Data Structures Test Bank 2015

Data Structures Balanced Binary Search Trees

Question 1

The worst case running time to search for an element in a balanced in a binary search tree with n2^n elements is

(A)

(B)

(C)

(D)

A A

B B

C C

D D

Question 1 Explanation:

Time taken to search an element is [Tex]\Theta (h) [/Tex] where h is the height of Binary Search Tree (BST). The growth of height of a balanced BST is logerthimic in terms of number of nodes. So the worst case time to search an element would be [Tex]\Theta (Log(n*2^n)) [/Tex] which is [Tex]\Theta (Log(n) + Log(2^n)) [/Tex] Which is [Tex]\Theta (Log(n) + n) [/Tex] which can be written as [Tex]\Theta (n) [/Tex].

Question 2

What is the maximum height of any AVL-tree with 7 nodes? Assume that the height of a tree with a single node is 0.

A 2

B 3

C 4

D 5

Data Structures Test Bank 2015

Question 2 Explanation:

AVL trees are binary trees with the following restrictions. 1) the height difference of the children is at most 1. 2) both children are AVL trees Following is the most unbalanced AVL tree that we can get with 7 nodes

```
                 a
               /   \
             /       \
           b           c
         /   \        /
       /       \    /
      d         e   g
     /
    /
   h
```

Question 3

What is the worst case possible height of AVL tree?

A 2Logn

Assume base of log is 2

B 1.44log n

Assume base of log is 2

C Depends upon implementation

D Theta(n)

Question 3 Explanation:

See http://lcm.csa.iisc.ernet.in/dsa/node112.html

Data Structures Test Bank 2015

Question 4

Which of the following is AVL Tree?

```
A
        100
     /      \
    50       200
   /           \
 10            300


B
           100
       /       \
     50        200
    /        /     \
  10       150     300
 /
5


C
            100
       /          \
     50            200
    /  \          /     \
  10    60      150     300
 /                 \        \
5                   180      400
```

A Only A

B A and C

C A, B and C

D Only B

Question 4 Explanation:

A Binary Search Tree is AVL if balance factor of every node is either -1 or 0 or 1. Balance factor of a node X is [(height of X->left) - (height of X->right)]. In Tree B, the node with value 50 has balance factor 2. That is why B is not an AVL tree.

Data Structures Test Bank 2015

Question 5

Consider the following AVL tree.

```
         60

      /     \

    20      100

           /   \

         80    120
```

Which of the following is updated AVL tree after insertion of 70

```
A
        70
      /    \
    60      100
   /       /    \
 20       80    120

B
        100
      /    \
    60      120
   /  \     /
 20   70   80


C
        80
      /    \
    60      100
   /  \       \
 20   70      120

D
        80
      /    \
    60      100
   /       /   \
 20       70    120
```

A A

B B

C C

D D

Question 5 Explanation:

Refer following for steps used in AVL insertion. AVL Tree | Set 1 (Insertion)

```
After insertion of 70, tree becomes following
         60
      /     \
    20      100
           /   \
         80    120
        /
       70
```

We start from 50 and travel up. We keep travelling up till we find an unbalanced node. In above case, we reach the node 60 and see 60 got unbalanced after insertion and this isRight Left Case. So we need to apply two rotations

```
         60                                60                             80
      /     \       Right Rotate(100)  /      \     Left Rotate(60)   /    \
    20      100    -----------------> 20        80 ---------------> 60      100
           /   \                              /   \                / \         \
         80    120                          70      100           20   70       120
        /                                             \
      70                                              120
```

Question 6

Which of the following is a self-adjusting or self-balancing Binary Search Tree

A Splay Tree

B AVL Tree

C Red Black Tree

D All of the above

Question 6 Explanation:

See Splay Tree, AVL Tree and Red Black Tree

Question 7

Consider the following left-rotate and right-rotate functions commonly used in self-adjusting BSTs

```
T1, T2 and T3 are subtrees of the tree rooted with y (on left side)

or x (on right side)

                y                               x

               / \     Right Rotation          /  \

              x   T3   - - - - - - - >        T1   y

             / \       < - - - - - - -            / \

            T1  T2     Left Rotation            T2  T3
```

Which of the following is tightest upper bound for left-rotate and right-rotate operations.

A O(1)

B O(Logn)

C O(LogLogn)

D O(n)

Question 7 Explanation:

The rotation operations (left and right rotate) take constant time as only few pointers are being changed there. Following are C implementations of left-rotate and right-rotate 1

Question 8

Which of the following is true

A The AVL trees are more balanced compared to Red Black Trees, but they may cause more rotations during insertion and deletion.

B Heights of AVL and Red-Black trees are generally same, but AVL Trees may cause more rotations during insertion and deletion.

C Red Black trees are more balanced compared to AVL Trees, but may cause more rotations during insertion and deletion.

D Heights of AVL and Red-Black trees are generally same, but Red Black rees may cause more rotations during insertion and deletion.

Question 8 Explanation:

Red Black Tree with n nodes has height <= 2Log2(n+1) AVL Tree with n nodes has height less than $\text{Log}_{\varphi}(\sqrt{5}(n+2)) - 2$. Therefore, the AVL trees are more balanced compared to Red Black Trees, but they may cause more rotations during insertion and deletion. So if your application involves many frequent insertions and deletions, then Red Black trees should be preferred. And if the insertions and deletions are less frequent and search is more frequent operation, then AVL tree should be preferred over Red Black Tree.

Question 9

Which of the following is true about Red Black Trees?

A The path from the root to the furthest leaf is no more than twice as long as the path from the root to the nearest leaf

B At least one children of every black node is red

C Root may be red

D A leaf node may be red

Question 9 Explanation:

See http://en.wikipedia.org/wiki/Red%E2%80%93black_tree

Question 10

Which of the following is true about AVL and Red Black Trees?

A In AVL tree insert() operation, we first traverse from root to newly inserted node and then from newly inserted node to root. While in Red Black tree insert(), we only traverse once from root to newly inserted node.

B In both AVL and Red Black insert operations, we traverse only once from root to newly inserted node,

C In both AVL and Red Black insert operations, we traverse twiceL first traverse root to newly inserted node and then from newly inserted node to root.

D None of the above

Data Structures Test Bank 2015

Question 11

What is the worst case possible height of Red-Black tree? Assume base of Log as 2 in all options

A 2Log(n+1)

B 1.44 Logn

C 4Logn

D None of the above

Question 12

Is the following statement valid? *A Red-Black Tree which is also a perfect Binary Tree can have all black nodes*

A Yes

B No

Question 12 Explanation:

A perfect BST with all black nodes doesn't violate any of the Red-Black tree properties.

Question 13

```
Which of the following operations are used by Red-Black trees to maintain balance during
insertion/deletion?


a) Recoloring of nodes

b) Rotation (Left and Right)
```

A Only a

B Only b

C Both a and b

D Neither a nor b

Question 13 Explanation:

Both recoloring and rotation operations are used during insertion and deletion.

Question 14

A program takes as input a balanced binary search tree with n leaf nodes and computes the value of a function g(x) for each node x. If the cost of computing g(x) is min{no. of leaf-nodes in left-subtree of x, no. of leaf-nodes in right-subtree of x} then the worst-case time complexity of the program is

A $\Theta(n)$

B $\Theta(n\text{Log}n)$

C $\Theta(n^2)$

D $\Theta(n^2\log n)$

Question 14 Explanation:

```
The recurrence relation for the recursive function is

T(N) = 2 * T(N/2) + n/2

Where N is the total no. of nodes in the tree.

T(N) = 2 * (2*T(N/2) + n/2) + n/2

     = 4 * T(N/2) + 3(n/2)

Solve this till T(1) i.e. till we reach the root.

T(N) = c * T(N / 2^i) + (2*i - 1) * (n/2)
```

```
Where i = lg(N)
= lg((2n - 1) / 2)
O(c * T(N / 2^i) + (2*i - 1) * (n/2)) reduces to
O((2*i - 1) * (n/2))
O((2*( lg((2n - 1) / 2)) - 1) * (n/2)) ...sub the value of i.
O(n * ln(n))
```

Source: http://www.nid.iitkgp.ernet.in/DSamanta/courses/IT60101_2/Archives/Assignment-%20IC%20Binary%20Trees%20Solutions.pdf

مع تحيات مبرمجون بلا حدود 2015

0786358466 تامر أبو الزينات

0785780440 محمود عماد