Binary Tree - Recursion

Discussion 06/29/2017

Recursion

- Recursion is the strategy for solving problems where a method calls itself.
- Approach

  - If the problem is straightforward, solve it directly (base case – the last step to stop the recursion).
  - Else (recursive step)
    1. Simplify the problem into smaller problems.
    2. Solve the simpler problems using the same algorithm.
    3. Combine the solutions of the smaller problems that solve the general problem.

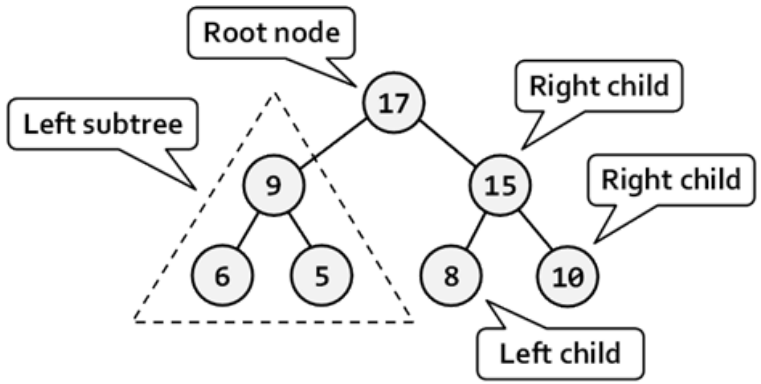
Recursion – Array Addition Example

- Given an array of [2, 3, 4, 5, 6, 7], implement a recursive method add(int[] a) that calculates the sum of all integers in the array.
- Answer:

```
public int add(int[] a) {
        return add_helper(a, 0);      //Need a helper method to access each
                                      //element
}
private int add_helper(int[] a, int index) {
        if (index == a.length - 1) return a[index]; //Base case
        else return a[index] + add_helper(a, ++index); //Recursive step
}
```

Binary Tree

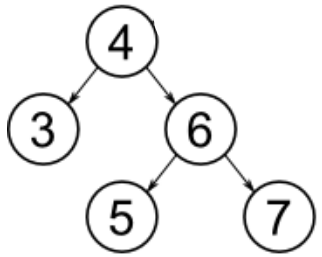
- Definition: Binary Tree is a data structure that has a root node and each node in the tree has at most two subtrees, which are referred to the left child and right child.
- Example:



Binary Tree Traversal

- Breadth-first traversal (BFS) visits node according to how far away from the root.
- Depth-first traversal (DFS) visits nodes as far ahead as possible before backing up. There are three types of DFS for binary trees:
  - Preorder traversal visits a node, then its left child, then its right child.
  - Inorder traversal visits a node's left child, then the node itself, then its right child.
  - Postorder traversal visits a node's left child, then its right child, then itself.

Binary Tree Traversal Example



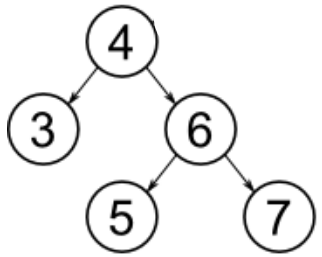
Breadth-first:

Preorder:

Inorder:

Postorder:

Binary Tree Traversal Example



Breadth-first: 4, 3, 6, 5, 7

Preorder: 4, 3, 6, 5, 7

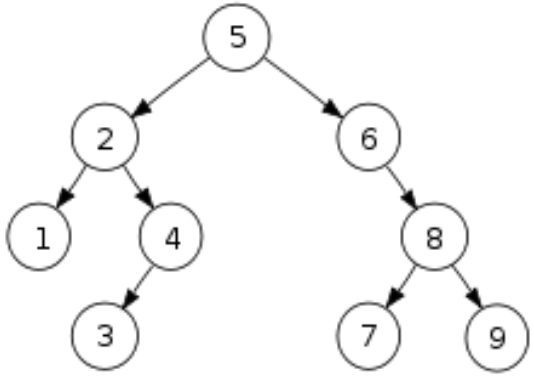
Inorder: 3, 4, 5, 6, 7

Postorder: 3, 5, 7, 6, 4

Visualization link: https://visualgo.net/en/bst

Which one of these is breadth-first traversal?

A. 1, 2, 3, 4, 5, 6, 7, 8, 9
B. 5, 2, 6, 1, 4, 8, 3, 7, 9
C. 1, 3, 4, 2, 7, 9, 8, 6, 5
D. 5, 3, 2, 4, 3, 6, 8, 7, 9



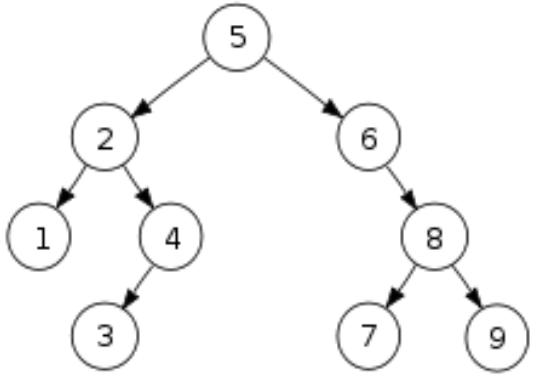
Which one of these is breadth-first traversal?

A. 1, 2, 3, 4, 5, 6, 7, 8, 9

B. 5, 2, 6, 1, 4, 8, 3, 7, 9

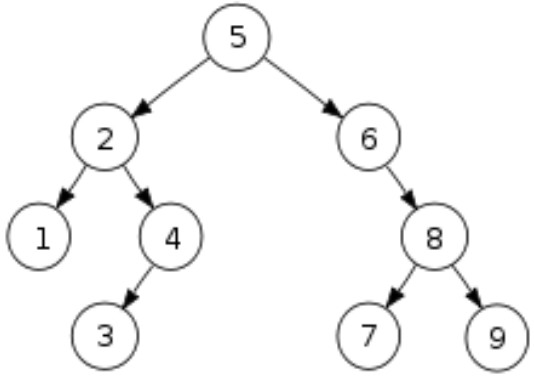
C. 1, 3, 4, 2, 7, 9, 8, 6, 5

D. 5, 3, 2, 4, 3, 6, 8, 7, 9



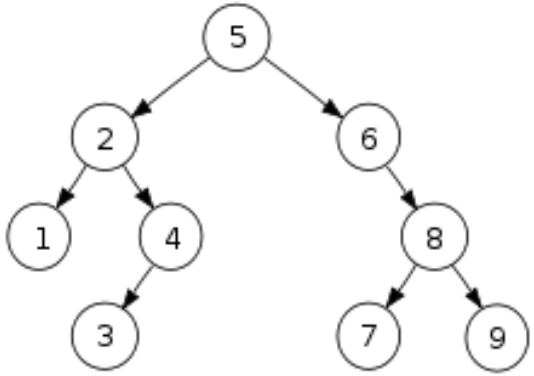
Which one of these is postorder traversal?

A. 1, 3, 4, 2, 7, 9, 8, 6, 5
B. 5, 2, 6, 1, 4, 8, 3, 7, 9
C. 1, 2, 3, 4, 5, 6, 7, 8, 9
D. 5, 3, 2, 4, 3, 6, 8, 7, 9



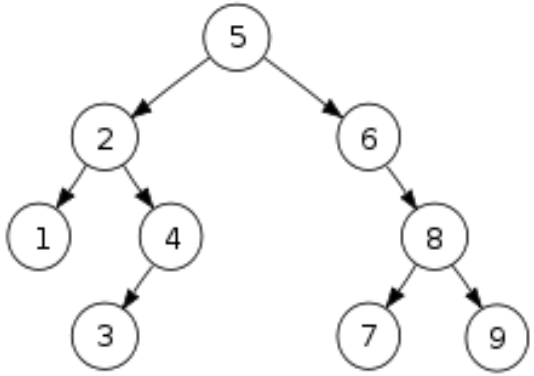
Which one of these is postorder traversal?

A. 1, 3, 4, 2, 7, 9, 8, 6, 5
B. 5, 2, 6, 1, 4, 8, 3, 7, 9
C. 1, 2, 3, 4, 5, 6, 7, 8, 9
D. 5, 3, 2, 4, 3, 6, 8, 7, 9



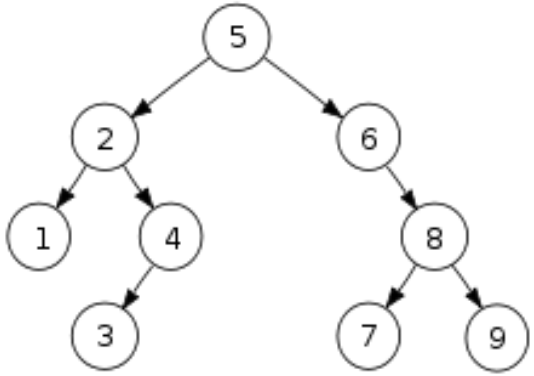
Which one of these is inorder traversal?

A. 1, 2, 3, 4, 5, 6, 7, 8, 9
B. 5, 2, 6, 1, 4, 8, 3, 7, 9
C. 1, 3, 4, 2, 7, 9, 8, 6, 5
D. 5, 3, 2, 4, 3, 6, 8, 7, 9



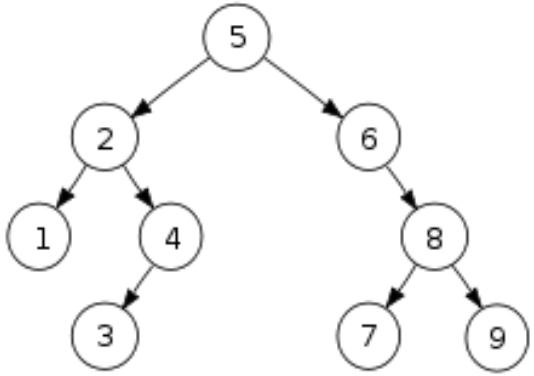
Which one of these is inorder traversal?

A. 1, 2, 3, 4, 5, 6, 7, 8, 9
B. 5, 2, 6, 1, 4, 8, 3, 7, 9
C. 1, 3, 4, 2, 7, 9, 8, 6, 5
D. 5, 3, 2, 4, 3, 6, 8, 7, 9



Which one of these is preorder traversal?

A. 1, 3, 4, 2, 7, 9, 8, 6, 5
B. 5, 2, 6, 1, 4, 8, 3, 7, 9
C. 1, 2, 3, 4, 5, 6, 7, 8, 9
D. 5, 3, 2, 4, 3, 6, 8, 7, 9



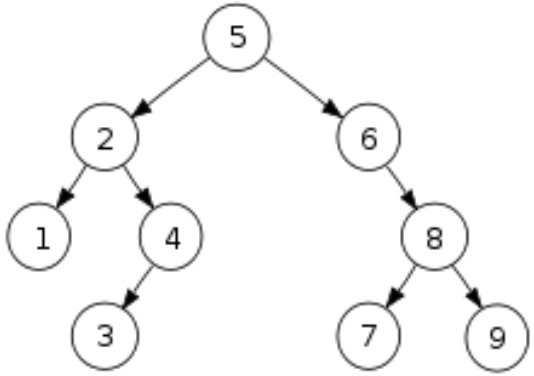
Which one of these is preorder traversal?

A. 1, 3, 4, 2, 7, 9, 8, 6, 5

B. 5, 2, 1, 4, 3, 6, 8, 7, 9

C. 1, 2, 3, 4, 5, 6, 7, 8, 9

D. 5, 3, 2, 4, 3, 6, 8, 7, 9



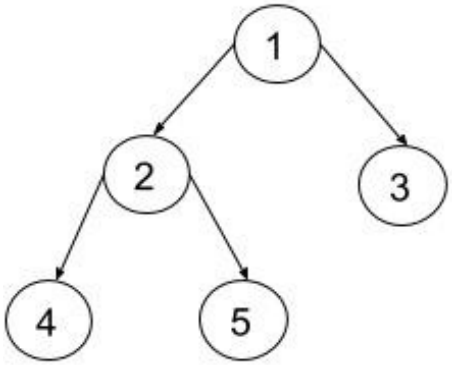
Tree Construction Algorithm

1. Get the root from the pre (post) order traversal.
2. Locate the root in the inorder traversal.
3. Determine the sizes of your left and right subtrees from the inorder traversal, and obtain the inorder traversal of the left and right subtrees.
4. Compute the split index of your pre (post) order traversal to get the left and right preorder traversal of your left and right subtrees.
5. Create the root node and solve recursively for its left and right subtrees.

Binary Tree

Exercise 1: Implement findMaxSum() method that find the maximum sum of all paths (each path has their own sum and find max sum of those sums). For example, the path 1->2->5 makes the max sum of 8 and 8 is the result.

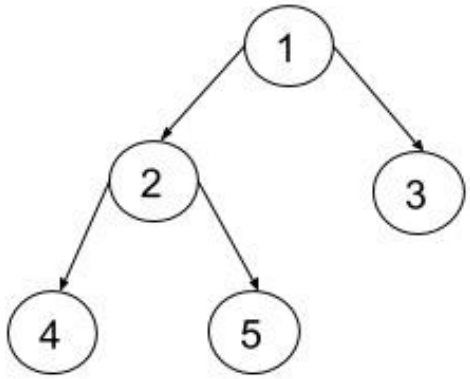
```
int findMaxSum(Node n) {


}
```



Binary Tree

Exercise 1: Implement findMaxSum() method that find the maximum sum of all paths (each path has their own sum and find max sum of those sums). For example, the path 1->2->5 makes sum of 8; 1->2>4 makes sum of 7; and 1->3 makes sum of 4. Therefore, 8 is the result.

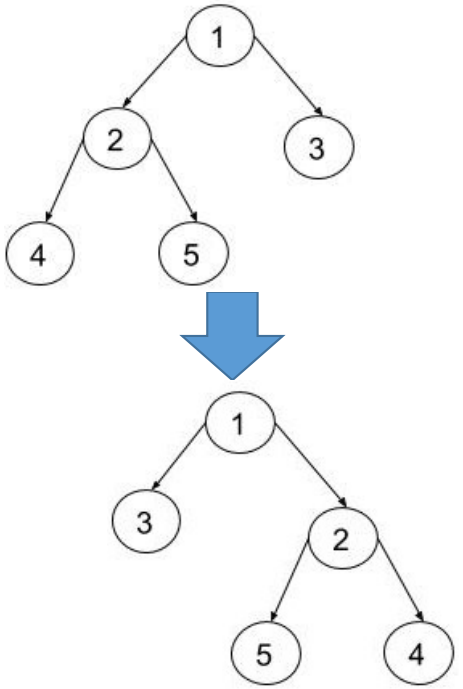
```
int findMaxSum(Node n) {
        if (n == null) return 0;
        else {
                int sumleft = findMaxSum(n.left);
                int sumright = findMaxSum(n.right);
                if (sumleft > sumright) return n.data + sumleft;
                else return n.data + sumright;
        }
}
```



Binary Tree

- Exercise 2: Implement mirror() method that replaces the current binary tree with its own mirror version.

```
void mirror(Node n) {


}
```



Binary Tree

- Exercise 2: Implement mirror() method that replaces the current binary tree with its own mirror version.

```
void mirror(Node n) {
        if (n != null) {
                mirror(n.left);
                mirror(n.right);
                Node temp = n.right;
                n.right = n.left;
                n.left = temp;
        }
}
```