b) Insertion and Deletion

Node insertion and deletion are easier as you just have to update the address present in the next pointer.

However, it is expensive in an array because you need to create room for new elements by shifting existing elements.

c) Implementation

Implementing stack and queue data structures using a linked list is easier.

With an array, implementing data structures is complex.

d) Memory Usage

A linked list can be decreased or increased based on the demand of the program, and memory is allocated only when needed, so there is no memory wastage.

An array cause more memory wastage. For example, if we store five elements in an array when the size is 10, then the remaining space will be wasted.

A linked list is used when:

- A large number of operations are added or removed.
- You want to insert an item in the middle of a list, like implementing a priority queue.
- You already know the upper limit of a number of elements.

Some common cases when an array is preferred over a linked list:

- You need to randomly access or index data elements.
- You need to iterate through all elements quickly.
- You already know the number of elements in an array, so you can allocate the correct amount of memory.
- In case memory is a major concern, arrays use less memory than linked lists because each element in an array is data, whereas each node in a linked list requires data and one or more pointers to other elements.

In a nutshell, it comes to time, space, and ease of implementation while deciding between a linked list and an array.

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

The doubly-linked list is a double-ended complex linked list with two links in a node. One of them connects to the next adjacent node in the sequence, while the second one connects to the previous node. This enables traversal in both directions. Some of the examples of a doubly-linked list are:

- Browser cache with back-forward visited pages

- Song playlist with next and previous buttons
- Undo and redo functionality
- **13. What is FIFO?**

FIFO stands for First in, First out order, representing the way and order in which data is accessed. The data element stored first in the list is the first entity to be removed from the data structure.

14. What is the working of LIFO?

LIFO stands for Last in, First out order and directly corresponds to how data is accessed and modified. The data element stored last in the stack is the first one to be worked on at any point in time. If the first one doesn't need to be accessed, you need to retrieve the data after that element.

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

a) Row-Major Order

Rows of a 2D array are stored contiguously. The first row is stored in the memory, then the second row of the array, then the third, and so on until the final row.

Data Structures and Algorithms (DSA) Interview Questions

16. What is an algorithm?

An algorithm shows a step-by-step process to solve a problem or manipulate data. It is a set of instructions to be executed in a specific order to get the desired results.

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

A problem can be resolved in multiple ways with different algorithms. Using algorithm analysis, we can estimate the required resources to solve a particular computation problem. We can determine the amount of time and space resources needed.

The time complexity determines the amount of time required for an algorithm to run as a function of input length, and the space complexity determines the amount of space or memory consumed by an algorithm.

18. What are different sorting algorithms? Which is the fastest of them?

Different sorting algorithms are:

- Quicksort
- Bubble sort
- Radix sort

- Balloon sort
- Merge sort

Among all the options available, putting one on the podium for enhanced and fastest performance is neither fair nor possible. Each sorting algorithm serves a specific purpose, and its performance varies according to the data, the way it's stored, and the reaction once the algorithm processes the data.

19. What is an asymptotic analysis of an algorithm?

Asymptotic analysis of an algorithm is a technique to determine its run-time performance according to the mathematical units.

It is used to find the best case (Omega Notation, Ω), worst case (Big Oh Notation, O), and average case (Theta Notation, θ) performances of an algorithm. It is not a deep learning method but an essential tool for programmers to analyse the efficiency of an algorithm.

20. What is a queue data structure?

A queue data structure supports systematic operations, i.e., elements are accessed and manipulated in a specific order. It follows FIFO (First in, First out) method, so elements are added to the queue one after the other from the rear end and are removed or processed on the front end.

It is similar to the literal queues in the real world. Unlike stack, it is open at both ends, with one end being used to insert elements and another one to remove them. Queues are commonly used when users want to store items for a long period.

21. What is a priority queue? What are the applications for the priority queue?

A priority queue is a type of abstract data resembling a normal queue, but here, each element is assigned a priority value. The order of elements in the priority queue determines their priority.

So, the elements with higher priority are processed first. In case two or more elements have the same priority, they are processed in the order they appear in the queue.

Some real-life applications of the priority queue are:

- Huffman code for data compression
- Graph algorithms like Prim's Minimum spanning tree
- Robotics to plan and execute tasks based on priorities.

Be prepared for this type of interview questions in data structures, as these are often asked to both freshers and experienced professionals.

22. List some applications of queue Data Structure.

A queue is used to manage a group of elements in FIFO (First in, First out) order. A few of its applications are:

- Call management in call centres
- Buffers in MP3 players and CD players
- Handling interruptions in real-time systems
- In graphs for a breadth-first search algorithm
- Waiting lists or serving requests for a single shared resource like CPU, image uploads, printer, etc.
- Operating system: CPU scheduling, job scheduling operations, and Disk scheduling.
- Manage a playlist in media players, for example adding or removing a song
- Asynchronous transfer of data

23. What operations can be performed on queues?

Different operations that can be performed on queues are:

- **dequeue()** removes an element from the queue. If the queue is empty, you get UNDERFLOW notification.
- **enqueue()** adds an element to the queue. If the queue is full, it will be an overflow condition.
- **isEmpty** is to check if the queue is empty or not.
- **init()** initialises the queue.
- **front** is used to find the value of the first element without removing it.
- **rear** returns the last or rear element from the queue.

24. What is a Deque?

A deque is a double-ended queue. It is like an array of items, but rather than adding or removing elements from only one end, a deque allows elements to be inserted at either end.

This feature makes deque suitable for various tasks, such as scheduling, keeping track of inventory, and handling a large amount of data. Moreover, it can be used as a stack and queue and performs better than stacks and linked lists.

25. What are the different types of deque? What are its applications?

There are two main types of deque:

a) Input Restricted Deque

Here, insertion is performed at one end, but deletion is performed at both ends.

b) Output Restricted Deque

Insertion is performed at both ends while deletion is performed at one end.

Some real-life applications of a deque data structure are:

- To store web browser history
- As it supports all data structure operations, it can be used as a stack and queue.
- Operating system job scheduling algorithm

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

Some major advantages of heap over stack are as follows:

- A heap is more flexible than a stack.
- With a heap, it is possible to allocate or de-allocate memory space dynamically.
- The heap is used to store data elements in Java, while the stack is used for variables.
- When you use recursion, heap memory size is more, but in the case of stack memory, it fills up quickly.
- Objects in a heap are visible to all threads, whereas variables are stored as private memory in the stack and are visible to only the owner.

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

Push and pop are the two key data structure activities that stand for Pushing and Popping. These operations signify the way data is stored and retrieved in a stack.

Push- It denotes that a user is adding an element to the stack.

- It requires the name of the stack and the value of the entity.
- In case the stack is filled but you still use the ‘Push’ command, you’ll get an ‘overflow’ error, indicating that the stack can no longer accommodate another element.

Pop- It is an activity to retrieve or pull elements from the stack.

- It only needs the name of the stack.
- When you try to pull an element from an empty stack, you see an ‘underflow’ error.

When we talk about adding or removing an element, it refers to the top-most available data.

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

Merge sort is based on the divide and conquer method. Here, the data is divided and sorted to attain the end goal. The adjacent data elements are merged and sorted in