Data Structures Trees Job Interview Questions And Answers

Interview Questions Answers

https://interviewquestionsanswers.org/

About Interview Questions Answers

Interview Questions Answers . ORG is an interview preparation guide of thousands of Job Interview Questions And Answers, Job Interviews are always stressful even for job seekers who have gone on countless interviews. The best way to reduce the stress is to be prepared for your job interview. Take the time to review the standard interview questions you will most likely be asked. These interview questions and answers on Data Structures Trees will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts.

If you find any **question or answer** is incorrect or incomplete then you can **submit your question or answer** directly with out any registration or login at our website. You just need to visit Data Structures Trees Interview Questions And Answers to add your answer click on the *Submit Your Answer* links on the website; with each question to post your answer, if you want to ask any question then you will have a link *Submit Your Question*; that's will add your question in Data Structures Trees category. To ensure quality, each submission is checked by our team, before it becomes live. This Data Structures Trees Interview preparation PDF was generated at **Wednesday 29th November, 2023**

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter for latest Jobs and interview preparation guides.
https://twitter.com/InterviewQA

If you need any further assistance or have queries regarding this document or its material or any of other inquiry, please do not hesitate to contact us.

Best Of Luck.

Interview Questions Answers.ORG Team
https://InterviewQuestionsAnswers.ORG/
Support@InterviewQuestionsAnswers.ORG

Data Structures Trees Interview Questions And Answers Guide.

Question - 1:

Can you explain implementation of deletion from a binary tree?

Ans:

To implement the deletion from a binary tree, there is a need to consider the possibilities of deleting the nodes. They are:
- Node is a terminal node: In case the node is the left child node of its parent, then the left pointer of its parent is set to NULL. In all other cases, if the node is right child node of its parent, then the right pointer of its parent is set to NULL.
- Node has only one child: In this scenario, the appropriate pointer of its parent is set to child node.
- Node has two children: The predecessor is replaced by the node value, and then the predecessor of the node is deleted.

View All Answers

Question - 2:

Do you know implementation of traversal of a binary tree?

Ans:

Binary tree traversal is a process of visiting each and every node of the tree. The two fundamental binary tree traversals are 'depth-first' and 'breadth-first'.
The depth-first traversal are classified into 3 types, namely, pre-order, in-order and post-order.
Pre-order: Pre-order traversal involves visiting the root node first, then traversing the left sub tree and finally the right sub tree.
In-order: In-order traversal involves visiting the left sub tree first, then visiting the root node and finally the right sub tree.
Post-order: Post-order traversal involves visiting the left sub tree first, then visiting the right sub tree and finally visiting the root node.
The breadth-first traversal is the 'level-order traversal'. The level-order traversal does not follow the branches of the tree. The first-in first-out queue is needed to traversal in level-order traversal.

View All Answers

Question - 3:

What is threaded binary tree. Explain its common uses?

Ans:

A threaded binary tree is structured in an order that, all right child pointers would normally be null and points to the 'in-order successor' of the node. Similarly, all the left child pointers would normally be null and points to the 'in-order predecessor' of the node.
Uses of Threaded binary tree:
- Traversal is faster than unthreaded binary trees
- More subtle, by enabling the determination of predecessor and successor nodes that starts from any node, in an efficient manner.
- No stack overload can be carried out with the threads.
- Accessibility of any node from any other node
- It is easy to implement to insertion and deletion from a threaded tree.

View All Answers

Question - 4:

Explain B+ tree? Explain its uses?

Ans:

B+ tree represents the way of insertion, retrieval and removal of the nodes in a sorted fashion. Every operation is identified by a 'key'. B+ tree has maximum and minimum bounds on the number of keys in each index segment, dynamically. All the records in a B+ tree are persisted at the last level, i.e., leaf level in the order of keys.
B+ tree is used to visit the nodes starting from the root to the left or / and right sub tree. Or starting from the first node of the leaf level. A bi directional tree traversal is possible in the B+ tree.

View All Answers

Question - 5:

What is pre-order and in-order tree traversal?

Ans:

A non-empty binary tree is traversed in 3 types, namely pre-order, in-order and post-order in a recursive fashion.
Pre-order:
Pre-order process is as follows:
- Visit the root node
- Traverse the left sub tree
- Traverse the right sub tree
In-Order:
In order process is as follows:
- Traverse the left sub tree
- Visit the root node
- Traverse the right sub tree

View All Answers

Question - 6:

How to find the depth of a binary tree?

Ans:

The depth of a binary tree is found using the following process:
1. Send the root node of a binary tree to a function
2. If the tree node is null, then return false value.
3. Calculate the depth of the left tree; call it 'd1' by traversing every node. Increment the counter by 1, as the traversing progresses until it reaches the leaf node. These operations are done recursively.
4. Repeat the 3rd step for the left node. Name the counter variable 'd2'.
5. Find the maximum value between d1 and d2. Add 1 to the max value. Let us call it 'depth'.
6. The variable 'depth' is the depth of the binary tree.

View All Answers

Question - 7:

Explain binary tree? Explain its uses?

Ans:

A binary tree is a tree structure, in which each node has only two child nodes. The first node is known as root node. The parent has two nodes namely left child and right child.
Uses of binary tree:
- To create sorting routine.
- Persisting data items for the purpose of fast lookup later.
- For inserting a new item faster

View All Answers

Question - 8:

Explain ree database. Explain its common uses?

Ans:

A tree is a data structure which resembles a hierarchical tree structure. Every element in the structure is a node. Every node is linked with the next node, either to its left or to its right. Each node has zero or more child nodes. The length of the longest downward path to a leaf from that node is known as the height of the node and the length of the path to its root is known as the depth of a node.
Common Uses:
- To manipulate hierarchical data
- Makes the information search, called tree traversal, easier.
- To manipulate data that is in the form of sorted list.
- To give visual effects for digital images using as a work flow by compositing them.

View All Answers

Question - 9:

Explain a B+ tree?

Ans:

It is a dynamic, multilevel index, with maximum and minimum bounds on the number of keys in each index segment. all records are stored at the lowest level of the tree; only keys are stored in interior blocks.

View All Answers

Question - 10:

What is threaded binary trees?

Ans:

In a threaded binary tree, if a node 'A' has a right child 'B' then B's left pointer must be either a child, or a thread back to A.
In the case of a left child, that left child must also have a left child or a thread back to A, and so we can follow B's left children until we find a thread, pointing back to A.
This data structure is useful when stack memory is less and using this tree the treversal around the tree becomes faster.

View All Answers

Question - 11:

Explain red-black trees?

Ans:

A red-black tree is a type of self-balancing binary search tree.
In red-black trees, the leaf nodes are not relevant and do not contain data.
Red-black trees, like all binary search trees, allow efficient in-order traversal of elements.
Each node has a color attribute, the value of which is either red or black.
Characteristics:
The root and leaves are black
Both children of every red node are black.
Every simple path from a node to a descendant leaf contains the same number of black nodes, either counting or not counting the null black nodes

View All Answers

Question - 12:

What is splay trees?

Ans:

A splay tree is a self-balancing binary search tree. In this, recently accessed elements are quick to access again
It is useful for implementing caches and garbage collection algorithms.
When we move left going down this path, its called a zig and when we move right, its a zag.
Following are the splaying steps. There are six different splaying steps.
1. Zig Rotation (Right Rotation)
2. Zag Rotation (Left Rotation)
3. Zig-Zag (Zig followed by Zag)
4. Zag-Zig (Zag followed by Zig)
5. Zig-Zig
6. Zag-Zag

View All Answers

Question - 13:

What is a B tree?

Ans:

A B-tree of order m (the maximum number of children for each node) is a tree which satisfies the following properties :
1. Every node has <= m children.
2. Every node (except root and leaves) has >= m/2 children.
3. The root has at least 2 children.
4. All leaves appear in the same level, and carry no information.
5. A non-leaf node with k children contains k - 1 keys

View All Answers

Question - 14:

Explain Trees using C++ with an example?

Ans:

Tree is a structure that is similar to linked list. A tree will have two nodes that point to the left part of the tree and the right part of the tree. These two nodes must be of the similar type.
The following code snippet describes the declaration of trees. The advantage of trees is that the data is placed in nodes in sorted order.

```
struct TreeNode
{
     int item; // The data in this node.
     TreeNode *left; // Pointer to the left subtree.
     TreeNode *right; // Pointer to the right subtree.
}
```

The following code snippet illustrates the display of tree data.

```
void showTree( TreeNode *root )
{
    if ( root != NULL ) { // (Otherwise, there's nothing to print.)
    showTree(root->left); // Print items in left sub tree.
    cout << root->item << " "; // Print the root item.
    showTree(root->right ); // Print items in right sub tree.
}
} // end inorderPrint()
```

View All Answers

Computer Programming Most Popular & Related Interview Guides