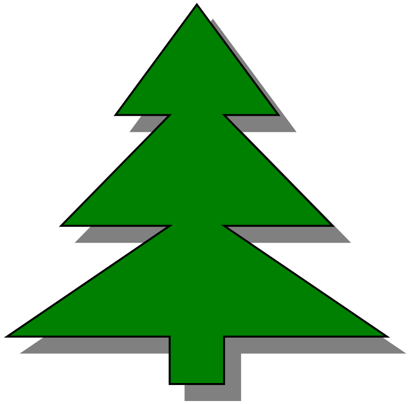
# Trees

CSIS 2226

# Trees

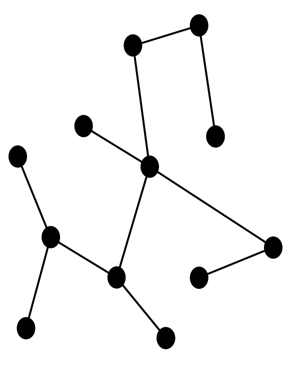


# Trees?

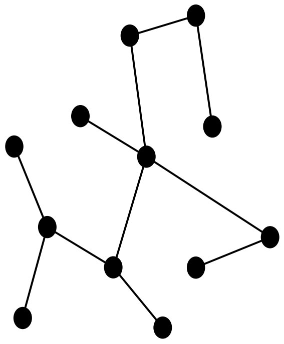
- Not those, but sort of….
- Our trees will have roots just like those
- Our trees will have branches just like those
- Our trees will have leaves just like those

# Tree: defined

- **Definition:** A tree is a connected undirected graph with no simple circuits.
- **Theorem:** An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices



## So, what is a tree?



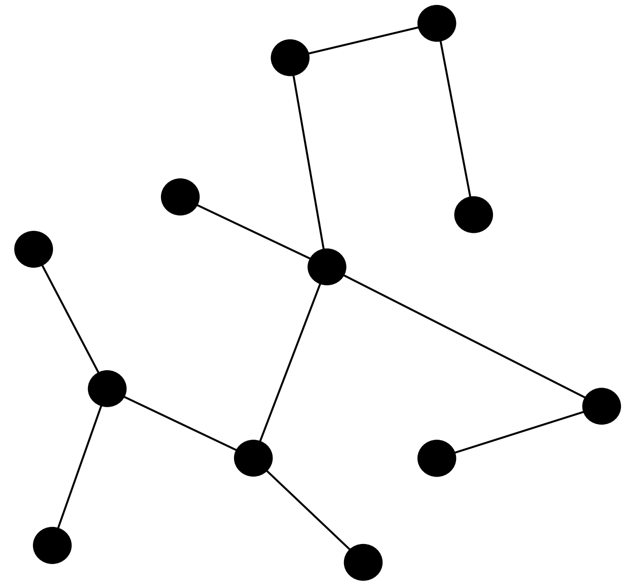
A tree is a connected undirected graph with no simple circuits

A tree is a connected graph with n-1 edges

A tree is a graph such that there is a unique simple path between any pair of vertices

All of the above say the same thing!

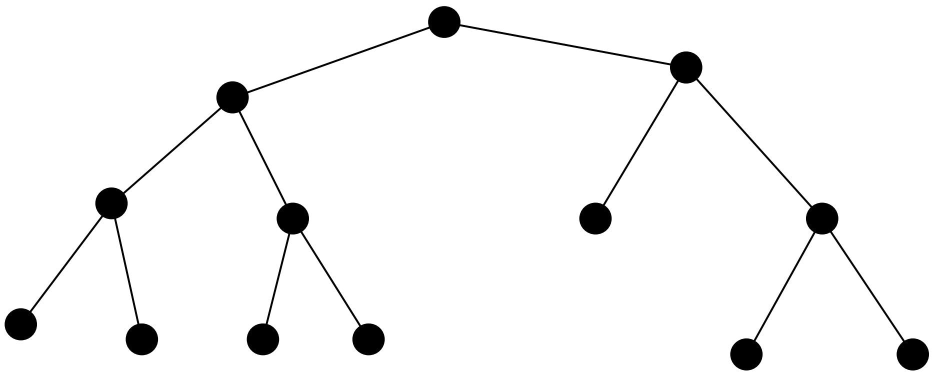
An unrooted tree



Kinds of nodes/vertices in a tree

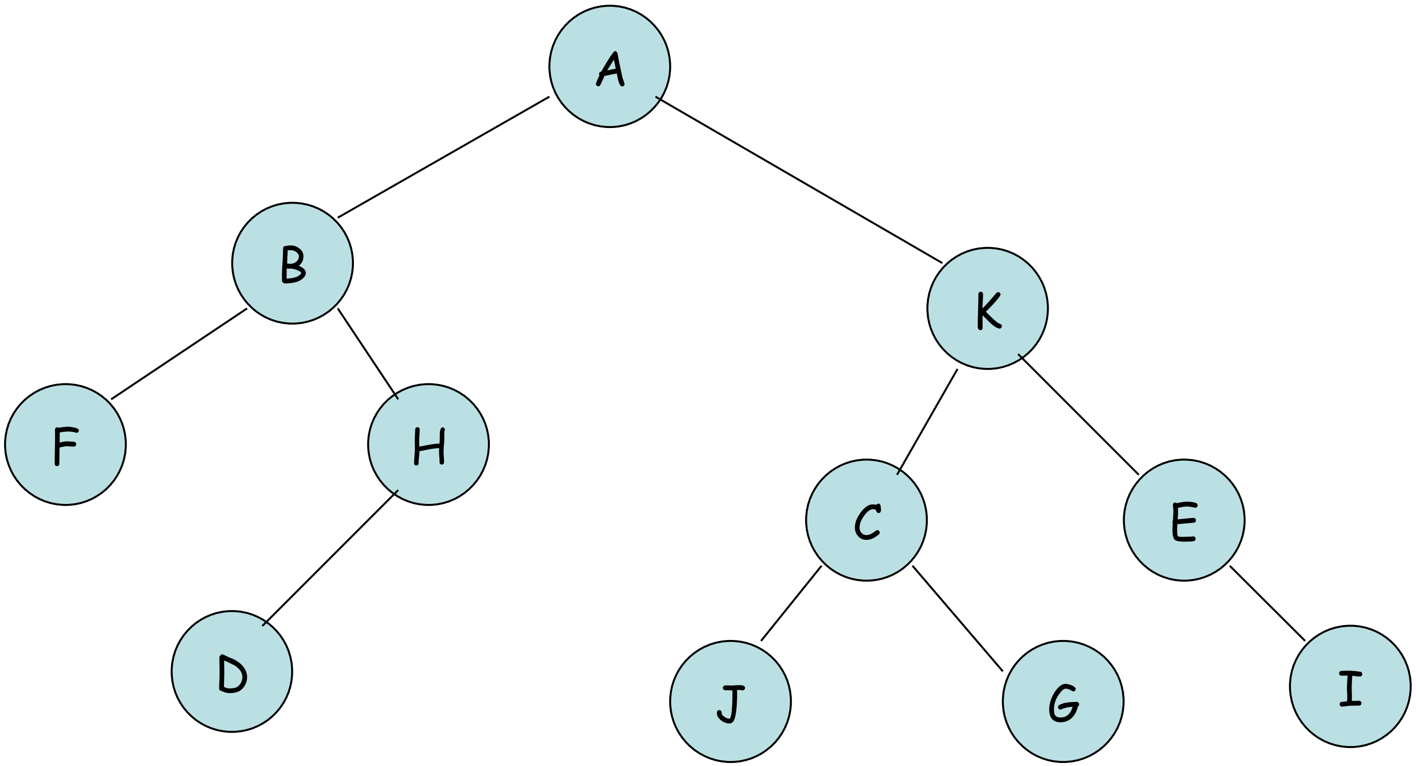
- the root (this tree doesn't have one)
- leaf nodes (there are 6 here)
- interior nodes (there are 5 here)

# A Rooted tree



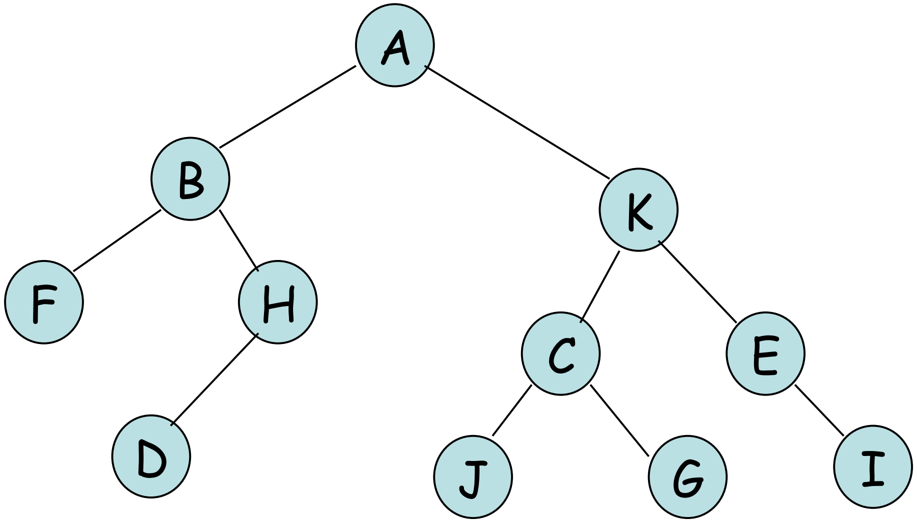
A rooted tree has one vertex designated as the root and every other edge is directed away from the root

The above tree is a binary tree
We put the root at the top by convention




The parent of H is B
The sibling of G is J
The ancestors of I are E, K, and A
C is a child of K
The descendants of B are F, H, and D
A is the root, and has no ancestors
The leaf nodes have no children
Again, the tree is a binary tree

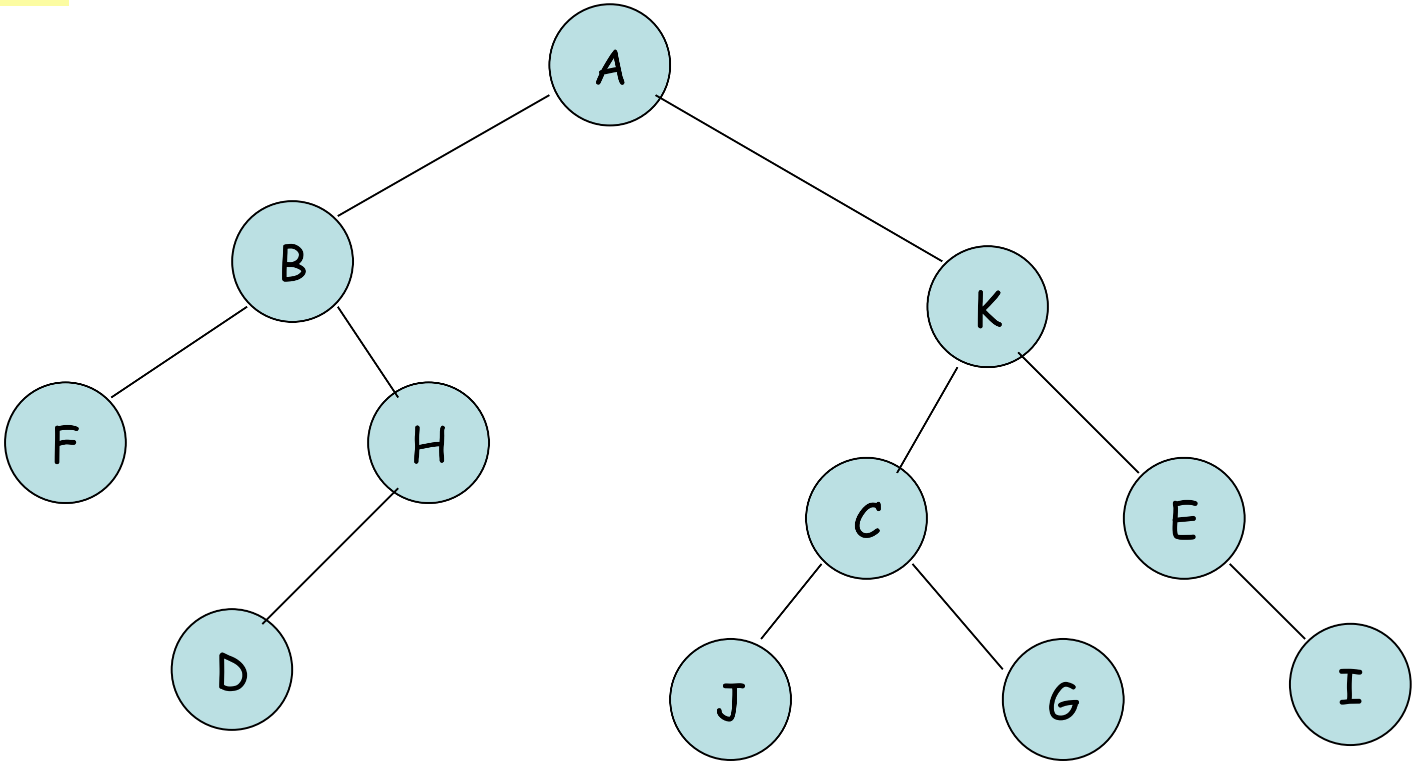
# The height of a binary tree




- The height of a leaf node is 0
- The height of an empty tree is 0
- The height of a node x is 1 + max(height(left(x)),height(right(x)))

Note: I've assumed that we have functions left, right, isNode, and isLeaf

# Traversals

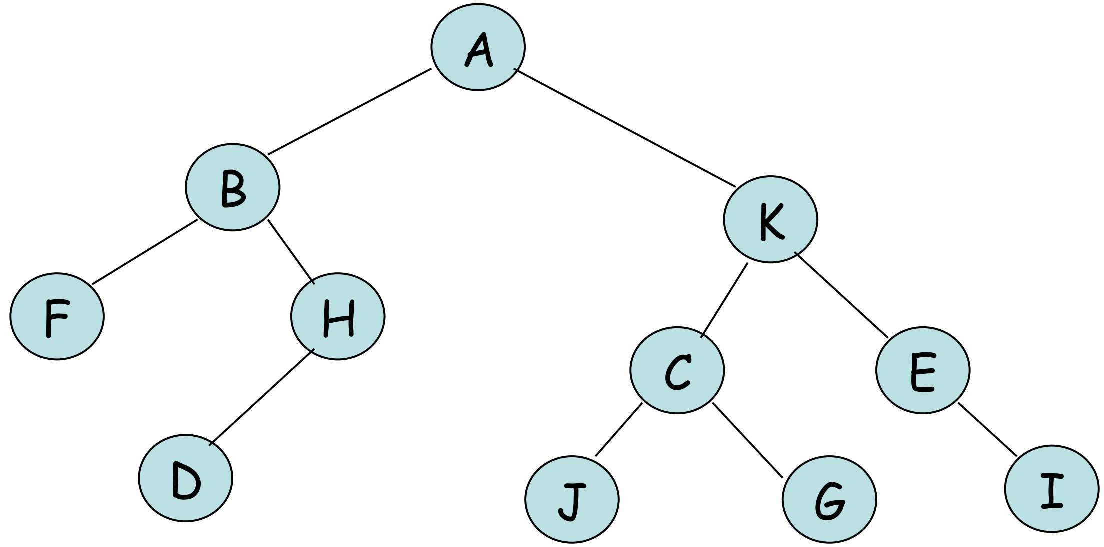



If you've got some structure one of the 1st things you want to be able to do is to traverse it!

Again, we'll stick to rooted binary trees

We have 3 traversals
2. preorder
3. inorder
4. postorder

Traversals




```
preorder(x)
 if isNode(x)
 then print(x),
        preorder(left(x)),
        preorder(right(x))
```

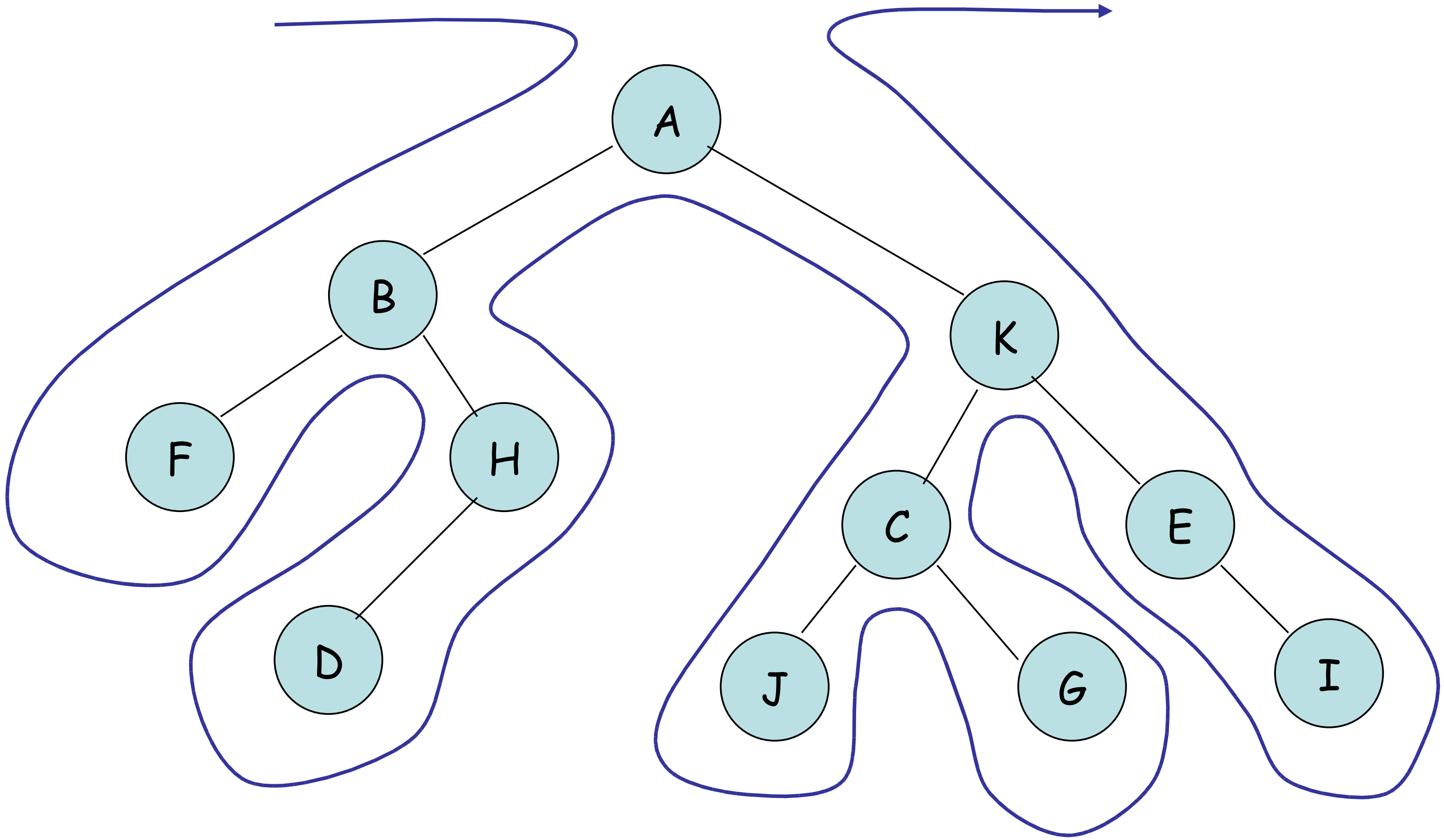
A,B,F,H,D,K,C,J,G,E,I

```
inorder(x)
 if isNode(x)
 then inorder(left(x)),
        print(x),
        inorder(right(x))
```

F,B,D,H,A,J,C,G,K,E,I

```
postorder(x)
 if isNode(x)
 then print(x),
        postorder(left(x)),
        postorder(right(x))
```

F,D,H,B,J,G,C,I,E,K,A




A walk round the tree

- if we "print" on 1st visit we get preorder
- if we "print" on 2nd visit we get inorder
- if we "print" on last visit we get postorder

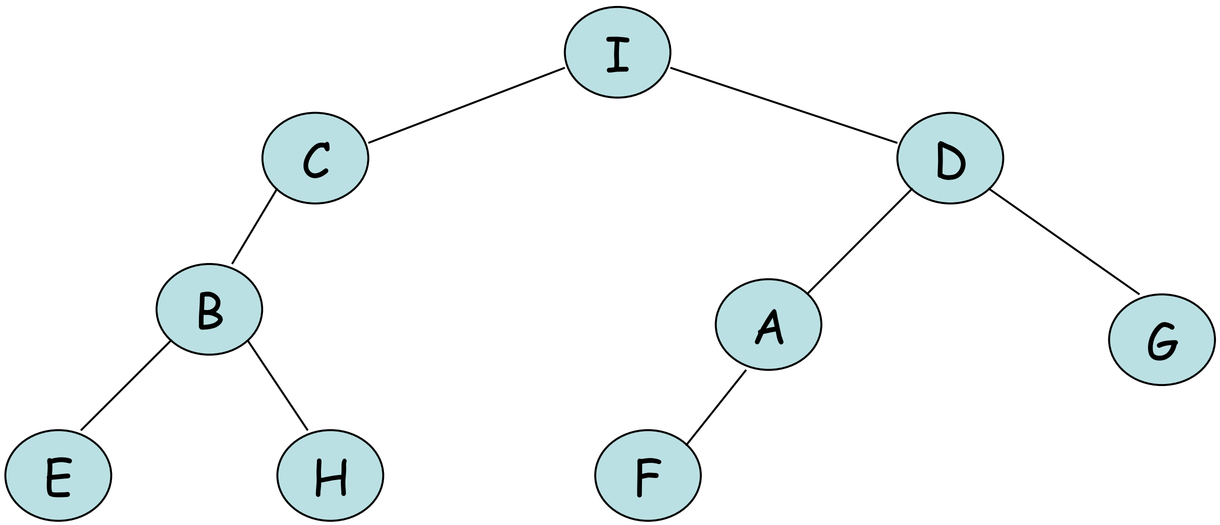
# Determine the tree from its traversals

1. Preorder: ICBEHDAFG
2. Inorder: EBHCIFADG
3. Postorder: EHBCFAGDI

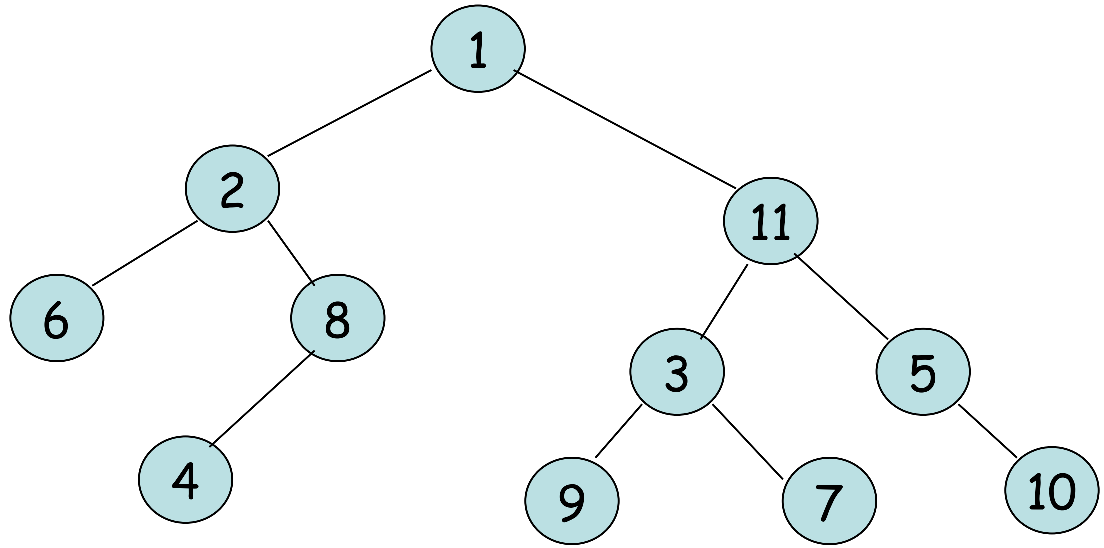
- (a) I is the root (from 1)
- (b) E, B, H, and C are to the left of I (from (a) and 2)
- (c) F, A, D, and G are to the right of I (from (a) 2)
- (d) C is the first left node of I (from (c) and 1)
- (e) D is the first right node of I (from (c) and 1)
- (f) possibly we have
  - B to the left of C,
  - E to the left of B,
  - H to the right of B ... as this satisfies 1 and 2
- (g) F and A are left of D, and G is right of D (from 2)
- (h) F must be left of A (from 1)
- (j) the tree is now fully defined

## Determine the tree from its traversals

1. Preorder: ICBEHDAFG
2. Inorder: EBHCIFADG
3. Postorder: EHBCFAGDI

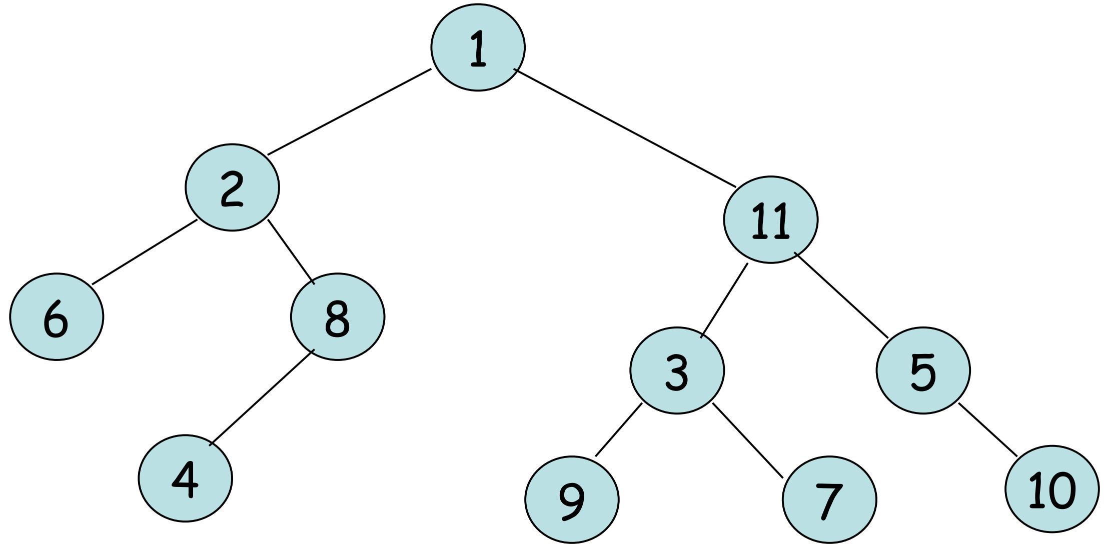


How would you represent a tree in a computer?



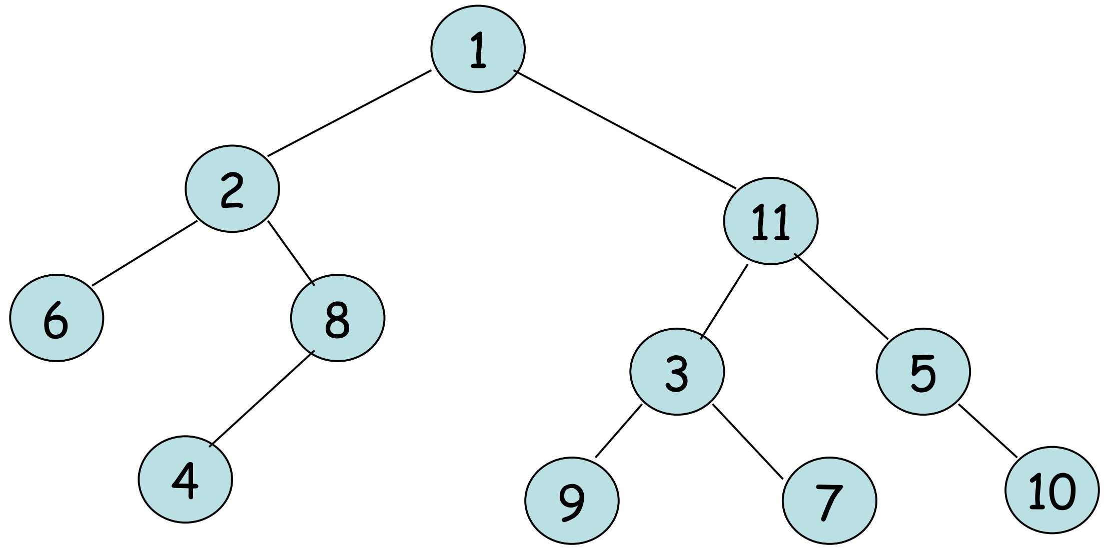

Might have a btree data structure with attributes

- data
  - the actual information in a node
- left
  - the btree to the left, or nil
- right
  - the btree to the right, or nil



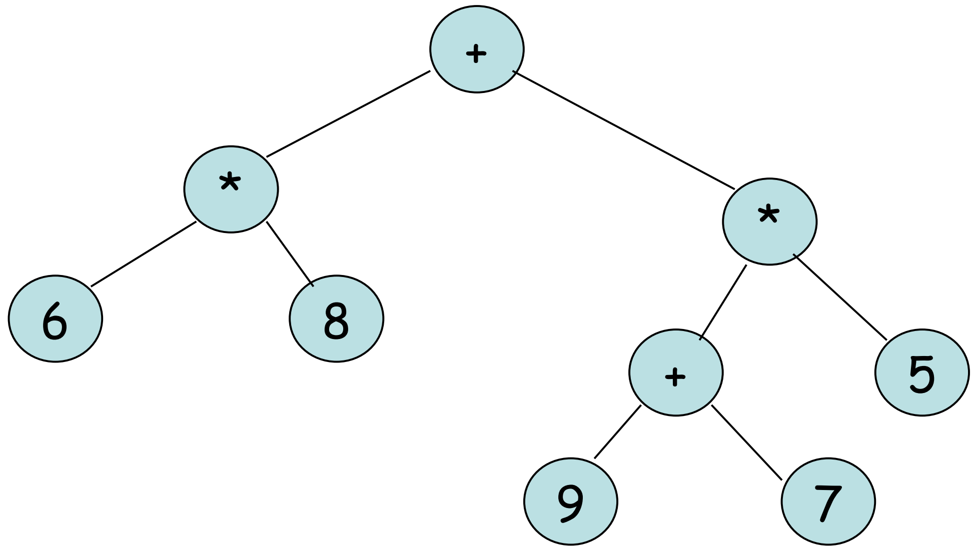

| | *data* | *left* | *right* |
|---|---|---|---|
| 1 | 1 | 2 | 11 |
| 2 | 2 | 6 | 8 |
| 3 | 3 | 9 | 7 |
| 4 | 4 | - 1 | - 1 |
| 5 | 5 | - 1 | 10 |
| 6 | 6 | - 1 | - 1 |
| 7 | 7 | - 1 | - 1 |
| 8 | 8 | 4 | - 1 |
| 9 | 9 | - 1 | - 1 |
| 10 | 10 | - 1 | - 1 |
| 11 | 11 | 3 | 5 |

Might use a 2d array




Might use a 1d array , giving parent of a node

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| - 1 | 1 | 11 | 8 | 11 | 2 | 3 | 2 | 3 | 5 | 1 |




An expression

(6 * 8) + ((9 + 7) * 5)

What would a preorder, inorder and postorder traversal of this tree looklike?