**Binary Tree**

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).

advertisement

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 2λ – 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 2k-1 total number of nodes. Number of levels is at least ceil(log(N+1)).

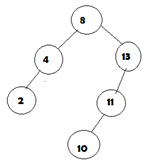
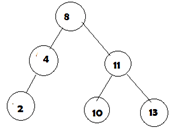
**AVL Tree**

1. What is an AVL tree?  
a) a tree which is balanced and is a height balanced tree  
b) a tree which is unbalanced and is a height balanced tree  
c) a tree with three children  
d) a tree with atmost 3 children  
View Answer

Answer: a  
Explanation: It is a self balancing tree with height difference atmost 1.

2. Why we need to a binary tree which is height balanced?  
a) to avoid formation of skew trees  
b) to save memory  
c) to attain faster memory access  
d) to simplify storing  
View Answer

Answer: a  
Explanation: In real world dealing with random values is often not possible, the probability that u are dealing with non random values(like sequential) leads to mostly skew trees, which leads to worst case. hence we make height balance by rotations.

3. Which of the below diagram is following AVL tree property?  
i.[](https://www.sanfoundry.com/wp-content/uploads/2017/08/data-structure-questions-answers-avl-tree-q3.png)  
ii.[](https://www.sanfoundry.com/wp-content/uploads/2017/08/data-structure-questions-answers-avl-tree-q3a.png)  
a) only i  
b) only i and ii  
c) only ii  
d) i is not a binary search tree  
View Answer

Answer: b  
Explanation: The property of AVL tree is it is height balanced tree with difference of atmost 1 between left and right subtrees. All AVL trees are binary search tree.

4. What is the maximum height of an AVL tree with p nodes?  
a) p  
b) log(p)  
c) log(p)/2  
d) p⁄2  
View Answer

Answer: b  
Explanation: Consider height of tree to be ‘he’, then number of nodes which totals to p can be written in terms of height as N(he)=N(he-1)+1+N(he-2). since N(he) which is p can be written in terms of height as the beside recurrence relation which on solving gives N(he)= O(logp) as worst case height.

5. To restore the AVL property after inserting a element, we start at the insertion point and move towards root of that tree. is this statement true?  
a) true  
b) false  
View Answer

Answer: a  
Explanation: It is interesting to note that after insertion, only the path from that point to node or only that subtrees are imbalanced interms of height.

**Join Sanfoundry@YouTube**

6. Given an empty AVL tree, how would you construct AVL tree when a set of numbers are given without performing any rotations?  
a) just build the tree with the given input  
b) find the median of the set of elements given, make it as root and construct the tree  
c) use trial and error  
d) use dynamic programming to build the tree  
View Answer

Answer: b  
Explanation: Sort the given input, find the median element among them, make it as root and construct left and right subtrees with elements lesser and greater than the median element recursively. this ensures the subtrees differ only by height 1.

7. What maximum difference in heights between the leafs of a AVL tree is possible?  
a) log(n) where n is the number of nodes  
b) n where n is the number of nodes  
c) 0 or 1  
d) atmost 1  
View Answer

Answer: a  
Explanation: At every level we can form a tree with difference in height between subtrees to be atmost 1 and so there can be log(n) such levels since height of AVL tree is log(n).

8. Consider the pseudo code:

**int** avl(binarysearchtree root):

**if**(not root)

**return** 0

left\_tree\_height = avl(left\_of\_root)

**if**(left\_tree\_height== -1)

**return** left\_tree\_height

right\_tree\_height= avl(right\_of\_root)

**if**(right\_tree\_height==-1)

**return** right\_tree\_height

Does the above code can check if a binary search tree is an AVL tree?  
a) yes  
b) no  
View Answer

9. Consider the below left-left rotation pseudo code where the node contains value pointers to left, right child nodes and a height value and Height() function returns height value stored at a particular node.

avltree leftrotation(avltreenode z):

avltreenode w =x-left

x-left=w-right

w-right=x

x-height=max(Height(x-left),Height(x-right))+1

w-height=max(missing)+1

**return** w

What is missing?  
a) Height(w-left), x-height  
b) Height(w-right), x-height  
c) Height(w-left), x  
d) Height(w-left)  
View Answer

10. Why to prefer red-black trees over AVL trees?  
a) Because red-black is more rigidly balanced  
b) AVL tree store balance factor in every node which costs space  
c) AVL tree fails at scale  
d) Red black is more efficient  
View Answer

Answer: b  
Explanation: Every node in an AVL tree need to store the balance factor (-1, 0, 1) hence space costs to O(n), n being number of nodes. but in red-black we can use the sign of number (if numbers being stored are only positive) and hence save space for storing balancing information. there are even other reasons where redblack is mostly prefered.

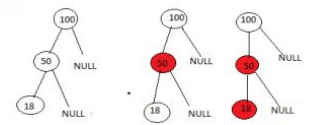
**Red Black Tree**

1. What is the special property of red-black trees and what root should always be?  
a) a color which is either red or black and root should always be black color only  
b) height of the tree  
c) pointer to next node  
d) a color which is either green or black  
View Answer

Answer: a  
Explanation: An extra attribute which is a color red or black is used. root is black because if it is red then one of red-black tree property which states that number of black nodes from root to null nodes must be same, will be violated.

2. Why do we impose restrictions like  
. root property is black  
. every leaf is black  
. children of red node are black  
. all leaves have same black  
a) to get logarithm time complexity  
b) to get linear time complexity  
c) to get exponential time complexity  
d) to get constant time complexity  
View Answer

Answer: a  
Explanation: We impose such restrictions to achieve self balancing trees with logarithmic complexities for insertions, deletions, search.

3. Cosider the below formations of red-black tree.  
[](https://www.sanfoundry.com/wp-content/uploads/2017/08/data-structure-questions-answers-red-black-tree-q3.png)  
All the above formations are incorrect for it to be a redblack tree. then what may be the correct order?  
a) 50-black root, 18-red left subtree, 100-red right subtree  
b) 50-red root, 18-red left subtree, 100-red right subtree  
c) 50-black root, 18-black left subtree, 100-red right subtree  
d) 50-black root, 18-red left subtree, 100-black right subtree  
View Answer

Answer: a  
Explanation: Considering all the properties of red-black tree, 50 must be the black root and there are two possibilities for subtrees. one is option “50-black root, 18-red left subtree, 100-red right subtree” and other is making all nodes of the tree to be black.

4. What are the operations that could be performed in O(logn) time complexity by red-black tree?  
a) insertion, deletion, finding predecessor, successor  
b) only insertion  
c) only finding predecessor, successor  
d) for sorting  
View Answer

Answer: a  
Explanation: We impose restrictions to achieve logarithm time complexities.  
impose restrictions are:  
. root property is black  
. every leaf is black  
. children of red node are black  
. all leaves have same black.

5. Which of the following is an application of Red-black trees and why?  
a) used to store strings efficiently  
b) used to store integers efficiently  
c) can be used in process schedulers, maps, sets  
d) for efficient sorting  
View Answer

Answer: c  
Explanation: RB tree is used for Linux kernel in the form of completely fair scheduler process scheduling algorithm. It is used for faster insertions, retrievals.

advertisement

6. When it would be optimal to prefer Red-black trees over AVL trees?  
a) when there are more insertions or deletions  
b) when more search is needed  
c) when tree must be balanced  
d) when log(nodes) time complexity is needed  
View Answer

Answer: a  
Explanation: Though both trees are balanced, when there are more insertions and deletions to make the tree balanced, AVL trees should have more rotations, it would be better to use red-black. but if more search is required AVL trees should be used.

7. Why Red-black trees are preferred over hash tables though hash tables have constant time complexity?  
a) no they are not preferred  
b) because of resizing issues of hash table and better ordering in redblack trees  
c) because they can be implemented using trees  
d) because they are balanced  
View Answer

Answer: b  
Explanation: Redblack trees have O(logn) for ordering elements in terms of finding first and next elements. also whenever table size increases or decreases in hash table you need to perform rehashing which can be very expensive in real time. also red black stores elements in sorted order rather than input order.

8. How can you save memory when storing color information in Red-Black tree?  
a) using least significant bit of one of the pointers in the node for color information  
b) using another array with colors of each node  
c) storing color information in the node structure  
d) using negative and positive numbering  
View Answer

Answer: a  
Explanation: The node pointers can be used to store color with the help of significant bits. the exceptions of this method are in languages like java where pointers are not used this may not work.

9. When to choose Red-Black tree, AVL tree and B-trees?  
a) many inserts, many searches and when managing more items respectively  
b) many searches, when managing more items respectively and many inserts respectively  
c) sorting, sorting and retrieval respectively  
d) retrieval, sorting and retrieval respectively  
View Answer

Answer: a  
Explanation: Red black when frequent inserts and deletes, AVL when less frequent inserts and deletes, B-tree when using paging from a slow storage device.