“Binary Tree Properties”.

1. The number of edges from the root to the node is called _____ of the tree.
a) Height
b) Depth
c) Length
d) Width
View Answer
Answer: b
Explanation: The number of edges from the root to the node is called depth of the tree.

2. The number of edges from the node to the deepest leaf is called _____ of the tree.
a) Height
b) Depth
c) Length
d) Width
View Answer
Answer: a
Explanation: The number of edges from the node to the deepest leaf is called height of the tree.

3. What is a full binary tree?
a) Each node has exactly zero or two children
b) Each node has exactly two children
c) All the leaves are at the same level
d) Each node has exactly one or two children
View Answer
Answer: a
Explanation: A full binary tree is a tree in which each node has exactly 0 or 2 children.

4. What is a complete binary tree?
a) Each node has exactly zero or two children
b) A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from right to left
c) A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right
d) A tree In which all nodes have degree 2
View Answer

Answer: c
Explanation: A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right is called complete binary tree. A Tree in which each node has exactly zero or two children is called full binary tree. A Tree in which the degree of each node is 2 except leaf nodes is called perfect binary tree.

5. What is the average case time complexity for finding the height of the binary tree?
a) h = O(loglogn)
b) h = O(nlogn)
c) h = O(n)
d) h = O(log n)
View Answer
Answer: d
Explanation: The nodes are either a part of left sub tree or the right sub tree, so we don't have to traverse all the nodes, this means the complexity is lesser than n, in the average case, assuming the nodes are spread evenly, the time complexity becomes O(logn).

Check this: Data Structure Books | Programming MCQs

6. Which of the following is not an advantage of trees?
a) Hierarchical structure
b) Faster search
c) Router algorithms
d) Undo/Redo operations in a notepad
View Answer
Answer: d
Explanation: Undo/Redo operations in a notepad is an application of stack. Hierarchical structure, Faster search, Router algorithms are advantages of trees.

7. In a full binary tree if number of internal nodes is I, then number of leaves L are?
a) L = 2*I
b) L = I + 1
c) L = I – 1
d) L = 2*I – 1
View Answer
Answer: b
Explanation: Number of Leaf nodes in full binary tree is equal to 1 + Number of Internal Nodes i.e L = I + 1

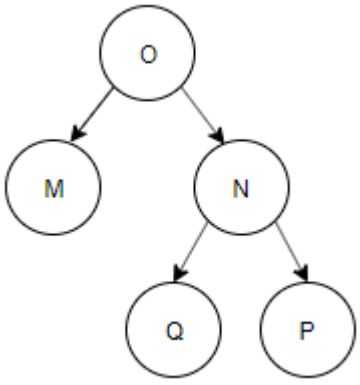
8. In a full binary tree if number of internal nodes is I, then number of nodes N are?
a) N = 2*I
b) N = I + 1

c) N = I – 1
d) N = 2*I + 1
View Answer
Answer: d
Explanation: Relation between number of internal nodes(I) and nodes(N) is N = 2*I+1.

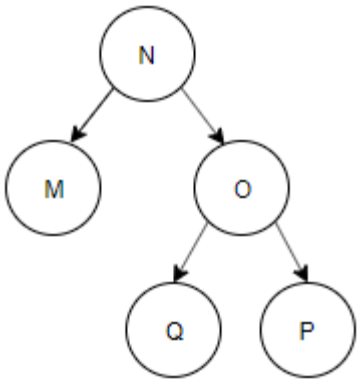
9. In a full binary tree if there are L leaves, then total number of nodes N are?
a) N = 2*L
b) N = L + 1
c) N = L – 1
d) N = 2*L – 1
View Answer
Answer: d
Explanation: The relation between number of nodes(N) and leaves(L) is N=2*L-1.

10. Which of the following is incorrect with respect to binary trees?
a) Let T be a binary tree. For every k ≥ 0, there are no more than 2k nodes in level k
b) Let T be a binary tree with λ levels. Then T has no more than $2^{\lambda - 1}$ nodes
c) Let T be a binary tree with N nodes. Then the number of levels is at least ceil(log (N + 1))
d) Let T be a binary tree with N nodes. Then the number of levels is at least floor(log (N + 1))
View Answer
Answer: d
Explanation: In a binary tree, there are atmost 2k nodes in level k and 2^{k-1} total number of nodes. Number of levels is at least ceil(log(N+1)).

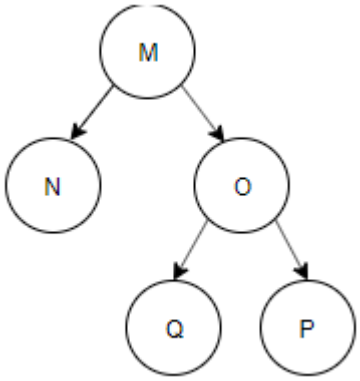
11. Construct a binary tree by using postorder and inorder sequences given below.
Inorder: N, M, P, O, Q
Postorder: N, P, Q, O, M



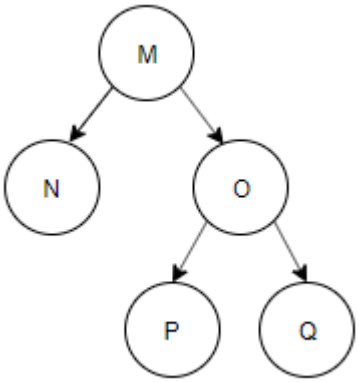
a)



b)



c)



d)

View Answer

Answer: d

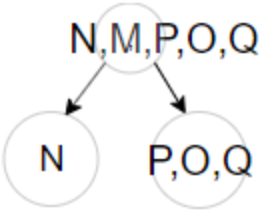
Explanation: Here,

Postorder Traversal: N, P, Q, O, M

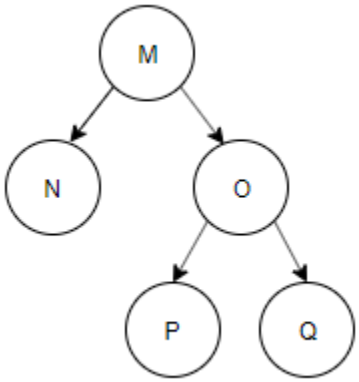
Inorder Traversal: N, M, P, O, Q

Root node of tree is the last visiting node in Postorder traversal. Thus, Root Node = ‘M’.

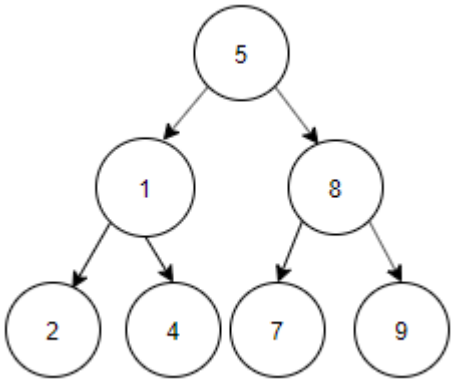
The partial tree constructed is:



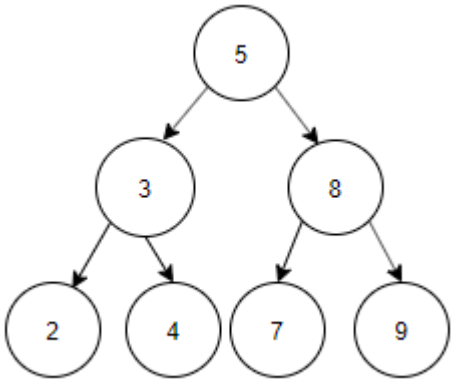
The second last node in postorder traversal is O. Thus, node P becomes left child of node O and node Q becomes right child of node Q. Thus, the final tree is:



12. Construct a binary search tree by using postorder sequence given below.
Postorder: 2, 4, 3, 7, 9, 8, 5.

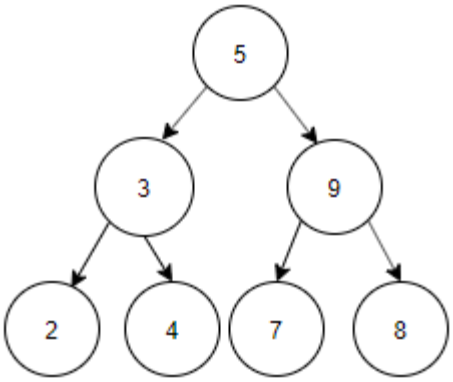


a)

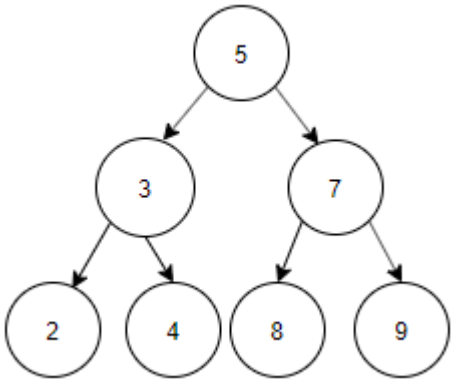


b)

c)



d)



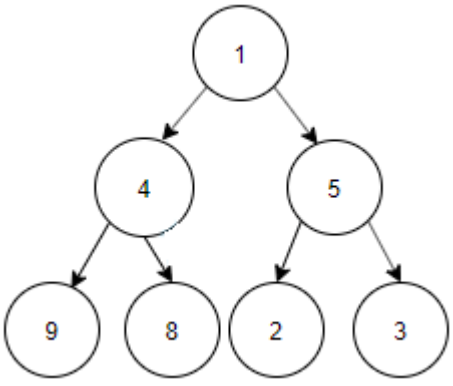
View Answer

Answer: b

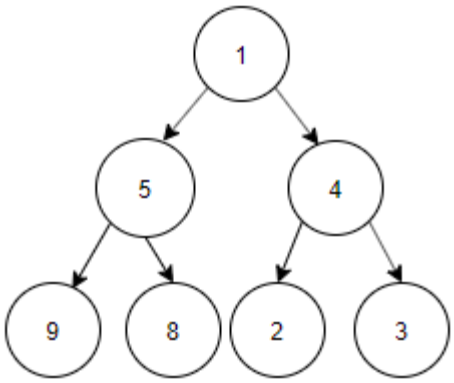
Explanation: Postorder sequence is 2, 4, 3, 7, 9, 8, 5.
Inorder sequence is the ascending order of nodes in Binary search tree. Thus, Inorder sequence is 2, 3, 4, 5, 7, 8, 9. The tree constructed using Postorder and Inorder sequence is

13. Construct a binary tree using inorder and level order traversal given below.
Inorder Traversal: 3, 4, 2, 1, 5, 8, 9
Level Order Traversal: 1, 4, 5, 9, 8, 2, 3

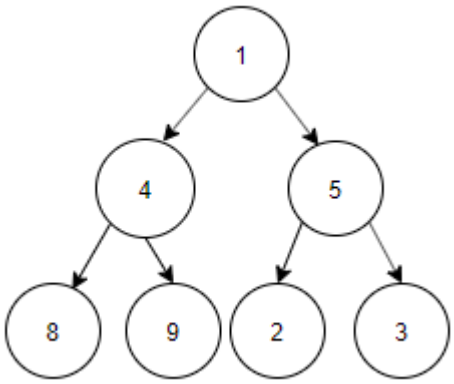
a)



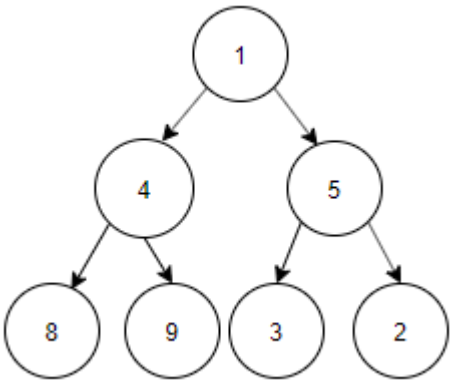
b)



c)



d)



View Answer

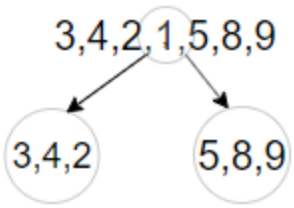
Answer: a

Explanation: Inorder Traversal: 3, 4, 2, 1, 5, 8, 9

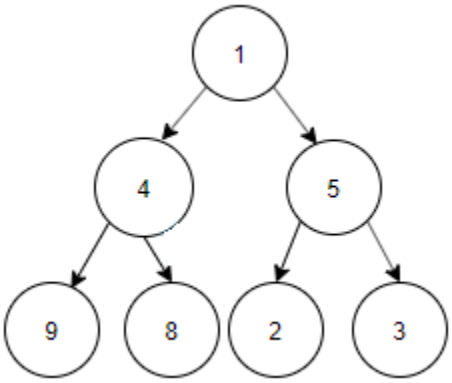
Level Order Traversal: 1, 4, 5, 9, 8, 2, 3

In level order traversal first node is the root node of the binary tree.

Thus the partially formed tree is:



In level order traversal, the second node is 4. Then, node 3 becomes left child of node 4 and node 2 becomes right child of node 4. Third node of level order traversal is 8. Then, node 5 becomes left child of node 8 and node 9 becomes right child of node 8. Thus, the final tree is:



"Binary Search Tree".

1. Which of the following is false about a binary search tree?
a) The left child is always lesser than its parent
b) The right child is always greater than its parent
c) The left and right sub-trees should also be binary search trees
d) In order sequence gives decreasing order of elements
View Answer
Answer: d
Explanation: In order sequence of binary search trees will always give ascending order of elements. Remaining all are true regarding binary search trees.

2. How to search for a key in a binary search tree?
a)

```
public Tree search(Tree root, int key)
{
```

```
          if( root == null || root.key == key )
        {
                  return root;
          }
          if( root.key < key )
        {
                  return search(root.right,key);
          }
          else
          return search(root.left,key);
}
```

b)

Sanfoundry Certification Contest of the Month is Live. 100+ Subjects. Participate Now!


```
public Tree search(Tree root, int key)
{
          if( root == null || root.key == key )
        {
                  return root;
          }
          if( root.key < key )
        {
                  return search(root.left,key);
          }
          else
          return search(root.right,key);
}
```

c)

Check this: Programming MCQs | Computer Science MCQs

```
public Tree search(Tree root, int key)
{
          if( root == null)
        {
                  return root;
          }
          if( root.key < key )
        {
```

```
                return search(root.right,key);
        }
        else
                return search(root.left,key);
}
```

d)

```
public Tree search(Tree root, int key)
{
        if( root == null)
       {
                return root;
        }
        if( root.key < key )
       {
                return search(root.right.right,key);
        }
        else
                return search(root.left.left,key);
}
```

View Answer

3. What is the speciality about the inorder traversal of a binary search tree?
a) It traverses in a non increasing order
b) It traverses in an increasing order
c) It traverses in a random fashion
d) It traverses based on priority of the node
View Answer
Answer: b
Explanation: As a binary search tree consists of elements lesser than the node to the left and the ones greater than the node to the right, an inorder traversal will give the elements in an increasing order.

4. What does the following piece of code do?

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

a) Preorder traversal
b) Inorder traversal
c) Postorder traversal
d) Level order traversal
View Answer
Answer: c
Explanation: In a postorder traversal, first the left child is visited, then the right child and finally the parent.

5. What does the following piece of code do?

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

a) Preorder traversal
b) Inorder traversal
c) Postorder traversal
d) Level order traversal
View Answer
Answer: a
Explanation: In a preorder traversal, first the parent is visited, then the left child and finally the right child.

6. How will you find the minimum element in a binary search tree?
a)

```
public void min(Tree root)
{
        while(root.left() != null)
        {
                root = root.left();
        }
        System.out.println(root.data());
}
```

b)

```
public void min(Tree root)
{
```

```
        while(root != null)
        {
                root = root.left();
        }
        System.out.println(root.data());
}
```

c)

```
public void min(Tree root)
{
        while(root.right() != null)
        {
                root = root.right();
        }
        System.out.println(root.data());
}
```

d)

```
public void min(Tree root)
{
        while(root != null)
        {
                root = root.right();
        }
        System.out.println(root.data());
}
```

View Answer

Answer: a

Explanation: Since all the elements lesser than a given node will be towards the left, iterating to the leftmost leaf of the root will give the minimum element in a binary search tree.

7. How will you find the maximum element in a binary search tree?

a)

```
public void max(Tree root)
{
        while(root.left() != null)
```

```
        {
                root = root.left();
        }
        System.out.println(root.data());
}
```

b)

```
public void max(Tree root)
{
        while(root != null)
        {
                root = root.left();
        }
        System.out.println(root.data());
}
```

c)

```
public void max(Tree root)
{
        while(root.right() != null)
        {
                root = root.right();
        }
        System.out.println(root.data());
}
```

d)

```
public void max(Tree root)
{
        while(root != null)
        {
                root = root.right();
        }
        System.out.println(root.data());
}
```

View Answer

Answer: c

Explanation: Since all the elements greater than a given node are towards the right, iterating through the tree to the rightmost leaf of the root will give the maximum element in a binary search tree.

8. What are the worst case and average case complexities of a binary search tree?
a) O(n), O(n)
b) O(logn), O(logn)
c) O(logn), O(n)
d) O(n), O(logn)
View Answer
Answer: d
Explanation: Worst case arises when the tree is skewed(either to the left or right) in which case you have to process all the nodes of the tree giving O(n) complexity, otherwise O(logn) as you process only the left half or the right half of the tree.

9. Given that 2 elements are present in the tree, write a function to find the LCA(Least Common Ancestor) of the 2 elements.
a)

```
public void lca(Tree root,int n1, int n2)
{
         while (root != NULL)
        {
            if (root.data() > n1 && root.data() > n2)
            root = root.right();
            else if (root.data() < n1 && root.data() < n2)
            root = root.left();
             else break;
        }
        System.out.println(root.data());
}
```

b)

```
public void lca(Tree root,int n1, int n2)
{
    while (root != NULL)
    {
        if (root.data() > n1 && root.data() < n2)
        root = root.left();
        else if (root.data() < n1 && root.data() > n2)
        root = root.right();
         else break;
    }
```

```
    System.out.println(root.data());
}
```

c)

```
public void lca(Tree root,int n1, int n2)
{
    while (root != NULL)
    {
        if (root.data() > n1 && root.data() > n2)
        root = root.left();
        else if (root.data() < n1 && root.data() < n2)
        root = root.right();
         else break;
    }
    System.out.println(root.data());
}
```

d)

```
public void lca(Tree root,int n1, int n2)
{
    while (root != NULL)
    {
        if (root.data() > n1 && root.data() > n2)
        root = root.left.left();
        else if (root.data() < n1 && root.data() < n2)
        root = root.right.right();
         else break;
    }
    System.out.println(root.data());
}
```

View Answer

Answer: c

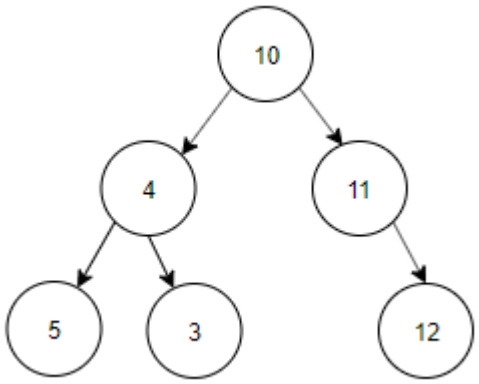
Explanation: The property of a binary search tree is that the lesser elements are to the left and greater elements are to the right, we use this property here and iterate through the tree such that we reach a point where the 2 elements are on 2 different sides of the node, this becomes the least common ancestor of the 2 given elements.

10. What are the conditions for an optimal binary search tree and what is its advantage?

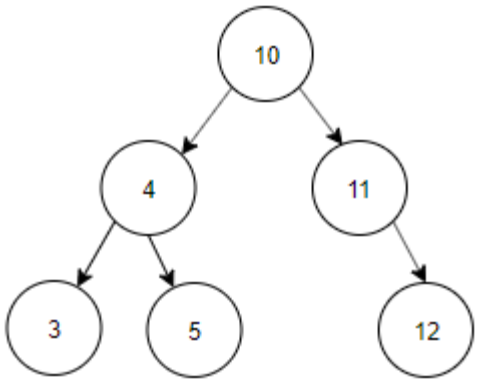
a) The tree should not be modified and you should know how often the keys are accessed, it improves the lookup cost
b) You should know the frequency of access of the keys, improves the lookup time
c) The tree can be modified and you should know the number of elements in the tree before hand, it improves the deletion time
d) The tree should be just modified and improves the lookup time
View Answer
Answer: a
Explanation: For an optimal binary search The tree should not be modified and we need to find how often keys are accessed. Optimal binary search improves the lookup cost.

11. Construct a binary search tree with the below information.
The preorder traversal of a binary search tree 10, 4, 3, 5, 11, 12.

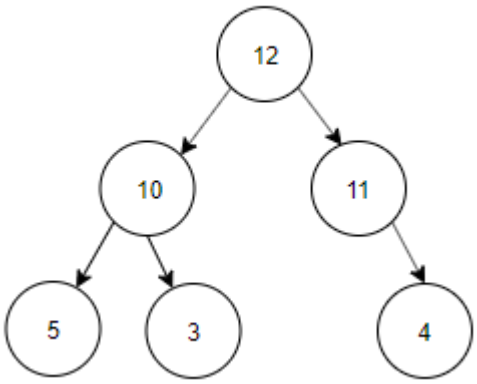
a)



b)



c)



d)

View Answer

Answer: c

Explanation: Preorder Traversal is 10, 4, 3, 5, 11, 12. Inorder Traversal of Binary search tree is equal to ascending order of the nodes of the Tree. Inorder Traversal is 3, 4, 5, 10, 11, 12. The tree constructed using Preorder and Inorder traversal is