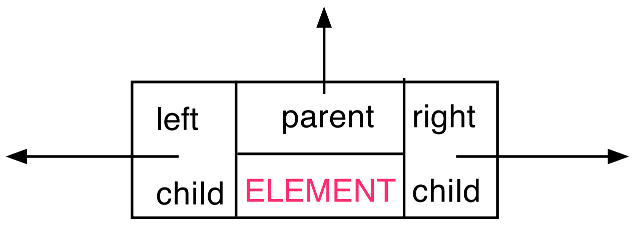
A full binary tree is

- A single node is a full binary tree
- A new root node that is connected with the roots of two full binary trees is also a full binary tree



Implementing a Binary Tree

Use a linked structure



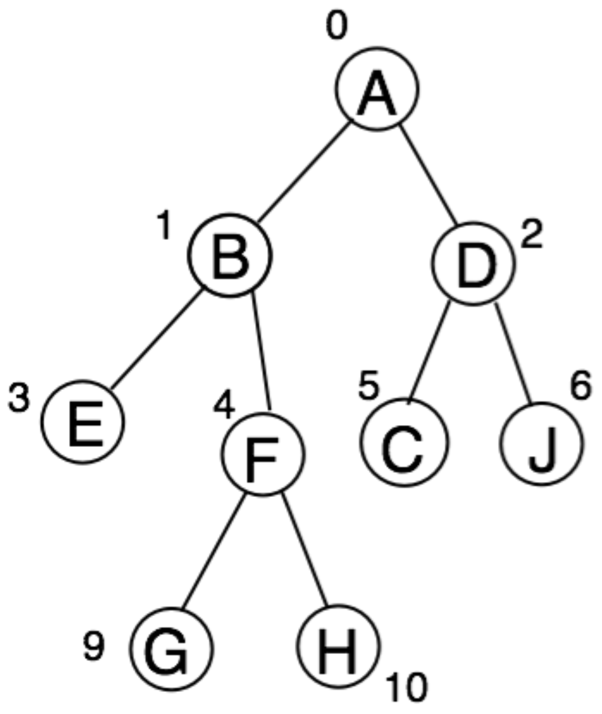
A single node

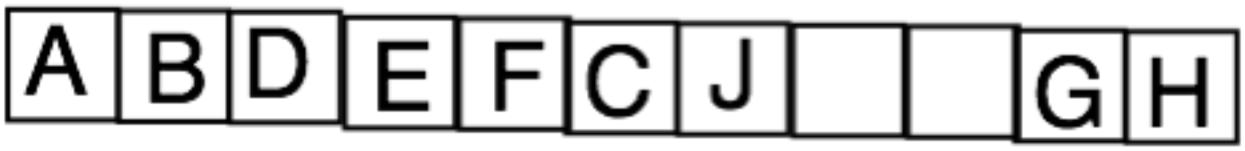
Now recall how a linked list was implemented.

Running times

Method	**Running time**
isEmpty	O(1)
Root, parent, children	O(1)
Depth (p)	$O(d_p+1)$
Height	O(n)

Array-based representation





0 1 2 3 4 5 6 9 10

Rank (root)=0

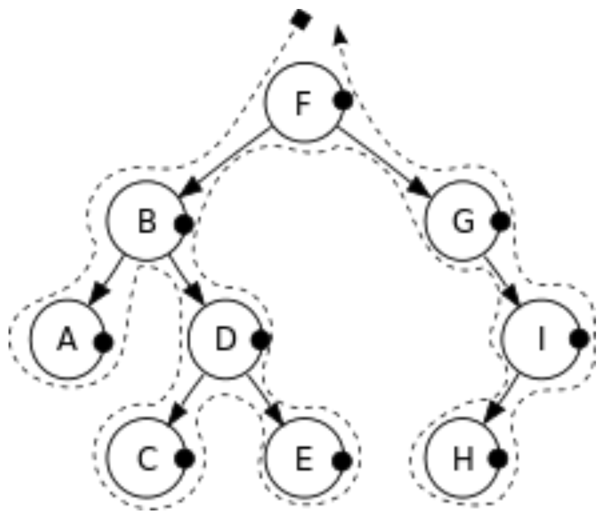
Rank (left child) = 2 x rank of parent+1

Rank (right child) = 2x rank of parent + 2

Traversal Algorithms of a Binary Tree

Visit each node of a tree in an organized manner.

PostOrder Traversal

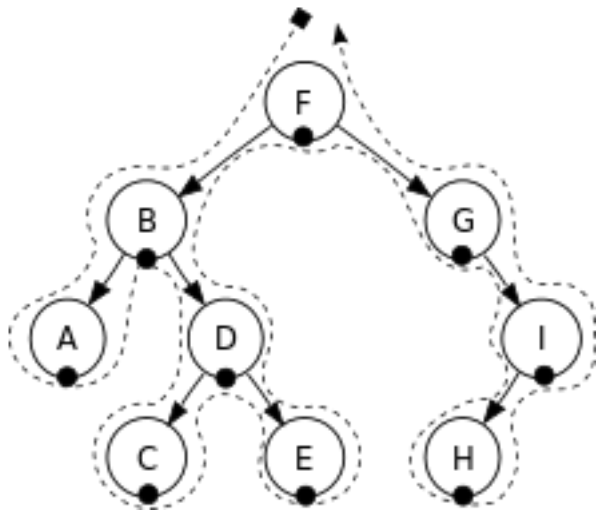


(Figure source: Wikipedia)

Order: A, C, E, D, B, H, I, G, F

```
Algorithm postOrder(v)
    for each child w of v
        postOrder (w)
    visit(v)
```

InOrder Traversal



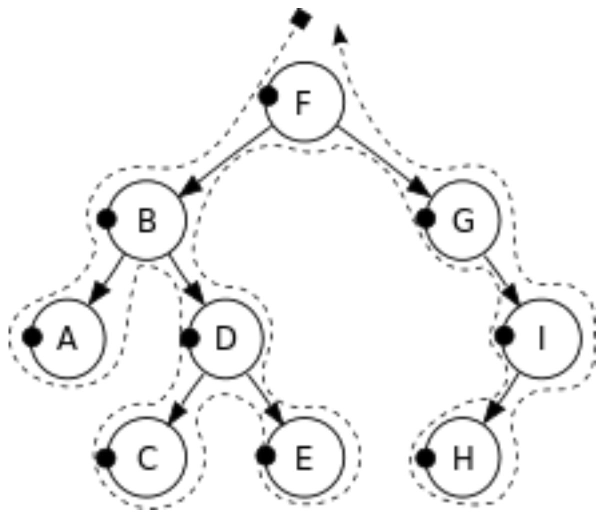
(Figure source: Wikipedia)

Order: A, B, C, D, E, F, G, H, I

```
Algorithm inOrder(v)
    if left (v) ≠ null
        inOrder (left (v))
    visit(v)
    if right(v) ≠ null
        inOrder (right (v))
```

We follow this in writing mathematical expressions.

Preorder traversal



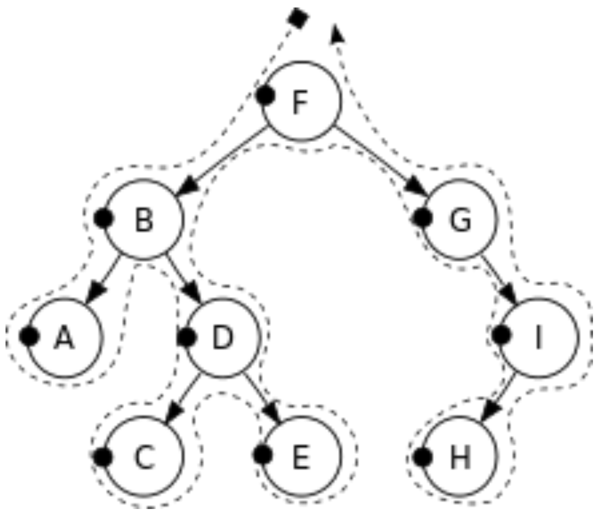
(Figure source: Wikipedia)

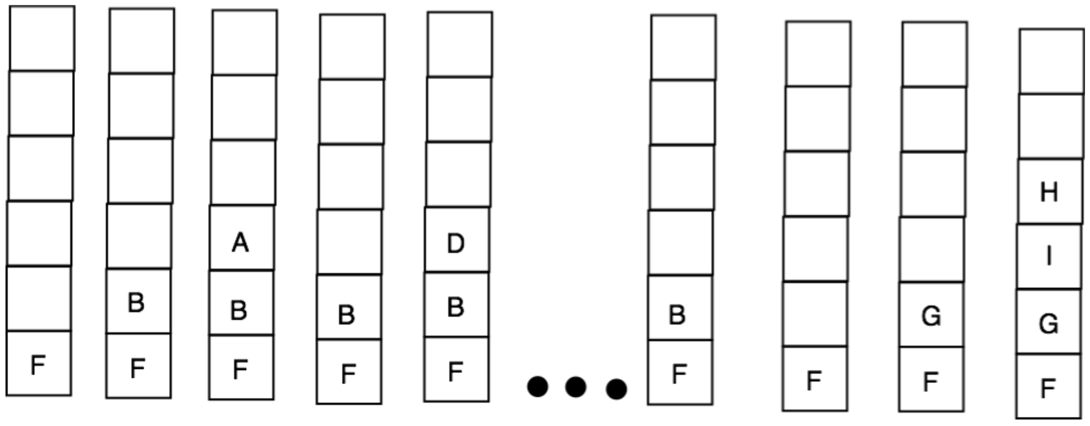
Order: F, B, A, D, C, E, G, I, H

Depth-first traversal and Breadth-first traversal

Depth-first traversal

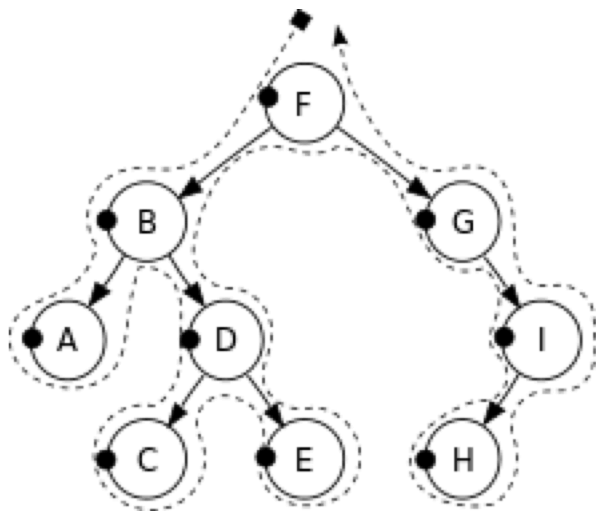
Pre-order traversal leads to a **depth-first traversal**. One can use a **stack** for generating this order.

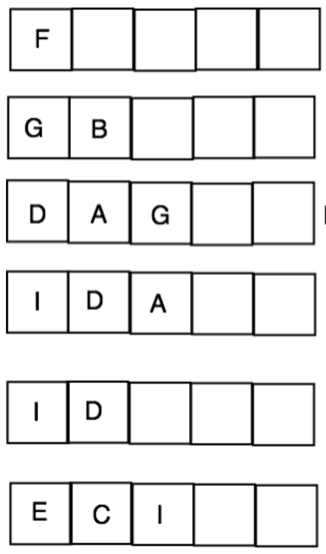




Breadth First traversal

Breadth-first order of traversal can be generated using a queue.





F dequeued, and its children are enqueued

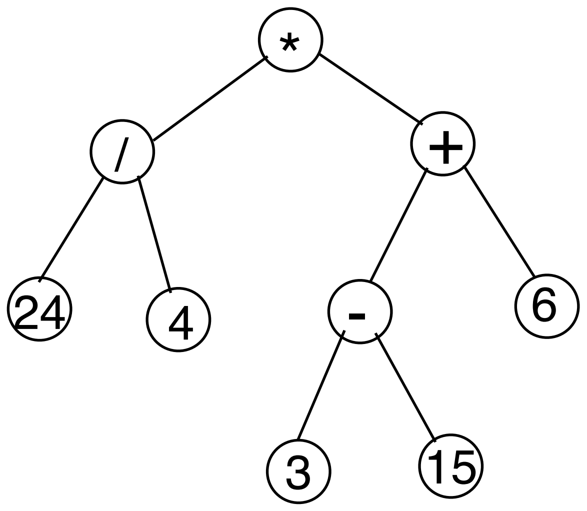
B dequeued, and its children are enqueued

G dequeued, and its child is enqueued

A dequeued, no child

D dequeued, and its child is enqueued

Representing Expressions



InOrder traversal:	(24/4) * ((3-15) + 6))
PostOrder traversal:	`24 4 / 3 15 - 6 + *`
	a.k.a. Reverse Polish Notation, no need for parentheses

How can you evaluate an expression presented in the Reverse Polish Notation? Use a single stack.

Evaluating an expression in the Reverse Polish Notation

Consider the input string as a stream of tokens.

24 4 / 3 15 - 6 + *

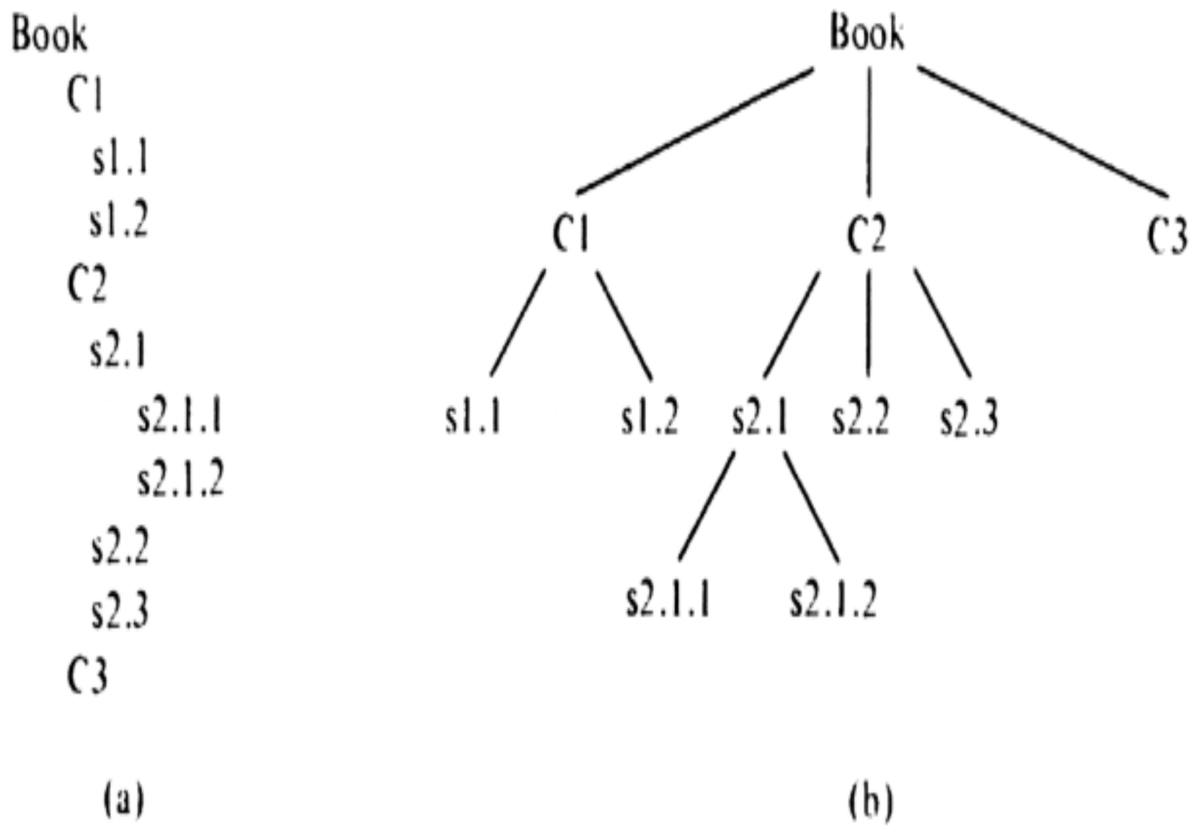
Assume that the input is a meaningful one.

1. If the token is a value, then push into the stack

2. If the token is an operator, then pop k values from the stack, (where k is the number of operands for that operator), apply it on the popped values, and push the result into the stack

3. After the last token, the stack will contain only one element, which is the result.

Consider the following tree showing the composition of a book.



What kind of tree traversal is this?