Data Structures Test Bank 2015

Data Structures Binary Search Trees

Question 1

What is the worst case time complexity for search, insert and delete operations in a general Binary Search Tree?

A O(n) for all

B O(Logn) for all

C O(Logn) for search and insert, and O(n) for delete

D O(Logn) for search, and O(n) for insert and delete

Question 1 Explanation:

In skewed Binary Search Tree (BST), all three operations can take O(n). See the following example BST and operations.

```
          10
        /
       20
     /
    30
  /
 40

Search 40.
Delete 40
```

```
Insert 50.
```

Data Structures Test Bank 2015

Question 2

In delete operation of BST, we need inorder successor (or predecessor) of a node when the node to be deleted has both left and right child as non-empty. Which of the following is true about inorder successor needed in delete operation?

A Inorder Successor is always a leaf node

B Inorder successor is always either a leaf node or a node with empty left child

C Inorder successor may be an ancestor of the node

D Inorder successor is always either a leaf node or a node with empty right child

Question 2 Explanation:

Let X be the node to be deleted in a tree with root as 'root'. There are three cases for deletion 1) X is a leaf node: We change left or right pointer of parent to NULL (depending upon whether X is left or right child of its parent) and we delete X 2) One child of X is empty: We copy values of non-empty child to X and delete the non-empty child 3) Both children of X are non-empty: In this case, we find inorder successor of X. Let the inorder successor be Y. We copy the contents of Y to X, and delete Y. Sp we need inorder successor only when both left and right child of X are not empty. In this case, the inorder successor Y can never be an ancestor of X. In this case, the inorder successor is the leftmost node in right subtree of X. Since it is leftmost node, the left child of Y must be empty.

Question 3

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? (GATE CS 2011)

A 0

B 1

C n!

D (1/(n+1)).2nCn

Question 3 Explanation:

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.

Data Structures Test Bank 2015

Question 4

How many distinct binary search trees can be created out of 4 distinct keys?

A 4

B 14

C 24

D 42

Question 4 Explanation:

See question 2 of http://www.geeksforgeeks.org/data-structures-and-algorithms-set-23/ for explanation. The link also has a generalized solution.

Question 5

Which of the following traversal outputs the data in sorted order in a BST?

A Preorder

B Inorder

C Postorder

D Level order

Question 5 Explanation:

Inorder traversal of a BST outputs data in sorted orde

Data Structures Test Bank 2015

Question 6

Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree?

A 7 5 1 0 3 2 4 6 8 9

B 0 2 4 3 1 6 5 9 8 7

C 0 1 2 3 4 5 6 7 8 9

D 9 8 6 4 2 3 0 1 5 7

Question 6 Explanation:

In-order traversal of a BST gives elements in increasing order. So answer c is correct without any doubt.

Question 7

The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the height of the binary search tree (the height is the maximum distance of a leaf node from the root)? (GATE CS 2004)

A 2

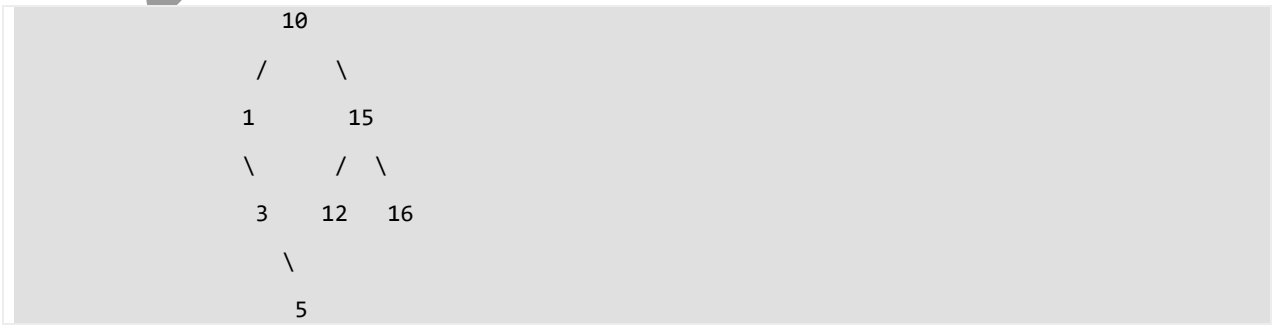
B 3

C 4

D 6

Question 7 Explanation:

Constructed binary search tree will be..



Data Structures Test Bank 2015

Question 8

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

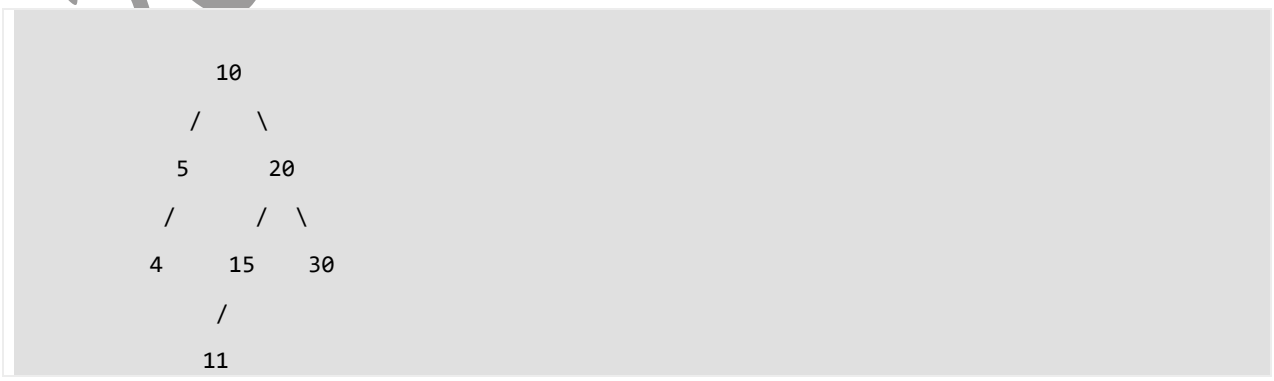
D 15, 10, 23, 25, 20, 35, 42, 39, 30

Question 8 Explanation:

The following is the constructed tree

Question 9

Consider the following Binary Search Tree



Data Structures Test Bank 2015

If we randomly search one of the keys present in above BST, what would be the expected number of comparisons?

A 2.75

B 2.25

C 2.57

D 3.25

Question 9 Explanation:

Expected number of comparisons = (1*1 + 2*2 + 3*3 + 4*1)/7 = 18/7 = 2.57

Question 10

Which of the following traversals is sufficient to construct BST from given traversals 1) Inorder 2) Preorder 3) Postorder

A Any one of the given three traversals is sufficient

B Either 2 or 3 is sufficient

C 2 and 3

D 1 and 3

Question 10 Explanation:

When we know either preorder or postorder traversal, we can construct the BST. Note that we can always sort the given traversal and get the inorder traversal. Inorder traversal of BST is always sorted.

Data Structures Test Bank 2015

Question 11

Consider the following code snippet in C. The function print() receives root of a Binary Search Tree (BST) and a positive integer k as arguments.

```
// A BST node
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);
    }
}
```

Run on IDE

What is the output of print(root, 3) where root represent root of the following BST.

A 10

B 16

C 20

D 20 10

Question 11 Explanation:

The code mainly finds out k'th largest element in BST, see K'th Largest Element in BST for details.

Data Structures Test Bank 2015

Question 12

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.

```
// A BST node
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

Question 12 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.

Data Structures Test Bank 2015

Question 13

You are given the postorder traversal, P, of a binary search tree on the n elements 1, 2, ..., n. You have to determine the unique binary search tree that has P as its postorder traversal. What is the time complexity of the most efficient algorithm for doing this?

A O(Logn)

B O(n)

C O(nLogn)

D none of the above, as the tree cannot be uniquely determined.

Question 13 Explanation:

See method 2 of http://www.geeksforgeeks.org/construct-bst-from-given-preorder-traversa/ Same technique can be used for postorder traversal.

Question 14

Suppose we have a balanced binary search tree T holding n numbers. We are given two numbers L and H and wish to sum up all the numbers in T that lie between L and H. Suppose there are m such numbers in T. If the tightest upper bound on the time to compute the sum is $O(n^a \log^b n + m^c \log^d n)$, the value of a + 10b + 100c + 1000d is ____.

A 60

B 110

C 210

D 50

Question 14 Explanation:

```
int getSum(node *root, int L, int H)
{
   // Base Case
   if (root == NULL)
      return 0;
```

```
   if (root->key < L)
        return getSum(root->right, L, H);

  if (root->key > H)
      return getSum(root->left, L, H)

   if (root->key >= L && root->key <=H)
      return getSum(root->left, L, H) + root->key +
             getSum(root->right, L, H);
}
```

The above always takes O(m + Logn) time. Note that the code first traverses across height to find the node which lies in range. Once such a node is found, it recurs for left and right children. Two recursive calls are made only if the node is in range. So for every node that is in range, we make at most one extra call (here extra call means calling for a node that is not in range).

Question 15

Let T(n) be the number of different binary search trees on n distinct elements.

$$T(n)=\sum_{k=1}^{n} T(k-1)\,T(x)$$

Then , where x is

A n-k+1

B n-k

C n-k-1

D n-k-2

Question 15 Explanation:

The idea is to make a key root, put (k-1) keys in one subtree and remaining n-k keys in other subtree.

Data Structures Test Bank 2015

Question 16

What are the worst-case complexities of insertion and deletion of a key in a binary search tree?

A Θ(logn) for both insertion and deletion

B Θ(n) for both insertion and deletion

C Θ(n) for insertion and Θ(logn) for deletion

D Θ(logn) for insertion and Θ(n) for deletion

Question 16 Explanation:

The time taken by search, insert and delete on a BST is always proportional to height of BST. Height may become O(n) in worst case.

Question 17

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

Question 17 Explanation:

71, 65, 84, 69, 67, 83

```
      71
    /    \
  65      84
    \     /
    69   83
    /
  67
```

Data Structures Test Bank 2015

مع تحيات مبرمجون بلا حدود 2015

تامر أبو الزينات 0786358466

محمود عماد 0785780440