

## Topic 18 Binary Trees

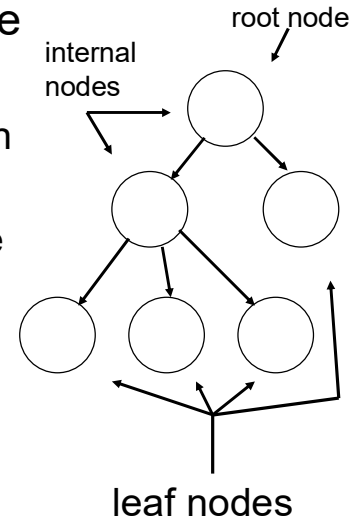
"A tree may grow a thousand feet tall, but its leaves will return to its roots."

-Chinese Proverb



## Definitions

- ▶ A *tree* is an abstract data type
  - one entry point, the **root**
  - Each node is either a **leaf** or an *internal node*
  - An internal node has 1 or more **children**, nodes that can be reached directly from that internal node.
  - The internal node is said to be the **parent** of its child nodes



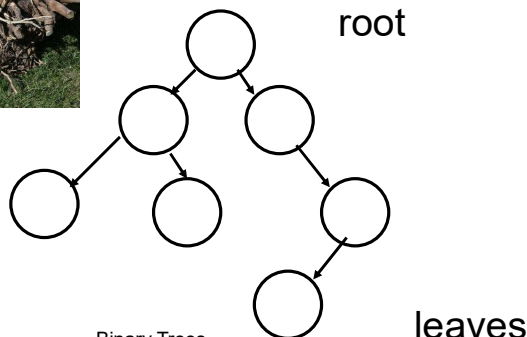
CS314

Binary Trees

2

## Properties of Trees

- ▶ Only access point is the root
- ▶ All nodes, except the root, have one parent
  - like the inheritance hierarchy in Java
- ▶ Traditionally trees drawn upside down



CS314

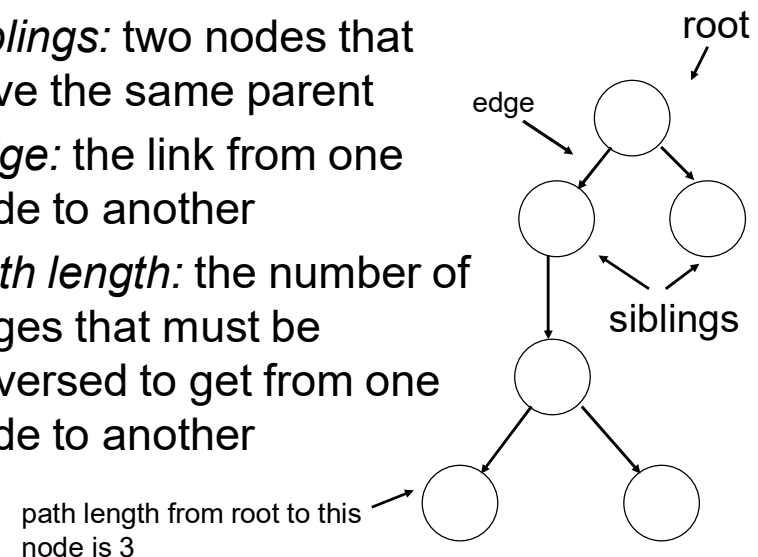
Binary Trees

leaves

3

## Properties of Trees and Nodes

- ▶ *siblings*: two nodes that have the same parent
- ▶ *edge*: the link from one node to another
- ▶ *path length*: the number of edges that must be traversed to get from one node to another



CS314

Binary Trees

4

## More Properties of Trees

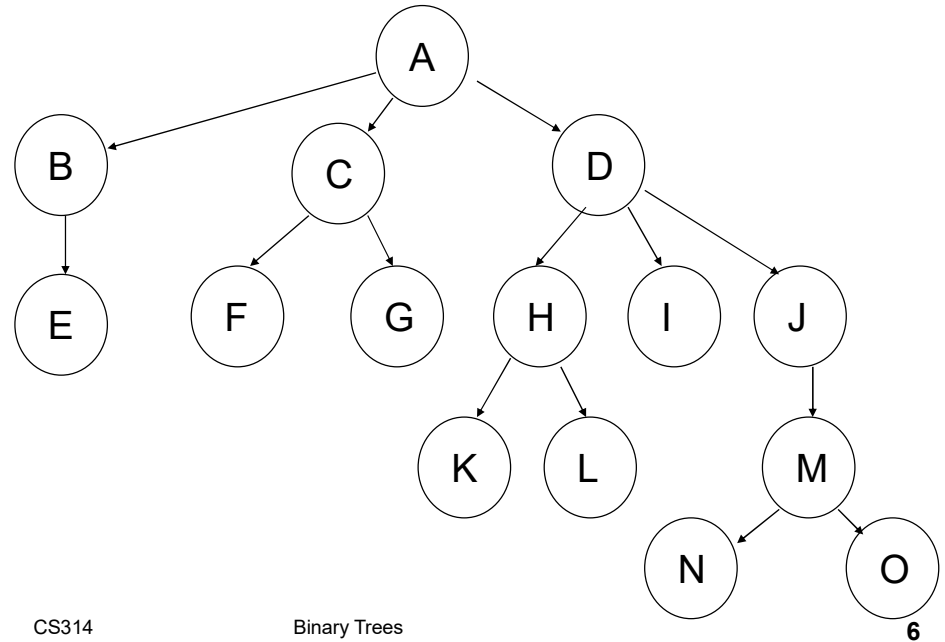
- ▶ **depth**: the path length from the root of the tree to this node
- ▶ **height of a node**: The maximum distance (path length) of any leaf from this node
  - a leaf has a height of 0
  - the height of a tree is the height of the root of that tree
- ▶ **descendants**: any nodes that can be reached via 1 or more edges from this node
- ▶ **ancestors**: any nodes for which this node is a descendant

CS314

Binary Trees

5

## Tree Visualization



CS314

Binary Trees

6

## Clicker 1

- ▶ What is the depth of the node that contains M on the previous slide?

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

**Clicker 2** - Same tree, same choices  
What is the height of the node that contains D?

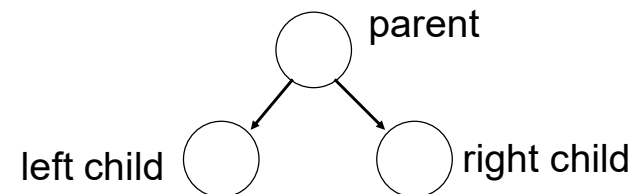
CS314

Binary Trees

7

## Binary Trees

- ▶ There are many variations on trees but we will start with *binary trees*
- ▶ **binary tree**: each node has at most two children
  - the possible children are usually referred to as the left child and the right child



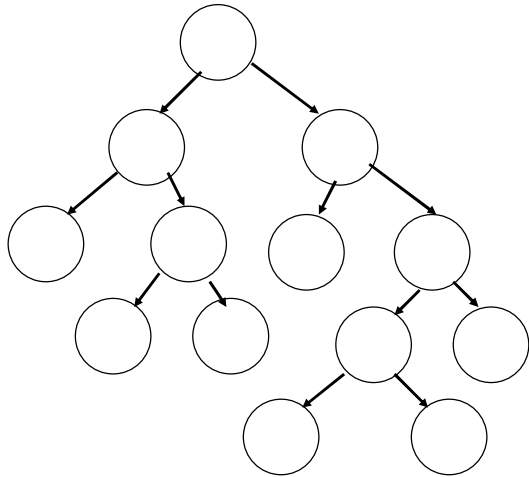
CS314

Binary Trees

8

## Full Binary Tree

- *full binary tree*: a binary tree in which each node has 2 or 0 children

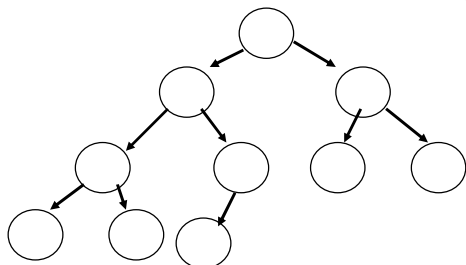


## Clicker 3

- What is the maximum height of a full binary tree with 11 nodes?
- A. 3  
B. 5  
C. 7  
D. 10  
E. Not possible to have full binary tree with 11 nodes.

## Complete Binary Tree

- *complete binary tree*: a binary tree in which every level, except possibly the deepest is completely filled. At depth  $n$ , the height of the tree, all nodes are as far left as possible



Where would the next node go to maintain a complete tree?

## Clicker 4

- What is the height of a complete binary tree that contains  $N$  nodes?
- A.  $O(1)$   
B.  $O(\log N)$   
C.  $O(N^{1/2})$   
D.  $O(N)$   
E.  $O(N \log N)$
- Recall, order can be applied to any function. It doesn't just apply to running time.

## Perfect Binary Tree

- ▶ *perfect binary tree*: a binary tree with all leaf nodes at the same depth. All internal nodes have exactly two children.
- ▶ a perfect binary tree has the maximum number of nodes for a given height
- ▶ a perfect binary tree has  $(2^{(n+1)} - 1)$  nodes where  $n$  is the height of the tree
  - height = 0 -> 1 node
  - height = 1 -> 3 nodes
  - height = 2 -> 7 nodes
  - height = 3 -> 15 nodes

## A Binary Node class

```
public class Bnode<E> {
    private E myData;
    private Bnode<E> myLeft;
    private Bnode<E> myRight;

    public BNode();
    public BNode(Bnode<E> left, E data,
                 Bnode<E> right)
    public E getData()
    public Bnode<E> getLeft()
    public Bnode<E> getRight()

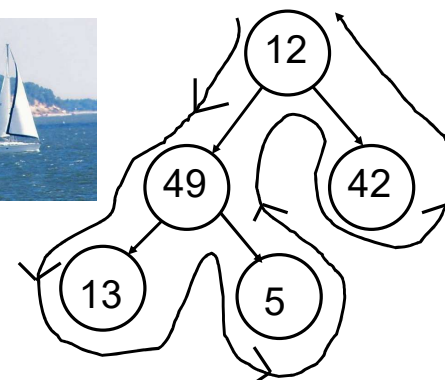
    public void setData(E data)
    public void setLeft(Bnode<E> left)
    public void setRight(Bnode<E> right)
}
```

## Binary Tree Traversals

- ▶ Many algorithms require all nodes of a binary tree be visited and the contents of each node processed or examined.
- ▶ There are 4 traditional types of traversals
  - preorder traversal: process the root, then process all sub trees (left to right)
  - in order traversal: process the left sub tree, process the root, process the right sub tree
  - post order traversal: process the left sub tree, process the right sub tree, then process the root
  - level order traversal: starting from the root of a tree, process all nodes at the same depth from left to right, then proceed to the nodes at the next depth.

## Results of Traversals

- ▶ To determine the results of a traversal on a given tree draw a path around the tree.
  - start on the left side of the root and trace around the tree. The path should stay close to the tree.

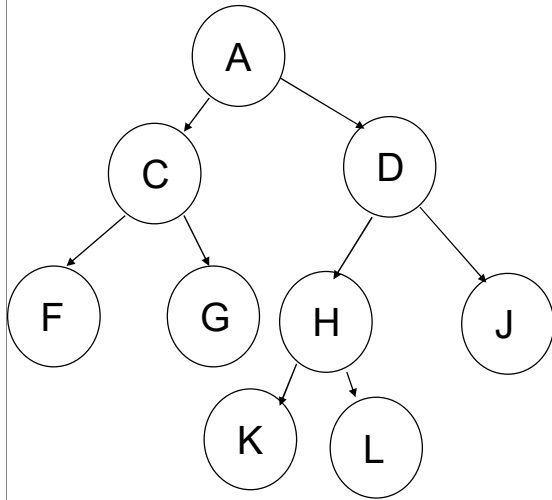


pre order: process when pass down left side of node  
12 49 13 5 42

in order: process when pass underneath node  
13 49 5 12 42

post order: process when pass up right side of node  
13 5 49 42 12

## Clicker 5 - Tree Traversals



What is the result of a post order traversal of the tree to the left?

- A. F C G A Q H L D J
- B. F G C Q L H J D A
- C. A C F G D H Q L J
- D. A C D F G H J Q L
- E. L Q J H G F D C A



Binary Trees

17

## Implement Traversals

- ▶ Implement preorder, inorder, and post order traversal
  - Big O time and space?
- ▶ Implement a level order traversal using a queue
  - Big O time and space?
- ▶ Implement a level order traversal without a queue
  - target depth

CS314

Binary Trees

18

## Breadth First Search Depth First Search

- ▶ from NIST - DADS
- ▶ **breadth first search:** Any search algorithm that considers neighbors of a *vertex* (node), that is, outgoing *edges* (links) of the vertex's predecessor in the search, before any outgoing edges of the vertex
- ▶ **depth first search:** Any search algorithm that considers outgoing *edges* (links of *children*) of a *vertex* (node) before any of the vertex's (node) *siblings*, that is, outgoing edges of the vertex's predecessor in the search. Extremes are searched first.

## Clicker 6

- ▶ Which traversal of a tree is a breadth first search?
  - A. Level order traversal
  - B. Pre order traversal
  - C. In order traversal
  - D. Post order traversal
  - E. More than one of these

CS314

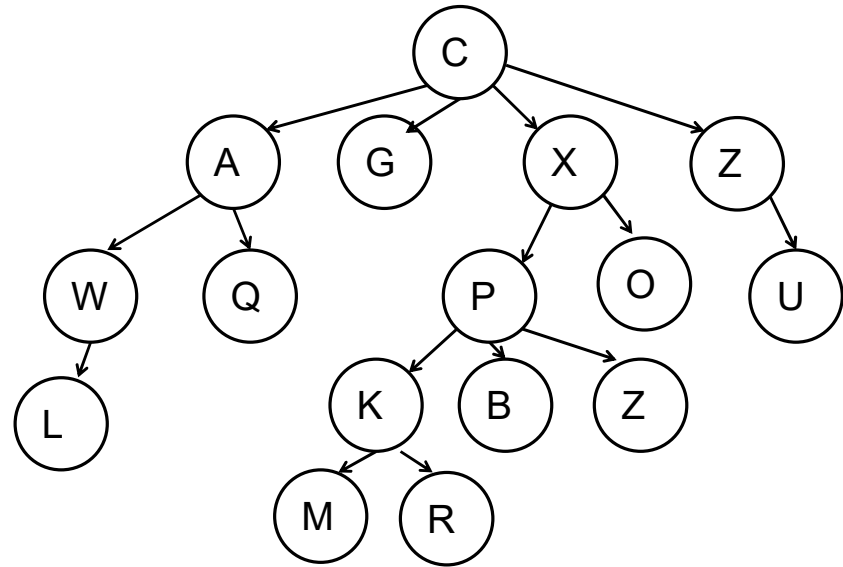
Binary Trees

20

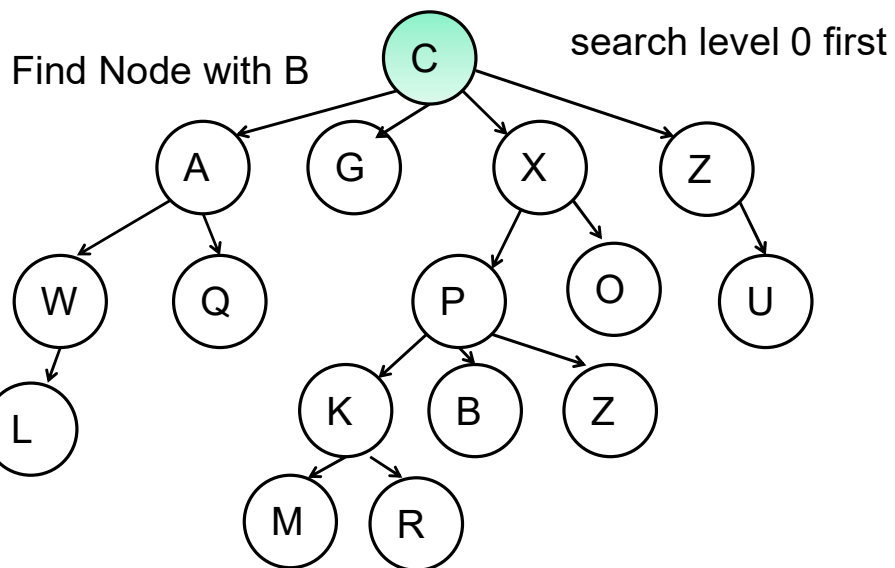
## Breadth First

- ▶ A level order traversal of a tree could be used as a breadth first search
- ▶ Search all nodes in a level before going down to the next level

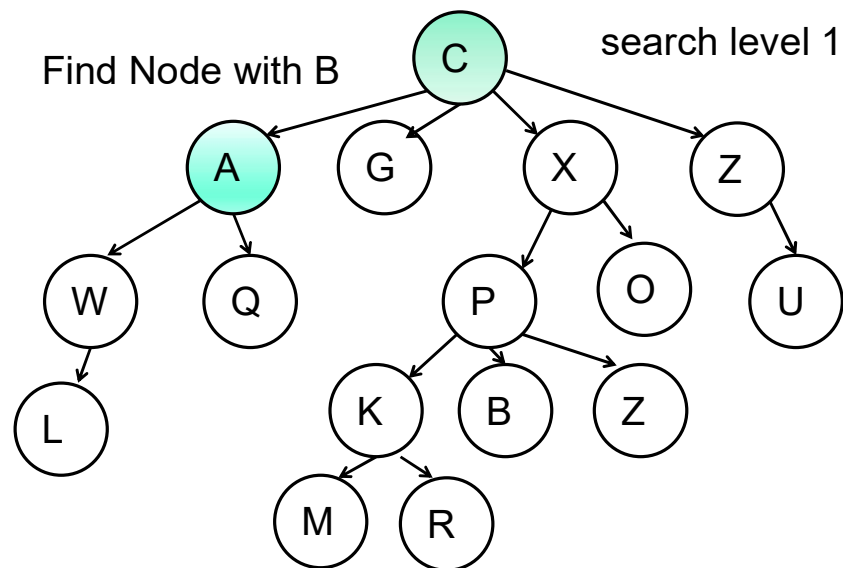
## Breadth First Search of Tree



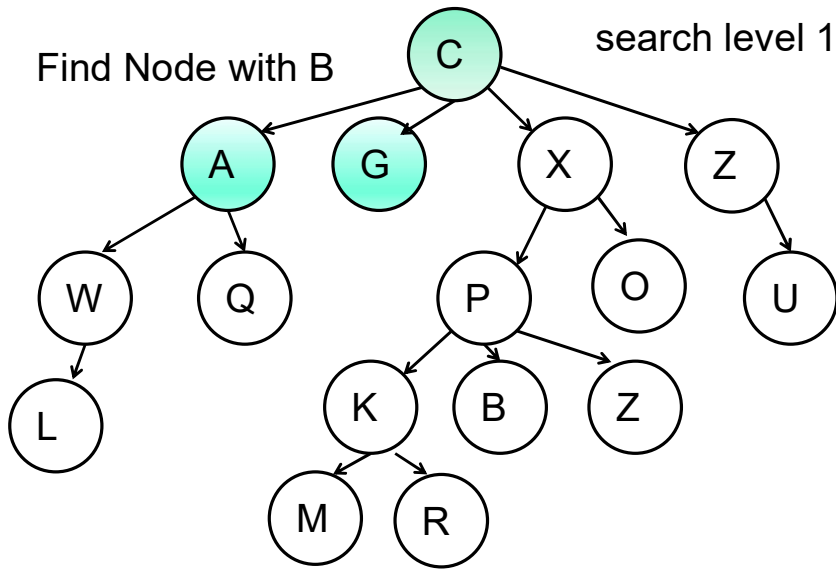
## Breadth First Search



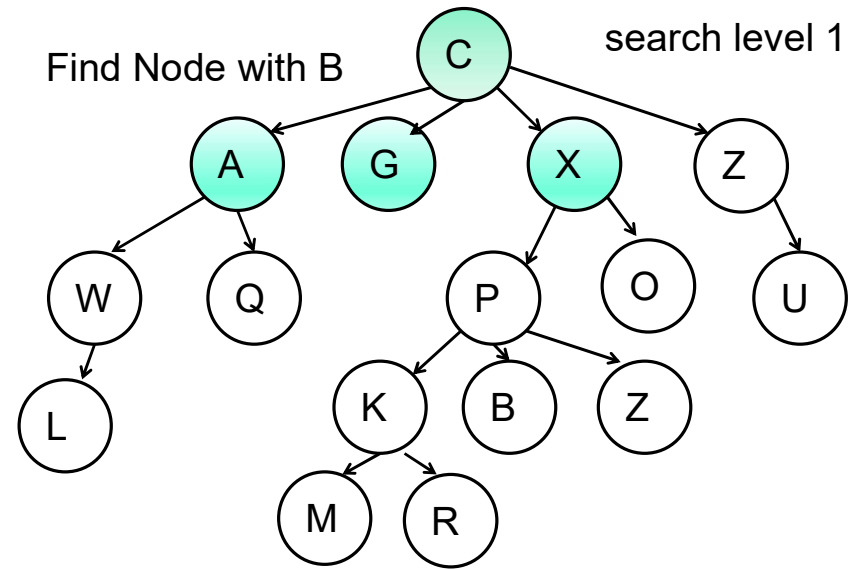
## Breadth First Search



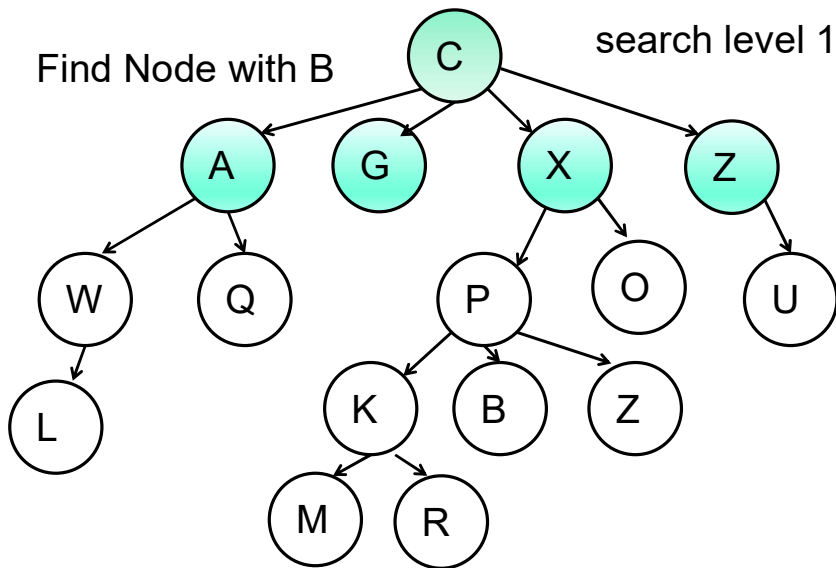
## Breadth First Search



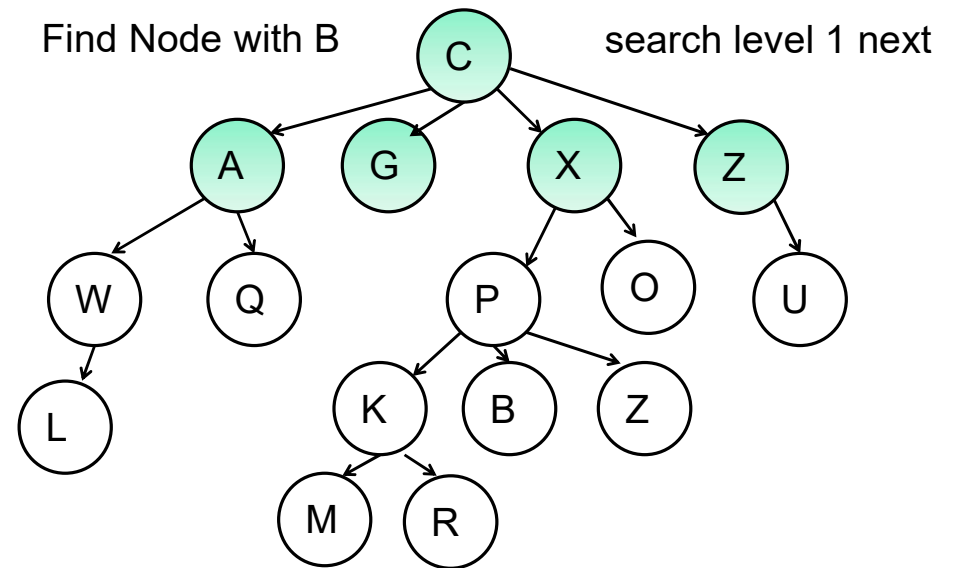
## Breadth First Search



## Breadth First Search



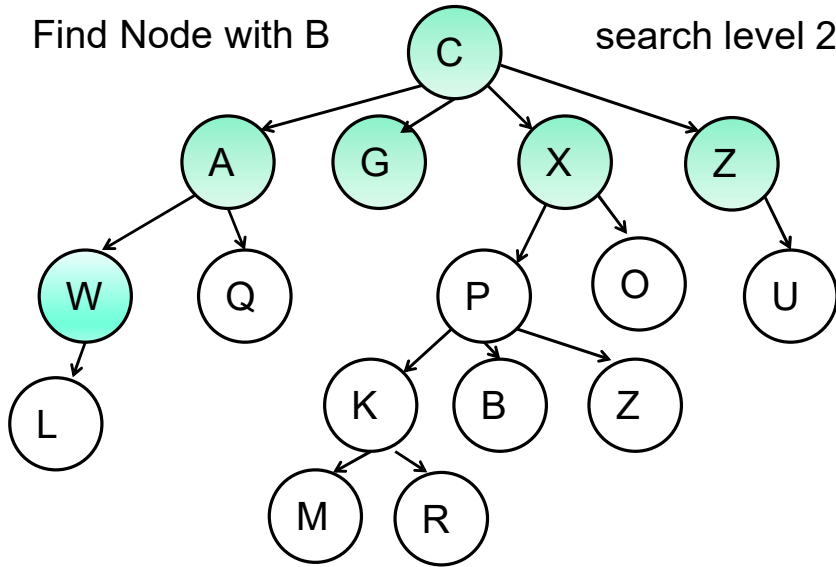
## Breadth First Search



## Breadth First Search

Find Node with B

search level 2 next



CS314

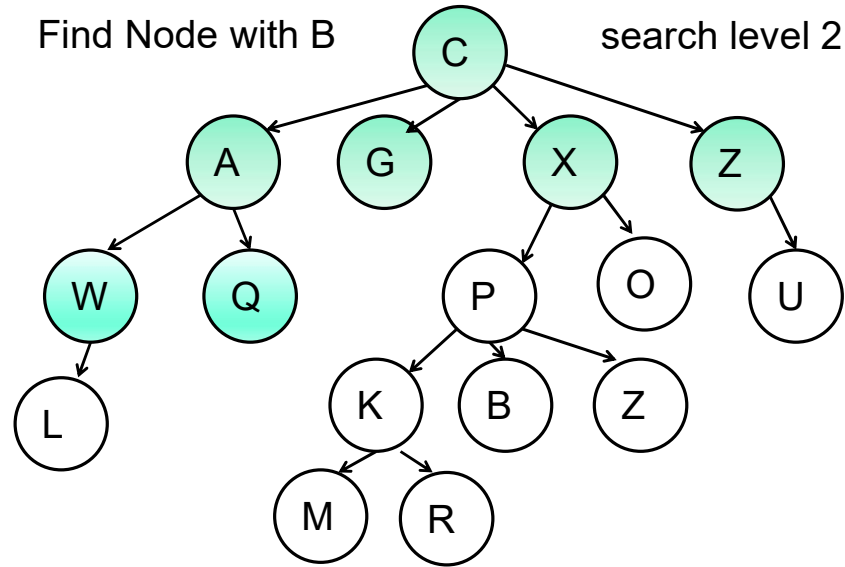
Binary Trees

29

## Breadth First Search

Find Node with B

search level 2 next



CS314

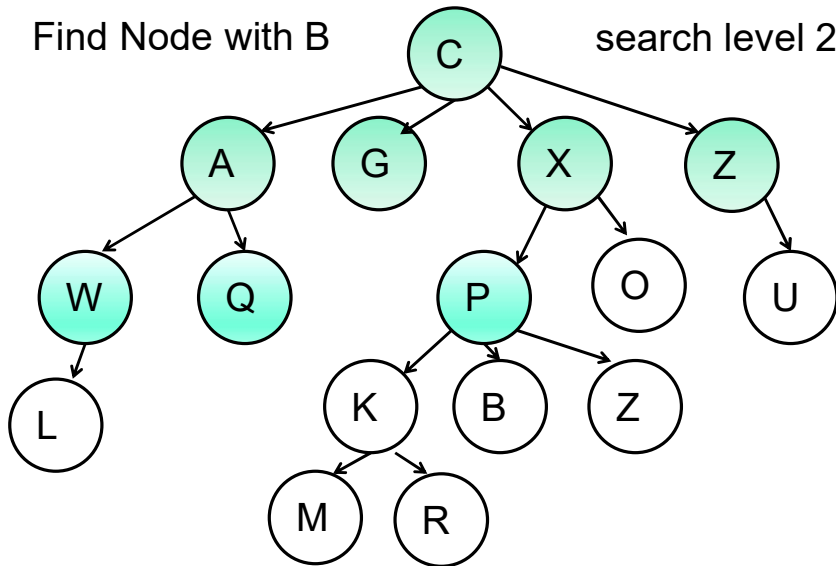
Binary Trees

30

## Breadth First Search

Find Node with B

search level 2 next



CS314

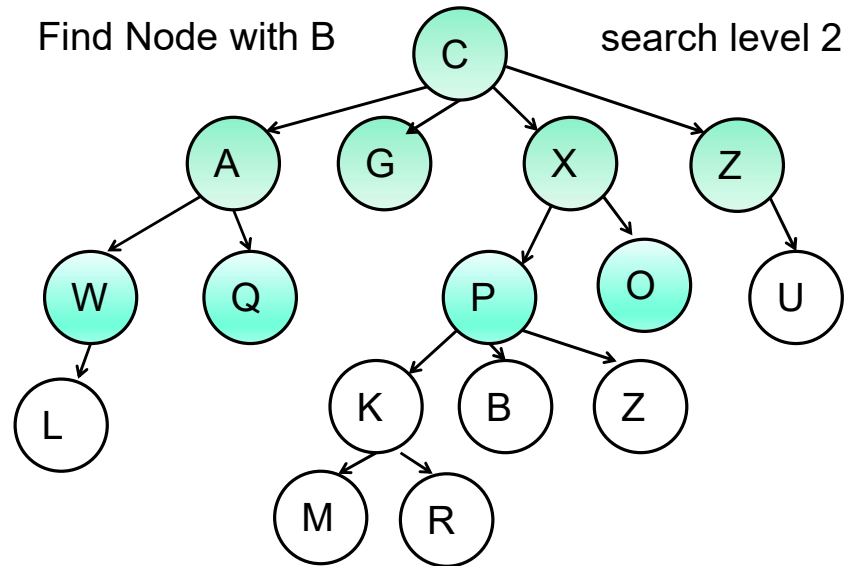
Binary Trees

31

## Breadth First Search

Find Node with B

search level 2 next



CS314

Binary Trees

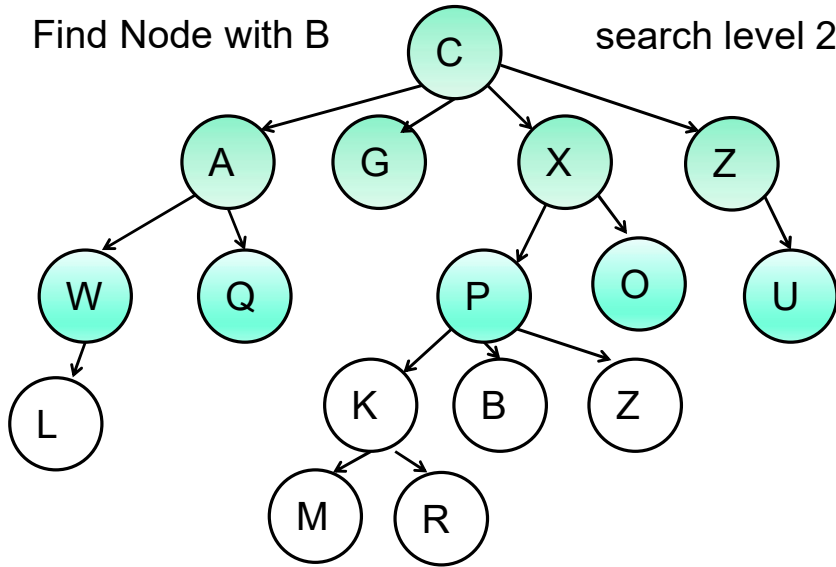
32



## Breadth First Search

Find Node with B

search level 2 next



CS314

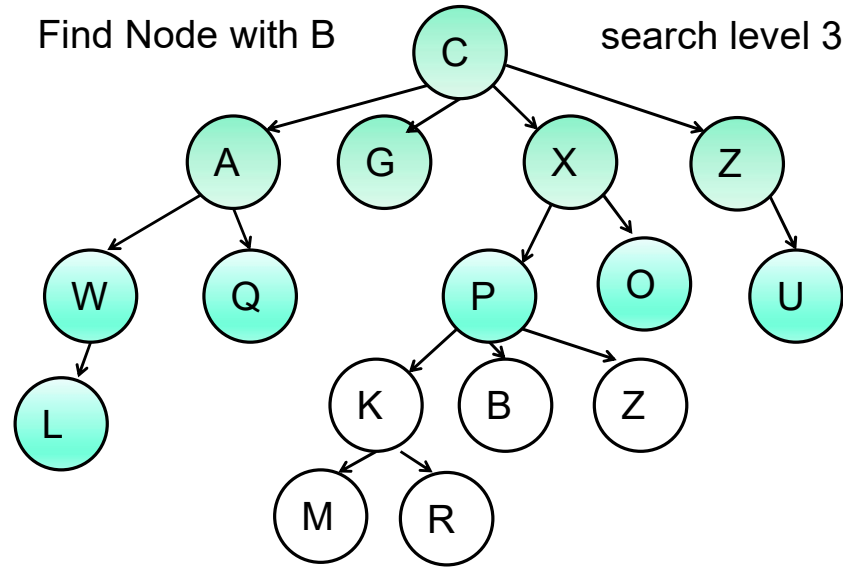
Binary Trees

33

## Breadth First Search

Find Node with B

search level 3 next



CS314

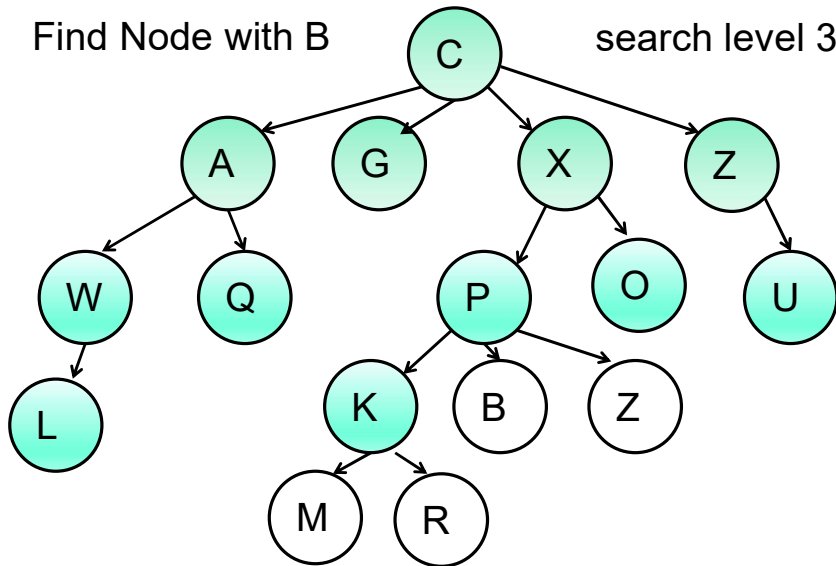
Binary Trees

34

## Breadth First Search

Find Node with B

search level 3 next



CS314

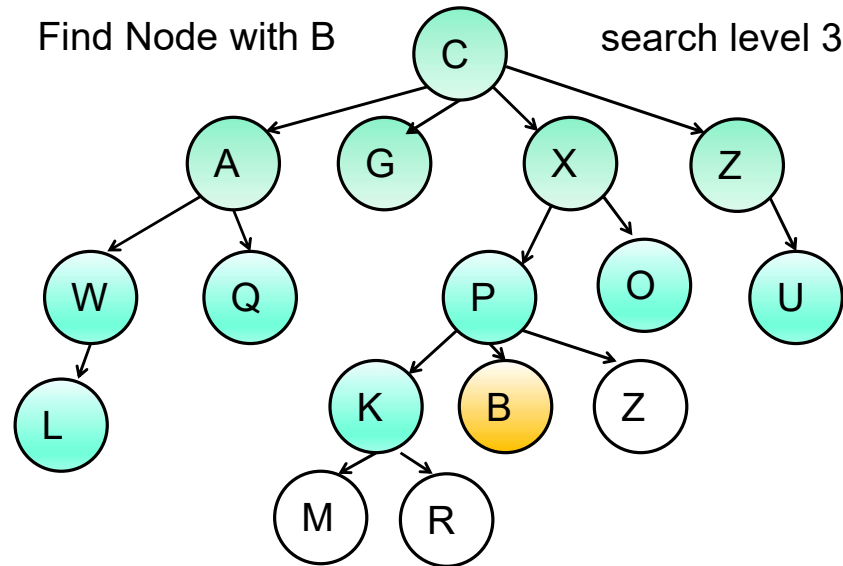
Binary Trees

35

## Breadth First Search

Find Node with B

search level 3 next



CS314

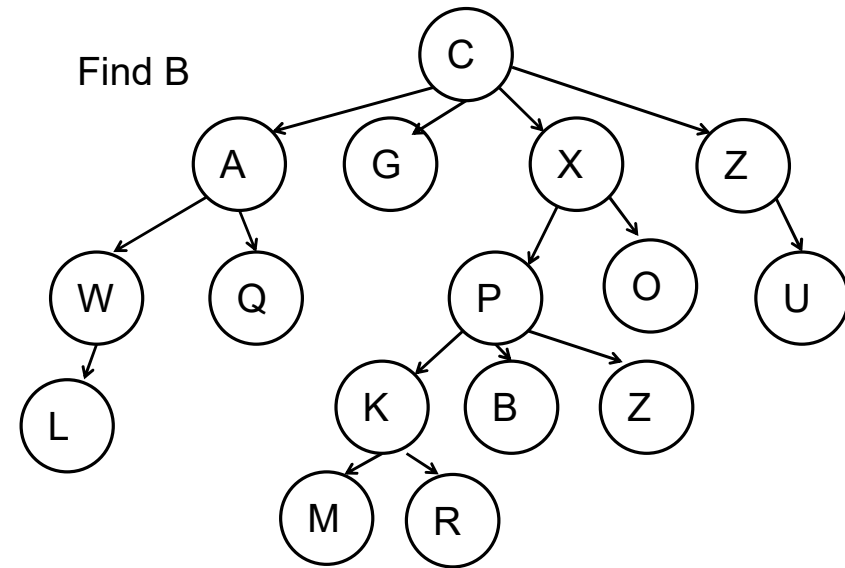
Binary Trees

36

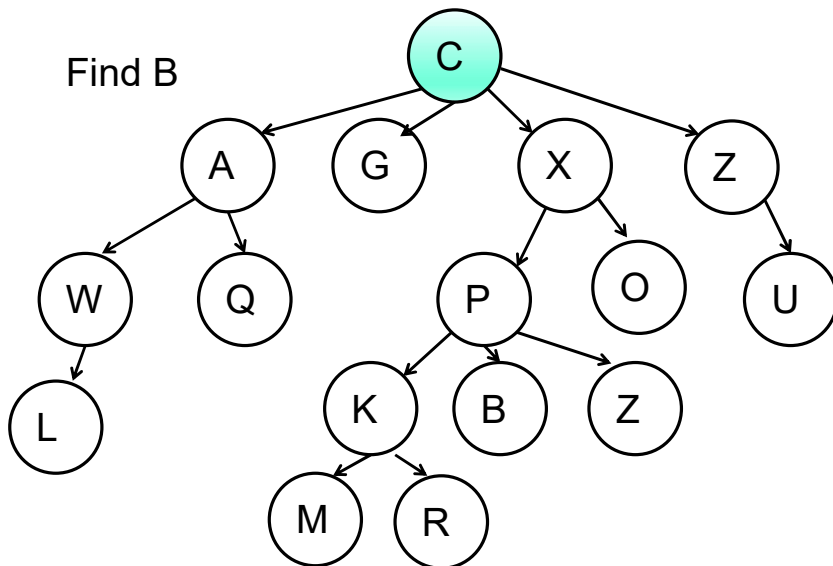
## BFS - DFS

- ▶ Breadth first search typically implemented with a Queue
- ▶ Depth first search typically implemented with a stack, implicit with recursion or iteratively with an explicit stack
- ▶ which technique do I use?
  - depends on the problem

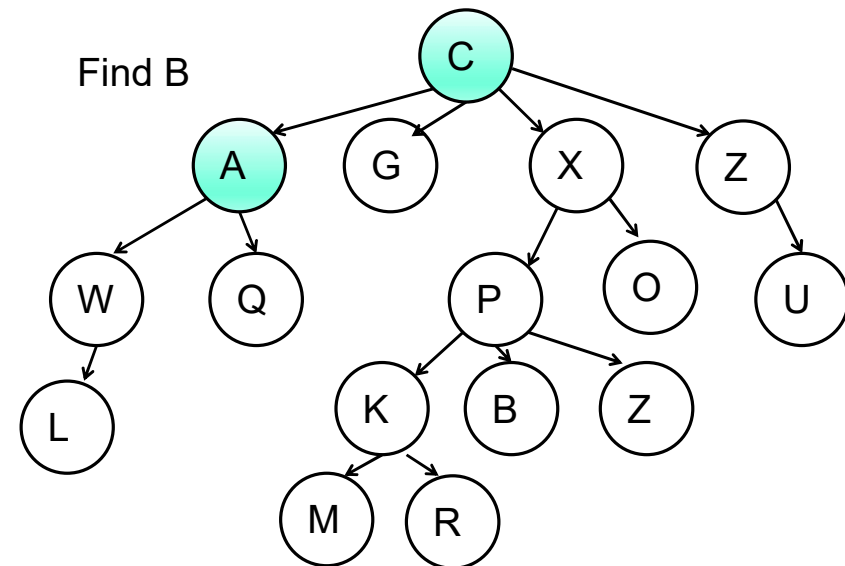
## Depth First Search of Tree



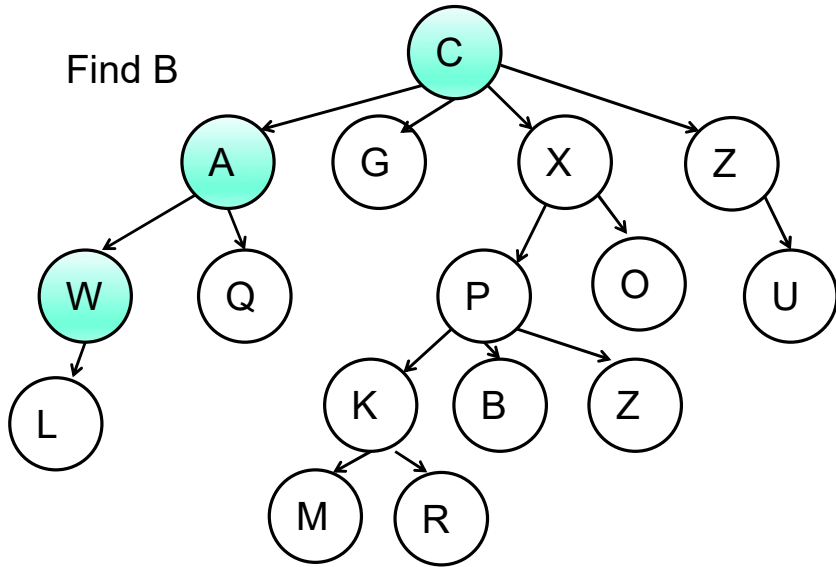
## Depth First Search of Tree



## Depth First Search of Tree



## Depth First Search of Tree

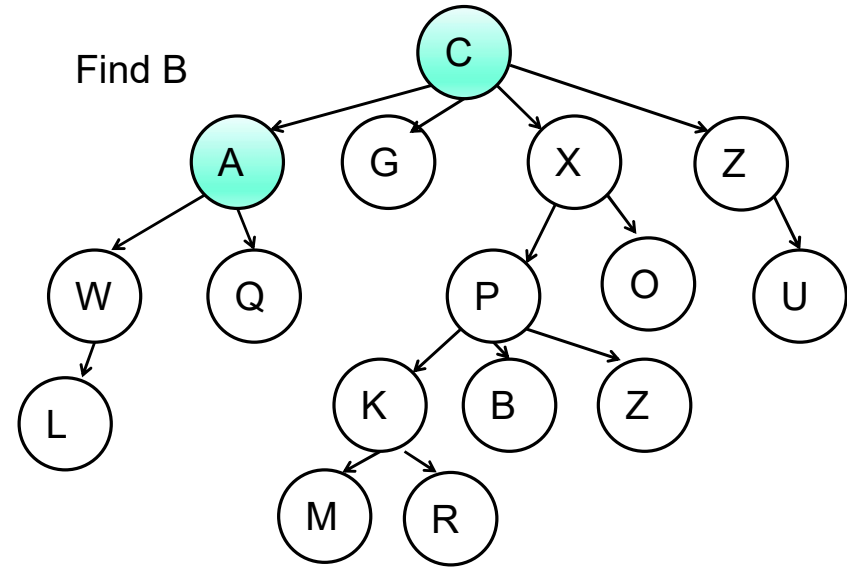


CS314

Binary Trees

41

## Depth First Search of Tree

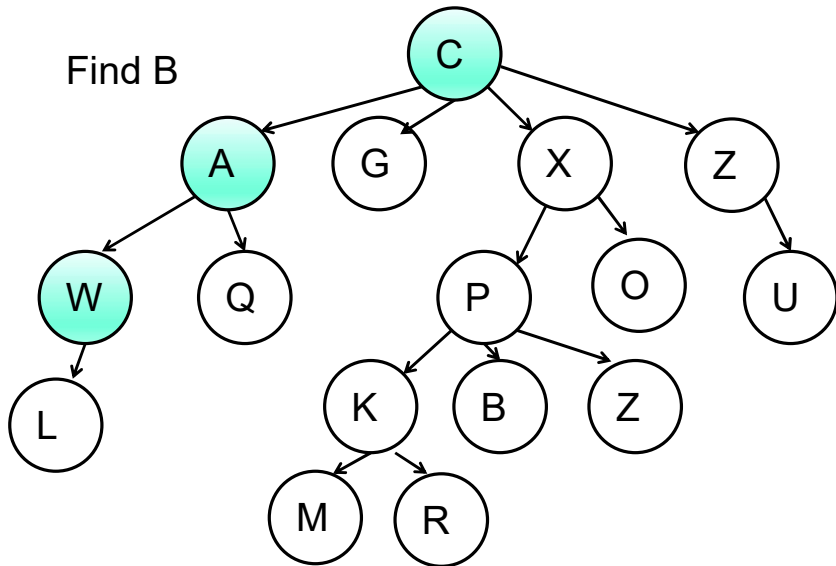


CS314

Binary Trees

42

## Depth First Search of Tree

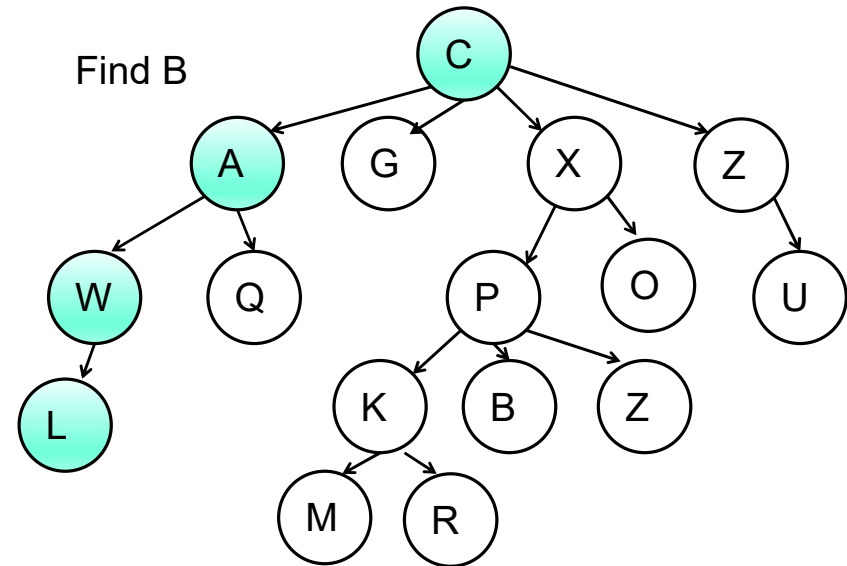


CS314

Binary Trees

43

## Depth First Search of Tree

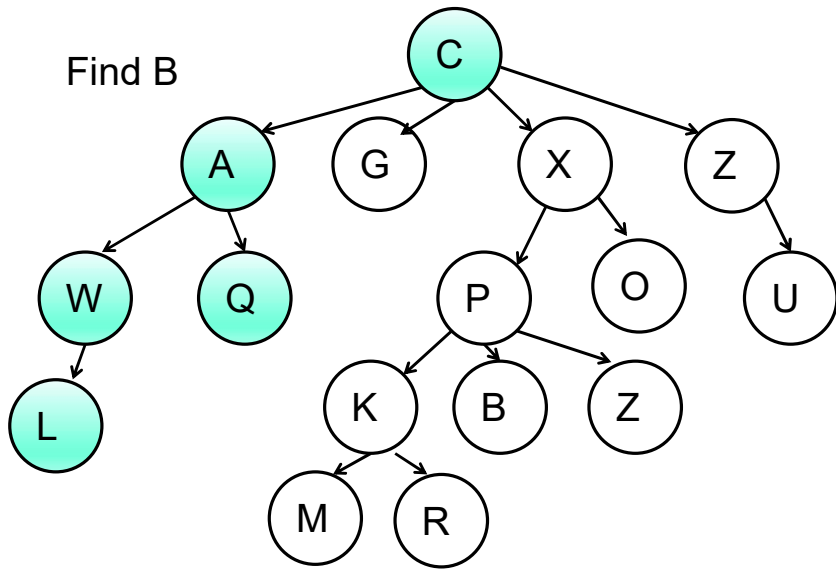


CS314

Binary Trees

44

## Depth First Search of Tree

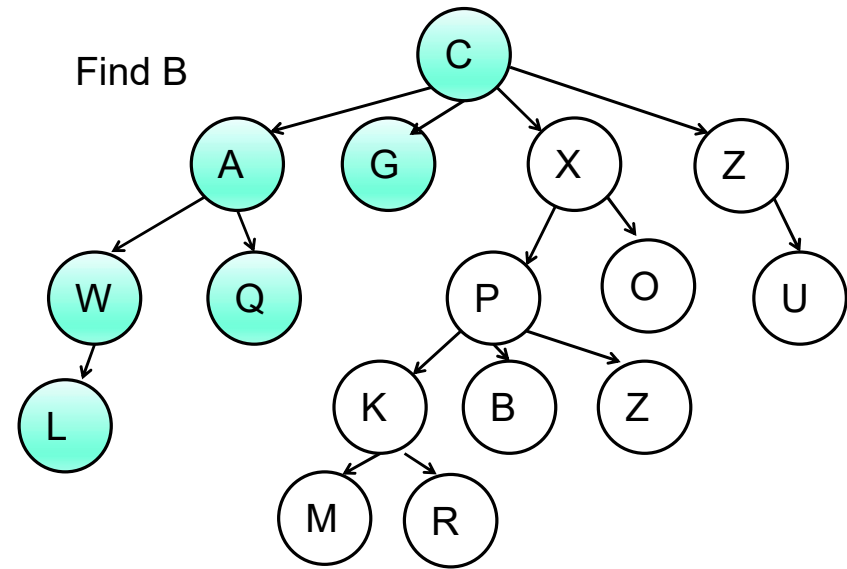


CS314

Binary Trees

45

## Depth First Search of Tree

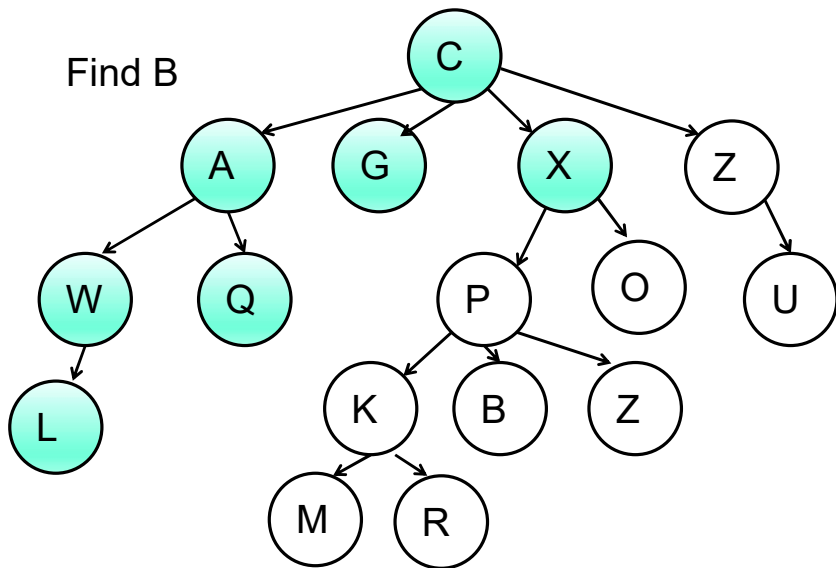


CS314

Binary Trees

46

## Depth First Search of Tree

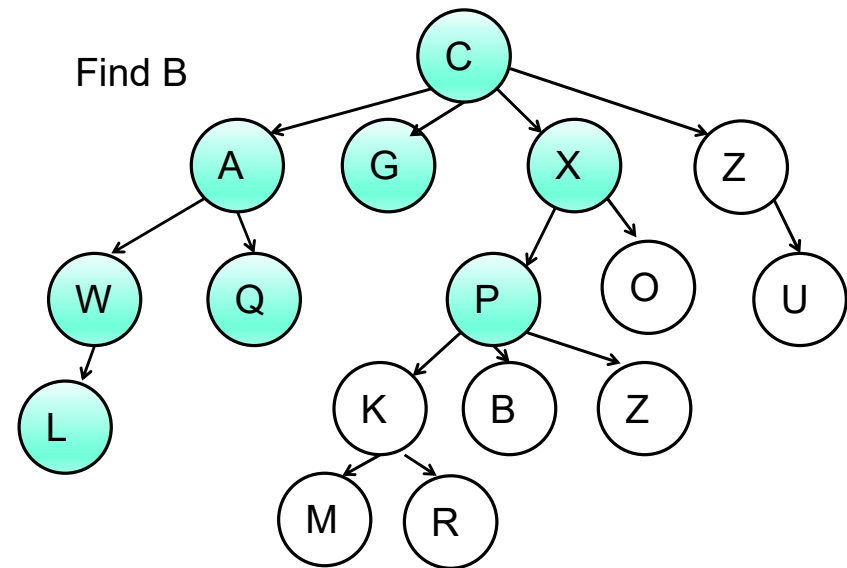


CS314

Binary Trees

47

## Depth First Search of Tree

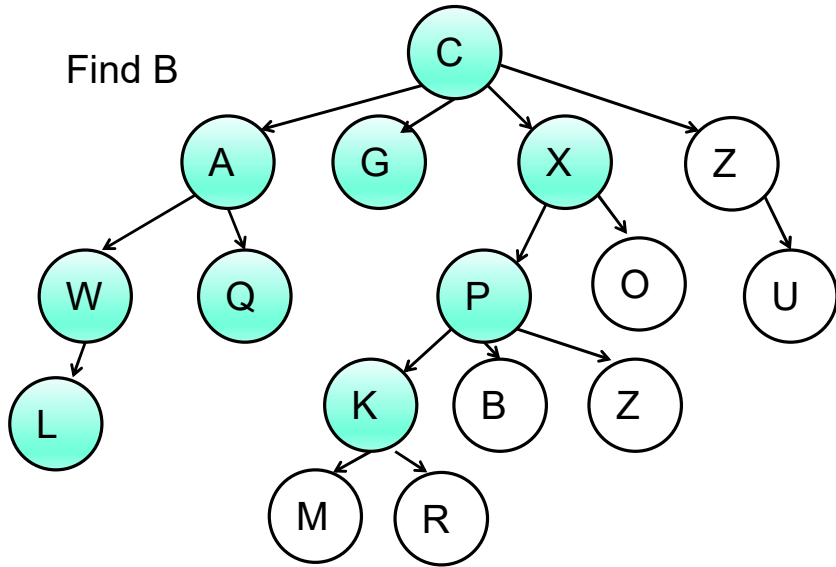


CS314

Binary Trees

48

## Depth First Search of Tree

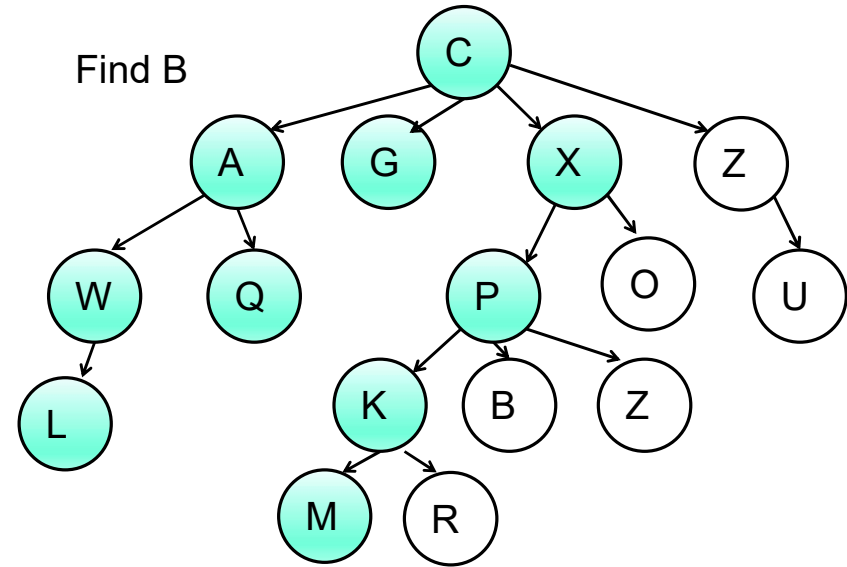


CS314

Binary Trees

49

## Depth First Search of Tree

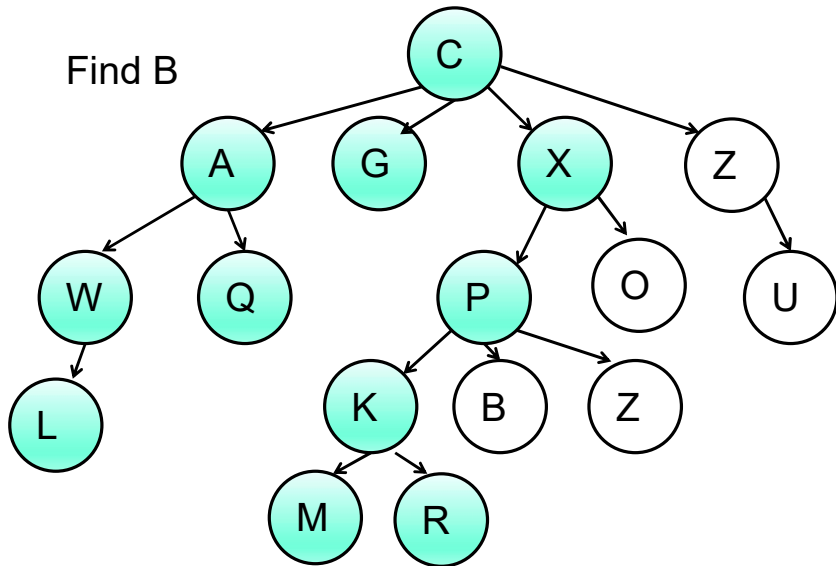


CS314

Binary Trees

50

## Depth First Search of Tree

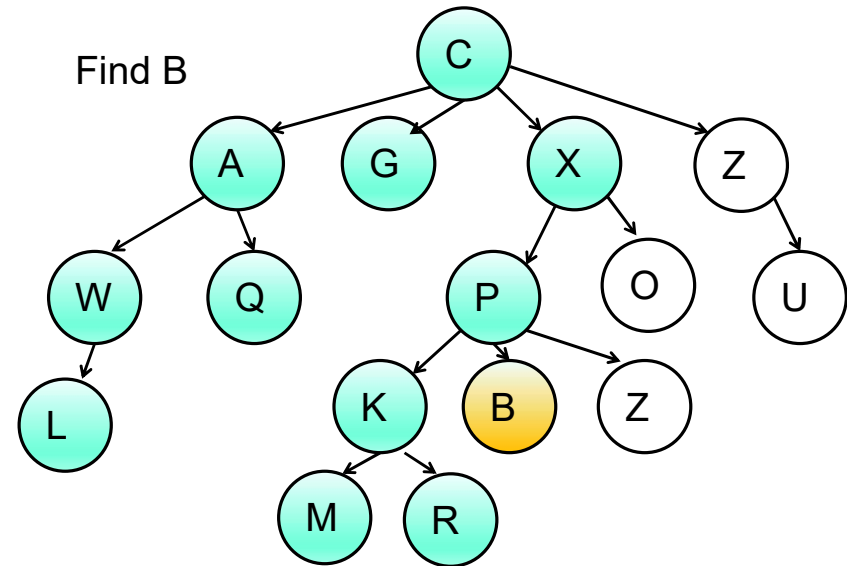


CS314

Binary Trees

51

## Depth First Search of Tree



CS314

Binary Trees

52