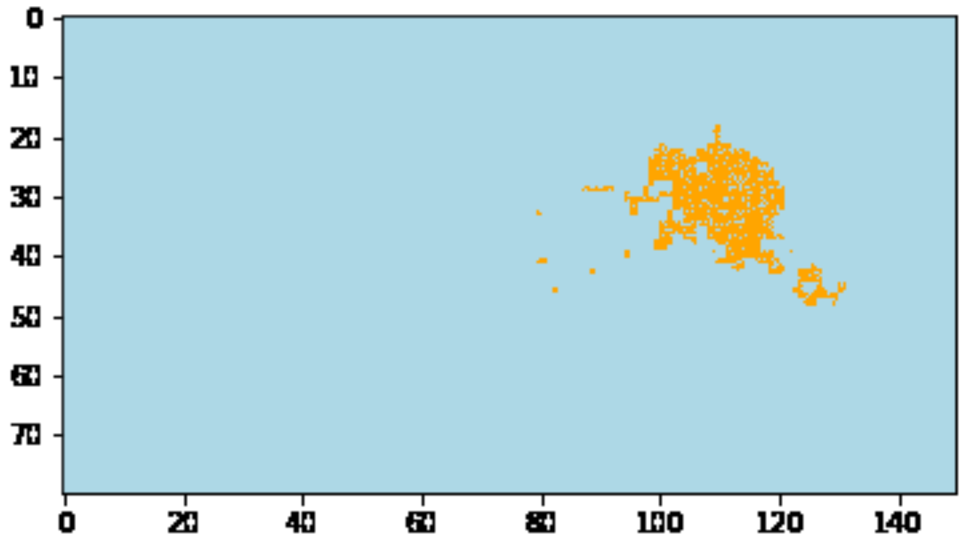
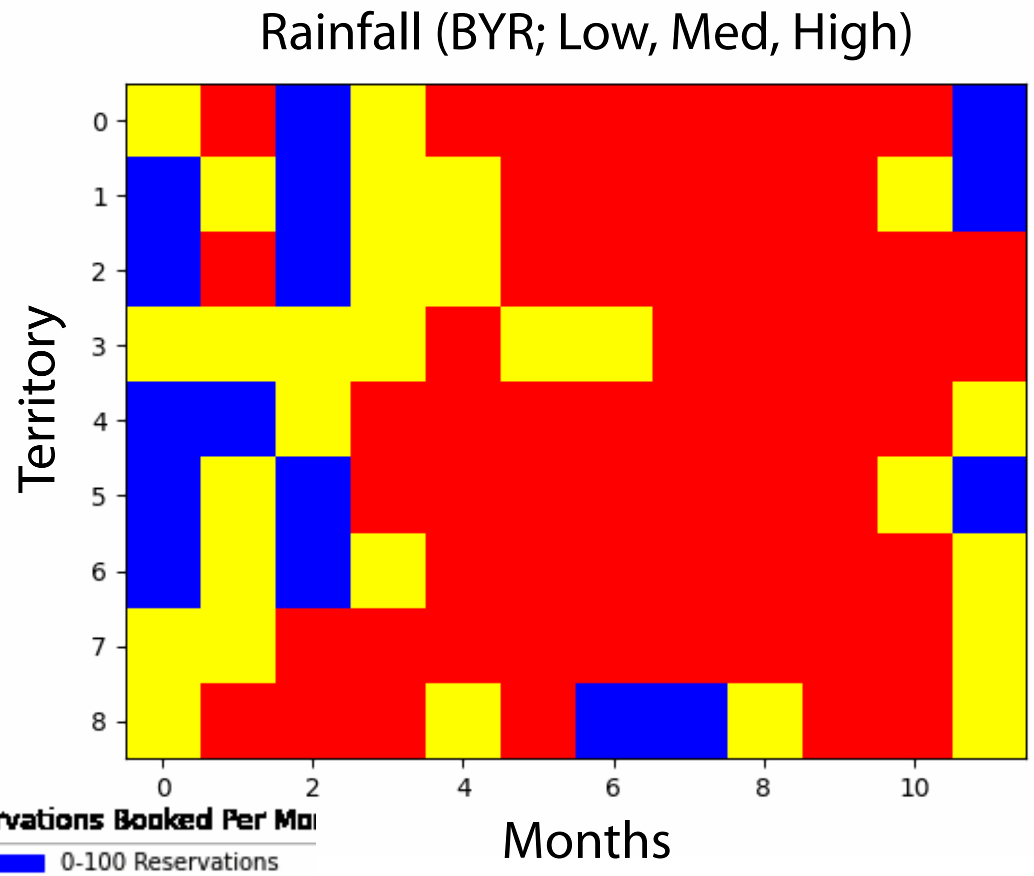
Algorithms and Data Structures for Data Science
Trees

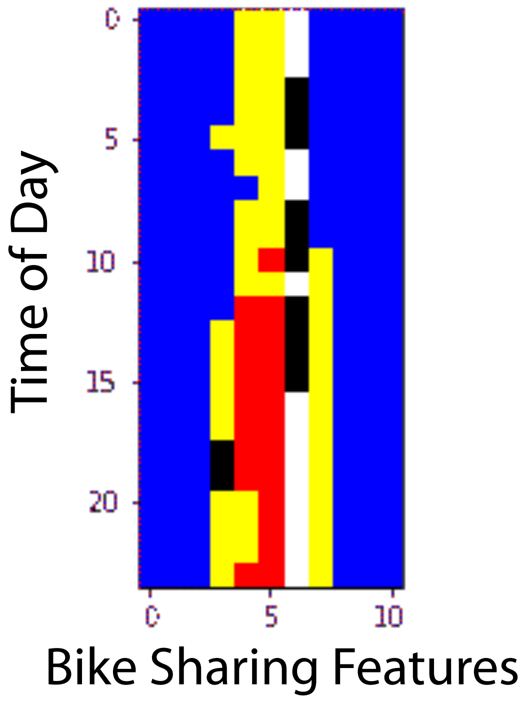
CS 277
Brad Solomon

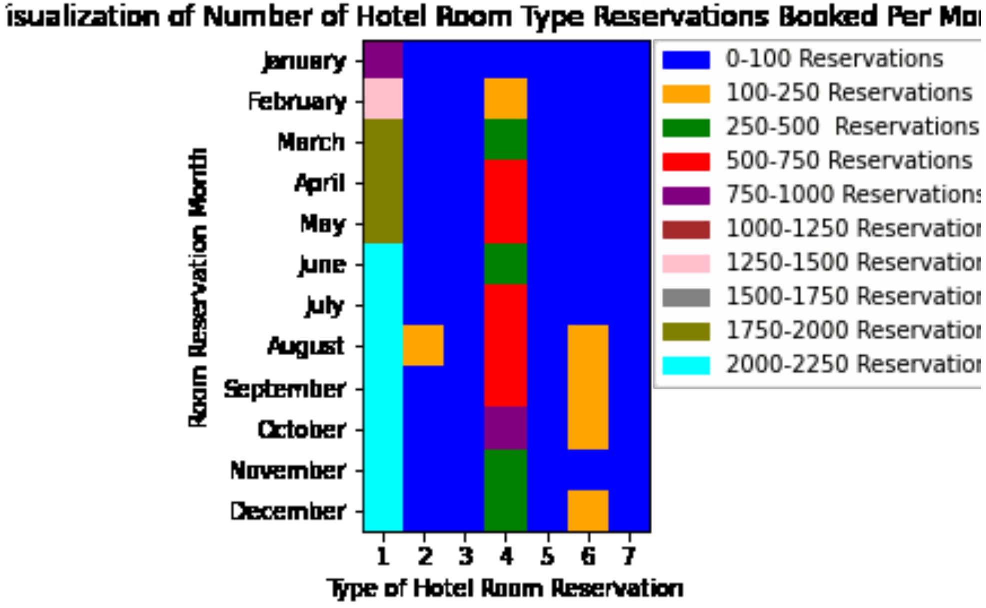
March 20, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science


0
10
20
30
40
50
60
70
0
20
40
60
80
100
120
140


Rainfall (BYR; Low, Med, High)
Territory
0
1
2
3
4
5
6
7
8
0
2
4
6
8
10
Months


Time of Day
0
5
10
15
20
0
5
10
Bike Sharing Features


isualization of Number of Hotel Room Type Reservations Booked Per Mo
Room Reservation Month
January
February
March
April
May
June
July
August
September
October
November
December
1
2
3
4
5
6
7
Type of Hotel Room Reservation
0-100 Reservations
100-250 Reservations
250-500 Reservations
500-750 Reservations
750-1000 Reservations
1000-1250 Reservatior
1250-1500 Reservatior
1500-1750 Reservatior
1750-2000 Reservatior
2000-2250 Reservatior

Assignment Extensions

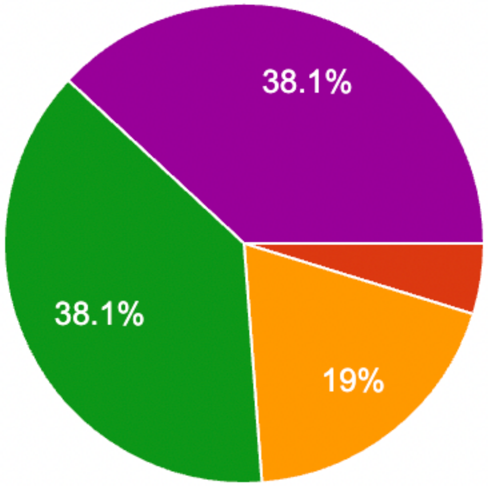
Everyone will have until Wednesday (3/22) to complete lab_search

Everyone will have until Friday (3/24) to complete mp_hash

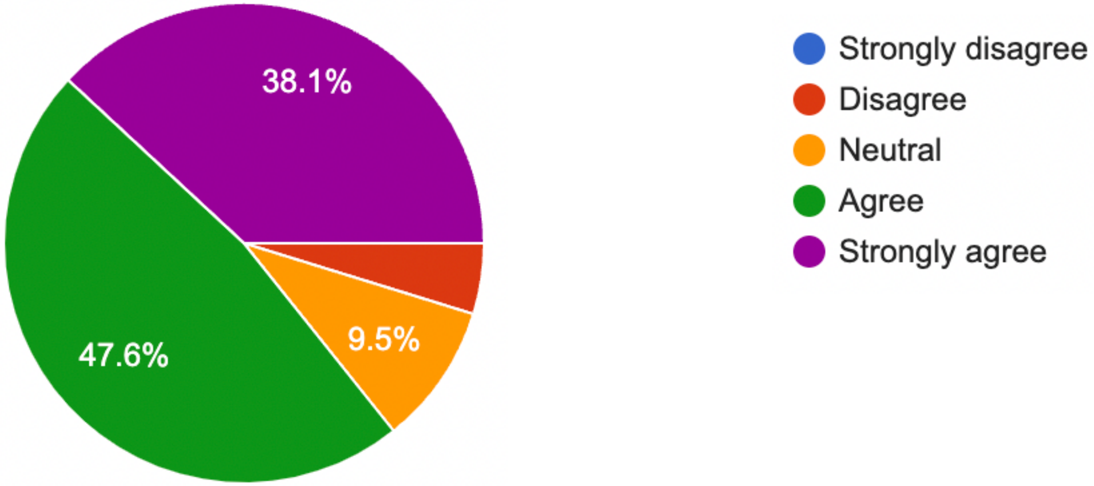
Informal Early Feedback

I feel that I can actively participate…

in lecture:



in class in general:

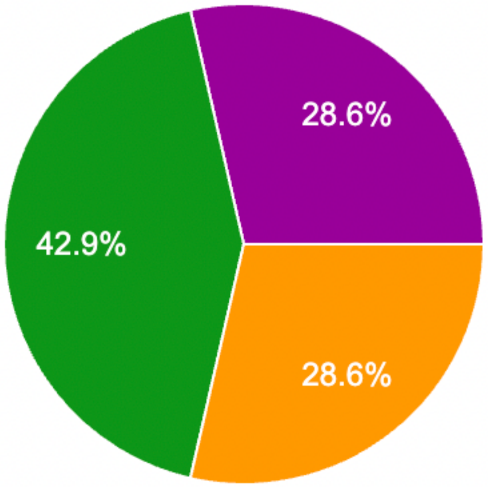


90.5% / 81% of class receive helpful and complete answers in / out of class

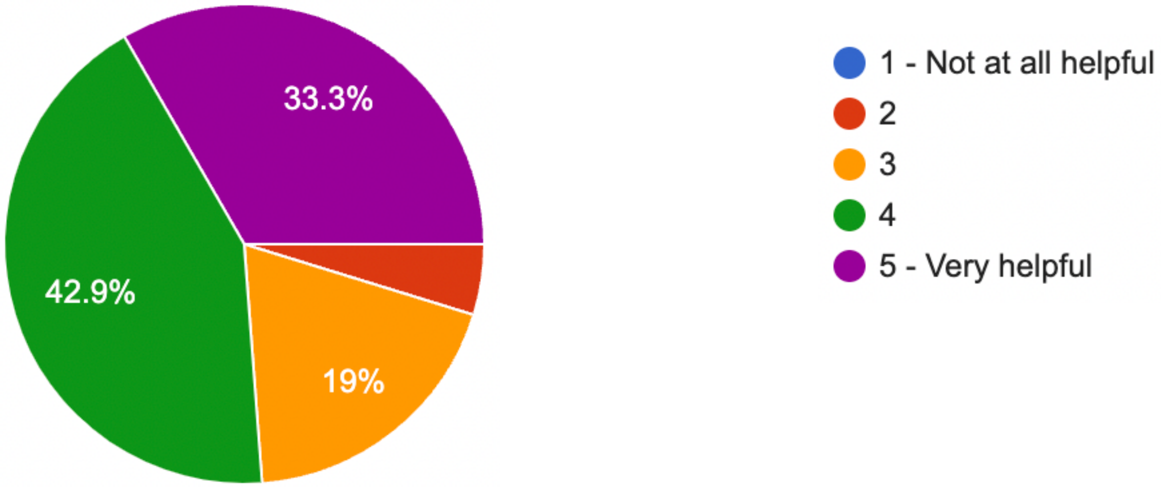
Informal Early Feedback

Helpful for learning material…

Lectures:

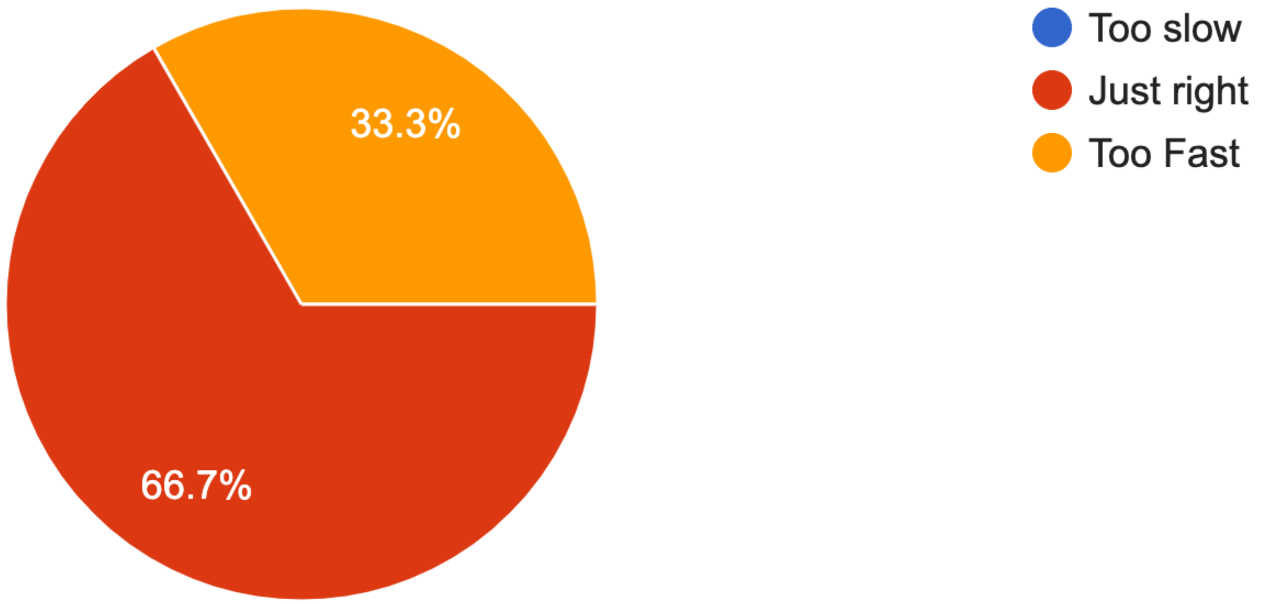


Assignments:



Informal Early Feedback

Pacing of the course:



Informal Early Feedback

Labs and lectures are 'most helpful' to most students; MPs not listed at all

Requests / Suggestions (in no particular order):

Extending deadline for labs (or opening labs earlier in week)

Providing annotated lecture slides after each lecture

Include more live coding demonstrations in the class

Giving optional coding questions for practice exercises

Providing practice exams / example exam questions

A final project instead of a final

Learning Objectives

Formally define the tree data structure

Explore properties of trees and the specifics of binary trees

Implement and understand traversals and search on trees

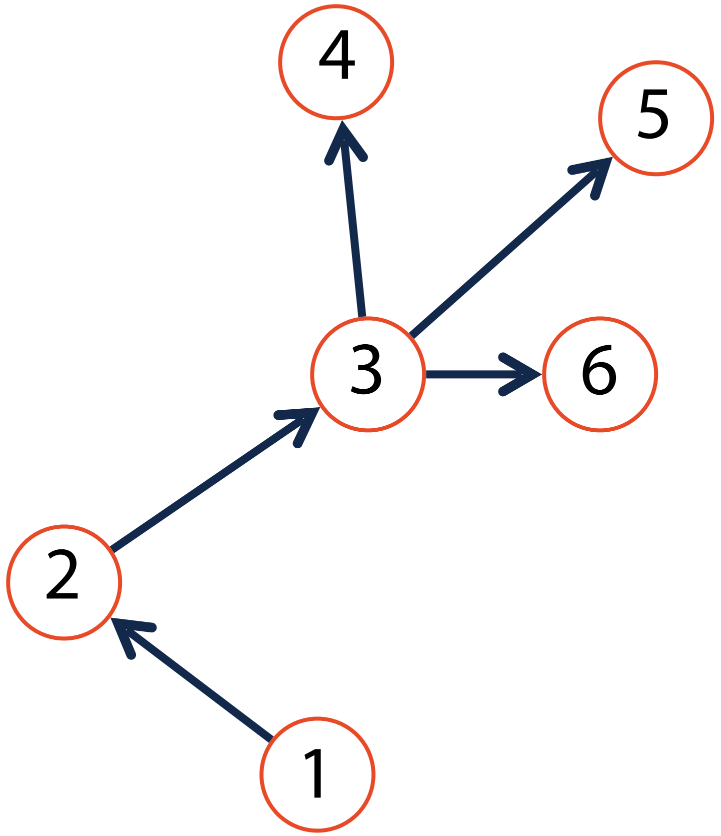
Trees

A non-linear data structure defined recursively as a collection of nodes where each node contains a value and zero or more connected nodes.

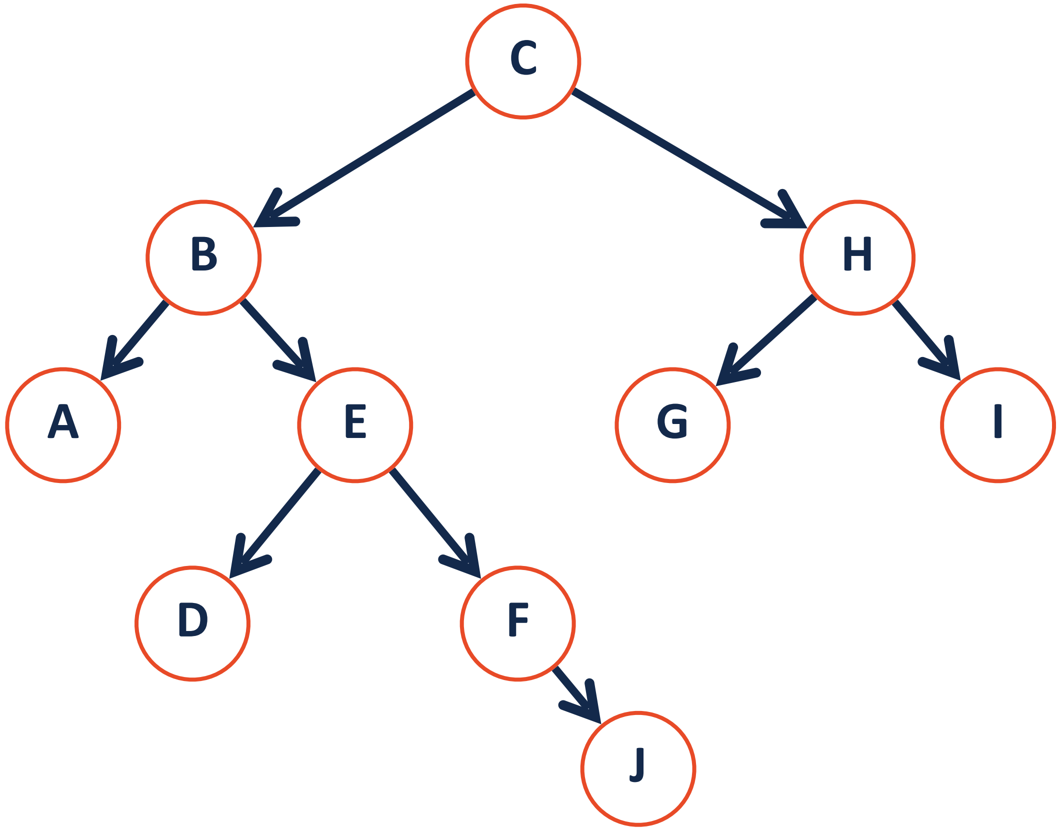
(In CS 277) a tree is also:

1) Acyclic

2) Rooted



Tree Terminology

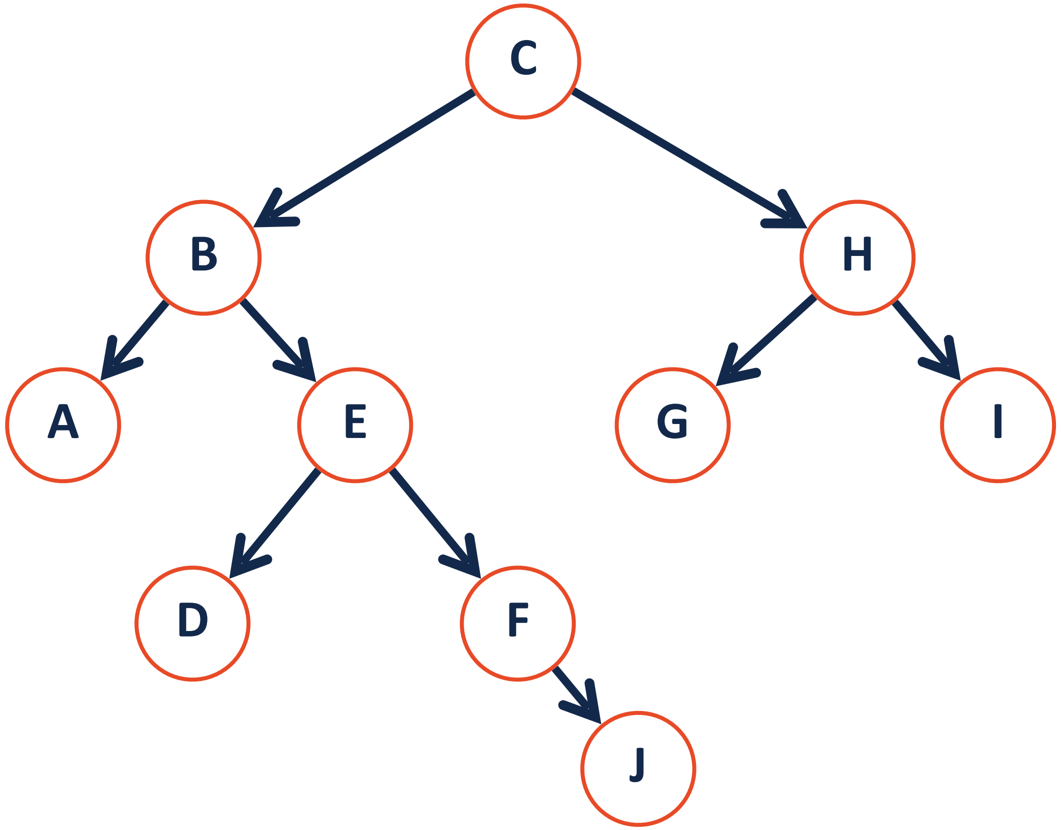


Node: The vertex of a tree

Edge: The [theoretical] connecting path between nodes

Path: A list of the edges (or nodes) traversed to go from node `start` to node `end`

Tree Terminology



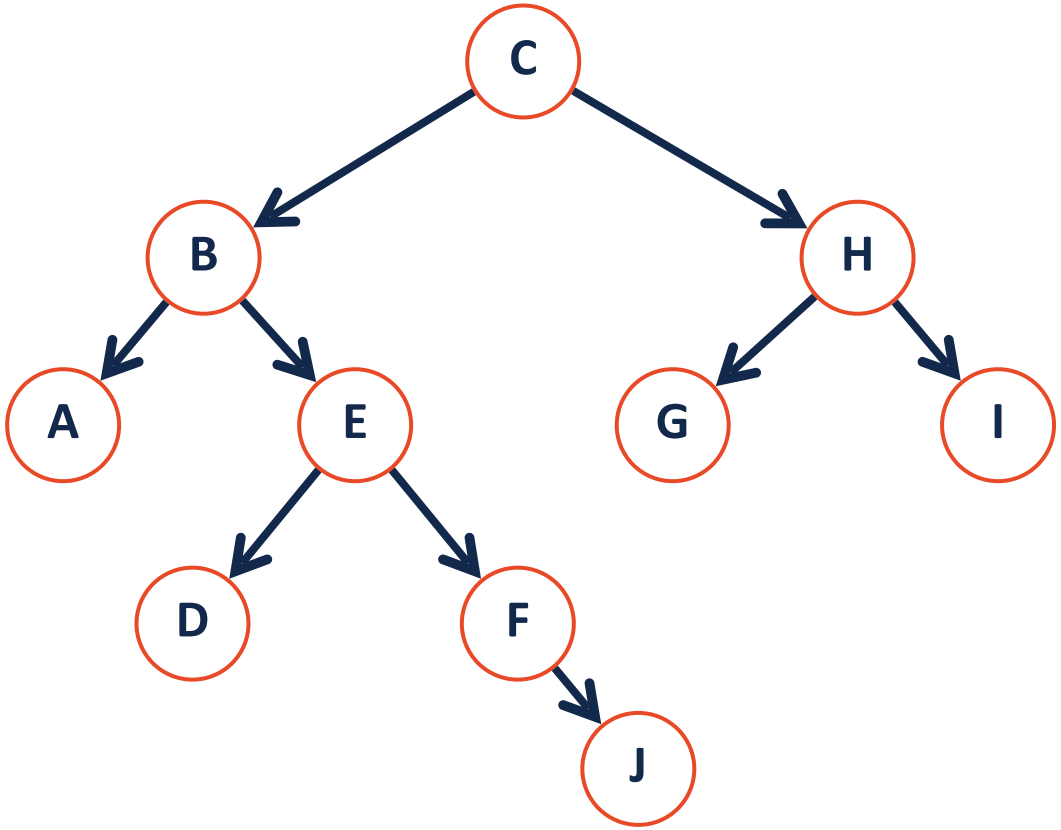
Parent: The precursor node to the current node is the 'parent'

Child: The nodes linked by the current node are it's 'children'

Neighbor: Parent or child

Degree: The number of children for a given node

Tree Terminology

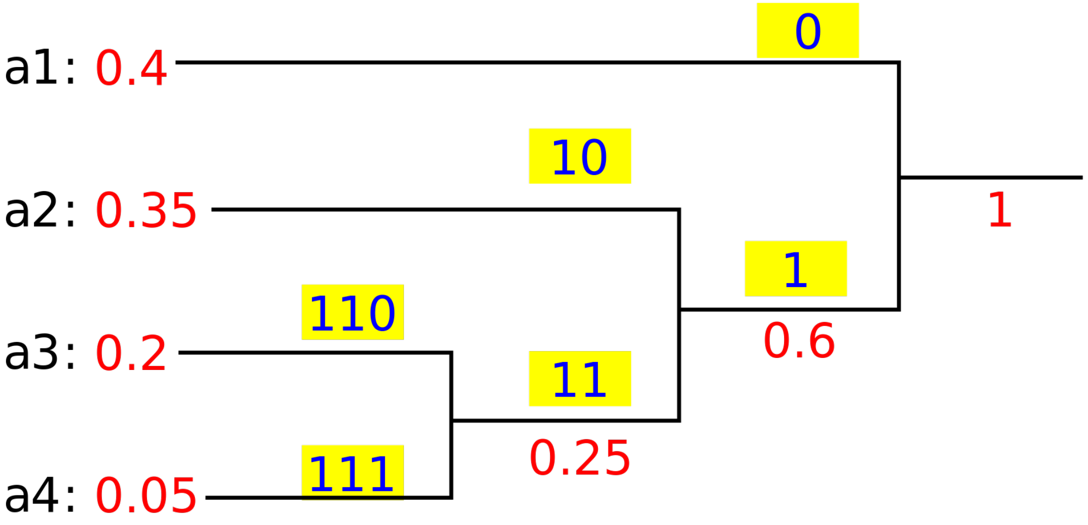


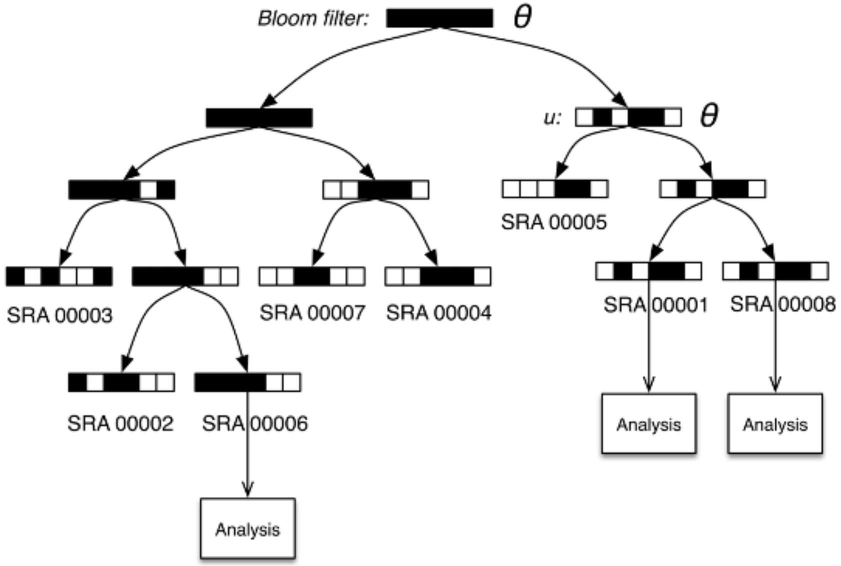
Root: The start of a tree (the only node with no parent).

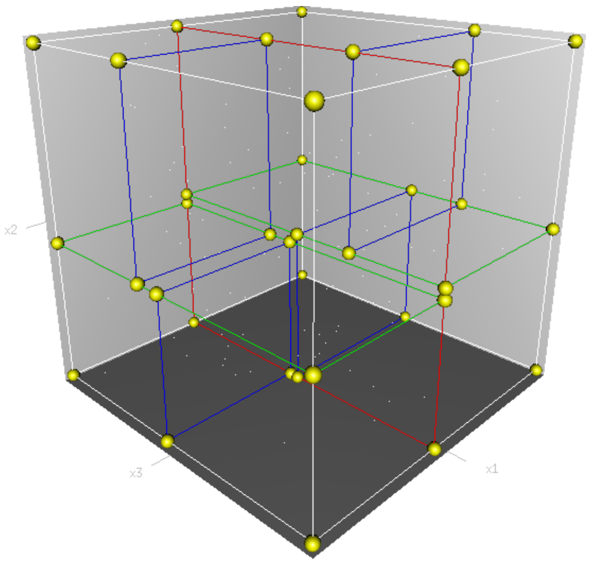
Leaf: The terminating nodes of a tree (have no children)

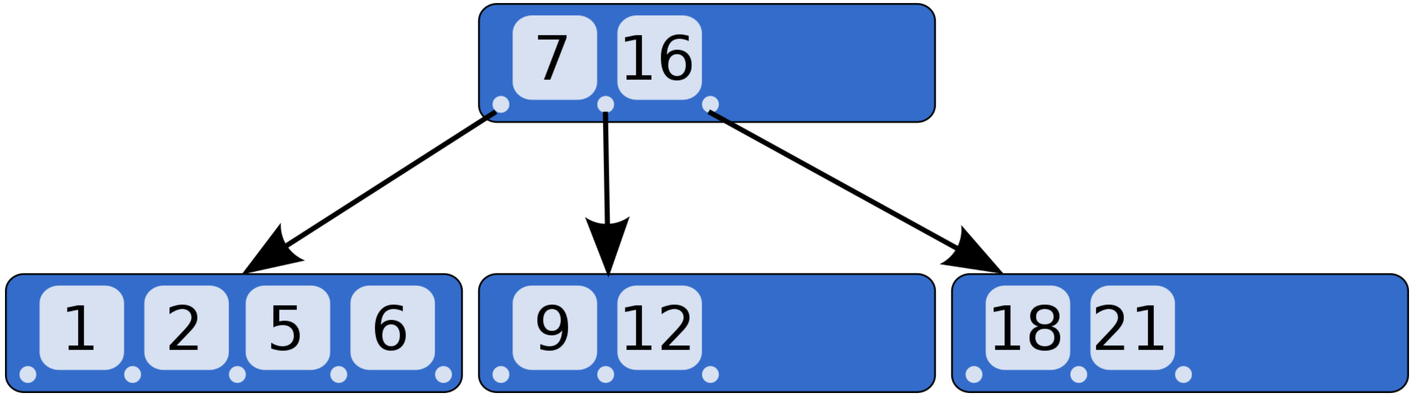
Internal: A node with at least one child

There are many *types* of trees









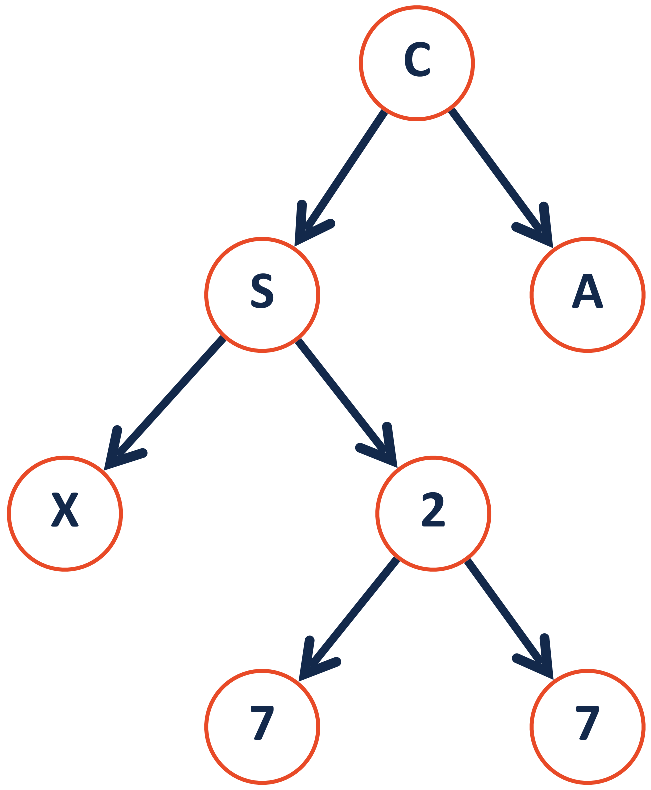
Binary Tree

A **binary tree** is a tree T such that:

$$T = None$$

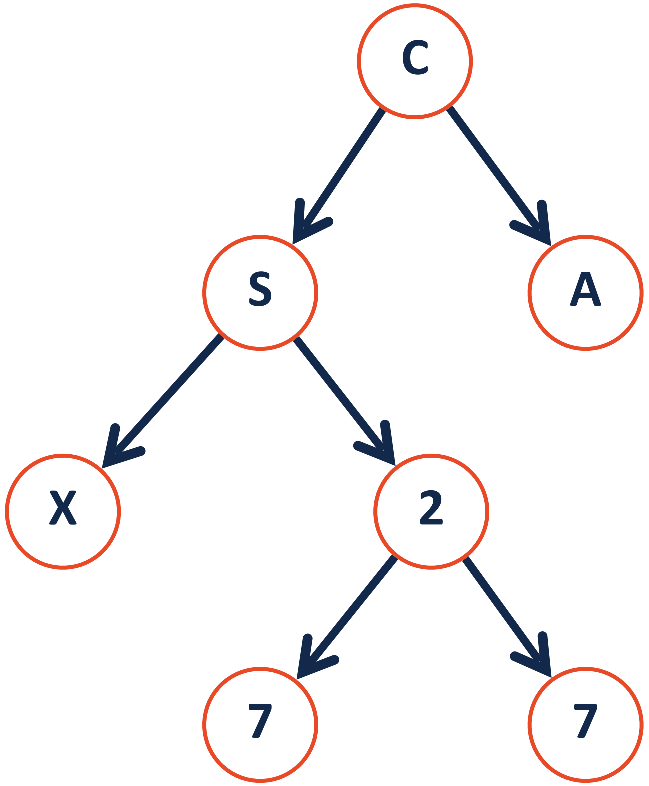
or

$$T = treeNode(val, T_L, T_R)$$



Binary Tree

```
class treeNode:
    def __init__(self, ______, ______, ______):


        ________________________



        ________________________



        ________________________


class binaryTree:
    def __init__(self, root=None):
        self.root = root
```

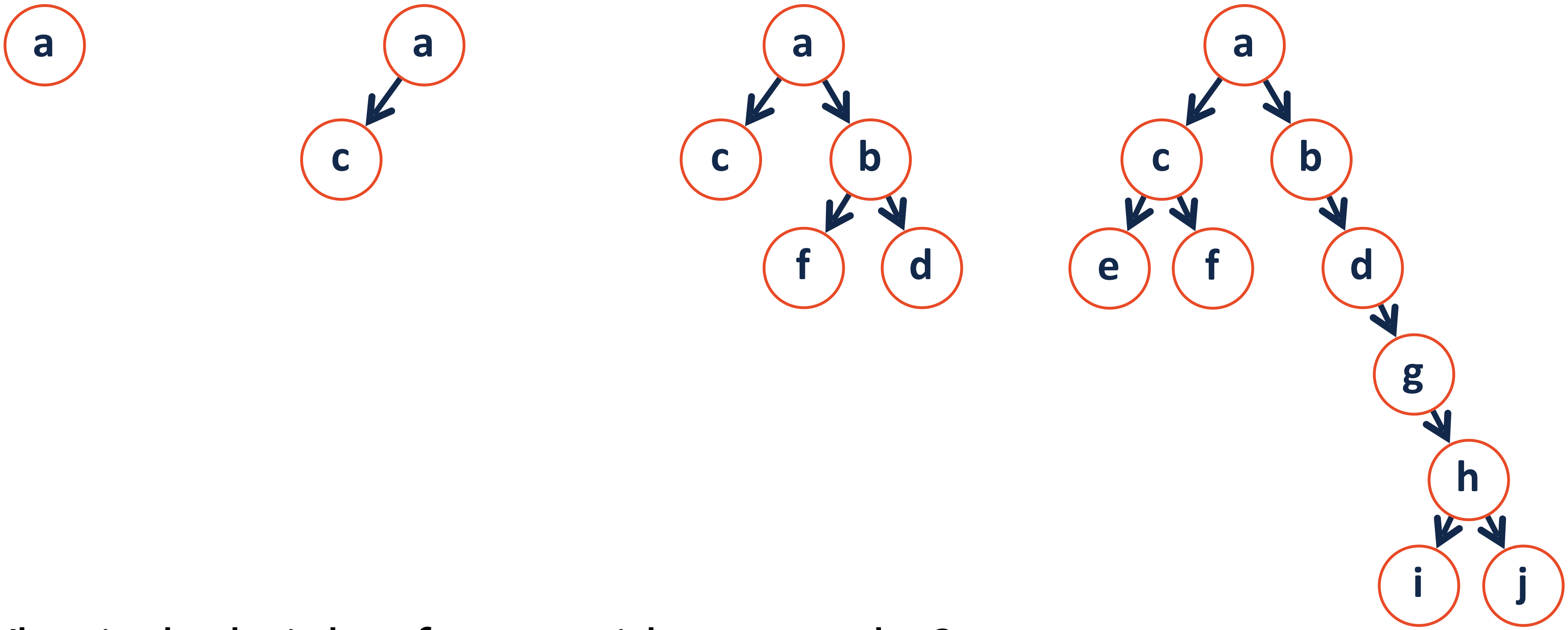


Defining a tree

```
class treeNode:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

tn1 = treeNode(1)

tn2 = treeNode(2)

tn3 = treeNode(3)

tn4 = treeNode(4)

tn5 = treeNode(5, tn1, tn2)

tn6 = treeNode(6, tn3, tn4)

tn7 = treeNode(7, tn5, tn6)

binaryTree(___________)
```

Tree Terminology

Height: the length of the longest path from the root to a leaf



What is the height of a tree with ***zero*** nodes?

Tree Height

```
height(T) =
```

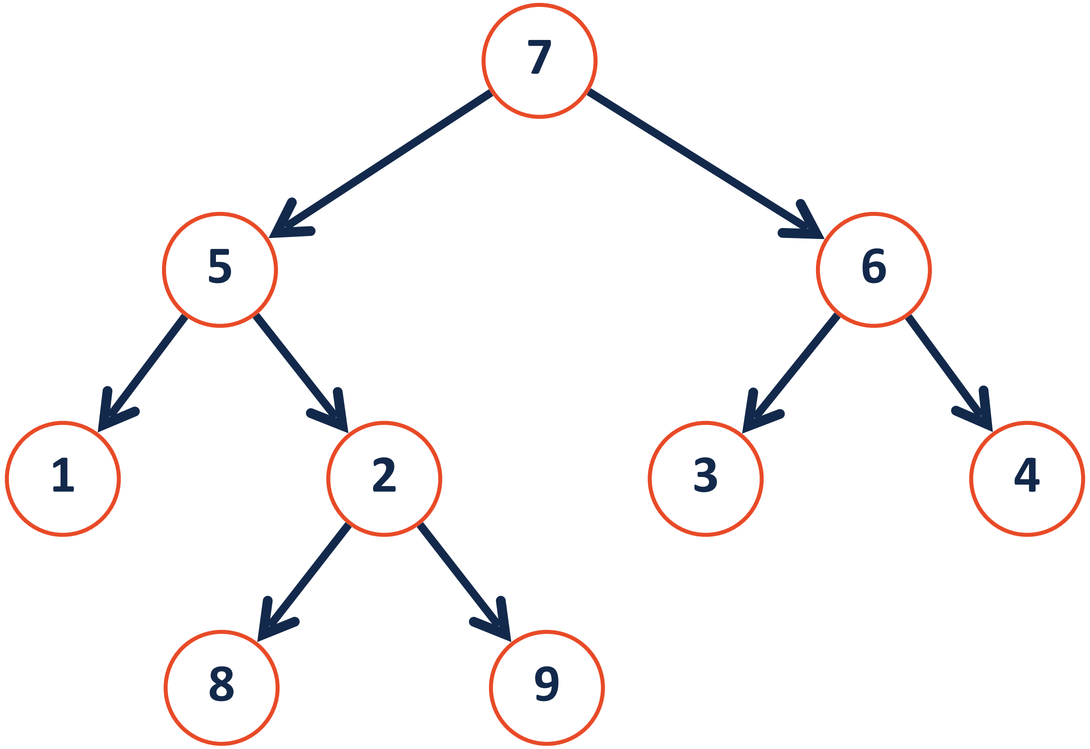
Base Case:

Recursive Step:

Combining:

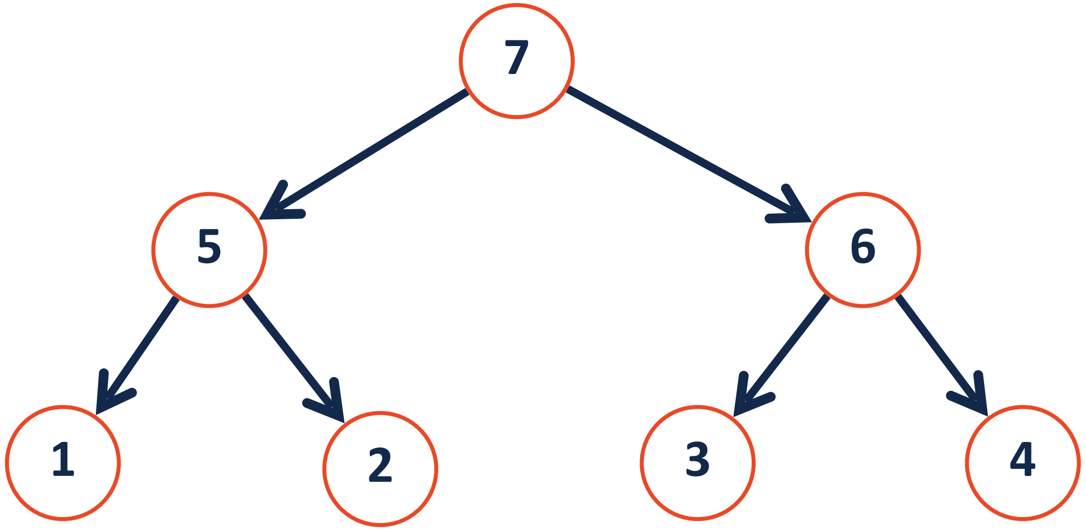
Tree Traversals

A **traversal** of a tree T is an ordered way of visiting every node once.



Pre-Order Traversal

1) Get current node's value
2) Recurse left
3) Recurse right

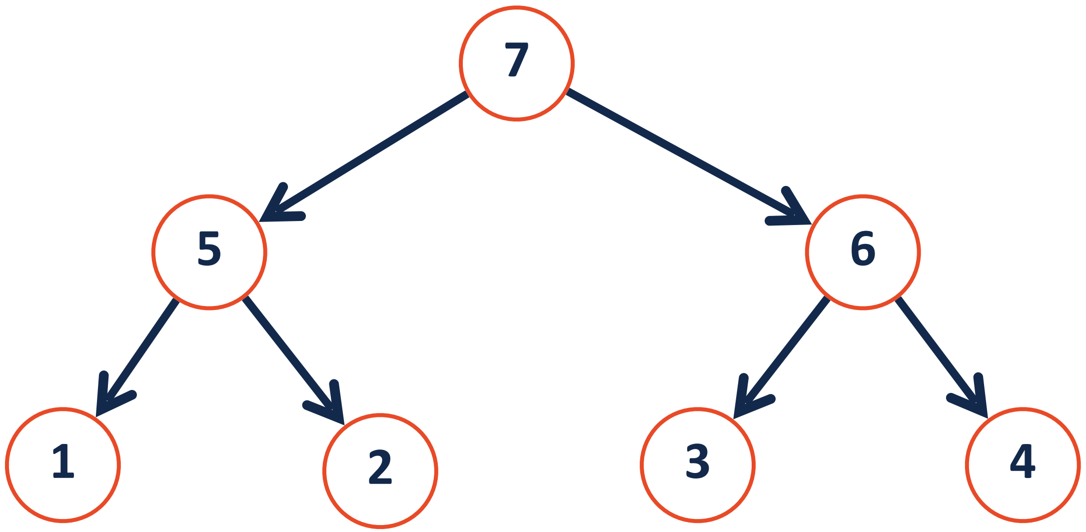


Post-Order Traversal

1) Recurse left

2) Recurse right

3) Get current nodes value

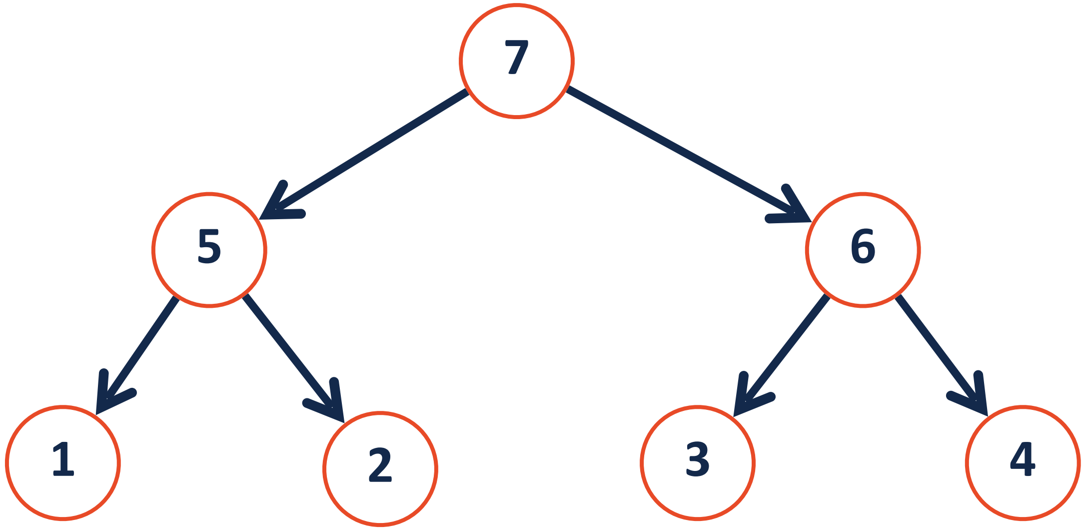


In-Order Traversal

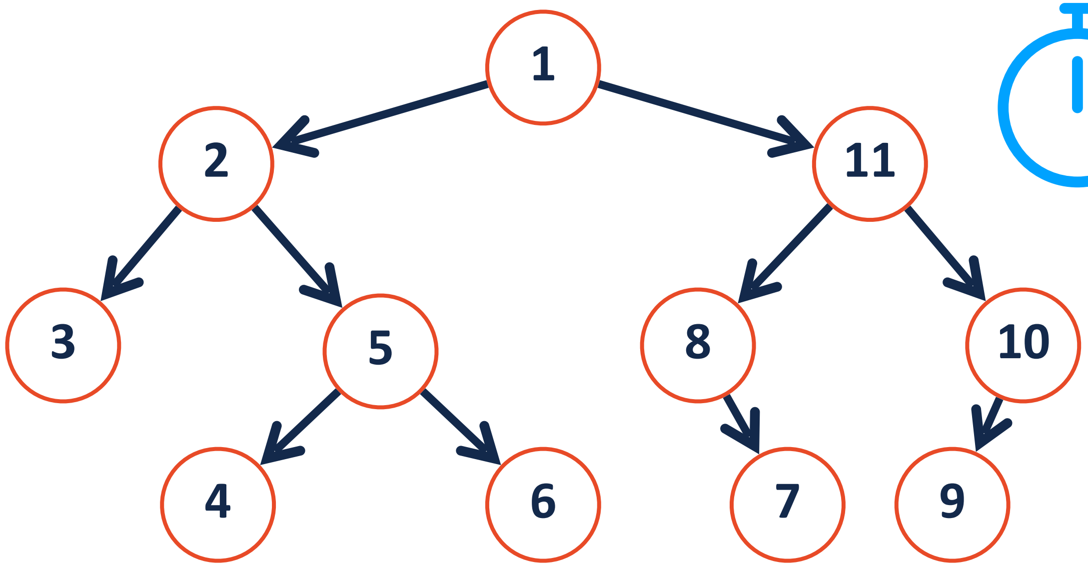
1) Recurse left

2) Get current nodes value

3) Recurse right



Tree Traversals



Pre-order:

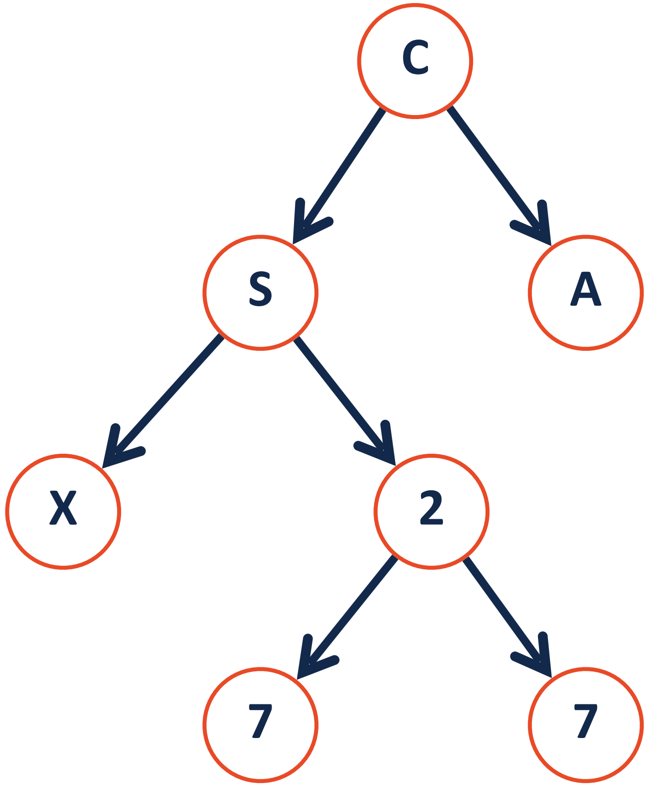
In-order:

Post-order:

Tree Traversals

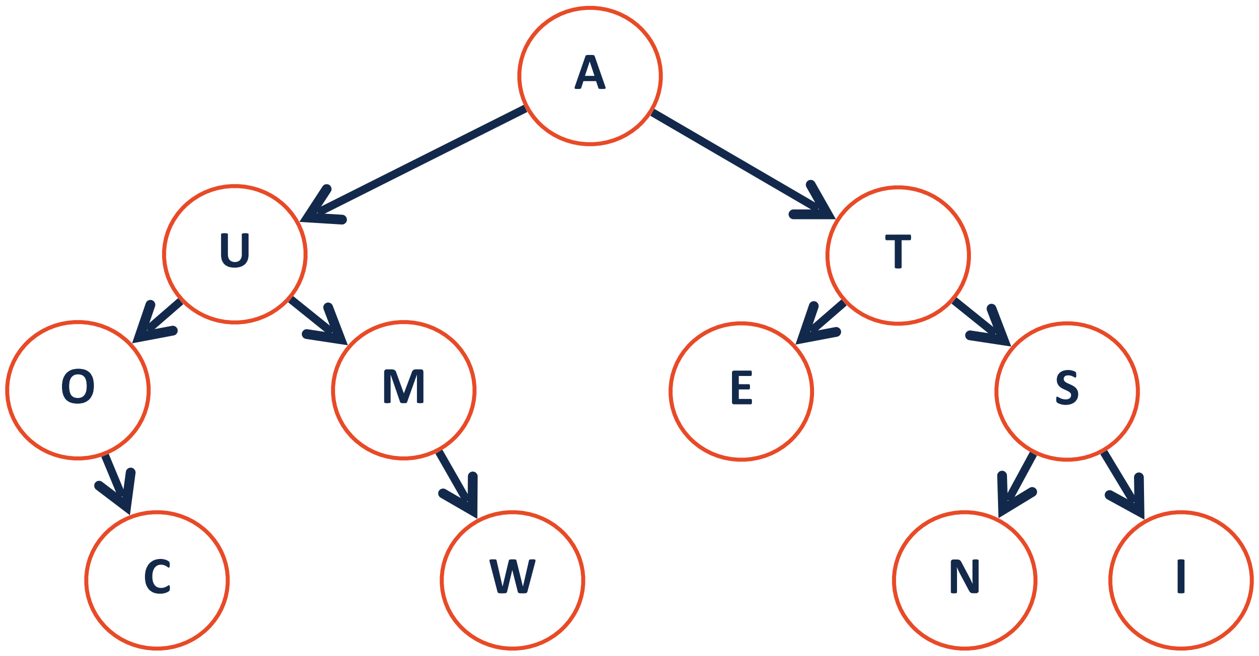
Pre-order: Ideal for copying trees

Post-order: Ideal for deleting trees



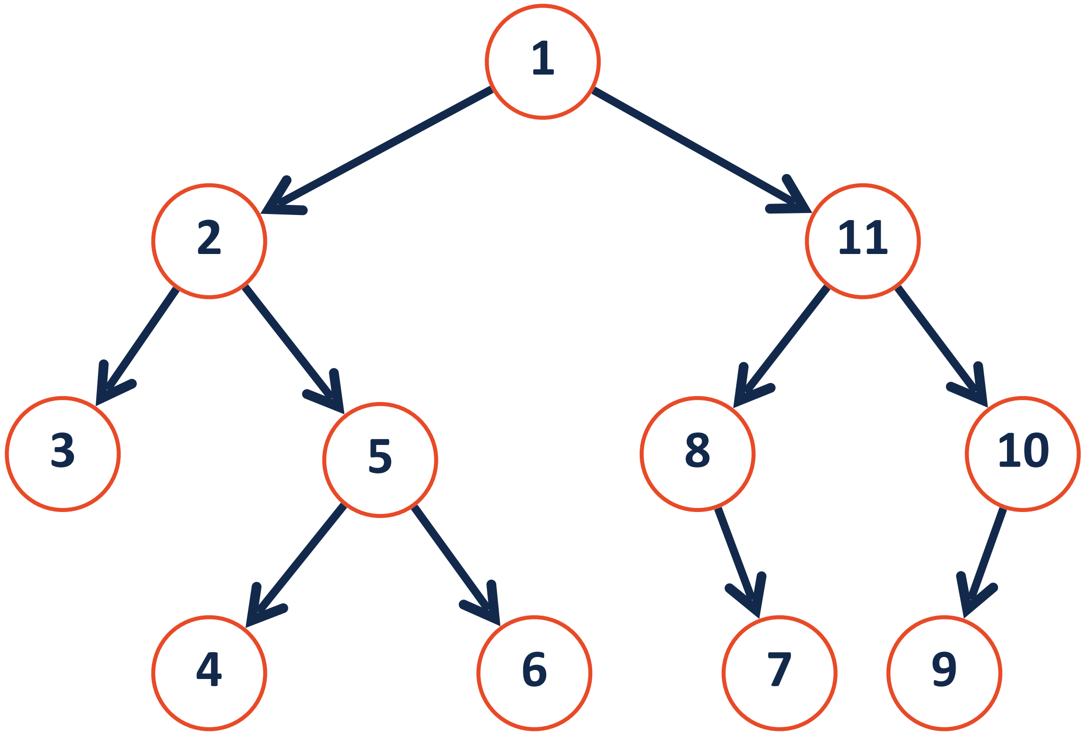
Searching a Binary Tree

There are two main approaches to searching a binary tree:



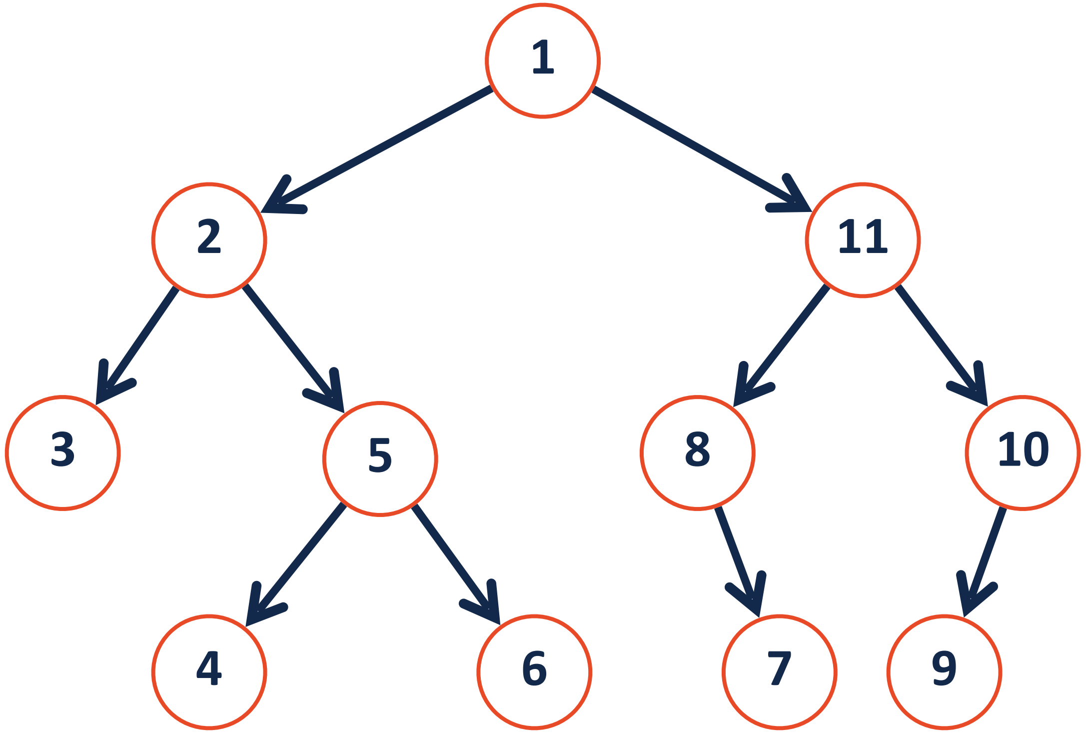
Depth First Search

Explore as far along one path as possible before backtracking



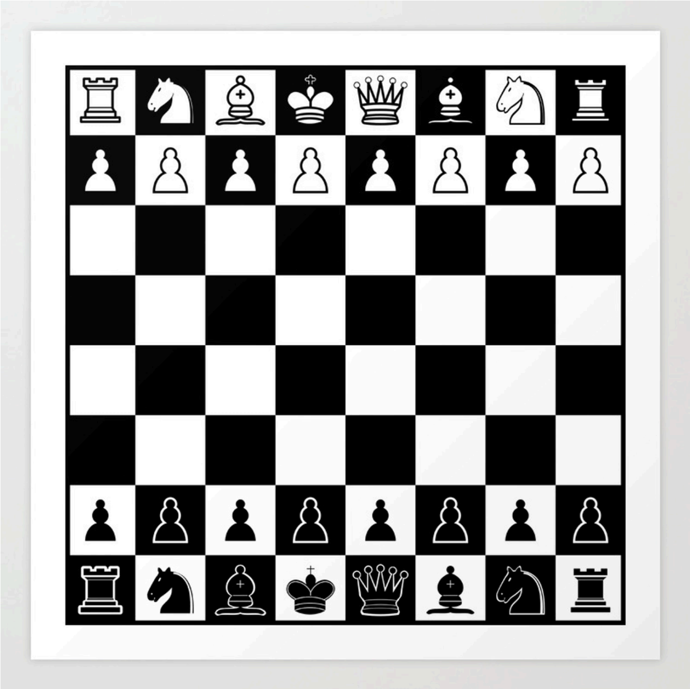
Breadth First Search

Fully explore depth i before exploring depth i+1



What search algorithm is best?

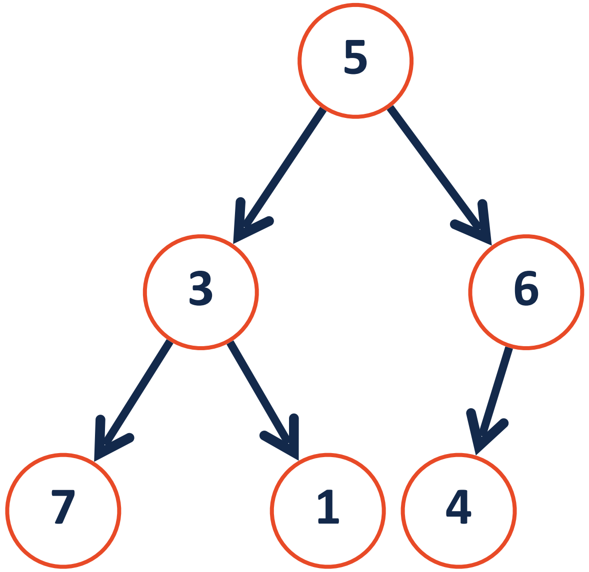
The average ‘branch factor’ for a game of chess is ~31. If you were searching a decision tree for chess, which search algorithm would you use?

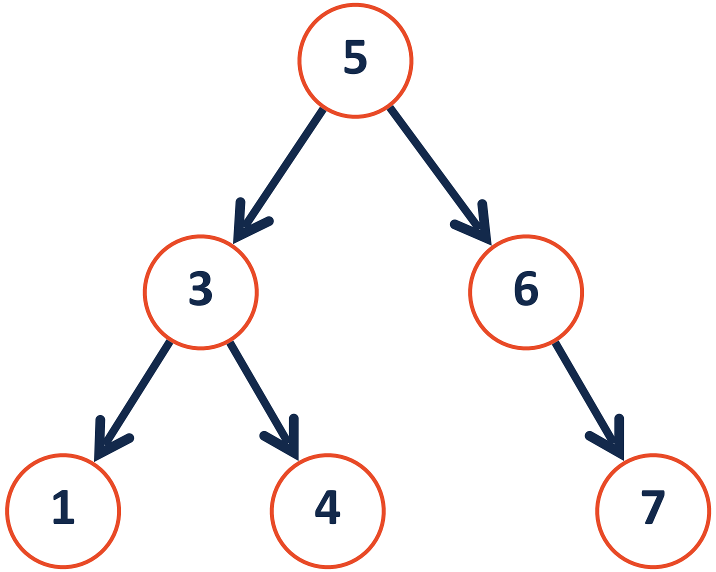


Improved search on a binary tree

5	3	6	7	1	4

1	3	4	5	6	7





Binary Search Tree (BST)

A **BST** is a binary tree $T = treeNode(val, T_L, T_r)$ such that:

$$\forall n \in T_L,\ n.val < T.val$$

$$\forall n \in T_R,\ n.val > T.val$$