Data Structures Viva Question & Answers

1. What is Data Structure?

A data structure is a model for organizing and storing data. Stacks, Queue, Linked Lists, and Trees are the examples of different data structures. Data structures are classified as either linear or non-linear:

- A data structure is said to be linear if only one element can be accessed from an element directly. Eg: Stacks, Lists, Queues.
- A data structure is non-linear if more than one elements can be accessed from an element directly. Eg: Trees, Graphs, Heap.

2. Define ADT?

An abstract data type is a set of objects together with a set of operations. Abstract data types are mathematical abstraction. Objects such as lists, sets, graphs, trees can be viewed as abstract data types.

3. What do you mean by LIFO and FIFO?

LIFO stands for 'Last In First Out', it says how the data to be stored and retrieved. It means the data or element which was stored last should come out first. Stack follows LIFO.

FIFO stands for 'First In First Out', here the data(or element) which was stored first should be the one to come out first. It is implemented in Queue.

4. What is Stack?

A stack is a list of elements in which insertion and deletions can take place only at one end, this end is called as stack 'top'. Only top element can be accessed. A stack data structure has LIFO (Last In First Out) property. A stack is an example of linear data structure.

5. What operations can be done on Stack?

The following operations can be performed on stack:

- **Push** operation inserts new element into stack.
- **Pop** operation deletes the top element from the stack.
- **Peek** returns or give the top element without deleting it.
- **Isempty()** operation returns whether stack is empty or not.

6. List some applications of stack?

Some well-known applications of stack are as follow:

1. Infix to Postfix conversion.

2. Evaluatiion of postfix expression.
3. Check for balanced parantheses in an expression.
4. Stack is very important data structure being used to implement function calls efficiently.
5. Parsing
6. Simulation of recursion function.
7. String reverse using stack.

7. What is a Queue?

A Queue is a particular data structure in which the elements in the collection are kept in order. Unlike Stack, queue is opened at both end. Queue follows 'FIFO' method where element is inserted from rear end and deleted or removed from front end.

8. What operations can be done on Queue?

The following operations can be performed on queue:

- **Enqueue** operation inserts new element in rear of the list.
- **Dequeue** operation deletes the front element from the list.
- **Isempty()** operation checks whether queue is empty or not.

9. Where do we use Queue?

Some well-known applications of queue are as follow:

1. For findind level order traversal efficiently.
2. For implementing BFS efficiently.
3. For implementing Kruskal algorithm efficiently.

10. What is Binary Tree?

A binary tree is a 2-ary tree in which each node(N) has atmost 2 children (either 0 or 1). The node with 2 children are called internal nodes, and the nodes with 0 children are called external nodes or leaf nodes.

11. What is Complete Binary Tree?

A Binary tree is complete binary tree if all levels are completely filled except possibly the last/lowest level are full and in the last/lowest level all the items are on the left.

12. What is Perfect Balanced Binary Tree?

A perfect balanced binary tree is a binary tree where each node has same number of nodes in both subtrees.

13. Define Height in a tree?

Height is a general measured as number of edges from the root to deepest node in the tree.

14. What is tree traversal?

Traversal is used to visit each node in the tree exactly once. A full traversal of a binary tree gives a linear ordering of the data in the tree. There are 3 different type of tree traversal:

1. Inorder Traversal
2. Preorder Traversal
3. Postorder Traversal

15. How does Inorder Traversal work?

1. Traverse the left sub tree of the root node R in inorder.
2. Visit the root node R.
3. Traverse the right sub tree of the root node R in inorder.

16. How does Preorder Traversal work?

1. Traverse the left sub tree of the root node R in preorder.
2. Traverse the right sub tree of the root node R in preorder.
3. Visit the root node R.

17. How does Postorder Traversal work?

1. Visit the root node R.
2. Traverse the left sub tree of the root node R in postorder.
3. Traverse the right sub tree of the root node R in postorder.

18. What is a Binary Search Tree?

A BST(Binary Search Tree) is a binary tree in which each node satisfies search property. Each node's key value must be greater than all values in its left subtree and must be less than all values in its right subtree.

19. What is a AVL Tree?

AVL tree is a self-balancing binary search tree with height-balanced condition. For every node the height of left subtree and right subtree can be differ by at most 1.

20. What is a B-tree?

A B-Tree is a tree data structure that keeps data sorted and allows searches, insertions, deletions, and sequential access in logarithmic amount of time.

In B-Tree, internal nodes can have a variable number of child nodes within some pre-defined range. A B-tree is kept balanced by requiring that all leaf nodes are at the same depth.

A B-tree of order m is a tree which satisfies the following properties:

1. Every node has atmost 2m children.
2. Every node (except root and leaves) has atleast m children.
3. The root has atleast two children if it is not a leaf node.
4. All leaves appear in the same level, and carry information.
5. A non-leaf node with k children contain k-1 keys.
6. Leaf nodes contain atleast m-1 children and atmost 2m-1 keys.

21. What is a B^+ Tree?

A B^+ Tree is a tree data structure which represents sorted data in a way that allows for efficient insertion, retrieval and removal of records, each of which is identified by a key. It is a dynamic multilevel index, with maximum and minimum bounds on the number of keys in each segment (usually called a 'block' or 'node')

22. What is a Binary Heap Tree?

A Binary Heap tree is a balanced binary tree where root node are compared with its children and placed accordingly. Heap have two properties namely, structure and order property.

Structure Property

It is a complete binary tree where all the levels except possibly the last/lower level are full and in the last/lower level all the items are on the left. Due to this fact that binary heap is a complete binary it can be implemented using a simple array.

Order Property

The heap order property for MinHeap is 'a parent is less than (or equal to) its children' whereas for MaxHeap 'a parent is greater than its children'.

23. What is a bubble sort?

Bubble sort is a simple sorting algorithm, it works by repeatedly stepping through the list to be sorted comparing each pair of adjacent items and swapping if they are in the worng order.

First pass bubbles out the largest element and places it in the last position and second pass places the second largest element in the second last position and so on. Thus in the last pass smallest element is placed in the first position.

The running time is the total number of comparisons that is n(n-1)/2 which implies $O(n^2)$ time complexity.

24. What is Insertion Sort?

Insertion sort is a simple comparison sorting algorithm, every iteration of insertion sort removes an element from the input data and insert it into the correct position in the already sorted list until no input element remains.

The running time is the total number of comparisons that is n(n-1)/2 which implies O(n^2) time complexity.

25. What is Selection Sort?

Insertion sort is a simple comparison sorting algorithm, the algorithm works as follows:

1. Find the minimum value in the list.
2. Swap it with the minimum value in the first position.
3. Repeat the steps above for the remainder of the list (starting at the second position and advancing each time).

The running time is the total number of comparisons that is n(n-1)/2 which implies O(n^2) time complexity.

26. What is Merge Sort?

Merge sort is an O(nlogn) comparison-based divide and conquer sorting algorithm, it works as follows:

1. If the list is of length 0 or 1, then it is already sorted.
2. Divide the unsorted list into two sub lists of about half the size.
3. Sort each sublist recursively by re-applying merge sort algorithm.
4. Merge the two sublists back into one sorted list.

If the running time of merge sort for a list of length n is T(n) then the recurrence relation is T(n) = 2T(n/2) + n, thus after simplifying the recurrence relation T(n) = O(nlogn).

27. What is Heap Sort?

Heap Sort is a comparison-based sorting algorithm which is much more efficient version of selection sort, it works by determining the largest (or smallest) element of the list, placing that at the end (or beginning)of the list, then continuing with the rest of the list, but accomplishes this task efficiently by using a data structure called a heap. Once the data list has been made into a heap, the root node is gurantedd to be the largest (or smallest) element.

28. What is Quick Sort?

Quick sort sorts by employing a divide and conquer strategy to divide a list into two sub-lists. The steps are:

1. Pick an element, called a pivot, from the list.
2. Reorder the list so the elements which are less than the pivot come before the pivot and the elements greater than pivot come after it. After this partitioning the pivot is in its final positon.
3. Recursively sort the sub-list of lesser elements and the sub-list of greater elements.

The running time complexity for worst case is O(n^2) and for best and average case it is same i.e., O(nlogn).

29. What is a Graph?

A graph is a pair of sets (V,E), where V is the sets of vertices and E is the set of edges connecting the pair of vertices.

30. What is a Directed and Undirected Graph?

A graph is directed if each edge of graph has a direction between vertices.

A graph is undirected if there are no direction between vertices.

31. What is a Connected Graph?

A undirected graph is connected if there is a path from every vertex to every other vertex. A directed graph with this property is called strongly connected.

32. What is a Complete Graph?

A complete graph is a graph in which every vertex is connected to every other vertex. That means each vertex's degree in undirected graph is n-1.

33. What is a Acyclic Graph?

An acyclic graph is a graph which does not have cycles.

34. How are graph represented?

A graph can be represented in two forms 'Adjacency matrix' and 'Adjacency list'.

Adjacency matrix is a two dimensional arrays where for each edge, (u,v) A[u,v] = true otherwise it will be false. The space complexity to represent a graph using this is O($|V|^2$).

Adjacency list are used where each list stores adjacent vertices of the corresponding vertex. The space complexity to represent a graph using this is O(|V|+|E|).

35. What is a Spanning Tree?

A Spanning tree is a graph which must include every vertex and if the graph contain n vertices, spanning tree must contain exactly (n-1) edges and is a subgraph of G.

36. What is a Minimum Spanning Tree?

A Minimum Spanning tree is a spanning tree such that the summ of all the weights of edges in spanning tree is minimum.

37. Explain algorithm for finding Minimum Spanning Tree?

There are 2 widely used algorithm for finding minimum spanning tree:

1. Prim's Algorithm
2. Kruskal Algorithm

Prims Algorithm computes the minimum spanning tree by including appropriate vertex and thus one edge into existing partially constructed tree in successive stages. The time complexity of the Prim’s Algorithm is O((V+E)logV).

Kruskal Algorithm computes the minimum spanning tree by adding edges one by one into a growing spanning tree.

Initially from the graph G, consider a forest of all the vertices without any edge.

1. Sort all the edges in ascending order to their costs.
2. Include the edge with minimum cost into the forest if it does not form a cycle in the partially condtructed tree.
3. Repeat step (2) until no edge can be added to the tree.

https://www.scribd.com/document/514549530/Data-Structure-Viva-Questions

Model Viva Questions for “Name of the Lab: Data Structure of lab”
Common to: CS 4th sem
Title of the Practical: Program to search an element of array using linear search.
Q1 Define searching process?
A1 searching is the process of finding an element within the list of elements stored in any order or randomly.
Q2 How many types of searching are there?
A2 There is basically two types of searching:-linear search and Binary search.
Q3Define: linear search?
A3 In linear search, we access each elements of an array one by one sequentially and see weather it is desired element
or not.
Q4 Why binary search method is more efficient then liner search?
A4 It is because less time is taken by linear search to search an element from the sorted list of elements.
Q5 Efficiency of linear search ?
A5 The time taken or the number of comparisons made in searching a record in a search table determines the efficiency
of the technique.
Q6 What do you understand by the term “linear search is unsuccessful”?
A6 search will be unsuccessful if all
the elements are accessed and the desired elements are not found.
Q7 What is worse case?
A7 In the worse case, the no. of average case we may have to scan half of the size
of the array (n/2).
Q8 What is the drawback of linear search?
A8 There is no requisite for the linear Search.
Q9 During linear search, when the record is present in first position then how many comparisons are made ?
A9 Only one comparison.
Q10 During linear search, when the record is present in last position then how many comparisons are made?
A10 n comparisons have to be made.
Title of the Practical: Program to reverse the element of array.
Insertion and deletion on array at specified position.
Q1 What is an array & how many types of arrays represented in memory?
A1 An array is a structured datatype made up of a finite, fixed size, collection of homogeneous ordered elements.
Types of array: One-Dimentional array, Two-Dimentional array, Multi-Dimentional array.
Q2 How can be declared an array?
A2 data_type var_name[Expression];
Q3 How can be insert an element in an array?
A3 Insertion of a new element in an array can be done in two ways:
* Insertion at the end of array.
* Insertion at required position.
Q4 How can be delete an element in an array?
A4- Deleting an element at the end of an array presents no difficulties, but deleting element somewhere in the middle of
the array.
Q5 How many types of implementation of a two-dimentional array?

A5 * Row-major implementation
* Column-major implementation
Q6 What is array of pointers?
A6 Array of pointer refers to homogeneous collection of pointer that is collection of pointer of same datatype.
Q7 What are limitations of array?
A7 * These are static structures.
* It is very time consuming.
Q8 Where the elements of the array are stored respectively in successive?
A8 Memory locations.
Q9 How can merge two arrays?
A9 Simplest way of merging two arrays is that first copy all elements of one array into a third empty array and the copy all
the elements of other array into third array.
Q10 What is the operations of array?
A10 Insertion, Deletion, Traversing, Merging.
Title of the Practical: Program based on structure union
Q1 What are structures?
A1 Structures are used to store different types.
Q2 What are arrays?
A2 Arrays are used to store similar data types.
Q3 Is array of structures possible?
A3 Is array of structures are possible.
Q4 How structures are declared?
A4 Declaration: struct book
{ char name;
Float price;
Int pages;
};
Q5 Why we use functions?
A5 Writing functions avoids rewriting the same code over and over.
Q6 Name some library functions?
A6 printf(), scanf() are examples of library function.
Q7 What are user defined functions?
A7 Functions declared by the user are called user defined functions.
Q8 Explain call by value?
A8 When the value is passed in the function is called call by value.
Q9 Explain call by reference?
A9 When the address of the value is passed is called call by reference.
Q10 Why we used keyword „break‟?
A10 When break is encountered inside any loop, control automatically passes to the first statement after the
loop.
Title of the Practical: Program to implement PUSH and POP operation on stack.
Q1 What is stack?
A1 Stack is an ordered collection of elements like array.
Q2 What are the operations performed on stack?
A2 Push for insertion and pop for deletion.
Q3 What is push operation?

A3 When an element is inserted into the stack is called push operation
Q4 What is pop operation.
A4 When an element is deleted from the stack is called pop operation.
Q5 How stacks are implemented?
A5 Stacks are implemented in two ways: static implementation –array
Dynamic implementation –pointers.
Q6 What are the applications of stack?
A6 infix , post fix and prefix notations are the applications of stack.
Q7 What is recursion.
A7 Recursion is defined as function calling itself.
Q8 What are the different types of stack
A8 Direct: a system calls itself from witin itself.
Indirect: two functions mutually calls one another
Q9 Define "Top of stack"
A9 Pointer indicating the top element of the stack.
Q10 Is stack is prinmitive or non primitive data structure ?
A10 Non primitive data structure.
Title of the Practical: Program based on infix to prefix and post fix notation.
Q1 What is Stack ?
A1 Stack is ordered collection of element like arrays out it has a special
feature that deletion and insertion of element can be done only from
one end called top of stack .It is also called LIFO(last in first out).
Q2 Application of Stack ?
A2 Infix
Prefix
Postfix
Q3 Terms used in Stack ?
A3 > Context
> Stack frames
> Maxsize
Q4 Explain infix in Stack ?
A4 The operator is written in between the operands.
Ex:- A+B
Q5 Explain Postfix in Stack ?
A5 The operator is written ofter the operands.It is also called suffix notation.
Ex:- AB+
Q6 Define operations on Stack ?
A6 The basic operation that can be performed on Stack are as follows:
>PUSH
>POP
Q7 Give postfix form for (A+B)*C/D
A7 AB+C*D/
Q8 Give postfix form for A+B/C-D
A8 ABC/+D-
Q9 Give prefix form for A/B^C+D
A9 +/A^BCD
Q10 Give prefix form for A*B+C
A10 +*ABC
Title of the Practical: Program based on queue & their operations for an application

Q1 Define queue.
A1Queue is an homogeneous collection of elements. It is logically a first in first out (FIFO) type of list.Queue means a
line.
Q2 In how many ways queue is implemented?
A2 Queues can be implemented in two ways: 1. static implementation (using array)
2. dynamic implementation (using pointers)
Q3 What is the rear end in a queue?
A3 in a queue new elements are added at one end called the rear end .
Q4 What is a front end?
A4 The existing elements in a queue are deleted from an end called the front end.
Q5 What is the value of front and rear end in an empty queue?
A5 Front =-1 and rear =-1.
Q6 Write an algorithm for deleting a node from a queue.
A6 1. If (front == null)

```
Write queue is empty and exit
Else
Temp = start
Value = temp ->no
Start = start -> next
Free (temp)
Return(value)
2. exit.
```

Q7 What are the different varirions in a queue?
A7 Major variations are: 1. circular queue 2. double ended queue 3. priority queue.
Q8 What is the full form of dequeue?
A8 DEQUEUE : Double ended queue.It is another form of queue in which both insertion and deletion are performed at the
either end.
Q9 When an element is added to the dequeue with n memory cells ,what happens to LEFT or RIGHT.
A9 If the element is added an the left , then LEFT is decreases by 1 (mod n) . IF the element is added on the right , then
RIGHT is increase by 1 (mod n)
Q10 What are the applications of queue.
A10 Applications are: 1. round robin technique for processor scheduling
2. All types of customer services(eg. RTC)
3. Printer server routines.
Title of the Practical: Program based on the implementation of circular queue.
Q1 What is Circular Queue?
A1 A Circular Queue is one in which the insertion of a new element
is done at the very first variation of the Queue if the last location
of the Queue is full.
Q2 Why use of Circular Queue?
A2 A Circular Queue over comes the problem of un utilized space in
linear Queue implemented as Array.
Q3 Explain Dequeue ?
A3 It is also a homogenous list of element in which insertion deletion
of element are perform both the ends we insert element from the
rear or from the front end.

Q4 What is Queue ?
A4 It is a non primitive linear data structure. In a Queue new element are added at the one end called rear end and the element are removed from another end called front end.
Q5 Variation in a Queue ?
A5 Circular Queue
-> Dequeue
->Priority Queue
Q6 What is Priority Queue ?
A6 Priority Queue determines the order in which the exist in the Queue the highest Priority items are removed first.
Q7 Application of Queue ?
A7 1> Customer service center
2> Ticket counter
3> Queue is used for determine space complexity & time complexity.
Q8 What is Queue implemention?
A8 >> Static implemention(Array)
>>Dynamic implemention(Pointer)
Q9 Define operations on queue?
A9 The basic operation that can be performed on queue are –
1>To insert an element in a Queue
2>To delete an element from the Queue
Q10 What is Stack ?
A10 Stack is ordered collection of element like arrays out it has a special feature that deletion and insertion of element can be done only from one end called top of stack .It is also called LIFO(last in first out).
Title of the Practical: Program based on list operations and its applications.
Q1 Define linked list.
A1 Linked list are special list of some data elements linked to one another .The logical ordering is represented by having
each element pointing to the next element. Each element is called a node.
Q2 What does a node define?
A2 Node has two parts: INFO – it stores the information and POINTER – which points to the next element.
Q3 What are the different types of linked list?
A3 Linked list are of four types: 1. singly linked list 2. doubly linked list 3. circular linked list 4. circular doubly linked list.
Q4 What are the different operations performed on a linked list?
A4 Basic operations are: creation, insertion, deletion , traversing, searching , concatenation and display.
Q5 What are the advantages of linked list?
A5 Advantages are:
1. linked lists are dynamic data structures
2. Efficient memory utilizations
3. Insertion and deletions are easier and efficient
Q6 What is null pointer?
A6 The link field of the last node contains NULL rather than a valid address. It is a null pointer and indicates the end of the
list.
Q7 What is external pointer?

A7 It is a pointer to the very first node in the linked list, it enables us to access the entire linked list.
Q8 What are the different notations used in a linked list.
A8 Node(p) : a node pointed to by the pointer p
Data (p) : data of the node pointed by p
Link (p) : address of the next node that follows yhe node pointed to by the pointer p.
Q9 What are advantages of circular linked list.
A9 1.Nodes can be accessed easily. 2. deletion of node is easier 3. concatenation and splitting of circular list is more
efficient.
Q10 A doubly linked list contains how many fields?
A10 Doubly linked list consists of three fields:
Data : contains the information
Next: contains the address of the next node
Prev: contains the address of the previous node
Title of the Practical: Program based on pointers in C.
Q1 Which of the following abstract data types are NOT used by Integer Abstract Data type group?
A1. A)Short
B)Int
C)float
D)long
Q2 What pointer type is used to implement the heterogeneous linked list in C?
A2 The answer is the void pointer. The heterogeneous linked list contains different data types in it's nodes and we need a
link, pointer, to connect them. Since we can't use ordinary pointers for this, we use the void pointer. Void pointer is a
generic pointer type, and capable of storing pointer to any type.
Q3: What issue do auto_ptr objects address?
A3: If you use auto_ptr objects you would not have to be concerned with heap objects not being deleted even if the exception is thrown.
Q4.: What is a dangling pointer?
A4: A dangling pointer arises when you use the address of an object after its lifetime is over. This may occur in situations like returning addresses of the automatic variables from a function or using the address of the memory block after it is freed.
Q5: What is the difference between a pointer and a reference?
A5: A reference must always refer to some object and, therefore, must always be initialized; pointers do not have such restrictions. A pointer can be reassigned to point to different objects while a reference always refers to an object with which it was initialized.
Q6: What is the difference between const char *myPointer and char *const myPointer?
A6: Const char *myPointer is a non constant pointer to constant data; while char *const myPointer is a constant pointer to non constant data.
Q7: From which is the pointer to object of a base class type compatible ?
A7: Pointers to object of a base class type is compatible with pointer to onject of a derived class . Therefore , we can use
single pointer variable to point to objects of base class as well as derived class.
Q8: What is a this pointer?
A8: A special pointer known as this pointer stores the address of the object that is currently invoking a member function.
Q9:How are objects passed to functions?
A9: Objects can be passed to functions through call-by –value as well as call-by- reference mechanism.

Q10: Why is Arrow operator (“-> “) used?
A10: The arrow operator is used to access the public members of the class with a pointer to an object.
Title of the Practical: Implementation of tree using linked list.
Q1 What is a tree?
A1 A tree is a non linear data structure in which items are arranged in a sorted sequence.It is a finite set of one or more
data items.
Q2 What is a root in a tree?
A2 A root is the first node in the hierarchical arrangement of data items.
Q3 What do you mean by degree of a tree?
A3 It is a maximum degree of nodes in a given tree .
Q4 What are non terminal nodes?
A4 Any node(Except the root node) is not zero is called non terminal node. Non terminal nodes are the intermediate
nodes in traversing the given tree.
Q5 Who are called siblings in a tree?
A5 The children nodes of a given parent node are called siblings. They are also called brothers.
Q6 . During linear search, when the record is present somewhere in search table, then how many comparisons are
made?
Ans- (n+1)/2.
Q7 How can access one-dimentional aray elements?
A7 array_name[index or subscript];
Q8 What is the traversing of an array?
A8 Traversing means to access all the elements of the array, starting from first element upto the last element in the array
one-by-one.
Q9Explain the types of searching?
A9There are two types of searching: - 1> linear searching, 2> binary searching.
Q10 What is the linear search?
A10 We access each element of and see whether. It is desire element or not a search will be unsuccessful.If the all
elements are access and the desired element is not found.
Title of the Practical: Implementation of different types of sorting techniques.
Q1 Explain the sorting?
A1 Searching and sorting and merging are three related operations which make the job of retrieval at data from the
storage device easier & speeder.
Q2 What are the different types of sorts in data structures ?
A2. 1> bubble sort 2> selection sort 3> insertion sort 4> quick sort 5> radix sort 6> merge sort 7> heap sort.
Q3 Define the bubble sort?
A3 In bubble sort each element is compared with its adduct element is larger than the second element is than the position
at the element are interchanging otherwise it is not change.
Q4 Define the selection sort?
A4 Selection sort technique is based upon the extension of the minimum maximum technique by means at a next at loop.
Q5 What is a bucket sort?

A5 The bucket sort is also called radix sort. It is a method that can be used to sort list at name alphabetically or
numerically.
Q6 What is insertion sort?
A6 An insertion sort is one that sorts a set of values by inserting values into an existing sorted file.
Q7 How many passes are required in selection sort?
A7 The selection sort makes first pass in n-1 comparisions, the second pass in n-2 comparisons and so on.
Q8 How does quick sort works?
A8 The quick sort works by partitioning the array to be sorted . And each partition is in turn sorted recursively.
Q9 How does bucket sort works?
A9 First or all the list of names is sorted according to the first letter of each name in26 buckets. In second pass , names
are arranged according to the second letter of each name and so on.
Q10 What is the efficiency of heap sort in worst case?
A10 O(n log n) is the efficiency of heap sort in worst case.
Title of the Practical: Implementation of binary search algorithm using binary tree
Q1 What do you understand by binary search?
A1 A binary search algorithm or binary chop is a technique for finding a particular value in a sorted list.
Q2 Explain the steps of algorithm of binary search?
A2 Step 1. Find the middle element of array f+1/2 = middle value
Step 2. Our derived element is greater than the main element
Q3 Define the complexity of binary search?
A3. In worst cause there is log(n+1) in the average causes.
Q4 Explain the difference between linear search and binary search?
A4 Linear search binary search
1. linear search can be applied on binary search can be applied on
any list of data. Sorted list of data.
2. It can be applied on linked list. It can not be applied on linked list
Q5 Define tree?
A5 A tree is a non linear data structures in which item are arranged in a sorted sequence.
Q6 Explain tree terminology?
1> root, 2> node.
A6 1> root: - it is the first in the hierarchical arrangement at data item.
2> node: - each item in a tree is called a node.
Q7 Explain the degree of node?
A7 It is the no. of sub tree of a node in a given tree.
Q8 Explain the terminal node & non terminal node?
A8 Terminal node:- a node with degree of 0 is called terminal node.
Non terminal node:- a node without degree of 0 is called non terminal node.
Q9 What do you understand by binary tree?
A9 The binary tree is a collection of finite node or set of data object.
Q10 Explain the application of binary tree?
A10 1> it is used in game programming.e.g. tic-tac-toe etc.
2> it is used to solve the problem of mathematical expression.
Title of the Practical: Assignment based on graph theory
Q1 What is agraph?

A1: A graph G consists of a set V of vertices (nodes) and a set E of edges . G= (V,E) is a finite and non empty set of
vertices.
Q2What do you mean by directed acyclic graph?
A2:Directed acyclic graph – A graph consists of no circular path and all the vertices are connected by the
directed edges.
Q3 What is adjacent matrix?
A3 Vertex v1 is said to be adjacent to a vertex v2 if there is an edge (v1, v2) or (v2,v1)
Q4 What is a path?
A4 A path from vertex w is a sequence of vertices , each adjacent to the next.
Q5 What do you mean by a cycle?
A5 A cycle is a path in which first and last vertices are the same.
Q6 What is a connected graph?
A6 A graph is called connected if there exists a path from any vertex to any other vertex.
Q7 What is a degree of a graph
A7 The number of edges incident on a vertex determine its degree. The degree of vertex u, is written as degree(u)
Q8 What do you understand by the term complete graph?
A8 A graph G is said to complete or fully connected if there is path from every vertex to every other vertex. A complete
graph with n vertices will have n(n-1)/2 edges.
Q9 What is a weighted graph?
A9 A graph is said to be weighted graph if every edge in a graph is assigned some weight or value. The weight of the
edge is a value representing the cost or distance between the vertices.
Q10 What ia a tree?
A10 A graph is a tree,iff it is connected and there are no cycles in the graph.

https://www.scribd.com/document/257045748/data-structure-viva-questions-doc

1) What is data structure?

Data structure refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with the data structure, we not only focus on one piece of data but the different set of data and how they can relate to one another in an organized manner.

---

2) Differentiate between file and structure storage structure.

The key difference between both the data structure is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

Free PDF Download: Data Structures Interview Questions & Answers

---

3) When is a binary search best applied?

A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is searched starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

---

4) What is a linked list?

A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link for data storage.

Don't Miss:

- Top 100 Tableau Interview Questions and Answers (2024)
- Top 100 Splunk Interview Questions and Answers (2024)
- Top 30 Data Analyst Interview Questions and Answers (2024)

---

5) How do you reference all the elements in a one-dimension array?

To reference all the elements in a one -dimension array, you need to use an indexed loop, So that, the counter runs from 0 to the array size minus one. In this manner, You can reference all the elements in sequence by using the loop counter as the array subscript.

---

6) In what areas do data structures are applied?

Data structures are essential in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

7) What is LIFO?

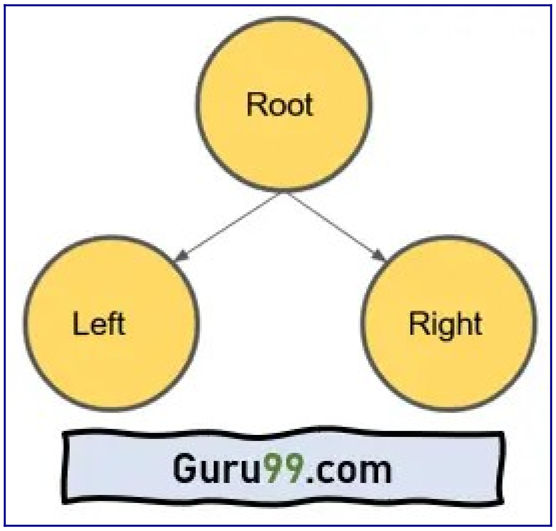
LIFO is a short form of Last In First Out. It refers how data is accessed, stored and retrieved. Using this scheme, data that was stored last should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

8) What is a queue?

A queue is a data structure that can simulate a list or stream of data. In this structure, new elements are inserted at one end, and existing elements are removed from the other end.

9) What are binary trees?

A binary tree is one type of data structure that has two nodes, a left node, and a right node. In programming, binary trees are an extension of the linked list structures.



10) Which data structures are applied when dealing with a recursive function?

Recursion, is a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

11) What is a stack?

A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

12) Explain Binary Search Tree

A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

13) What are multidimensional arrays?

Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structures?

It depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?

Apart from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?

FIFO stands for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?

An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

18) What is merge sort?

Merge sort, is a divide-and-conquer approach for sorting the data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continues until you have one single sorted list.

19) Differentiate NULL and VOID

Null is a value, whereas Void is a data type identifier. A variable that is given a Null value indicates an empty value. The void is used to identify pointers as having no initial size.

20) What is the primary advantage of a linked list?

A linked list is an ideal data structure because it can be modified easily. This means that editing a linked list works regardless of how many elements are in the list.

21) What is the difference between a PUSH and a POP?

Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being “pushed” into the stack. On the other hand, a pop denotes data retrieval, and in particular, refers to the topmost data being accessed.

---

22) What is a linear search?

A linear search refers to the way a target key is being searched in a sequential data structure. In this method, each element in the list is checked and compared against the target key. The process is repeated until found or if the end of the file has been reached.

---

23) How does variable declaration affect memory allocation?

The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

---

24) What is the advantage of the heap over a stack?

The heap is more flexible than the stack. That’s because memory space for the heap can be dynamically allocated and de-allocated as needed. However, the memory of the heap can at times be slower when compared to that stack.

---

25) What is a postfix expression?

A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

---

26) What is Data abstraction?

Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

---

27) How do you insert a new item in a binary search tree?

Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

---

28) How does a selection sort work for an array?

The selection sort is a fairly intuitive sorting algorithm, though not necessarily efficient. In this process, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position.

The smallest element remaining in the subarray is then located next to subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

---

29) How do signed and unsigned numbers affect memory?

In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (an unsigned 8-bit number has a range 0-255, while the 8-bit signed number has a range -128 to +127.

---

30) What is the minimum number of nodes that a binary tree can have?

A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

---

31) What are dynamic data structures?

Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

---

32) In what data structures are pointers applied?

Pointers that are used in linked list have various applications in the data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

---

33) Do all declaration statements result in a fixed reservation in memory?

Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

---

34) What are ARRAYs?

When dealing with arrays, data is stored and retrieved using an index that refers to the element number in the data sequence. This means that data can be accessed in any order. In programming, an array is declared as a variable having a number of indexed elements.

---

35) What is the minimum number of queues needed when implementing a priority queue?

The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is used for actual storage of data.

---

36) Which sorting algorithm is considered the fastest?

There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

---

37) Differentiate STACK from ARRAY.

Stack follows a LIFO pattern. It means that data access follows a sequence wherein the last data to be stored when the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

---

38) Give a basic algorithm for searching a binary search tree.

1.if the tree is empty, then the target is not in the tree, end search
2. if the tree is not empty, the target is in the tree
3. check if the target is in the root item
4. if a target is not in the root item, check if a target is smaller than the root’s value
5. if a target is smaller than the root’s value, search the left subtree
6. else, search the right subtree

---

39) What is a dequeue?

A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

---

40) What is a bubble sort and how do you perform it?

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values “bubble” to the top of the list, while the larger value sinks to the bottom.

---

41) What are the parts of a linked list?

A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes. All these nodes are linked sequentially.

---

42) How does selection sort work?

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

---

43) What is a graph?

A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs and are used to connect nodes where data can be stored and retrieved.

---

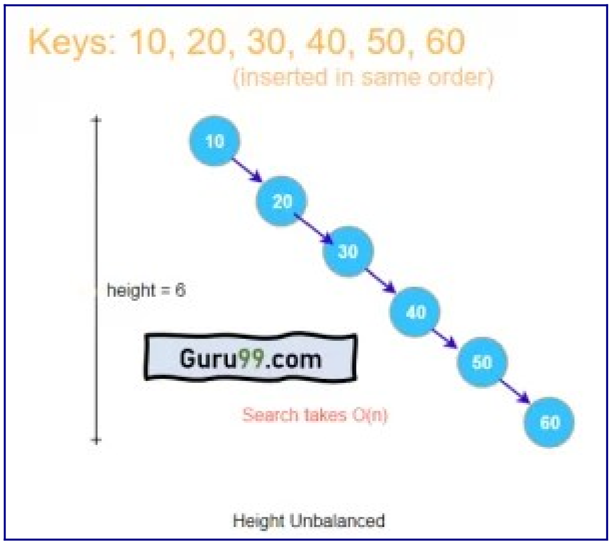
44) Differentiate linear from a nonlinear data structure.

The linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks, and queues. On the other hand, a non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of nonlinear data structure include trees and graphs.

---

45) What is an AVL tree?

An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.



46) What are doubly linked lists?

Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and another one that connects to the previous node.

47) What is Huffman's algorithm?

Huffman's algorithm is used for creating extended binary trees that have minimum weighted path lengths from the given weights. It makes use of a table that contains the frequency of occurrence for each data element.

48) What is Fibonacci search?

Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can significantly reduce the time needed in order to reach the target element.

49) Briefly explain recursive algorithm.

Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

50) How do you search for a target key in a linked list?

To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.

https://lastmomenttuitions.com/engineering-viva-questions/data-structure/

1. What is a Data Structure?

The Data Structure is the way data is organized (stored) and manipulated for retrieval and access. It also defines the way different sets of data relate to one another, establishing relationships and forming algorithms.

2. Describe the types of Data Structures?

The following are the types of data structures:

1. **Lists:** A collection of related things linked to the previous or/and following data items.
2. **Arrays**: A collection of values that are all the same.
3. **Records**: A collection of fields, each of which contains data from a single data type.
4. **Trees**: A data structure that organizes data in a hierarchical framework. This form of data structure follows the ordered order of data item insertion, deletion, and modification.
5. **Tables**: The data is saved in the form of rows and columns. These are comparable to records in that the outcome or alteration of data is mirrored across the whole table.

3. What is a Linear Data Structure? Name a few examples.

A data structure is linear if all its elements or data items are arranged in a sequence or a linear order. The elements are stored in a non-hierarchical way so that each item has successors and predecessors except the first and last element in the list.

Examples of linear data structures are Arrays, Stack, Strings, Queue, and Linked List.

4. What are some applications of Data Structures?

In terms of data structure interview questions, this is one of the most frequently asked question.

Numerical analysis, operating system, AI, compiler design, database management, graphics, statistical analysis, and simulation.

5. What is the difference between file structure and storage structure?

The difference lies in the memory area accessed. Storage structure refers to the data structure in the memory of the computer system, whereas file structure represents the storage structure in the auxiliary memory.

6. What is a multidimensional array?

A multidimensional array is a multidimensional array with more than one dimension. It is an array of arrays or an array with numerous layers. The 2D array, or two-dimensional array, is the most basic multidimensional array. As you'll see in the code, it's technically an array of arrays. A 2D array is also referred to as a matrix or a table with rows and columns. Declaring a multidimensional array is the same as saying a one-dimensional array. We need to notify C that we have two dimensions for a two-dimensional array.

7. How are the elements of a 2D array stored in the memory?

1. **Row-Major Order:** -In row-major ordering, all of the rows of a 2D array are stored in memory in a contiguous manner.

First, the first row of the array is entirely stored in memory, followed by the second row of the array, and so on until the final row.

1. **Column-Major Order**: In column-major ordering, all of the columns of a 2D array are stored in memory in the same order. The first column of the array is entirely saved in memory, followed by the second row of the array, and so on until the last column of the array is wholly recorded in memory.

8. What is a linked list Data Structure?

This is one of the most frequently asked data structure interview questions where the interviewer expects you to give a thorough answer. Try to explain as much as possible rather than finishing your answer in a sentence!

It’s a linear Data Structure or a sequence of data objects where elements are not stored in adjacent memory locations. The elements are linked using pointers to form a chain. Each element is a separate object, called a node. Each node has two items: a data field and a reference to the next node. The entry point in a linked list is called the head. Where the list is empty, the head is a null reference and the last node has a reference to null.

A linked list is a dynamic data structure, where the number of nodes is not fixed, and the list has the ability to grow and shrink on demand.

It is applied in cases where:

- We deal with an unknown number of objects or don’t know how many items are in the list
- We need constant-time insertions/deletions from the list, as in real-time computing where time predictability is critical
- Random access to any elements is not needed
- The algorithm requires a data structure where objects need to be stored irrespective of their physical address in memory
- We need to insert items in the middle of the list as in a priority queue

Some implementations are stacks and queues, graphs, directory of names, dynamic memory allocation, and performing arithmetic operations on long integers.

Full Stack Java Developer Masters Program

- Kickstart Full Stack Java Developer career with industry-aligned curriculum by experts
- Hands-on practice through 20+ projects, assessments, and tests

7 months months

View Program

- **Full Stack Developer - MERN Stack Masters Program**

  - 40+ micro skilling exercises & 6+ work-like professional projects
  - Develop expertise in 10+ full stack development tools and frameworks

  6 Months months

  View Program

Here's what learners are saying regarding our programs:

- **Jonathan Mabiala**

  Java Software Developer, Desjardins

  I chose to upskill after moving from the United States to Canada in 2019. My journey in programming began during engineering, but I lost touch after college. Realizing I needed certification to advance, I enrolled in Simplilearn. The live classes boosted my confidence, allowing me to transition to a Java Software Developer role.

- **Mayur Kharad**

  Product Engineer, IKS Health

  During the lockdown, I realized I needed to upskill myself, and my journey with Simplilearn has been fantastic. I learned many things during the full stack java developer course, thanks to trainer Virendra Sharma. I've always wanted to work in this sector, and after completing my certification in Fullstack Java Development, I got placed at IKS Health through Simplilearn.

Not sure what you’re looking for?View all Related Programs

9. Are linked lists considered linear or non-linear Data Structures?

Linked lists are considered both linear and non-linear data structures depending upon the application they are used for. When used for access strategies, it is considered as a linear data-structure. When used for data storage, it is considered a non-linear data structure.

10. What are the advantages of a linked list over an array? In which scenarios do we use Linked List and when Array?

This is another frequently asked data structure interview questions! Advantages of a linked list over an array are:

1. Insertion and Deletion

Insertion and deletion of nodes is an easier process, as we only update the address present in the next pointer of a node. It's expensive to do the same in an array as the room has to be created for the new elements and existing elements must be shifted.

2. Dynamic Data Structure

As a linked list is a dynamic data structure, there is no need to give an initial size as it can grow and shrink at runtime by allocating and deallocating memory. However, the size is limited in an array as the number of elements is statically stored in the main memory.

3. No Wastage of Memory

As the size of a linked list can increase or decrease depending on the demands of the program, and memory is allocated only when required, there is no memory wasted. In the case of an array, there is memory wastage. For instance, if we declare an array of size 10 and store only five elements in it, then the space for five elements is wasted.

4. Implementation

Data structures like stack and queues are more easily implemented using a linked list than an array.

Some scenarios where we use linked list over array are:

- When we know the upper limit on the number of elements in advance
- When there are a large number of add or remove operations
- When there are no large number of random access to elements
- When we want to insert items in the middle of the list, such as when implementing a priority queue

Some scenarios in which we use array over the linked list are:

- When we need to index or randomly access elements
- When we know the number of elements in the array beforehand, so we can allocate the correct amount of memory
- When we need speed when iterating through all the elements in the sequence
- When memory is a concern; filled arrays use less memory than linked lists, as each element in the array is the data but each linked list node requires the data as well as one or more pointers to the other elements in the linked list

In summary, we consider the requirements of space, time, and ease of implementation to decide whether to use a linked list or array.

11. What is a doubly-linked list? Give some examples.

It is a complex type (double-ended LL) of a linked list in which a node has two links, one that connects to the next node in the sequence and another that connects to the previous node. This allows traversal across the data elements in both directions.

Examples include:

- A music playlist with next and previous navigation buttons
- The browser cache with BACK-FORWARD visited pages
- The undo and redo functionality on a browser, where you can reverse the node to get to the previous page

12. How do you reference all of the elements in a one-dimension array?

Using an indexed loop, we may access all of the elements in a one-dimensional array. The counter counts down from 0 to the maximum array size, n, minus one. The loop counter is used as the array subscript to refer to all items of the one-dimensional array in succession.

13. What are dynamic Data Structures? Name a few.

They are collections of data in memory that expand and contract to grow or shrink in size as a program runs. This enables the programmer to control exactly how much memory is to be utilized.

Examples are the dynamic array, linked list, stack, queue, and heap.

14. What is an algorithm?

An algorithm is a step by step method of solving a problem or manipulating data. It defines a set of instructions to be executed in a certain order to get the desired output.

15. Why do we need to do an algorithm analysis?

A problem can be solved in more than one way using several solution algorithms. Algorithm analysis provides an estimation of the required resources of an algorithm to solve a specific computational problem. The amount of time and space resources required to execute is also determined.

The time complexity of an algorithm quantifies the amount of time taken for an algorithm to run as a function of the length of the input. The space complexity quantifies the amount of space or memory taken by an algorithm, to run as a function of the length of the input.

16. What is a stack?

A stack is an abstract data type that specifies a linear data structure, as in a real physical stack or piles where you can only take the top item off the stack in order to remove things. Thus, insertion (push) and deletion (pop) of items take place only at one end called top of the stack, with a particular order: LIFO (Last In First Out) or FILO (First In Last Out).

17. Where are stacks used?

- Expression, evaluation, or conversion of evaluating prefix, postfix, and infix expressions
- Syntax parsing
- String reversal
- Parenthesis checking
- Backtracking

18. What are the operations that can be performed on a stack?

A stack is a linear data structure that operates on the same concept, in that components in a stack are added and deleted only from one end, referred to as the TOP. As a result, a stack is known as a LIFO (Last-In-First-Out) data structure because the piece that was put last is the first to be removed.

A stack may perform three fundamental operations:

1. **PUSH**: The push action inserts a new element into the stack. The new feature is placed at the top of the stack. However, before inserting the value, we must first verify if TOP=MAX–1, since if so, the stack is filled, and no more insertions are possible. An OVERFLOW message is printed if an attempt is made to put a value into an existing stack.
2. **POP:** The pop operation is performed to remove the stack's topmost element. However, before removing the value, we must first verify if TOP=NULL, since if it is, the stack is empty, and no further deletions are permitted. An UNDERFLOW notice is produced if an attempt is made to erase a value from a stack that is already empty.
3. **PEEK:** A peek action returns the value of the stack's topmost element without removing it from the stack. On the other hand, the Peek operation first checks if the stack is empty, i.e., if TOP = NULL, then an appropriate message is written. Otherwise, a value is returned.

Data Structure Interview Questions for Experienced

19. What is a postfix expression?

A postfix expression is made up of operators and operands, with the operator coming after the operands. That is, in a postfix expression, the operator comes after the operands. Likewise, what is the proper postfix form? The correct postfix phrase is A B + C *.

20. What is a queue Data Structure?

In this type of data structure interview questions, you can also discuss your experience and situations using queue. A queue is an abstract data type that specifies a linear data structure or an ordered list, using the First In First Out (FIFO) operation to access elements. Insert operations can be performed only at one end called REAR and delete operations can be performed only at the other end called FRONT.

21. List some applications of queue Data Structure.

To prioritize jobs as in the following scenarios:

- As waiting lists for a single shared resource in a printer, CPU, call center systems, or image uploads; where the first one entered is the first to be processed
- In the asynchronous transfer of data; or example pipes, file IO, and sockets
- As buffers in applications like MP3 media players and CD players
- To maintain the playlist in media players (to add or remove the songs)

22. What is a Dequeue?

Dequeue is a double-ended queue, or a data structure, where the elements can be inserted or deleted at both ends (FRONT and REAR).

23. What operations can be performed on queues?

- enqueue() adds an element to the end of the queue
- dequeue() removes an element from the front of the queue
- init() is used for initializing the queue
- isEmpty tests for whether or not the queue is empty
- The front is used to get the value of the first data item but does not remove it
- The rear is used to get the last item from a queue

-

Jonathan Mabiala

Java Software Developer, Desjardins

I chose to upskill after moving from the United States to Canada in 2019. My journey in programming began during engineering, but I lost touch after college. Realizing I needed certification to advance, I enrolled in Simplilearn. The live classes boosted my confidence, allowing me to transition to a Java Software Developer role.

-

Mayur Kharad

Product Engineer, IKS Health

During the lockdown, I realized I needed to upskill myself, and my journey with Simplilearn has been fantastic. I learned many things during the full stack java developer course, thanks to trainer Virendra Sharma. I've always wanted to work in this sector, and after completing my certification in Fullstack Java Development, I got placed at IKS Health through Simplilearn.

Not sure what you're looking for?View all Related Programs

24. What are the advantages of the heap over a stack?

In this data structure interview questions, try giving various advantages, along with examples, if possible. It will show the interviewer your domain expertise. Generally, both heap and stack are part of memory and used in Java for different needs:

- Heap is more flexible than the stack because memory space can be dynamically allocated and de-allocated as needed
- Heap memory is used to store objects in Java, whereas stack memory is used to store local variables and function call
- Objects created in the heap are visible to all threads, whereas variables stored in stacks are only visible to the owner as private memory
- When using recursion, the size of heap memory is more whereas it quickly fill-ups stack memory

25. Where can stack Data Structure be used?

- Expression evaluation
- Backtracking
- Memory management
- Function calling and return

26. What is the difference between a PUSH and a POP?

In terms of data structure interview questions, this is one of the most frequently asked question.

The acronyms stand for Pushing and Popping operations performed on a stack. These are ways data is stored and retrieved.

- PUSH is used to add an item to a stack, while POP is used to remove an item.
- PUSH takes two arguments, the name of the stack to add the data to and the value of the entry to be added. POP only needs the name of the stack.
- When the stack is filled and another PUSH command is issued, you get a stack overflow error, which means that the stack can no longer accommodate the last PUSH. In POP, a stack underflow error occurs when you're trying to POP an already empty stack.

27. Which sorting algorithm is considered the fastest? Why?

A single sorting algorithm can't be considered best, as each algorithm is designed for a particular data structure and data set. However, the QuickSort algorithm is generally considered the fastest because it has the best performance for most inputs.

Its advantages over other sorting algorithms include the following:

- Cache-efficient: It linearly scans and linearly partitions the input. This means we can make the most of every cache load.
- Can skip some swaps: As QuickSort is slightly sensitive to input that is in the right order, it can skip some swaps.
- Efficient even in worst-case input sets, as the order is generally random.
- Easy adaption to already- or mostly-sorted inputs.
- When speed takes priority over stability.

28. What is the merge sort? How does it work?

Merge sort is a divide-and-conquer algorithm for sorting the data. It works by merging and sorting adjacent data to create bigger sorted lists, which are then merged recursively to form even bigger sorted lists until you have one single sorted list.

29. How does the Selection sort work?

This is one of the most frequently asked data structure interview questions. Selection sort works by repeatedly picking the smallest number in ascending order from the list and placing it at the beginning. This process is repeated moving toward the end of the list or sorted subarray.

Scan all items and find the smallest. Switch over the position as the first item. Repeat the selection sort on the remaining N-1 items. We always iterate forward (i from 0 to N-1) and swap with the smallest element (always i).

Time complexity: best case O(n2); worst O(n2)

Space complexity: worst O(1)

30. What is an asymptotic analysis of an algorithm?

Asymptotic analysis is the technique of determining an algorithm's running time in mathematical units to determine the program's limits, also known as "run-time performance." The purpose is to identify the best case, worst case, and average-case times for completing a particular activity. While not a deep learning training technique, Asymptotic analysis is an essential diagnostic tool for programmers to analyze an algorithm's efficiency rather than its correctness.

31. What are asymptotic notations?

Asymptotic Notation represents an algorithm's running time - how long an algorithm takes with a given input, n. Big O, large Theta (), and big Omega () are the three distinct notations. When the running time is the same in all circumstances, big- is used, big-O for the worst-case running time, and big- for the best case running time.

32. What are some examples of divide and conquer algorithms?

Quicksort is the name of a sorting algorithm. The method selects a pivot element and rearranges the array elements so that all items less than the pivot chosen element go to the left side of the pivot and all elements more significant than the pivot element move to the right side.

Merge Sort is a sorting algorithm as well. The algorithm divides the array into two halves, sorts them recursively, and then combines the two sorted halves. The goal of points that are closest together is to identify the nearest pair of points in an x-y plane collection of points. The issue may be solved in O(n2) time by computing the distances between each pair of locations and comparing them to determine the shortest distance.

33. Define the graph Data Structure?

Graph Data structure is a type of non-linear data structure that consists of vertices or nodes connected by edges or arcs to enable storage or retrieval of data. Edges may be directed or undirected.

34. What are the applications of graph Data Structure?

- Transport grids where stations are represented as vertices and routes as the edges of the graph
- Utility graphs of power or water, where vertices are connection points and edge the wires or pipes connecting them
- Social network graphs to determine the flow of information and hotspots (edges and vertices)
- Neural networks where vertices represent neurons and edge the synapses between them

35. List the types of trees?

Data structure interview questions like this are very common and frequently asked

- The General Tree

A tree is referred to as a generic tree if its hierarchy is not constrained. In the General Tree, each node can have an endless number of offspring, and all other trees are subsets of the tree.

- The Binary Tree

The binary tree is a type of tree in which each parent has at least two offspring. The children are referred to as the left and right youngsters. This tree is more popular than most others. When specific limitations and features are given to a Binary tree, various trees such as AVL tree, BST (Binary Search Tree), RBT tree, and so on are also utilized.

- Tree of Binary Search

Binary Search Tree (BST) is a binary tree extension that includes numerous optional constraints. In BST, a node's left child value should be less than or equal to the parent value, while the correct child value should always be higher than or equal to the parent's value.

- The AVL Tree

The AVL tree is a self-balancing binary search tree. The term AVL is given in honor of the inventors Adelson-Velshi and Landis. This was the first tree to achieve dynamic equilibrium. Each node in the AVL tree is assigned a balancing factor based on whether the tree is balanced or not. The node kids have a maximum height of one AVL vine.

- Red and Black Tree

Red-black trees are another type of auto-balancing tree. The red-black term is derived from the qualities of the red-black tree, which has either red or black painted on each node. It helps to keep the forest in balance. Even though this tree is not perfectly balanced, the searching process takes just O (log n) time.

- The N-ary Tree

In this sort of tree with a node, N is the maximum number of children. A binary tree is a two-year tree since each binary tree node has no more than two offspring. A full N-ary tree is one in which the children of each node are either 0 or N.

36. What are Binary trees?

A binary tree is a tree data structure made up of nodes, each of which has two offspring, known as the left and right nodes. The tree begins with a single node called the root.

Each node in the tree carries the following information:

Data

A pointing device indicates the left kid.

An arrow pointing to the correct child

37. What are the differences between the B tree and the B+ tree?

The B tree is a self-balancing m-way tree, with m defining the tree's order. Depending on the number of m, Btree is an extension of the Binary Search tree in which a node can have more than one key and more than two children. The data is provided in the B tree in a sorted manner, with lower values on the left subtree and higher values on the right subtree.

The B+ tree is an advanced self-balanced tree since every path from the tree's root to its leaf is the same length. The fact that all leaf nodes are the same length indicates that they all occur at the same level. Specific leaf nodes can't appear at the third level, while others appear at the second level.

38. What are the advantages of binary search over a linear search?

In a sorted list:

- A binary search is more efficient than a linear search because we perform fewer comparisons. With linear search, we can only eliminate one element per comparison each time we fail to find the value we are looking for, but with the binary search, we eliminate half the set with each comparison.
- Binary search runs in O(log n) time compared to linear search's O(n) time. This means that the more of the elements present in the search array, the faster is binary search compared to a linear search.

39. What is an AVL tree?

An AVL (Adelson, Velskii, and Landi) tree is a height balancing binary search tree in which the difference of heights of the left and right subtrees of any node is less than or equal to one. This controls the height of the binary search tree by not letting it get skewed. This is used when working with a large data set, with continual pruning through insertion and deletion of data.

40. Differentiate NULL and VOID

- Null is a value, whereas Void is a data type identifier
- Null indicates an empty value for a variable, whereas void indicates pointers that have no initial size
- Null means it never existed; Void means it existed but is not in effect

41. Do dynamic memory allocations help in managing data? How?

Dynamic memory allocation stores simple structured data types at runtime. It has the ability to combine separately allocated structured blocks to form composite structures that expand and contract as needed, thus helping manage data of data blocks of arbitrary size, in arbitrary order.

42. Name the ways to determine whether a linked list has a loop.

- Using hashing
- Using the visited nodes method (with or without modifying the basic linked list data structure)
- Floyd's cycle-finding algorithm

43. List some applications of multilinked structures?

- Sparse matrix
- Index generation

44. Explain the jagged array.

It is an array whose elements themselves are arrays and may be of different dimensions and sizes.

45. Explain the max heap Data Structure.

It is a type of heap data structure where the value of the root node is greater than or equal to either of its child nodes.

46. How do you find the height of a node in a tree?

The height of the node equals the number of edges in the longest path to the leaf from the node, where the depth of a leaf node is 0

Question 1: What is an array?

Answer: An array is a data structure consisting of a collection of elements, each identified by at least one array index or key.

Question 2: Can an array be resized at runtime?

Answer: In some programming languages, arrays can be resized dynamically, while in others, such as C, the size is fixed.

Question 3: What is the time complexity for accessing an element in an array?

Answer: The time complexity for accessing an element in an array is **O(1),** as it can be accessed directly using its index.

Question 4: What is the difference between an array and a linked list?

Answer: An array is a static data structure, while a linked list is a dynamic data structure. Arrays have a fixed size, and elements are stored consecutively in memory, while linked lists can grow and do not require contiguous memory allocation.

Question 5: How would you find the smallest and largest element in an array?

Answer: To find the smallest and largest elements in an array, one common approach is to iterate through the array and keep track of the smallest and largest elements encountered so far.

Question 6: Explain the concept of a multi-dimensional array.

Answer: A multi-dimensional array is an array that contains other arrays. For example, a 2D array is an array of arrays, representing a matrix.

Question 7: What is an array index out of bounds exception?

Answer: Answer: This error occurs when an attempt is made to access an element at an index that is outside the bounds of the array (e.g., negative index or greater than the array size).

Question 8: How would you reverse an array in-place in linear time and constant space?

Answer: One approach is to use two pointers starting from the beginning and end of the array and swap the elements until they meet in the middle.

Question 9: Explain the concept of a jagged array.

Answer: A jagged array is an array of arrays, where each sub-array could be of a different length.

Question 10: How can you find duplicate elements in an array?

Answer: One way to find duplicate elements in an array is to use a hash set or to sort the array and then iterate through it to find consecutive duplicates.

Question 11: Discuss the advantages and disadvantages of using arrays.

Answer:

- **Advantages:** Constant time access, simple implementation, and efficient storage for contiguous data.
- **Disadvantages:** Fixed size, no support for dynamic growth, inefficient for insertions and deletions.

Question 12: Explain the concept of a sparse array.

Answer: A sparse array is an array in which most of the elements have the same value. It can be represented using a data structure that only stores the non-default (non-zero) values.

Question 13: What is the difference between an array and a list?

Answer: An array is a static data structure with a fixed size, while a list is a dynamic data structure that can grow and shrink during runtime.

Commonly Asked Data Structure Interview Questions on Linked List

Question 1: What is a linked list?

Answer: A linked list is a linear data structure consisting of a sequence of elements, where each element points to the next one, forming a chain.

Question 2: What are the different types of linked lists?

Answer: Singly linked list, doubly linked list, and circular linked list.

Question 3: What are the advantage of Linked List?

Answer: Advantages of Linked Lists:

- Dynamic memory allocation
- Efficient insertion and deletion
- Can represent complex data structures
- Can be used to implement queues and stacks
- Can be used for memory management and caching
- Can be used for garbage collection

Question 4: What are the disadvantage of Linked List?

Answer: Disadvantages of Linked Lists:

- Slow random access
- More memory overhead
- Difficult to debug
- Not cache-friendly
- Can suffer from memory leaks

Question 5: What is a cycle/loop in Singly Linked List:

Answer: A cycle, also known as a loop, in a singly-linked list occurs when a node in the list points back to a previous node, creating a circular path. This means that if you start traversing the list from any node, you will eventually come back to the same node, forming an infinite loop.

Question 6: What is time complexity of Linked List operations?

Answer: The time complexity of common operations on a singly-linked list are as follows:

Insertion:

- At the beginning: O(1)
- At the end: O(n)
- At a specific position: O(n)

Deletion:

- At the beginning: O(1)
- At the end: O(n)
- At a specific position: O(n)

Search: O(n)
Traversal: O(n)

Question 7: How would you compare Dynamic Arrays Vs Linked Lists?

Answer: Dynamic Array Advantages:

- Fast random access (O(1))
- Efficient for large data sets
- Contiguous memory allocation

Dynamic Array Disadvantages:

- Slow insertion and deletion in the middle (O(n))
- Fixed size, can lead to memory waste or reallocation

Linked Lists Advantages:

- Efficient insertion and deletion in the middle (O(1))

- Can grow and shrink dynamically
- Can represent complex data structures

Linked Lists Disadvantages:

- Slow random access (O(n))
- More memory overhead due to pointers
- Not cache-friendly

Dynamic arrays are more efficient for random access and large data sets, while linked lists are more efficient for operations that involve insertion and deletion in the middle. Linked lists are also more flexible and can represent complex data structures.

Commonly Asked Data Structure Interview Questions on Stack:

Question 1: What is a stack?

Answer: A stack is a linear data structure that follows the **Last-In-First-Out (LIFO)** principle.

Question 2: What are the operations performed on a stack?

Answer: The common operations on a stack are push (insert an element), pop (remove the top element), and peek (view the top element).

Question 3: How is a stack implemented in an array?

Answer: A stack can be implemented using an array by maintaining a pointer to the top of the stack.

Question 4: What is the time complexity of stack operations?

Answer: Push, pop, and peek operations have a time complexity of O(1).

Question 5: What are the applications of a stack?

Answer: Stacks are used in various applications, such as function calls, recursion, expression evaluation, and parsing.

Question 6: What is a stack overflow?

Answer: A stack overflow occurs when the stack exceeds its allocated memory.

Question 7: What is a stack underflow?

Answer: A stack underflow occurs when the stack is empty and an attempt is made to pop an element.

Question 8: What is a postfix expression?

Answer: A postfix expression is an expression where the operator follows the operands.

Question 9: How can a stack be used to evaluate a postfix expression?

Answer: By pushing operands onto the stack and performing operations when operators are encountered.

Question 10: What is a prefix expression?

Answer: A prefix expression is an expression where the operator precedes the operands.

Question 11: How can a stack be used to evaluate a prefix expression?

Answer: By pushing operators onto the stack and performing operations when operands are encountered.

Question 12: How can a stack be used to check if a parenthesis expression is balanced?

Answer: A stack can be used to check if a parenthesis expression is balanced by following these steps:

- Push the opening parenthesis onto the stack.
- When an closing parenthesis is encountered, pop the top element from the stack and check if it matches the closing parenthesis.
- If the stack is empty at the end of the expression, then the expression is balanced.
- If the stack is not empty, then the expression is not balanced.

Commonly Asked Data Structure Interview Questions on Queue

Question 1: What is a Queue?

Answer: A queue is a linear data structure that follows the **First-In-First-Out (FIFO)** principle, where elements are added at the rear (enqueue) and removed from the front (dequeue).

Question 2: What are the different types of Queues?

Answer:

- Simple Queue
- Circular Queue
- Priority Queue
- Double-Ended Queue (Deque)

Question 3: How is a Queue implemented in an array?

Answer: An array can be used to implement a simple queue by maintaining two pointers: front and rear. Front points to the first element, and rear points to the next available position.

Question 4: How is a Queue implemented in a linked list?

Answer: A linked list can be used to implement a queue by creating a node for each element and maintaining a head and tail pointer. Enqueueing adds a node to the tail, and dequeueing removes a node from the head.

Question 5: What is the time complexity of enqueue and dequeue operations in a Queue?

Answer:

- **Enqueue:** O(1)
- **Dequeue:** O(1) for simple and circular queues, **O(n)** for priority queues

Question 6: What is the difference between a Queue and a Stack?

Answer: A queue follows **FIFO**, while a stack follows **Last-In-First-Out (LIFO)**.

Question 7: What are the applications of Queues?

Answer:

- Task scheduling
- Message passing
- Simulation of real-world scenarios

Question 8: How do you handle overflow and underflow conditions in a Queue?

Answer:

- **Overflow:** When the queue is full, throw an exception or return an error code.
- **Underflow:** When the queue is empty, throw an exception or return a null value.

Question 9: What is a circular queue?

Answer: A circular queue is a variation of a simple queue where the rear pointer wraps around to the beginning of the array after reaching the end.

Question 10: What is a priority queue?

Answer: A priority queue is a queue where elements are assigned priorities and are dequeued based on their priorities.

Question 11: How is a priority queue implemented?

Answer: Priority queues can be implemented using a binary heap or a self-balancing binary search tree.

Question 12: What is a double-ended queue (Deque)?

Answer: A deque is a queue that allows insertions and deletions from both ends.

Question 13: How is a deque implemented?

Answer: A deque can be implemented using two stacks or a circular buffer.

Question 14: What are the advantages of using a Queue?

Answer:

- Simple and efficient FIFO implementation
- Easy to enqueue and dequeue elements
- Supports multiple producers and consumers

Question 15: What are the disadvantages of using a Queue?

Answer:

- Limited access to elements (only from the front or rear)
- Can be inefficient if elements need to be accessed in a non-sequential order

Commonly Asked Data Structure Interview Questions on Heap Data Structure

Question 1: What is a heap data structure?

Answer: A heap is a complete binary tree that satisfies the heap property: each node's value is greater than or equal to its children's values.

Question 2: What are the two types of heaps?

Answer: Max-heap and min-heap. In a max-heap, the root node has the maximum value, while in a min-heap, the root node has the minimum value.

Question 3: What is the time complexity of inserting an element into a heap?

Answer: O(log n), where n is the number of elements in the heap.

Question 4: What is the time complexity of deleting an element from a heap?

Answer: O(log n), where n is the number of elements in the heap.

Question 5: What is the time complexity of finding the minimum or maximum element in a heap?

Answer: O(1), as the root node always contains the minimum or maximum element.

Question 6: What are the applications of heaps?

Answer: Heap applications:

- Priority queues
- Sorting
- Finding the median
- Implementing Dijkstra's algorithm
- Network routing
- Huffman coding

Question 7: What is the difference between a heap and a binary search tree (BST)?

Answer: A heap is a complete binary tree that satisfies the heap property, while a BST is a partially ordered binary tree that satisfies the BST property.

Question 8: How do you convert a BST into a heap?

Answer: By performing an in-order traversal of the BST and inserting the elements into a heap.

Question 9: How do you merge two heaps?

Answer: By creating a new heap and inserting the elements from both heaps into the new heap while maintaining the heap property.

Question 10: What is the difference between a heap and a priority queue?

Answer: A heap is a data structure, while a priority queue is an abstract data type that can be implemented using a heap.

Question 11: What are the advantages of using a heap?

Answer: Advantages of using a heap:

- Efficient insertion and extraction (O(log n))
- Can be used to implement priority queues
- Can be used for sorting (O(n log n))
- Useful for other applications, such as finding the median and implementing Dijkstra's algorithm

Commonly Asked Data Structure Interview Questions on Hash Data Structure

Question 1: What is a hash data structure?

Answer: A hash data structure is a data structure that stores key-value pairs, where the keys are hashed to determine the location of the value in the data structure.

Question 2: What is a hash table?

Answer: A hash table is a data structure that implements an associative array, allowing fast retrieval of values based on unique keys. It uses a hash function to map keys to indices in an array, providing constant-time average access (O(1)) if collisions are minimized.

Question 3: What is a hash function?

Answer: A hash function is a function that takes an input of any size and produces an output of a fixed size. The output is called a **hash value or hash code**.

Question 4: Explain how a hash function works.

Answer: A hash function takes an input key and maps it to a fixed-size index (hash value) within the hash table's array. Ideally, the function distributes keys evenly across the array to minimize collisions. Common hash functions include **modulo division, bitwise operations,** and **polynomial hashing**.

Question 5: What is a collision?

Answer: A collision occurs when two different keys hash to the same value.

Question 6: Describe different collision resolution techniques.

Answer:

- **Open addressing:** Use probing techniques (linear, quadratic, double hashing) to find the next available slot when a collision occurs.
- **Separate chaining:** Store key-value pairs in linked lists at each index, leading to better performance for larger data sets.

Question 7: How are collisions handled in a hash data structure?

Answer: Collisions can be handled using various techniques, such as **chaining, open addressing,** and **cuckoo hashing**.

Question 8: What is chaining?

Answer: Chaining is a collision resolution technique where colliding keys are stored in a linked list at the same hash value.

Question 9: What is open addressing?

Answer: Open addressing is a collision resolution technique where colliding keys are stored in the same hash table, but at different locations.

Question 10: What is separate chaining?

Answer: Separate chaining is a collision resolution technique used in hash tables. When two or more keys hash to the same index in the hash table, instead of overwriting the existing data, separate chaining stores each key-value pair in a linked list at that index. This allows for efficient retrieval of all elements that hash to the same index, even if there are collisions.

Question 11: What are the trade-offs between open addressing and separate chaining?

Answer:

- Open addressing: Less memory overhead, but search performance degrades with collisions.
- Separate chaining: More memory usage, but faster search performance even with collisions.

Question 12: What is cuckoo hashing?

Answer: Cuckoo hashing is a collision resolution technique that uses two hash functions to store keys in a hash table.

Question 13: What is the load factor of a hash table?

Answer: The load factor of a hash table is the ratio of the number of keys stored in the table to the size of the table.

Question 14: What is the optimal load factor for a hash table?

Answer: The optimal load factor for a hash table depends on the collision resolution technique used, but it is typically around 0.7.

Question 15: Explain the concept of load factor and its impact on performance.

Answer: Load factor (number of elements / size of hash table) measures how full the table is. Higher load factors increase collision frequency and impact performance. Optimal values vary based on implementation and trade-offs.

Question 16: What are the advantages of using a hash data structure?

Answer: Hash data structures offer fast lookup, insertion, and deletion operations. They are also space-efficient and can handle large datasets.

Question 17: What are the disadvantages of using a hash data structure?

Answer: Hash data structures can suffer from collisions, which can slow down lookup operations. They also require a hash function that is both efficient and effective.

Question 18: Explain bloom filters and their applications.

Answer: Bloom filters use multiple hash functions to probabilistically represent set membership, offering efficient space-time trade-offs for membership queries but not supporting direct value retrieval. Used in caching, network security, and other applications.

Commonly Asked Data Structure Interview Questions on Tree Data Structures:

Question 1: What is a Tree?

Answer: A tree is a non-linear data structure consisting of nodes connected by edges. Each node contains data and references to its child nodes. It has one special node called the root, with no parent, and leaf nodes with no children.

Question 2: Explain different types of trees.

Answer:

- Binary Tree: Each node has at most two children (left and right).
- Full Binary Tree: Every node except leaves has two children.
- Complete Binary Tree: All levels are filled except possibly the last, and nodes are filled left to right.
- Perfect Binary Tree: Every node has two children, and all leaves are at the same level.
- AVL Tree: Self-balancing binary search tree with a height difference of at most 1 between subtrees.
- Red-Black Tree: Self-balancing binary search tree with specific coloring rules to maintain balance.
- B-Tree: Generalization of a binary search tree with more than two children per node.

Question 3: What are the basic operations performed on a tree?

Answer:

- **Insertion:** Add a new node to the tree while maintaining its properties (e.g., ordering in search trees).
- **Deletion:** Remove a node from the tree while preserving its structure.
- **Traversal:** Visit each node in the tree exactly once in a specific order (preorder, inorder, postorder).
- **Searching:** Find a specific node with a given value based on search criteria.

Question 4: What are the different ways to represent a tree in memory?

Answer:

- **Node-based representation:** Each node stores its data and references to child nodes.
- **Array-based representation:** Use an array to store node data with calculations to find child nodes based on their positions.

Question 5: What are the advantages and disadvantages of using trees?

Advantages:

- Efficient for hierarchical data representation and organization.

- Fast searching and traversal in balanced trees.

Disadvantages:

- Memory overhead due to storing pointers or references.
- Not efficient for storing large amounts of unstructured data.

Question 6: When would you choose a tree over other data structures like arrays or linked lists?

Answer: Trees are ideal for hierarchical data, maintaining relationships between elements, and efficient searching based on order. Arrays or linked lists are better for simple linear data or frequent insertions/deletions at specific positions.

Question 7: Explain the concept of a binary search tree.

Answer: A binary search tree has a specific ordering property: the data in the left subtree is less than the root, and the data in the right subtree is greater than the root. This allows for efficient searching by comparing values with the root and navigating left or right accordingly.

Question 8: How do self-balancing trees like AVL or Red-Black trees work?

Answer: These trees automatically adjust their structure after insertions or deletions to maintain a balanced height, ensuring efficient search and insertion/deletion operations. They achieve this through specific rules and rotations based on node heights and colors.

Question 9: Describe the different tree traversal methods (preorder, inorder, postorder).

Answer:

- Preorder: Visit root, then left subtree, then right subtree.
- Inorder: Visit left subtree, then root, then right subtree.
- Postorder: Visit left subtree, then right subtree, then root.

Each traversal method has different purposes. Inorder is useful for printing sorted elements in a binary search tree, while preorder might be used for copying tree structure.

Question 10: How can you convert a binary search tree into a sorted array?

Answer: One efficient way is to use an inorder traversal of the tree. Since the tree is sorted, visiting nodes in this order will result in a sorted array.

Question 11: Explain the concept of a minimum spanning tree.

Answer: A minimum spanning tree is a subgraph of a connected, undirected graph that includes all vertices but with the minimum total edge weight, connecting all nodes without cycles. It has applications in network routing and clustering algorithms.

Question 12: Describe the use of trees in real-world scenarios.

Answer: Trees are used in various domains, including:

- File systems (directory structure)
- XML or JSON data representation
- Decision trees for machine learning
- Game AI (representing game states and possible actions)
- Social networks (representing user relationships

Commonly Asked Data Structure Interview Questions on Graph:

Question 1: What is a graph?

Answer: A graph is a data structure consisting of a set of vertices (nodes) and a set of edges that connect pairs of vertices. Graphs are used to represent relationships between objects, such as social networks, road networks, and computer networks.

Question 2: Explain common graph representations.

Answer:

- **Adjacency matrix:** A 2D array where rows and columns represent nodes, and values indicate the existence of an edge between them. Efficient for space usage, but can be slow for sparse graphs.
- **Adjacency list:** An array of linked lists or other data structures, where each list stores nodes connected to a specific node. Efficient for sparse graphs and adjacency queries, but may require more space.

Question 3: Differentiate between directed and undirected graphs.

Answer:

- **Directed graphs:** Edges have a direction, signifying one-way relationships.
- **Undirected graphs:** Edges have no direction, representing bidirectional relationships.

Question 4: Describe common graph types.

Answer:

- Simple graphs: Undirected, no loops or multiple edges between nodes.
- Complete graphs: Every node is connected to every other node.
- Trees: No cycles, a single root node connects to child nodes that don’t form cycles.
- Bipartite graphs: Can be divided into two disjoint sets where only nodes in different sets connect.

Question 5: Discuss time and space complexity of basic graph operations.

Answer:

- **Traversal (DFS, BFS):** O(V + E) for both time and space, where V is the number of nodes and E is the number of edges.
- **Insertion:** O(1) for constant-time operations in both adjacency list and matrix, although insertion into sparse matrices requiring reallocation might have amortized cost.
- **Deletion**: O(degree of the node) for adjacency lists, O(V^2) for adjacency matrices due to potential row/column shifts.

Question 6: Explain Dijkstra's algorithm and its applications.

Answer: This algorithm finds the shortest path between two nodes in a weighted graph. It is used in route planning, network optimization, and other problems involving finding minimum-cost paths.

Question 7: Compare DFS and BFS algorithms: strengths, weaknesses, and use cases.

Answer:

- **DFS:** Explores deeply before exploring breadth-wise, efficient for finding connected components, good for detecting cycles.
- **BFS:** Explores breadth-wise, efficient for finding shortest paths in unweighted graphs, useful for level-order traversals.

Question 8: Describe topological sorting: algorithm and applications.

Answer:

- **Topological sorting:** Orders nodes such that edges always point from earlier nodes to later ones, required for tasks with dependencies.
- **Application:** Used in job scheduling, software dependency management, and circuit design.

Question 9: Explain minimum spanning trees: algorithms and their significance.

Answer:

- **Minimum spanning trees:** Finds a subset of edges that connects all nodes with minimum total weight while avoiding cycles.
- **Significance:** Used in network communication, clustering.