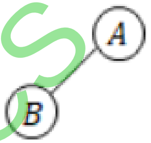
Data Structure
Chapter 4: Tree

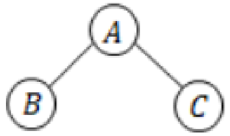
GATE CS PYQ

Solved By

Monalisa Pradhan

- GATE 2010,Q10,1 Mark: In a binary tree with n nodes, every node has an odd number of descendants. Every node is considered to be its own descendant. What is the number of nodes in the tree that have exactly one child?
- **(A)** 0 **(B)** 1 **(C)** (n-1)/2 **(D)** n-1





- Given a Binary Tree with n nodes.
- Every node has an odd number of descendants.
- Also given every node is considered to be its own descendant.
- This is even number of descendants (2), because A is also considered as a descendant
- This is odd number of descendants (3), A, B, C are descendants here.
  Condition satisfied – odd number, but number of nodes in a tree that has exactly one child is 0.
- Ans **(A)** 0

- GATE 2011,Q29,2 Mark:We are given a set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree?
- (A) 0 (B) 1 (C) n! (D) $(1/(n+1)).{}^{2n}C_n$
- There is only one way.
- The minimum value has to go to the leftmost node and the maximum value to the rightmost node.
- Recursively, we can define for other nodes.
- Ans : (B) 1
- We can populate $(1/(n+1)).{}^{2n}C_n$ binary tree but one binary search tree.

- GATE 2012,Q5,1 Mark: The worst case running time to search for an element in a balanced binary search tree with $n2^n$ elements is
- (A) $\Theta(n \log n)$ (B) $\Theta(n2^n)$ (C) $\Theta(n)$ (D) $\Theta(\log n)$
- Worst case running time to search for an element in a balanced binary search tree of 'n' elements is $\Theta(\log n)$.
- No of elements = $n.2^n$ then search time = $\Theta(\log n.2^n)$
- $= (\log n + \log 2^n)$
- $= (\log n + n \log 2)$
- $= \Theta(n)$
- Ans : (C) $\Theta(n)$

- GATE 2012,Q47,2 Mark:The height of a tree is defined as the number of edges on the longest path in the tree. The function shown in the pseudocode below is invoked as height (root) to compute the height of a binary tree rooted at the tree pointer root.

```
int height(treeptr n)
{if(n == NULL) return -1;
   if(n → left == NULL)
      if(n → right == NULL) return 0;
      else return B1; // Box 1
   else{h1 = height(n → left);
      if(n → right == NULL) return (1+h1);
      else{h2 = height(n → right);
         return B2; // Box 2   } } }
```

- The box B1 gets executed when left sub tree of n is NULL & right subtree is not NULL.
- Height of n will be the height of the right subtree +1.
- The box B2 gets executed when both left & right subtrees of n are not NULL.
- Height of n will be max of heights of left & right subtrees of n+1.

- The appropriate expression for the two boxes B1 and B2 are

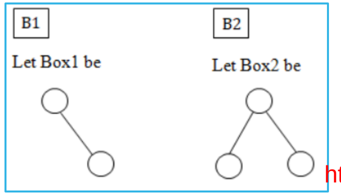
Ans : (A)

- **(A)** B1 : (1 + height(n→right)), B2 : (1 + max(h1,h2))

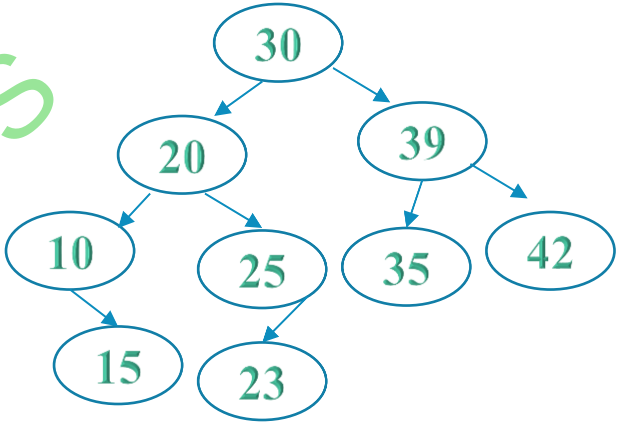
(B) B1 : (height(n→right)), B2 : (1 + max(h1,h2))

(C) B1 : height(n→right), B2 : max(h1,h2)

(D) B1 : (1 + height(n→right)), B2 : max(h1,h2)

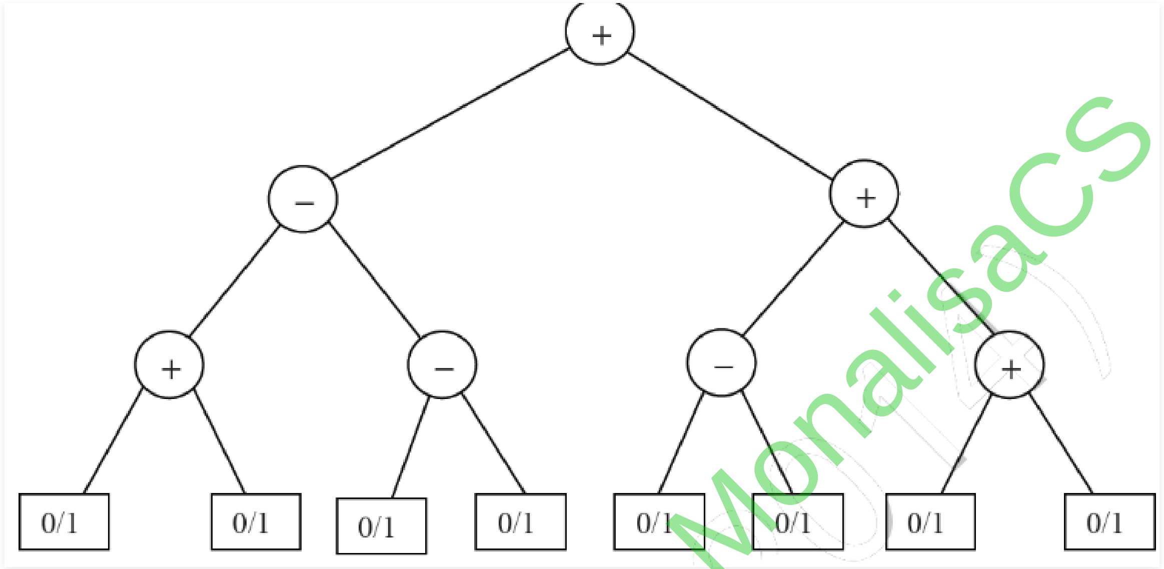


- GATE 2013,Q43,2 Mark:The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree?
- **(A)** 10, 20, 15, 23, 25, 35, 42, 39, 30
- **(B)** 15, 10, 25, 23, 20, 42, 35, 39, 30
- **(C)** 15, 20, 10, 23, 25, 42, 35, 39, 30
- **(D)** 15, 10, 23, 25, 20, 35, 42, 39, 30
- Inorder :Left,Root,Right
- Preorder: Root,Left,Right
- Postorder:Left,Right,Root
- Inorder : 10, 15, 20, 23, 25, 30, 35, 39, 42
- Preorder: 30, 20, 10, 15, 25, 23, 39, 35, 42
- Postorder:15, 10, 23, 25, 20, 35, 42, 39, 30
- Ans : **(D)** 15, 10, 23, 25, 20, 35, 42, 39, 30



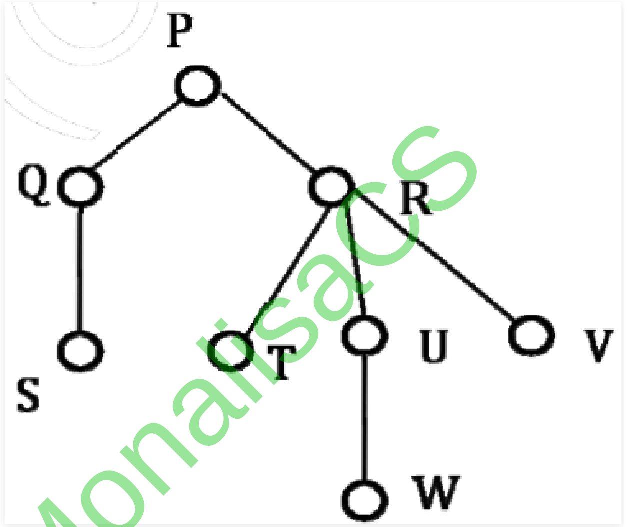
- GATE 2014 Set-1,Q12,1 Mark:Consider a rooted n node binary tree represented using pointers. The best upper bound on the time required to determine the number of subtrees having exactly 4 nodes is O($n^a \log^b n$). Then the value of a+10b is ________.
- We can find the subtree with 4 nodes in O(n) time.
- Following can be a simple approach.
- 1) Traverse the tree in bottom up manner and find size of subtree rooted with current node
- 2) If size becomes 4, then print the current node.
- Time complexity =O(n) where a=1,b=0
- a+10b=1+10*0=1
- Ans :1

- GATE 2014 Set-2,Q39,2 Mark:Consider the expression tree shown. Each leaf represents a numerical value, which can either be 0 or 1. Over all possible choices of the values at the leaves, the maximum possible value of the expression represented by the tree is ____.



- An Expression Tree is a binary tree in which each internal node corresponds to operator and each leaf node corresponds to operand .
- ((1+1)-(0-1))+((1-0)+(1+1))=2-(-1)+1+2=3+3=6
- Ans : 6

- GATE 2014,Set-3,Q12,1 Mark:Consider the following rooted tree with the vertex P labeled as root .The order in which the nodes are visited during in-order traversal is
- **(A)** SQPTRWUV
  (B) SQPTURWV
  (C) SQPTWUVR
  (D) SQPTRUWV
- In-order :left,root,right
- Ans : **(A)** SQPTRWUV



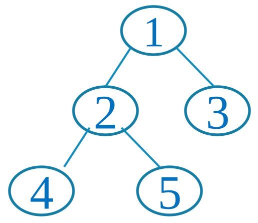
- GATE 2014,Set-3,Q41,2 Mark: Consider the pseudocode given below. The function DoSomething() takes as argument a pointer to the root of an arbitrary tree represented by the leftMostChild-rightSibling representation. Each node of the tree is of type treeNode.

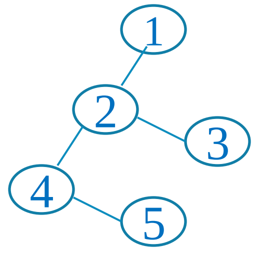
```
typedef struct treeNode* treeptr;
struct treeNode { treeptr leftMostChild, rightSibling; };
int DoSomething (treeptr tree)
{  int value=0;
    if (tree != NULL)
    { if (tree→leftMostChild == NULL)
        value = 1;
     else
        value = DoSomething(tree→leftMostChild);
        value = value + DoSomething(tree→rightSibling);
     } return(value); }
```

- d(1),v=0
- else v=d(2) *
- d(2) else v=d(4)*
- d(4) if true v=1

v=v+d(5)
v=v+d(3)

- v=v+d(5)=1+1=2
- v=v+d(3)=2+1=3
- ans :**(d)** number of leaf nodes in the tree.





- When the pointer to the root of a tree is passed as the argument to DoSomething, the value returned by the function corresponds to the
- **(A)** number of internal nodes in the tree.
- **(B)** height of the tree.
- **(C)** number of nodes without a right sibling in the tree.
- **(D)** number of leaf nodes in the tree.

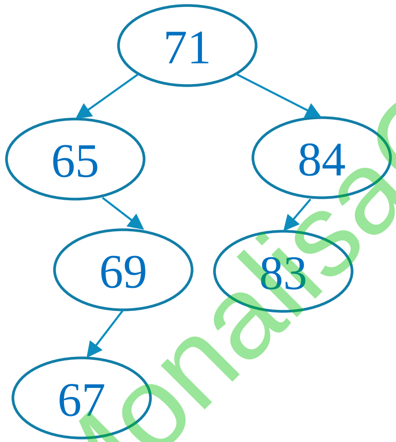
- GATE 2015,Set-1,Q10,1 mark:Which of the following is/are correct inorder traversal sequence(s) of binary search tree(s)?
- I. 3, 5, 7, 8, 15, 19, 25
- II. 5, 8, 9, 12, 10, 15, 25
- III. 2, 7, 10, 8, 14, 16, 20
- IV. 4, 6, 7, 9, 18, 20, 25
- **(A)** I and IV only **(B)** II and III only **(C)** II and IV only **(D)** II only
- An Inorder traversal of a Binary Search Tree must be in increasing order.
- **Answer:** **(A)** I and IV only

- GATE 2015,Set-1,Q25,1 mark:The height of a tree is the length of the longest root-to-leaf path in it. The maximum and minimum number of nodes in a binary tree of height 5 are
- **(A)** 63 and 6, respectively
  (B) 64 and 5, respectively
  (C) 32 and 6, respectively
  (D) 31 and 5, respectively
- Maximum number of nodes in a binary tree of height h is,
  $2^{h+1} - 1 = 2^{5+1} - 1 = 63$
- Minimum number of nodes in a binary tree of height h is
  $h + 1 = 5 + 1 = 6$
- Ans : **(A)** 63 and 6, respectively

- GATE 2015,Set-2Q10,1 mark: A binary tree T has 20 leaves. The number of nodes in T having two children is __________.
- Number of nodes in T having two children=leaves-1=20-1=19
- Ans: 19

- GATE 2015,Set-3,Q25,1 Mark:Consider a binary tree T that has 200 leaf nodes. Then, the number of nodes in T that have exactly two children are _____ .
- Number of nodes in T having exactly two children =leaf-1
- 200-1=199
- Ans : 199

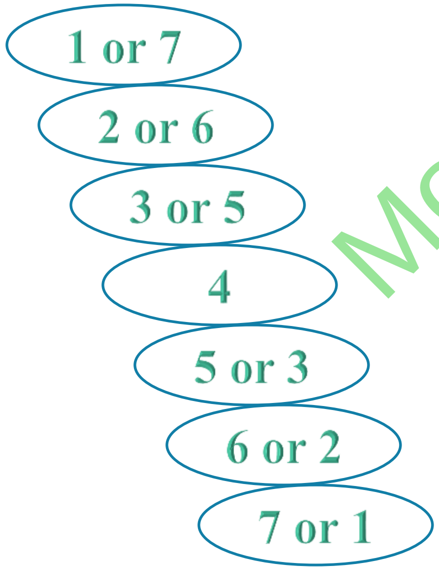
- GATE 2015,Set-3,Q13,1 mark:While inserting the elements 71, 65, 84, 69, 67, 83 in an empty binary search tree (BST) in the sequence shown, the element in the lowest level is
- (A)65 (B)67 (C)69 (D)83



- Ans : (B)67

- GATE 2016,Set-2,Q36,2 Mark:Consider the following New-order strategy for traversing a binary tree:
- Visit the root;
- Visit the right subtree using New-order;
- Visit the left subtree using New-order;
- The New-order traversal of the expression tree corresponding to the reverse polish expression 3 4 * 5 - 2 ˆ 6 7 * 1 + - is given by:
- **(A)** + – 1 6 7 * 2 ˆ 5 – 3 4 * **(B)** – + 1 * 6 7 ˆ 2 – 5 * 3 4
- **(C)** – + 1 * 7 6 ˆ 2 – 5 * 4 3 **(D)** 1 7 6 * + 2 5 4 3 * – ˆ –
- Reverse Polish expression is derived through Post-Order :Left, Right , Root.
- Post-Order Expression : 3 4 * 5 – 2 ˆ 6 7 * 1 + –
- Acc. to Ques. New Order algorithm is :Root , Right ,Left
- ie. New Order Expression will be a total reverse of the Post-Order algorithm
- Hence , New Order Expression : – + 1 * 7 6 ^ 2 – 5 * 4 3
- Ans : **(C)** – + 1 * 7 6 ˆ 2 – 5 * 4 3

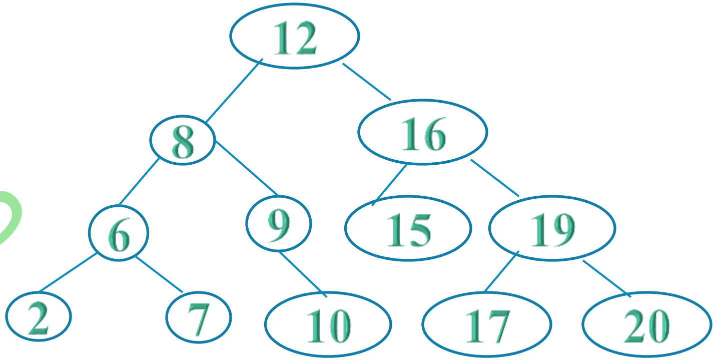
- GATE 2016,Set-2,Q40,2 Mark:The number of ways in which the numbers 1, 2, 3, 4, 5, 6, 7 can be inserted in an empty binary search tree, such that the resulting tree has height 6, is __________ .
- Note: The height of a tree with a single node is 0.
- There are 2 ways to select for each edge,whether it should be a right skewed edge or left skewed edge.
- Each node has 2 choices 1 or 7 on it's end,2 or 6 on its end,3 or 5 on its end and the rest 4 is the only choice at end,there are totally 2*2*2*2*2*2*1 choices.
- 2^6 =64
- Ans :64



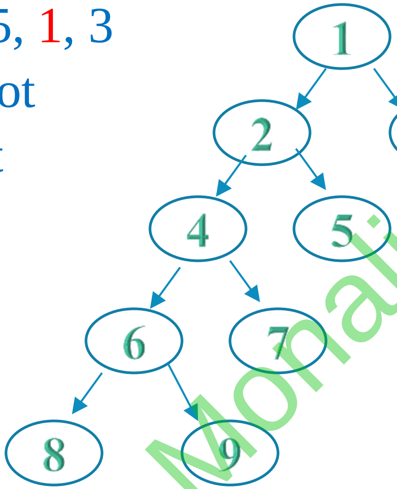
- GATE 2017,Set-1,Q6,1 mark:Let T be a binary search tree with 15 nodes. The minimum and maximum possible heights of T are:
- **Note:** The height of a tree with a single node is 0.
  (A) 4 and 15 respectively
  (B) 3 and 14 respectively
  (C) 4 and 14 respectively
  (D) 3 and 15 respectively
- Min height of a binary search tree = $\log_2(n+1) – 1 = \log_2(15+1) – 1 = 4 – 1 = 3$
- Max height of the binary search tree = n-1 = 15 – 1 = 14, where the tree is Skewed tree
- Ans : **(B)** 3 and 14 respectively

- GATE 2017,Set-1,Q20,1 mark:Let *T* be a tree with 10 vertices. The sum of the degrees of all the vertices in *T* is __________.
- A tree with 10 vertices has 9 edges.
- Edges=vertices-1
- Edges=10-1=9
- $\Sigma d(v) = 2|E|$
- =2*9=18
- Ans :18

- GATE 2017,Set-2,Q36,2 Mark: The pre-order traversal of a binary search tree is given by 12, 8, 6, 2, 7, 9, 10, 16, 15, 19, 17, 20.Then the post-order traversal of this tree is:
- **(A)** 2, 6, 7, 8, 9, 10, 12, 15, 16, 17, 19, 20
- **(B)** 2, 7, 6, 10, 9, 8, 15, 17, 20, 19, 16, 12
- **(C)** 7, 2, 6, 8, 9, 10, 20, 17, 19, 15, 16, 12
- **(D)** 7, 6, 2, 10, 9, 8, 15, 16, 17, 20, 19, 12
- Inorder :Left,Root,Right
- Preorder: Root,Left,Right
- Postorder:Left,Right,Root
- Inorder : 2, 6, 7, 8, 9, 10, 12, 15, 16, 17, 19, 20
- Preorder: 12, 8, 6, 2, 7, 9, 10, 16, 15, 19, 17, 20
- Postorder: 2, 7, 6, 10 , 9, 8, 15, 17, 20, 19, 16, 12
- Ans : **(B) 2, 7, 6, 10, 9, 8, 15, 17, 20, 19, 16, 12**

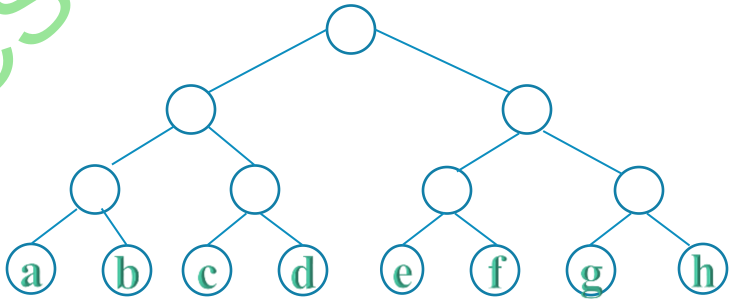


- GATE 2018,Q20,1 mark:The postorder traversal of a binary tree is 8, 9, 6, 7, 4, 5, 2, 3, 1. The inorder traversal of the same tree is 8, 6, 9, 4, 7, 2, 5, 1, 3. The height of a tree is the length of the longest path from the root to any leaf. The height of the binary tree above is ______.
- Postorder : 8, 9, 6, 7, 4, 5, 2, 3, 1
- Inorder :8, 6, 9, 4, 7, 2, 5, 1, 3
- Postorder:Left,Right,Root
- Inorder :Left,Root,Right

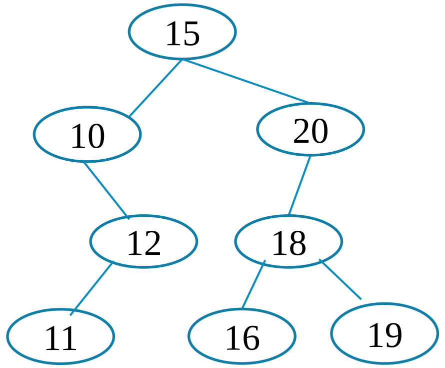


- Ans : 4

- GATE 2019,Q46,2 mark:Let T be a full binary tree with 8 leaves. (A full binary tree has every level full.) Suppose two leaves a and b of T are chosen uniformly and independently at random. The expected value of the distance between a and b in T (i.e., the number of edges in the unique path between a and b) is (rounded off to 2 decimal places) ___________ .
- Two leaf nodes can be selected in 8*8 = 64 ways.
- Path=0,probability =8/64=1/8 [(a,a)(b,b)... 8]
- Path =2,probability=8/64=1/8
- [(a,b)(b,a)(c,d)(d,c)…8]
- Path=4,probability =16/64=1/4
- [(a,c)(a,d)(b,c)(b,d)(c,a)(c,b)………16]
- Path=6 ,probability =32/64=1/2 [(a,e)(a,f)(a,g)(a,h)………32]
- ∴ Expected value = Σ (Path length × Probability of selecting path)
- = 0×1/8 +2×1/8 + 4×1/4 + 6×1/2
- = 0+1/4+1+3=1/4+4
- = 17/4 = 4.25
- Ans :4.25



- GATE 2020,Q5,1 mark:The preorder traversal of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19.Which one of the following is the postorder traversal of the tree?
- **(A)** 10, 11, 12, 15, 16, 18, 19, 20
  (B) 11, 12, 10, 16, 19, 18, 20, 15
  (C) 20, 19, 18, 16, 15, 12, 11, 10
  (D) 19, 16, 18, 20, 11, 12, 10, 15

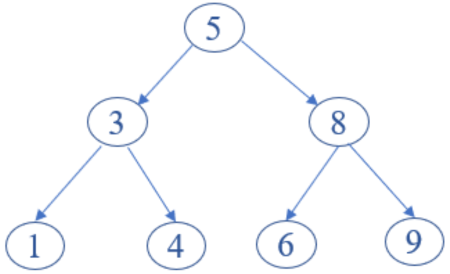


- Inorder : (Left,Root,Right)
- Preorder :(Root ,Left,Right)
- Postorder :(Left,Right,Root)
- Inorder : 10, 11, 12, 15, 16, 18, 19, 20
- Preorder : 15, 10, 12, 11, 20, 18, 16, 19
- Postorder :11, 12, 10, 16, 19, 18, 20, 15
- Ans : (B) 11,12,10,16,19,18,20,15

- GATE 2020,Q6,1 mark:What is the worst case time complexity of inserting n^2 elements into an AVL-tree with n elements initially?
- (A) $\Theta(n^4)$ (B) $\Theta(n^2)$ (C) $\Theta(n^2 \log n)$ (D) $\Theta(n^3)$
- AVL Tree all operations(insert, delete and search) will take $\Theta(\log n)$ time.
- For one element $\Theta(\log n)$
- Inserting n^2 elements. In worst case it will take $\Theta(n^2 \log n)$ time.
- Ans: (C) $\Theta(n^2 \log n)$

- GATE 2020 CS,Q41,2 mark:In a balanced binary search tree with n elements, what is the worst case time complexity of reporting all elements in range [a,b]? Assume that the number of reported elements is k.
- (A) Θ(logn) (B) Θ(logn+k) (C) Θ(klogn) (D) Θ(nlogk)

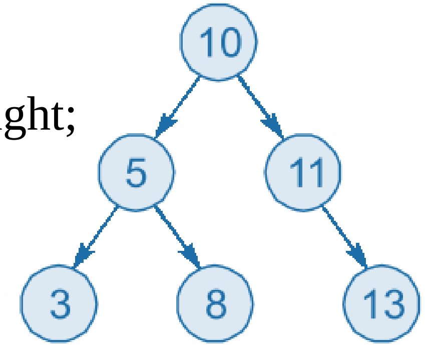
- Let n=7,a=3,b=8,k=5
- First, you'll have to check if the elements a and b are present in the BST or not.
- a(3),b(8) are present ,It will take 2log(n) time =Θ(log n)
- Traversing the k elements would take additional Θ(k) time.
- It will just traverse inorder and will get k element in range[a,b]
- So total running time =Θ(log n+k)
- Ans : (B) Θ(log n+k)



GATE CS 2023 | Question: 37

- Consider the C function foo and the binary tree shown.

```
typedef struct node {
    int val;
    struct node *left, *right;
} node;
int foo(node *p) {
    int retval;
    if (p == NULL)
        return 0;
    else {
        retval =p→val+foo(p →left)+foo(p →right);
     printf("%d ", retval);
     return retval;
        }
}
```



- When foo is called with a pointer to the root node of the given binary tree, what will it print?
- (A) 3 8 5 13 11 10 (B) 3 5 8 10 11 13
- (C) 3 8 16 13 24 50 (D) 3 16 8 50 24 13
- p points to root node 10.

- foo(10) retval = 10+ foo(5) + foo (11);
- foo(5) retval = 5+ foo(3) + foo (8);
- foo(3) retval = 3+ 0+ 0; print (3)
- foo(8) retval = 8+ 0+ 0; print (8)
- foo(5) retval = 5+ 3+8; print (16)
- foo(11) retval = 11+ 0+ foo (13);
- foo (13) retval = 13+ 0+ 0; print (13)
- foo(11) retval = 11+ 0+ 13; print (24)
- foo(10) retval = 10+ 16 + 24; print (50)
- Ans : (C) 3 8 16 13 24 50