1. Consider the same code as given in above question. What does the function print() do in general? The function print() receives root of a Binary Search Tree (BST) and a positive integer k as arguments.

```
struct node {

   int data;

   struct node *left, *right;

};

int count = 0;

void print(struct node *root, int k){

   if (root != NULL && count <= k)    {

       print(root->right, k);

       count++;

       if (count == k)

         printf("%d ", root->data);

      print(root->left, k);

   }

}
```

A. Prints the kth smallest element in BST

B. Prints the kth largest element in BST

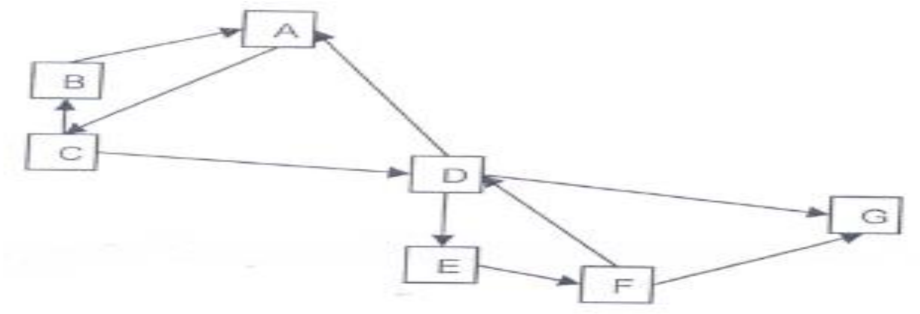
C. Prints the leftmost node at level k from root

D. Prints the rightmost node at level k from root

Answer: B

Explanation: The function basically does reverse inorder traversal of the given Binary Search Tree. The reverse inorder traversal produces data in reverse sorted order. Whenever a nod is visited, count is incremented by 1 and data of a node is printed only when count becomes k.

2. Which of the following is invalid path for the following graph?



A. C B A

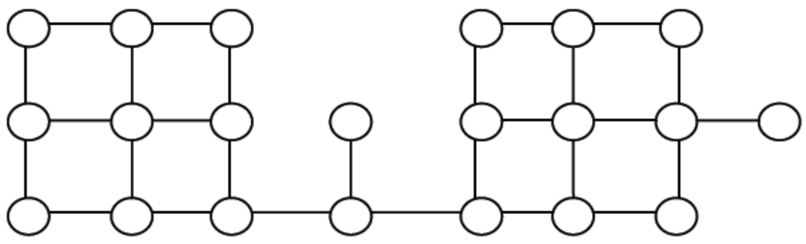
B. C D A C B A

C. C D A

D. C D E F G

Answer: B

3. Suppose depth first search is executed on the graph below starting at some unknown vertex. Assume that a recursive call to visit a vertex is made only after first checking that the vertex has not been visited earlier. Then the maximum possible recursion depth (including the initial call) is _________.



A. 17

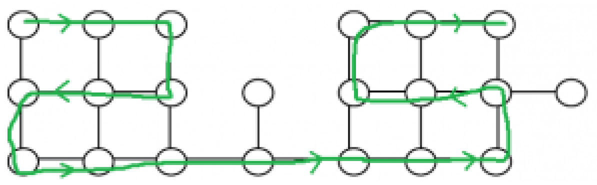
B. 18

C. 19

D. 20

Answer: C

Explanation: The following diagram shows



The worst case situation where the recursion tree has maximum depth. GATECS2014Q20Sol So the recursion depth is 19 (including the first node).

4. In an adjacency list representation of an undirected simple graph G = (V, E), each edge (u, v) has two adjacency list entries: [v] in the adjacency list of u, and [u] in the adjacency list of v. These are called twins of each other. A twin pointer is a pointer from an adjacency list entry to its twin. If |E| = m and |V | = n, and the memory size is not a constraint, what is the time complexity of the most efficient algorithm to set the twin pointer in each entry in each adjacency list?

A. Θ(n2)

B. Θ(m+n)

C. Θ(m2)

D. Θ(n4)

Answer: B

Explanation: First you need to find twins of each node. You can do this using level order traversal (i.e., BFS) once. Time complexity of BFS is Θ(m +n). And you have to use linked list for representation which is extra space (but memory size is not a constraint here). Final, time complexity is Θ(m + n) to set twin pointer

5. An articulation point in a connected graph is a vertex such that removing the vertex and its incident edges disconnects the graph into two or more connected components. Let T be a DFS tree obtained by doing DFS in a connected undirected graph G. Which of the following options is/are correct?

A. Root of T can never be an articulation point in G.

B. Root of T is an articulation point in G if and only if it has 2 or more children.

C. A leaf of T can be an articulation point in G.

D. If u is an articulation point in G such that x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u.

Answer: B,D

Explanation: To find all articulation points

DFS- based-approach:

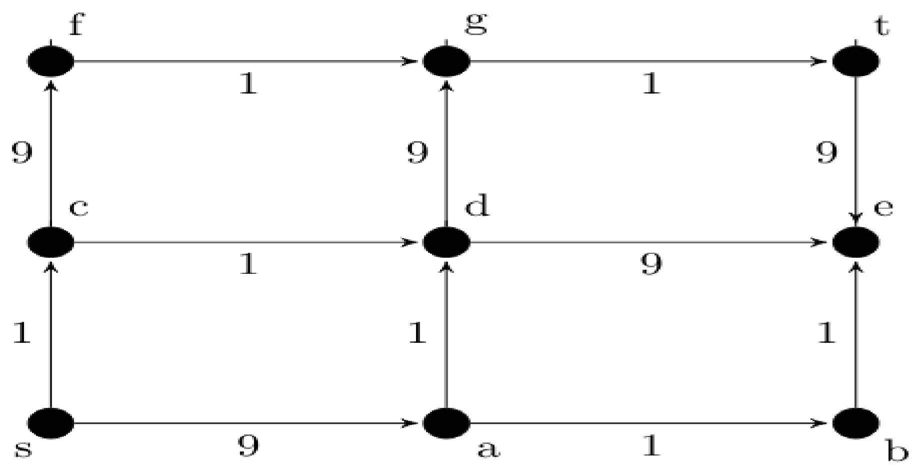
We can prove following properties:

The root of a DFS-tree is an articulation point if and only if it has at least two children.

Leaf of a DFS-tree are never articulation points.

D is not true because, it is not necessary to have a path through articulation point only. There can a be a direct path from x to y as well in a graph G.

6. In a directed acyclic graph with a source vertex s, the quality-score of a directed path is defined to be the product of the weights of the edges on the path. Further, for a vertex v other than s, the quality-score of v is defined to be the maximum among the quality-scores of all the paths from s to v. The quality-score of s is assumed to be 1.



The sum of the quality-scores of all vertices on the graph shown above is _______ .

A. 929

B. 81

C. 729

D. 1023

Answer: A

Explanation: Let Q(V) denote the quality-score of vertex V.

Q(S) = 1 (Given)

Q(C) = 1 (S → C)

Q(F) = 1 * 9 (S → C → F)

Q(A) = 9 (S → A)

Q(D) = 9*1 (S → A → D)

Q(G) = 9 * 1 * 9 (S → A → D → G)

Q(B) = 9 * 1 (S → A → B)

Q(E) = 9 * 1 * 9 (S → A → D → E)

Q(T) = 9*1*9*9 (S → A → D → E → T)

Sum of quality-score of all vertices,

= 1 + 1 + 9 + 9 + 9 + 81 + 9 + 81 + 729 = 929

7. A Binary Search Tree (BST) stores values in the range 37 to 573. Consider the following sequence of keys.

I. 81, 537, 102, 439, 285, 376, 305

II. 52, 97, 121, 195, 242, 381, 472

III. 142, 248, 520, 386, 345, 270, 307

IV. 550, 149, 507, 395, 463, 402, 270

Which of the following statements is TRUE?

a. II is an inorder sequence of some BST where 121 is the root and 52 is a leaf

b. I is a preorder sequence of some BST with 439 as the root

c. IV is a postorder sequence of some BST with 149 as the root

d. I, II and IV are inorder sequences of three different BSTs

Answer: A

8. 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. 3

B. 5

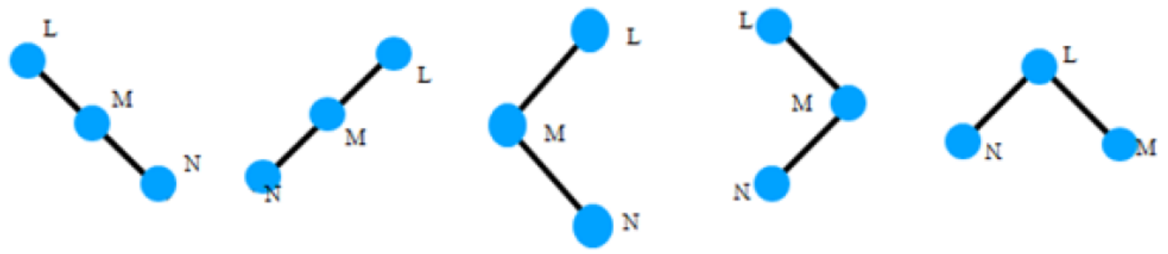
C. 8

D. 15

Answer: B

Explanation:

5 binary trees are possible and they are,



No of trees with given post order traversal = No of trees with given

preorder traversal = No of trees with given

inorder traversal = Catalan Number [C(2n,n) / (n+1)]

9. When searching for the key value 60 in a binary search tree, nodes containing the key values 10, 20, 40, 50, 70 80, 90 are traversed, not necessarily in the order given. How many different orders are possible in which these key values can occur on the search path from the root to the node containing the value 60?

A. 64

B. 5040

C. 35

D. 128

Answer: C

Explanation:

Explanation 1: There are two set of values, smaller than 60 and greater than 60. Smaller values 10, 20, 40 and 50 are visited, means they are visited in order. Similarly, 90, 80 and 70 are visited in order. = 7!/(4!3!) = 35

Explanation 2: The four lesser keys must appear in ascending order while the three greater ones must appear in descending order otherwise some keys will be left out in the traversal. Note that in the traversal, these less er keys might not be continuous and can be separated by greater keys.
For eg: 10, 90, 20, 80, 40, 70, 50 but the order of both groups of keys (lesser and greater) remains the same individually.
Now, out of total seven positions, the lesser keys acquire four positions which can be selected in 7C4 ways. Once we know the places of these four keys, the places of the three greater keys only have one permutation.
For eg: if we know that
10, _, 20, 30, _, 40, _, 5 0
Then there's only one per mutation of places for 90, 80 & 70 which is place number 2, 5 & 7 respectively. Therefore for each combinatio n of lesser keys, there's a unique combination of greater keys. So total combinations= 7C4*1 =35 ways