"Binary Trees using Array".

1. How many children does a binary tree have?
a) 2
b) any number of children
c) 0 or 1 or 2
d) 0 or 1
View Answer
Answer: c
Explanation: Can have atmost 2 nodes.

2. What is/are the disadvantages of implementing tree using normal arrays?
a) difficulty in knowing children nodes of a node
b) difficult in finding the parent of a node
c) have to know the maximum number of nodes possible before creation of trees
d) difficult to implement
View Answer
Answer: c
Explanation: The size of array is fixed in normal arrays. We need to know the number of nodes in the tree before array declaration. It is the main disadvantage of using arrays to represent binary trees.

3. What must be the ideal size of array if the height of tree is 'l'?
a) 2^l-1
b) l-1
c) l
d) 2l
View Answer
Answer: a
Explanation: Maximum elements in a tree (complete binary tree in worst case) of height 'L' is 2^L-1. Hence size of array is taken as 2^L-1.

4. What are the children for node 'w' of a complete-binary tree in an array representation?
a) 2w and 2w+1
b) 2+w and 2-w
c) w+1/2 and w/2
d) w-1/2 and w+1/2
View Answer

Answer: a
Explanation: The left child is generally taken as 2*w whereas the right child will be taken as 2*w+1 because root node is present at index 0 in the array and to access every index position in the array.

5. What is the parent for a node ‘w’ of a complete binary tree in an array representation when w is not 0?
a) floor(w-1/2)
b) ceil(w-1/2)
c) w-1/2
d) w/2
View Answer
Answer: a
Explanation: Floor of w-1/2 because we can’t miss a node.

6. If the tree is not a complete binary tree then what changes can be made for easy access of children of a node in the array?
a) every node stores data saying which of its children exist in the array
b) no need of any changes continue with 2w and 2w+1, if node is at i
c) keep a seperate table telling children of a node
d) use another array parallel to the array with tree
View Answer
Answer: a
Explanation: Array cannot represent arbitrary shaped trees. It can only be used in case of complete trees. If every node stores data saying that which of its children exists in the array then elements can be accessed easily.

7. What must be the missing logic in place of missing lines for finding sum of nodes of binary tree in alternate levels?

```
//e.g:-consider -complete binary tree:-height-3, [1,2,3,4,5,6,7]-answer must be 23
n=power(2,height)-1; //assume input is height and a[i] contains tree elements
for(i=1;i<=n;)
{
      //present level is initialized to 1 and sum is initialized to  0
      for(j=1;j<=pow(2,currentlevel-1);j++)
      {
         sum=sum+a[i];
         i=i+1;
      }
```

```
    //missing logic
}
```

a)

```
i=i+pow(2,currentlevel);
currentlevel=currentlevel+2;
j=1;
```

b)

```
i=i+pow(2,currentlevel);
currentlevel=currentlevel+2;
j=0;
```

c)

```
i=i-pow(2,currentlevel);
currentlevel=currentlevel+2;
j=1;
```

d)

```
i=i+pow(2,currentlevel);
currentlevel=currentlevel+1;
j=1;
```

View Answer

Answer: a

Explanation: The i value must skip through all nodes in the next level and current level must be one+next level.

8. Consider a situation of writing a binary tree into a file with memory storage efficiency in mind, is array representation of tree is good?

a) yes because we are overcoming the need of pointers and so space efficiency

b) yes because array values are indexable

c) No it is not efficient in case of sparse trees and remaning cases it is fine

d) No linked list representation of tree is only fine

View Answer

Answer: c

Explanation: In case of sparse trees (where one node per level in worst cases), the array

size (2^h)-1 where h is height but only h indexes will be filled and (2^h)-1-h nodes will be left unused leading to space wastage.

9. Why is heap implemented using array representations than tree(linked list) representations though both tree representations and heaps have same complexities?

```
for binary heap
-insert: O(log n)
-delete min: O(log n)

for a tree
-insert: O(log n)
-delete: O(log n)
```

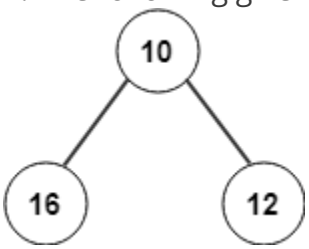
Then why go with array representation when both are having same values ?
a) arrays can store trees which are complete and heaps are not complete
b) lists representation takes more memory hence memory efficiency is less and go with arrays and arrays have better caching
c) lists have better caching
d) In lists insertion and deletion is difficult
View Answer
Answer: b
Explanation: In memory the pointer address for next node may not be adjacent or nearer to each other and also array have wonderful caching power from os and manipulating pointers is a overhead. Heap data structure is always a complete binary tree.

10. Can a tree stored in an array using either one of inorder or post order or pre order traversals be again reformed?
a) Yes just traverse through the array and form the tree
b) No we need one more traversal to form a tree
c) No in case of sparse trees
d) Yes by using both inorder and array elements
View Answer
Answer: b
Explanation: We need any two traversals for tree formation but if some additional stuff or techniques are used while storing a tree in an array then one traversal can facilitate like also storing null values of a node in array.

“Binary Tree Operations”.

1. What is the maximum number of children that a binary tree node can have?
a) 0
b) 1
c) 2
d) 3
View Answer
Answer: c
Explanation: In a binary tree, a node can have atmost 2 nodes (i.e.) 0,1 or 2 left and right child.

2. The following given tree is an example for?



a) Binary tree
b) Binary search tree
c) Fibonacci tree
d) AVL tree
View Answer
Answer: a
Explanation: The given tree is an example for binary tree since has got two children and the left and right children do not satisfy binary search tree’s property, Fibonacci and AVL tree.

3. A binary tree is a rooted tree but not an ordered tree.
a) true
b) false
View Answer
Answer: b
Explanation: A binary tree is a rooted tree and also an ordered tree (i.e) every node in a binary tree has at most two children.

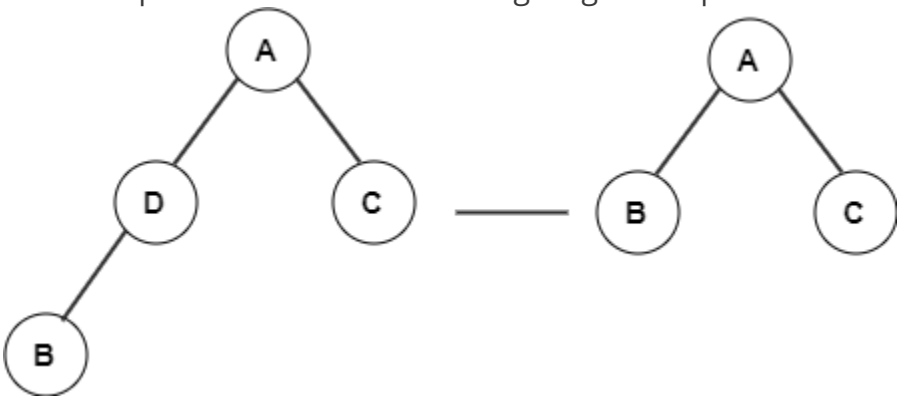
4. How many common operations are performed in a binary tree?
a) 1
b) 2
c) 3
d) 4
View Answer
Answer: c
Explanation: Three common operations are performed in a binary tree- they are insertion, deletion and traversal.

5. What is the traversal strategy used in the binary tree?
a) depth-first traversal
b) breadth-first traversal
c) random traversal
d) Priority traversal
View Answer
Answer: b
Explanation: Breadth first traversal, also known as level order traversal is the traversal strategy used in a binary tree. It involves visiting all the nodes at a given level.

Check this: Programming Books | Programming MCQs

6. How many types of insertion are performed in a binary tree?
a) 1
b) 2
c) 3
d) 4
View Answer
Answer: b
Explanation: Two kinds of insertion operation is performed in a binary tree- inserting a leaf node and inserting an internal node.

7. What operation does the following diagram depict?



a) inserting a leaf node
b) inserting an internal node
c) deleting a node with 0 or 1 child
d) deleting a node with 2 children
View Answer
Answer: c
Explanation: The above diagram is a depiction of deleting a node with 0 or 1 child since the node D which has 1 child is deleted.

8. General ordered tree can be encoded into binary trees.
a) true
b) false
View Answer
Answer: a
Explanation: General ordered tree can be mapped into binary tree by representing them in a left-child-right-sibling way.

9. How many bits would a succinct binary tree occupy?
a) n+O(n)
b) 2n+O(n)
c) n/2
d) n
View Answer
Answer: b
Explanation: A succinct binary tree occupies close to minimum possible space established by lower bounds. A succinct binary tree would occupy 2n+O(n) bits.

10. The average depth of a binary tree is given as?
a) O(N)
b) O(√N)
c) $O(N^2)$
d) O(log N)
View Answer
Answer: d
Explanation: The average depth of a binary tree is given as O(√N). In case of a binary search tree, it is O(log N).

11. How many orders of traversal are applicable to a binary tree (In General)?
a) 1
b) 4
c) 2

d) 3
View Answer
Answer: d
Explanation: The three orders of traversal that can be applied to a binary tree are in-order, pre-order and post order traversal.

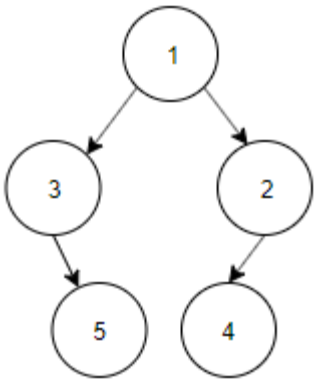
12. If binary trees are represented in arrays, what formula can be used to locate a left child, if the node has an index i?
a) 2i+1
b) 2i+2
c) 2i
d) 4i
View Answer
Answer: a
Explanation: If binary trees are represented in arrays, left children are located at indices 2i+1 and right children at 2i+2.

13. Using what formula can a parent node be located in an array?
a) (i+1)/2
b) (i-1)/2
c) i/2
d) 2i/2
View Answer
Answer: b
Explanation: If a binary tree is represented in an array, parent nodes are found at indices (i-1)/2.

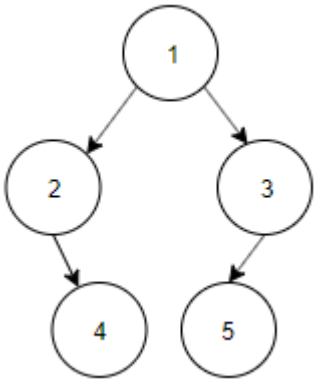
14. Which of the following properties are obeyed by all three tree – traversals?
a) Left subtrees are visited before right subtrees
b) Right subtrees are visited before left subtrees
c) Root node is visited before left subtree
d) Root node is visited before right subtree
View Answer
Answer: a
Explanation: In preorder, inorder and postorder traversal the left subtrees are visited before the right subtrees. In Inorder traversal, the Left subtree is visited first then the Root node then the Right subtree. In postorder traversal, the Left subtree is visited first, then Right subtree and then the Root node is visited.

15. Construct a binary tree using the following data.
The preorder traversal of a binary tree is 1, 2, 5, 3, 4. The inorder traversal of the same binary tree is 2, 5, 1, 4, 3.

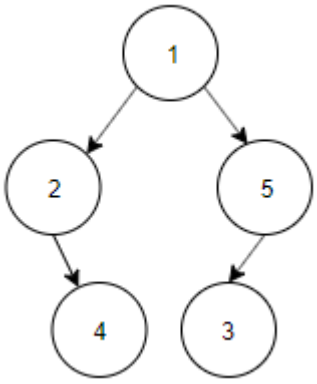
a)



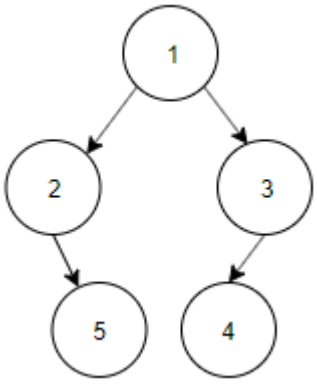
b)



c)

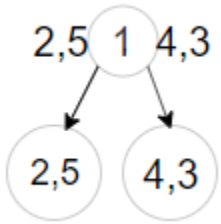


d)

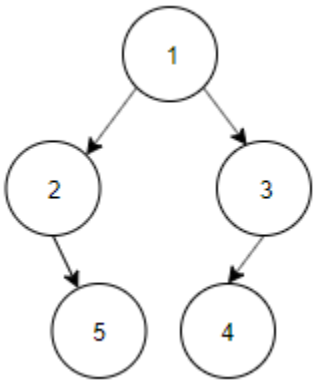


View Answer

Answer: d
Explanation: Here,
Preorder Traversal is 1, 2, 5, 3, 4
Inorder Traversal is 2, 5, 1, 4, 3
Root node of binary tree is the first node in Preorder traversal.
The rough sketch of tree is:

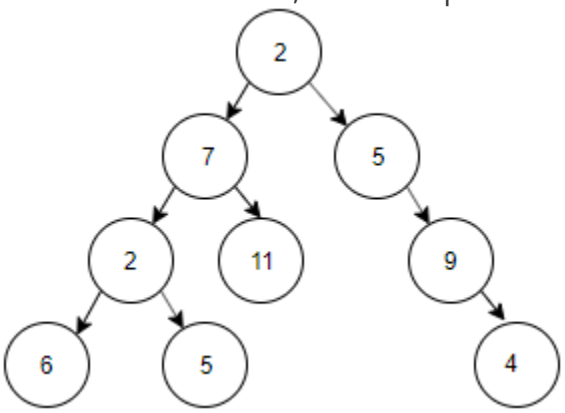


Second node in preorder traversal is 2. This makes 5 as right child to node 2. The fourth node in preorder traversal is 3. This makes 4 as right child to node 3. Thus the final tree is:



Preorder Traversal".

1. For the tree below, write the pre-order traversal.

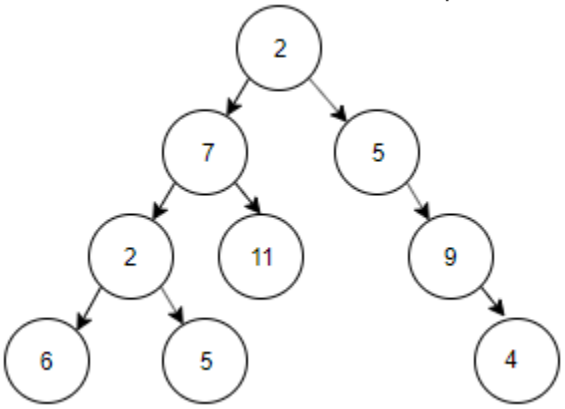


a) 2, 7, 2, 6, 5, 11, 5, 9, 4
b) 2, 7, 5, 2, 6, 9, 5, 11, 4
c) 2, 5, 11, 6, 7, 4, 9, 5, 2
d) 2, 7, 5, 6, 11, 2, 5, 4, 9

View Answer

Answer: a
Explanation: Pre order traversal follows NLR(Node-Left-Right).

2. For the tree below, write the post-order traversal.



a) 6, 2, 7, 2, 5, 11, 9, 5, 4
b) 6, 5, 11, 2, 7, 5, 9, 4, 2
c) 6, 5, 2, 11, 7, 4, 9, 5, 2
d) 6, 2, 7, 2, 11, 5, 5, 9, 4

View Answer

Answer: c
Explanation: Post order traversal follows LRN(Left-Right-Node).

3. Select the code snippet which performs pre-order traversal.
a)

```
public void preorder(Tree root)
{
          System.out.println(root.data);
          preorder(root.left);
          preorder(root.right);
}
```

b)


```
public void preorder(Tree root)
{
          preorder(root.left);
          System.out.println(root.data);
          preorder(root.right);
}
```

c)

```
public void preorder(Tree root)
{
          System.out.println(root.data);
          preorder(root.right);
          preorder(root.left);
}
```

d)

```
public void preorder(Tree root)
{
          preorder(root.right);
          preorder(root.left);
        System.out.println(root.data);
}
```

View Answer

Answer: a

Explanation: Pre-order traversal follows NLR(Node-Left-Right).

4. Select the code snippet which performs post-order traversal.

a)

```
public void postorder(Tree root)
{
          System.out.println(root.data);
          postorder(root.left);
          postorder(root.right);
}
```

b)

```
public void postorder(Tree root)
{
          postorder(root.left);
          postorder(root.right);
          System.out.println(root.data);
}
```

c)

```
public void postorder(Tree root)
{
          System.out.println(root.data);
          postorder(root.right);
          postorder(root.left);
}
```

d)

```
public void postorder(Tree root)
{
          postorder(root.right);
         System.out.println(root.data);
          postorder(root.left);
}
```

View Answer

Answer: b

Explanation: Post order traversal follows NLR(Left-Right-Node).

5. Select the code snippet that performs pre-order traversal iteratively.

a)

```
public void preOrder(Tree root)
{
        if (root == null) return;
         Stack<Tree> stk = new Stack<Tree>();
        st.add(root);
         while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
                        if (node.left != null) stk.push(node.left);
            if (node.right != null) stk.push(node.right);
        }
}
```

b)

```
public void preOrder(Tree root)
{
        if (root == null) return;
                  Stack<Tree> stk = new Stack<Tree>();
         while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
            if (node.right != null) stk.push(node.right);
            if (node.left != null) stk.push(node.left);
        }
}
```

c)

```
public void preOrder(Tree root)
{
        if (root == null) return;
         Stack<Tree> stk = new Stack<Tree>();
        st.add(root);
         while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
            if (node.right != null) stk.push(node.right);
```

```
            if (node.left != null) stk.push(node.left);
        }
}
```

d)

```
public void preOrder(Tree root)
{
        if (root == null) return;
         Stack<Tree> stk = new Stack<Tree>();
        st.add(root);
         while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
            if (node.right != null) stk.push(node.left);
            if (node.left != null) stk.push(node.right);
        }
}
```

View Answer

Answer: c

Explanation: Add the root to the stack first, then continously check for the right and left children of the top-of-the-stack.

6. Select the code snippet that performs post-order traversal iteratively.

a)

```
public void postorder(Tree root)
{
        if (root == null)
        return;
         Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
         while (!stk.isEmpty() || node != null)
        {
            while (node != null)
            {
                if (node.right != null)
                    stk.add(node.left);
                stk.add(node);
                node = node.right;
```

```
            }
             node = stk.pop();
             if (node.right != null && !stk.isEmpty() && node.right == stk.peek())
            {
                Trees nodeRight = stk.pop();
                stk.push(node);
                node = nodeRight;
            } else
            {
                System.out.print(node.data + " ");
                node = null;
            }
        }
}
```

b)

```
public void postorder(Tree root)
{
        if (root == null)
        return;
         Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
         while (!stk.isEmpty() || node != null)
        {
            while (node != null)
            {
                if (node.right != null)
                    stk.add(node.right);
                stk.add(node);
                node = node.left;
            }
            node = stk.pop();
             if (node.right != null && !stk.isEmpty() && node.right == stk.peek())
            {
                Trees nodeRight = stk.pop();
                stk.push(node);
                node = nodeRight;
            } else
            {
                System.out.print(node.data + " ");
                node = null;
            }
```

```
        }
}
```

c)

```
public void postorder(Tree root)
{
        if (root == null)
            return;
         Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
         while (!stk.isEmpty() || node != null)
        {
            while (node != null)
            {
                if (node.right != null)
                    stk.add(node.right);
                stk.add(node);
                node = node.left;
            }
             node = stk.pop();
            if (node.right != null)
            {
                Trees nodeRight = stk.pop();
                stk.push(node);
                node = nodeRight;
            } else
            {
                System.out.print(node.data + " ");
                node = null;
            }
        }
}
```

d)

```
public void postorder(Tree root)
{
        if (root == null)
            return;
         Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
         while (!stk.isEmpty() || node != null)
```

```
        {
            while (node != null)
            {
                if (node.right != null)
                    stk.add(node.left);
                stk.add(node);
                node = node.left;
            }
             node = stk.pop();
            if (node.right != null)
            {
                Trees nodeRight = stk.pop();
                stk.push(node);
                node = nodeLeft;
            } else
            {
                System.out.print(node.data + " ");
                node = null;
            }
        }
}
```

View Answer

Answer: b

Explanation: The left and right children are added first to the stack, followed by the node, which is then popped. Post-order follows LRN policy.

7. What is the time complexity of pre-order traversal in the iterative fashion?

a) O(1)

b) O(n)

c) O(logn)

d) O(nlogn)

View Answer

Answer: b

Explanation: Since you have to go through all the nodes, the complexity becomes O(n).

8. What is the space complexity of the post-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)

a) O(1)

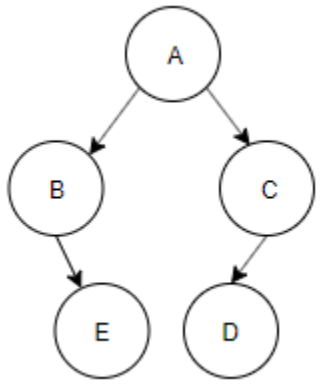
b) O(nlogd)

c) O(logd)

d) O(d)
View Answer
Answer: d
Explanation: In the worst case we have d stack frames in the recursive call, hence the complexity is O(d).

9. To obtain a prefix expression, which of the tree traversals is used?
a) Level-order traversal
b) Pre-order traversal
c) Post-order traversal
d) In-order traversal
View Answer
Answer: b
Explanation: As the name itself suggests, pre-order traversal can be used.

10. Consider the following data. The pre order traversal of a binary tree is A, B, E, C, D. The in order traversal of the same binary tree is B, E, A, D, C. The level order sequence for the binary tree is _____
a) A, C, D, B, E
b) A, B, C, D, E
c) A, B, C, E, D
d) D, B, E, A, C
View Answer
Answer: c
Explanation: The inorder sequence is B, E, A, D, C and Preorder sequence is A, B, E, C, D. The tree constructed with the inorder and preorder sequence is



The levelorder traversal (BFS traversal) is A, B, C, E, D.

11. Consider the following data and specify which one is Preorder Traversal Sequence, Inorder and Postorder sequences.
S1: N, M, P, O, Q
S2: N, P, Q, O, M

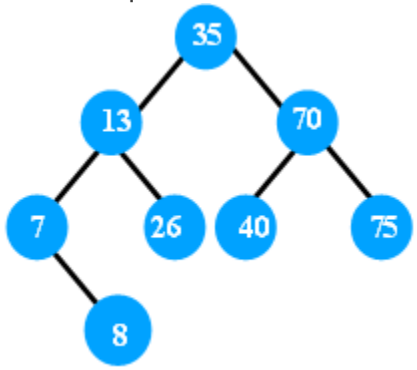
S3: M, N, O, P, Q
a) S1 is preorder, S2 is inorder and S3 is postorder
b) S1 is inorder, S2 is preorder and S3 is postorder
c) S1 is inorder, S2 is postorder and S3 is preorder
d) S1 is postorder, S2 is inorder and S3 is preorder
View Answer
Answer: c
Explanation: Preorder traversal starts from the root node and postorder and inorder starts from the left child node of the left subtree. The first node of S3 is different and for S1 and S2 it's the same. Thus, S3 is preorder traversal and the root node is M. Postorder traversal visits the root node at last. S2 has the root node(M) at last that implies S2 is postorder traversal. S1 is inorder traversal as S2 is postorder traversal and S3 is preorder traversal. Therefore, S1 is inorder traversal, S2 is postorder traversal and S3 is preorder traversal.

"Postorder Traversal".

1. In postorder traversal of binary tree right subtree is traversed before visiting root.
a) True
b) False
View Answer
2. What is the possible number of binary trees that can be created with 3 nodes, giving the sequence N, M, L when traversed in post-order.
a) 15
b) 3
c) 5
d) 8
View Answer
3. The post-order traversal of a binary tree is O P Q R S T. Then possible pre-order traversal will be _____
a) T Q R S O P
b) T O Q R P S
c) T Q O P S R
d) T Q O S P R
View Answer
Answer: c
Explanation: The last, second last nodes visited in post-order traversal are root and it's

right child respectively. Option T Q R S O P can't be a pre-order traversal, because nodes O, P are visited after the nodes Q, R, S. Option T O Q R P S, can't be valid, because the pre-order sequence given in option T O Q R P S and given post-order traversal creates a tree with node T as root and node O as left subtree. Option T Q O P S R is valid. Option T Q O S P R is not valid as node P is visited after visiting node S.

4. A binary search tree contains values 7, 8, 13, 26, 35, 40, 70, 75. Which one of the following is a valid post-order sequence of the tree provided the pre-order sequence as 35, 13, 7, 8, 26, 70, 40 and 75?
a) 7, 8, 26, 13, 75, 40, 70, 35
b) 26, 13, 7, 8, 70, 75, 40, 35
c) 7, 8, 13, 26, 35, 40, 70, 75
d) 8, 7, 26, 13, 40, 75, 70, 35
View Answer
Answer: d
Explanation: The binary tree contains values 7, 8, 13, 26, 35, 40, 70, 75. The given pre-order sequence is 35, 13, 7, 8, 26, 70, 40 and 75. So, the binary search tree formed is



Thus post-order sequence for the tree is 8, 7, 26, 13, 40, 75, 70 and 35.

5. Which of the following pair's traversals on a binary tree can build the tree uniquely?
a) post-order and pre-order
b) post-order and in-order
c) post-order and level order
d) level order and preorder
View Answer
Answer: b
Explanation: A binary tree can uniquely be created by post-order and in-order traversals.

6. A full binary tree can be generated using _____
a) post-order and pre-order traversal
b) pre-order traversal
c) post-order traversal
d) in-order traversal
View Answer
Answer: a
Explanation: Every node in a full binary tree has either 0 or 2 children. A binary tree can be generated by two traversals if one of them is in-order. But, we can generate a full binary tree using post-order and pre-order traversals.

7. The maximum number of nodes in a tree for which post-order and pre-order traversals may be equal is _____
a) 3
b) 1
c) 2
d) any number
View Answer
Answer: b
Explanation: The tree with only one node has post-order and pre-order traversals equal.

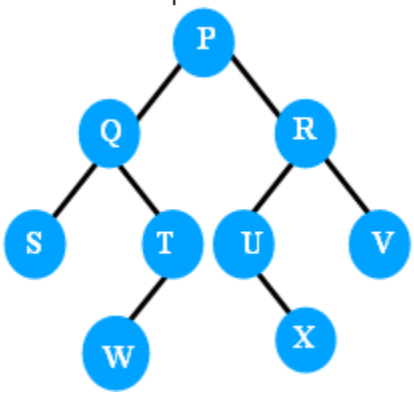
8. The steps for finding post-order traversal are traverse the right subtree, traverse the left subtree or visit the current node.
a) True
b) False
View Answer
Answer: b
Explanation: Left subtree is traversed first in post-order traversal, then the right subtree is traversed and then the output current node.

9. The pre-order and in-order are traversals of a binary tree are T M L N P O Q and L M N T O P Q. Which of following is post-order traversal of the tree?
a) L N M O Q P T
b) N M O P O L T
c) L M N O P Q T
d) O P L M N Q T
View Answer
10. For a binary tree the first node visited in in-order and post-order traversal is same.
a) True
b) False
View Answer

Answer: b
Explanation: Consider a binary tree,

Its in-order traversal – 13 14 16 19
Its post-order traversal- 14 13 19 16. Here the first node visited is not same.

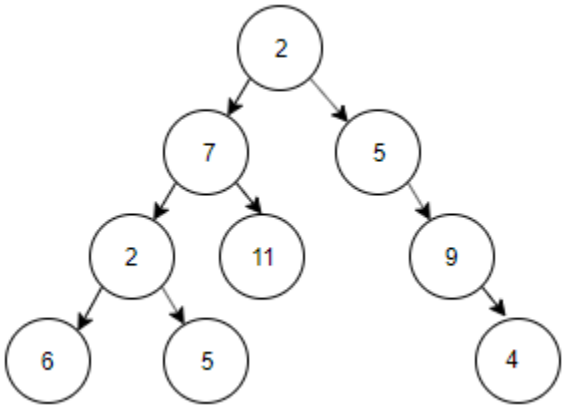
11. Find the postorder traversal of the binary tree shown below.



a) P Q R S T U V W X
b) W R S Q P V T U X
c) S W T Q X U V R P
d) S T W U X V Q R P
View Answer
Answer: c
Explanation: In postorder traversal the left subtree is traversed first and then the right subtree and then the current node. So, the posturer traversal of the tree is, S W T Q X U V R P.

“Inorder Traversal”.

1. For the tree below, write the in-order traversal.

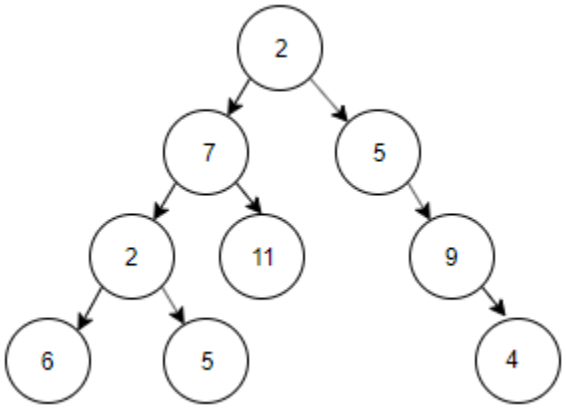


a) 6, 2, 5, 7, 11, 2, 5, 9, 4
b) 6, 5, 2, 11, 7, 4, 9, 5, 2
c) 2, 7, 2, 6, 5, 11, 5, 9, 4
d) 2, 7, 6, 5, 11, 2, 9, 5, 4

View Answer

Answer: a
Explanation: In-order traversal follows LNR(Left-Node-Right).

2. For the tree below, write the level-order traversal.



a) 2, 7, 2, 6, 5, 11, 5, 9, 4
b) 2, 7, 5, 2, 11, 9, 6, 5, 4
c) 2, 5, 11, 6, 7, 4, 9, 5, 2
d) 2, 7, 5, 6, 11, 2, 5, 4, 9

View Answer

Answer: b
Explanation: Level order traversal follows a breadth first search approach.

3. Select the code snippet which performs in-order traversal.
a)

```
public void inorder(Tree root)
{
         System.out.println(root.data);
         inorder(root.left);
         inorder(root.right);
}
```

b)


```
public void inorder(Tree root)
{
         inorder(root.left);
         System.out.println(root.data);
         inorder(root.right);
}
```

c)

```
public void inorder(Tree root)
{
         System.out.println(root.data);
         inorder(root.right);
         inorder(root.left);
}
```

d)

```
public void inorder(Tree root)
{
         inorder(root.right);
         inorder(root.left);
         System.out.println(root.data);
}
```

View Answer

Answer: b

Explanation: In-order traversal follows LNR(Left-Node-Right).

4. Select the code snippet which performs level-order traversal.
a)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.left!=null)
        queue.add(tempNode.left);
        if(tempNode.right!=null)
        queue.add(tempNode.right);
    }
}
```

b)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.left!=null)
        queue.add(tempNode.right);
        if(tempNode.right!=null)
        queue.add(tempNode.left);
    }
}
```

c)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
```

```
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.right!=null)
        queue.add(tempNode.left);
        if(tempNode.left!=null)
        queue.add(tempNode.right);
    }
}
```

d)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.right!=null)
        queue.add(tempNode.left.left);
        if(tempNode.left!=null)
        queue.add(tempNode.right.right);
    }
}
```

View Answer

Answer: a

Explanation: Firstly add the root node to the queue. Then for all the remaining nodes, pop the front end of the queue and print it, add the left and right children of the popped node to the queue.

5. What is the space complexity of the in-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)

a) O(1)

b) O(nlogd)

c) O(logd)

d) O(d)

View Answer

Answer: d
Explanation: In the worst case we have d stack frames in the recursive call, hence the complexity is O(d).

6. What is the time complexity of level order traversal?
a) O(1)
b) O(n)
c) O(logn)
d) O(nlogn)
View Answer
Answer: b
Explanation: Since you have to go through all the nodes, the complexity becomes O(n).

7. Which of the following graph traversals closely imitates level order traversal of a binary tree?
a) Depth First Search
b) Breadth First Search
c) Depth & Breadth First Search
d) Binary Search
View Answer
Answer: b
Explanation: Both level order tree traversal and breadth first graph traversal follow the principle that visit your neighbors first and then move on to further nodes.

8. In a binary search tree, which of the following traversals would print the numbers in the ascending order?
a) Level-order traversal
b) Pre-order traversal
c) Post-order traversal
d) In-order traversal
View Answer
Answer: d
Explanation: In a binary search tree, a node's left child is always lesser than the node and right child is greater than the node, hence an in-order traversal would print them in a non decreasing fashion.