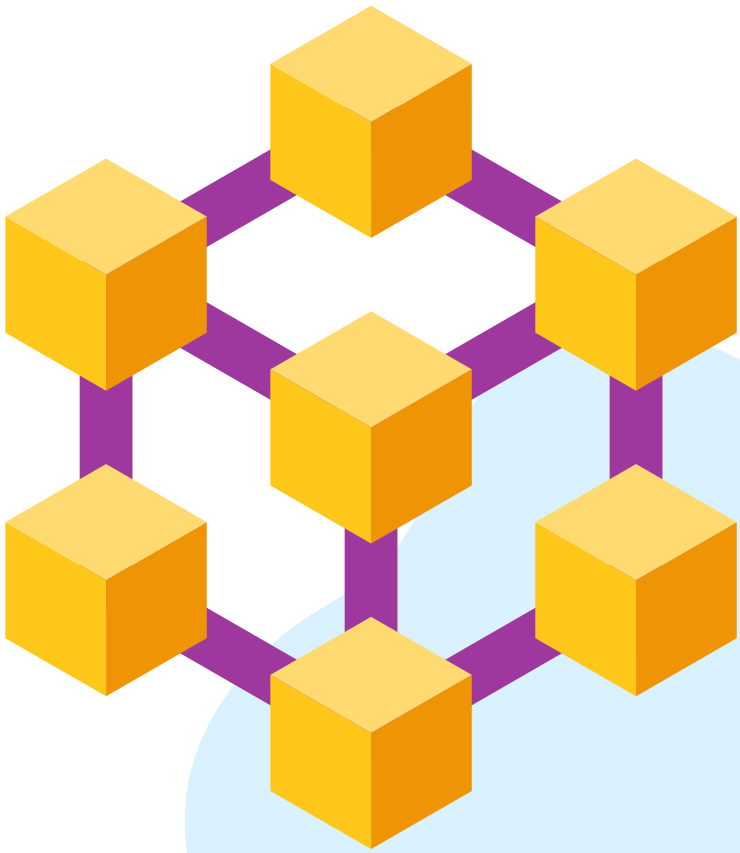
TalentBattle

Frequently Asked DSA Interview Questions with answers



Basic Level 

1. What are Data Structures?

Answer:

A data structure is a mechanical or logical way that data is organized within a program. The organization of data is what determines how a program performs. There are many types of data structures, each with its own uses. When designing code, we need to pay particular attention to the way data is structured. If data isn't stored efficiently or correctly structured, then the overall performance of the code will be reduced.

2. Describe the types of Data Structures?

Answer:

Data Structures are mainly classified into two types:

Linear Data Structure: A data structure is called linear if all of its elements are arranged in the sequential order. In linear data structures, the elements are stored in a non-hierarchical way where each item has the successors and predecessors except the first and last element.

Non-Linear Data Structure: The Non-linear data structure does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in the sequential structure.

3. Explain the process behind storing a variable in memory.

Answer:

- ·A variable is stored in memory based on the amount of memory that is needed. Following are the steps followed to store a variable:

i.The required amount of memory is assigned first.
ii.Then, it is stored based on the data structure being used.

- ·Using concepts like dynamic allocation ensures high efficiency and that the storage units can be accessed based on requirements in real-time.

4. What is a stack data structure? What are the applications of stack?

Answer:

A stack is a data structure that is used to represent the state of an application at a particular point in time. The stack consists of a series of items that are added to the top of the stack and then removed from the top. It is a linear data structure that follows a particular order in which operations are performed. LIFO (Last In First Out) or FILO (First In Last Out) are two possible orders. A stack consists of a sequence of items. The element that's added last will come out first, a real-life example might be a stack of clothes on top of each other. When we remove the cloth that was previously on top, we can say that the cloth that was added last comes out first.

Following are some applications for stack data structure:

- It acts as temporary storage during recursive operations
- Redo and Undo operations in doc editors
- Reversing a string
- Parenthesis matching
- Postfix to Infix Expressions
- Function calls order

5. Which data structure is used to perform recursion?

Answer:

Stack data structure is used in recursion due to its last in first out nature. Operating system maintains the stack in order to save the iteration variables at each function call.

6. List the area of applications where stack data structure can be used?

Answer:

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

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

Answer:

- Push Operations
- Pop Operations
- Peek Operations

8. What is the difference between PUSH and POP?

Answer:

PUSH and POP operations specify how data is stored and retrieved in a stack.

- PUSH: PUSH specifies that data is being "inserted" into the stack.
- POP: POP specifies data retrieval. It means that data is being deleted from the stack.

9. What is a postfix expression?

Answer:

An expression in which operators follow the operands is known as postfix expression. The main benefit of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

The expression "a + b" will be represented as "ab+" in postfix notation.

10. What is an array?

Answer:

Arrays are defined as the collection of similar types of data items stored at contiguous memory locations. It is the simplest data structure in which each data element can be randomly accessed by using its index number.

11. What is a linked list data structure? What are the applications for the Linked list?

Answer:

A linked list can be thought of as a series of linked nodes (or items) that are connected by links (or paths). Each link represents an entry into the linked list, and each entry points to the next node in the sequence. The order in which nodes are added to the list is determined by the order in which they are created.

Following are some applications of linked list data structure:

- Stack, Queue, binary trees, and graphs are implemented using linked lists.
- Dynamic management for Operating System memory.
- Round robin scheduling for operating system tasks.
- Forward and backward operation in the browser.

12. What is an asymptotic analysis of an algorithm?

Answer:

Asymptotic analysis of an algorithm defines the run-time performance as per its mathematical boundations. Asymptotic analysis helps us articulate the best case(Omega Notation, Ω), average case(Theta Notation, θ), and worst case(Big Oh Notation, O) performance of an algorithm.

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

Answer:

There are two techniques by using which, the elements of a 2D array can be stored in the memory.

- **Row-Major Order:** In row-major ordering, all the rows of the 2D array are stored into the memory contiguously. First, the 1st row of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last row.
- **Column-Major Order:** In column-major ordering, all the columns of the 2D array are stored into the memory contiguously. first, the 1st column of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last column of the array.

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

Answer:

A linked list is considered both linear and non-linear data structure depending upon the situation.

- On the basis of data storage, it is considered as a non-linear data structure.
- On the basis of the access strategy, it is considered as a linear data-structure.

15. What are the advantages of Linked List over an array?

Answer:

- The size of a linked list can be incremented at runtime which is impossible in the case of the array.
- The List is not required to be contiguously present in the main memory, if the contiguous space is not available, the nodes can be stored anywhere in the memory connected through the links.
- The List is dynamically stored in the main memory and grows as per the program demand while the array is statically stored in the main memory, size of which must be declared at compile time.
- The number of elements in the linked list are limited to the available memory space while the number of elements in the array is limited to the size of an array.

16. If you are using C language to implement the heterogeneous linked list, what pointer type should be used?

Answer:

The heterogeneous linked list contains different data types, so it is not possible to use ordinary pointers for this. For this purpose, you have to use a generic pointer type like void pointer because the void pointer is capable of storing a pointer to any type.

17. What is doubly linked list?

Answer:

The doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence. In a doubly linked list, a node consists of three parts:

- node data
- pointer to the next node in sequence (next pointer)
- pointer to the previous node (previous pointer).

18. What is hashmap in data structure?

Answer:

Hashmap is a data structure that uses an implementation of a hash table data structure which allows access to data in constant time (O(1)) complexity if you have the key.

19. List some applications of queue data structure.

Answer:

The Applications of the queue is given as follows:

- Queues are widely used as waiting lists for a single shared resource like a printer, disk, CPU.
- Queues are used in the asynchronous transfer of data (where data is not being transferred at the same rate between two processes) for eg. pipes, file IO, sockets.
- Queues are used as buffers in most of the applications like MP3 media player, CD player, etc.
- Queues are used to maintain the playlist in media players to add and remove the songs from the play-list.
- Queues are used in operating systems for handling interrupts.

20. What are the drawbacks of array implementation of Queue?

Answer:

Memory Wastage: The space of the array, which is used to store queue elements, can never be reused to store the elements of that queue because the elements can only be inserted at front end and the value of front might be so high so that, all the space before that, can never be filled.

Array Size: There might be situations in which, we may need to extend the queue to insert more elements if we use an array to implement queue, It will almost be impossible to extend the array size, therefore deciding the correct array size is always a problem in array implementation of queue.

Intermediate Level 

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

Answer:

Priority Queue is an abstract data type that is similar to a queue in that each element is assigned a priority value. The order in which elements in a priority queue are served is determined by their priority (i.e., the order in which they are removed). If the elements have the same priority, they are served in the order they appear in the queue.

Following are some real-time applications for priority queue:

- Used in graph algorithms like Dijkstra, Prim's Minimum spanning tree etc.
- Huffman code for data compression
- Finding Kth Largest/Smallest element

22. List some applications of queue data structure.

Answer:

The Applications of the queue is given as follows:

- Queues are widely used as waiting lists for a single shared resource like a printer, disk, CPU.
- Queues are used in the asynchronous transfer of data (where data is not being transferred at the same rate between two processes) for eg. pipes, file IO, sockets.
- Queues are used as buffers in most of the applications like MP3 media player, CD player, etc.
- Queues are used to maintain the playlist in media players to add and remove the songs from the play-list.
- Queues are used in operating systems for handling interrupts.

23. Define the tree data structure.

Answer:

The Tree is a recursive data structure containing the set of one or more data nodes where one node is designated as the root of the tree while the remaining nodes are called as the children of the root. The nodes other than the root node are partitioned into the nonempty sets where each one of them is to be called sub-tree.

24. List the types of tree.

Answer:

There are six types of tree given as follows.

- General Tree
- Forests
- Binary Tree
- Binary Search Tree
- Expression Tree
- Tournament Tree

25. What are Binary trees?

Answer:

A binary Tree is a special type of generic tree in which, each node can have at most two children. Binary tree is generally partitioned into three disjoint subsets, i.e. the root of the node, left sub-tree and Right binary sub-tree.

26. How can AVL Tree be useful in all the operations as compared to Binary search tree?

Answer:

AVL tree controls the height of the binary search tree by not letting it be skewed. The time taken for all operations in a binary search tree of height h is O(h). However, it can be extended to O(n) if the BST becomes skewed (i.e. worst case). By limiting this height to log n, AVL tree imposes an upper bound on each operation to be O(log n) where n is the number of nodes.

27. State the properties of B Tree.

Answer:

A B tree of order m contains all the properties of an M way tree. In addition, it contains the following properties.

- Every node in a B-Tree contains at most m children.
- Every node in a B-Tree except the root node and the leaf node contain at least m/2 children.
- The root nodes must have at least 2 nodes.
- All leaf nodes must be at the same level.

28. List some applications of Tree-data structure?

Answer:

Applications of Tree- data structure:

- The manipulation of Arithmetic expression,
- Symbol Table construction,
- Syntax analysis
- Hierarchal data model

29. Define the graph data structure?

Answer:

A graph G can be defined as an ordered set G(V, E) where V(G) represents the set of vertices and E(G) represents the set of edges which are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent-child relations.

30. Differentiate among cycle, path, and circuit?

Answer:

Path: A Path is the sequence of adjacent vertices connected by the edges with no restrictions.
Cycle: A Cycle can be defined as the closed path where the initial vertex is identical to the end vertex. Any vertex in the path can not be visited twice
Circuit: A Circuit can be defined as the closed path where the intial vertex is identical to the end vertex. Any vertex may be repeated.

31. What are the applications of Graph data structure?

Answer:

The graph has the following applications:

- Graphs are used in circuit networks where points of connection are drawn as vertices and component wires become the edges of the graph.
- Graphs are used in transport networks where stations are drawn as vertices and routes become the edges of the graph.
- Graphs are used in maps that draw cities/states/regions as vertices and adjacency relations as edges.
- Graphs are used in program flow analysis where procedures or modules are treated as vertices and calls to these procedures are drawn as edges of the graph.

32. What are the advantages of Binary search over linear search?

Answer:

There are relatively less number of comparisons in binary search than that in linear search. In average case, linear search takes O(n) time to search a list of n elements while Binary search takes O(log n) time to search a list of n elements.

33. What are the advantages of Selection Sort?

Answer:

- It is simple and easy to implement.
- It can be used for small data sets.
- It is 60 per cent more efficient than bubble sort.

34. What is the difference between NULL and VOID?

Answer:

- Null is actually a value, whereas Void is a data type identifier.
- A null variable simply indicates an empty value, whereas void is used to identify pointers as having no initial size.

35. Differentiate between stack and queue data structure.

Answer:

Stack	Queue
Stack is a linear data structure where data is added and removed from the top.	Queue is a linear data structure where data is ended at the rear end and removed from the front.
Stack is based on LIFO(Last In First Out) principle	Queue is based on FIFO(First In First Out) principle
Insertion operation in Stack is known as push.	Insertion operation in Queue is known as eneque.
Delete operation in Stack is known as pop.	Delete operation in Queue is known as dequeue.
Only one pointer is available for both addition and deletion: top()	Two pointers are available for addition and deletion: front() and rear()
Used in solving recursion problems	Used in solving sequential processing problems

36. Elaborate on different types of array data structure

Answer.

There are several different types of arrays:

- **One-dimensional array:** A one-dimensional array stores its elements in contiguous memory locations, accessing them using a single index value. It is a linear data structure holding all the elements in a sequence.
- **Two-dimensional array**: A two-dimensional array is a tabular array that includes rows and columns and stores data. An M × N two-dimensional array is created by grouping M rows and N columns into N columns and rows.
- **Three-dimensional array:** A three-dimensional array is a grid that has rows, columns, and depth as a third dimension. It comprises a cube with rows, columns, and depth as a third dimension. The three-dimensional array has three subscripts for a position in a particular row, column, and depth. Depth (dimension or layer) is the first index, row index is the second index, and column index is the third index.

37. Elaborate on different types of Linked List data structures?

Answer:

Following are different types of linked lists:

1. **Singly Linked List:** A singly linked list is a data structure that is used to store multiple items. The items are linked together using the key. The key is used to identify the item and is usually a unique identifier. In a singly linked list, each item is stored in a separate node. The node can be a single object or it can be a collection of objects. When an item is added to the list, the node is updated and the new item is added to the end of the list. When an item is removed from the list, the node that contains the removed item is deleted and its place is taken by another node. The key of a singly linked list can be any type of data structure that can be used to identify an object. For example, it could be an integer, a string, or even another singly linked list. Singly-linked lists are useful for storing many different types of data. For example, they are commonly used to store lists of items such as grocery lists or patient records. They are also useful for storing data that is time sensitive such as stock market prices or flight schedules.

2. Doubly Linked List: A doubly linked list is a data structure that allows for two-way data access such that each node in the list points to the next node in the list and also points back to its previous node. In a doubly linked list, each node can be accessed by its address, and the contents of the node can be accessed by its index. It's ideal for applications that need to access large amounts of data in a fast manner. A disadvantage of a doubly linked list is that it is more difficult to maintain than a single-linked list. In addition, it is more difficult to add and remove nodes than in a single-linked list.

3. Circular Linked List: A circular linked list is a unidirectional linked list where each node points to its next node and the last node points back to the first node, which makes it circular.

4. Doubly Circular Linked List: A doubly circular linked list is a linked list where each node points to its next node and its previous node and the last node points back to the first node and first node's previous points to the last node.

5. Header List: A list that contains the header node at the beginning of the list, is called the header-linked list. This is helpful in calculating some repetitive operations like the number of elements in the list etc.

38. Difference between Array and Linked List.

Answer:

Arrays	**Linked Lists**
An array is a collection of data elements of the same type.	A linked list is a collection of entities known as nodes. The node is divided into two sections: data and address.
It keeps the data elements in a single memory.	It stores elements at random, or anywhere in the memory.
The memory size of an array is fixed and cannot be changed during runtime.	The memory size of a linked list is allocated during runtime.
An array's elements are not dependent on one another.	Linked List elements are dependent on one another.
It is easier and faster to access an element in an array.	In the linked list, it takes time to access an element.
Memory utilization is ineffective in the case of an array.	Memory utilization is effective in the case of an array.
Operations like insertion and deletion take longer time in an array.	Operations like insertion and deletion are faster in the linked list.

39. What is linear searching?

Answer:

Linear Search is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found, otherwise the search continues till the end of the data set. It is the easiest searching algorithm. This method can be performed on a sorted or an unsorted list (usually arrays).

Linear Search Algorithm

Step 1: First, read the search element (Target element) in the array.

Step 2: In the second step compare the search element with the first element in the array.

Step 3: If both are matched, display “Target element is found” and terminate the Linear Search function.

Step 4: If both are not matched, compare the search element with the next element in the array.

Step 5: In this step, repeat steps 3 and 4 until the search (Target) element is compared with the last element of the array.

Step 6 – If the last element in the list does not match, the Linear Search Function will be terminated, and the message “Element is not found” will be displayed.

40. What is bubble sort and how bubble sort works?

Answer:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Time Complexity: O(N2)

Auxiliary Space: O(1)

We assume list is an array of n elements. We further assume that swap function swaps the values of the given array elements.

```
begin BubbleSort(list)
for all elements of list
if list[i] > list[i+1]
swap(list[i], list[i+1])
    end if
end for
return list
end BubbleSort
```

41. What is selection sort?

Answer:

In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. It is also the simplest algorithm. It is an in-place comparison sorting algorithm. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right.

The average and worst-case complexity of selection sort is O(n2), where n is the number of items. Due to this, it is not suitable for large data sets.

Selection sort is generally used when -

- A small array is to be sorted
- Swapping cost doesn't matter
- It is compulsory to check all elements

Algorithm

SELECTION SORT(arr, n)
Step 1: Repeat Steps 2 and 3 for i = 0 to n-1
Step 2: CALL SMALLEST(arr, i, n, pos)
Step 3: SWAP arr[i] with arr[pos]
[END OF LOOP]
Step 4: EXIT
SMALLEST (arr, i, n, pos)
Step 1: [INITIALIZE] SET SMALL = arr[i]
Step 2: [INITIALIZE] SET pos = i
Step 3: Repeat for j = i+1 to n
if (SMALL > arr[j])
SET SMALL = arr[j]
SET pos = j
[END OF if]
[END OF LOOP]
Step 4: RETURN pos

42. How insertion sort and selection sorts are different?

Answer:

Insertion Sort

- In Insertion Sort, the values are inserted in a list/array wherein some of the values are previous sorted.
- It is a stable sorting algorithm.
- The best-case time complexity is O(N) (when the list has been sorted in ascending order).
- A key value is defined, and the array is iterated from the beginning to the end.
- The current element during iteration is compared to the key value.
- The number of comparison operations made during insertion sort is less than the number of the element swapping that is done.
- If the key element is smaller than the element with which it is being compared, the previous elements are compared with it.
- The elements greater than key are moved one position up to make space for the element that is swapped.
- The elements are known before hand, only their position is determined during insertion sort.

Selection sort

- In Selection Sort, at first, the minimum or the maximum number from the list is obtained.
- The list is sorted in ascending or descending order.
- It is considered as an unstable sorting algorithm.
- The time complexity in all cases is O(n^2).
- It is less efficient in comparison to insertion sort.
- The number of comparisons made during iterations is more than the element swapping that is done.
- The location of every element in the list is previously known.
- This means the user only searches for the element that needs to be inserted at the specific position.

43. What are some key operations performed on the Deque data structure?

Answer:

Following are the key operations available deque:

- insertFront(): This adds an element to the front of the Deque.
- insertLast(): This adds an element to the rear of the Deque.
- deleteFront(): This deletes an element from the front of the Deque.
- deleteLast():This deletes an element from the front of the Deque.
- getFront(): This gets an element from the front of the Deque.
- getRear(): This gets an element from the rear of the Deque.
- isEmpty(): This checks whether Deque is empty or not.
- isFull(): This checks whether Deque is full or not.

44. Compare different implementations of priority queue

Answer:

Operations	peek	insert	delete
Linked List	O(1)	O(n)	O(1)
Binary Heap	O(1)	O(log n)	O(log n)
Binary Search Tree	O(1)	O(log n)	O(log n)

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

Answer:

Priority Queue is an abstract data type that is similar to a queue in that each element is assigned a priority value. The order in which elements in a priority queue are served is determined by their priority (i.e., the order in which they are removed). If the elements have the same priority, they are served in the order they appear in the queue.

Following are some real-time applications for priority queue:

- Used in graph algorithms like Dijkstra, Prim's Minimum spanning tree etc.
- Huffman code for data compression
- Finding Kth Largest/Smallest element

Advanced Level 

46. What is the difference between the Breadth First Search (BFS) and Depth First Search (DFS)?

Answer:

Breadth First Search (BFS)	Depth First Search (DFS)
It stands for "Breadth First Search"	It stands for "Depth First Search"
BFS (Breadth First Search) finds the shortest path using the Queue data structure.	DFS (Depth First Search) finds the shortest path using the Stack data structure.
We walk through all nodes on the same level before passing to the next level in BFS.	DFS begins at the root node and proceeds as far as possible through the nodes until we reach the node with no unvisited nearby nodes.

When compared to DFS, BFS is slower.	When compared to BFS, DFS is faster.
BFS performs better when the target is close to the source.	DFS performs better when the target is far from the source.
BFS necessitates more memory.	DFS necessitates less memory.
Nodes that have been traversed multiple times are removed from the queue.	When there are no more nodes to visit, the visited nodes are added to the stack and then removed.
Backtracking is not an option in BFS.	The DFS algorithm is a recursive algorithm that employs the concept of backtracking.
It is based on the FIFO principle (First In First Out).	It is based on the LIFO principle (Last In First Out).

47. What is a B-tree data structure? What are the applications for B-trees?

Answer:

The B Tree is a type of m-way tree that is commonly used for disc access. A B-Tree with order m can only have m-1 keys and m children. One of the primary reasons for using a B tree is its ability to store a large number of keys in a single node as well as large key values while keeping the tree's height relatively small.

Following are the key properties of a B-tree data structure:

- All of the leaves are at the same height.
- The term minimum degree 't' describes a B-Tree. The value of t is determined by the size of the disc block.
- Except for root, every node must have at least t-1 keys. The root must contain at least one key.
- All nodes (including root) can have no more than 2*t - 1 keys.

- The number of children of a node is equal to its key count plus one.
- A node's keys are sorted in ascending order. The child of two keys k1 and k2 contains all keys between k1 and k2.
- In contrast to Binary Search Tree, B-Tree grows and shrinks from the root.

Following are real-time applications of a B-Tree data structure:

- It is used to access data stored on discs in large databases.
- Using a B tree, you can search for data in a data set in significantly less time.
- The indexing feature allows for multilevel indexing.
- The B-tree approach is also used by the majority of servers.

48. Define Segment Tree data structure and its applications.

Answer:

A segment Tree is a binary tree that is used to store intervals or segments. The Segment Tree is made up of nodes that represent intervals. Segment Tree is used when there are multiple range queries on an array and changes to array elements.

Following are key operations performed on the Segment tree data structure:

- Building Tree: In this step, we create the structure and initialize the segment tree variable.
- Updating the Tree: In this step, we change the tree by updating the array value at a point or over an interval.
- Querying Tree: This operation can be used to run a range query on the array.

Following are real-time applications for Segment Tree:

- Used to efficiently list all pairs of intersecting rectangles from a list of rectangles in the plane.
- The segment tree has become popular for use in pattern recognition and image processing.
- Finding range sum/product, range max/min, prefix sum/product, etc
- Computational geometry
- Geographic information systems

49. Define Trie data structure and its applications

Answer:

The word "Trie" is an abbreviation for "retrieval." Trie is a data structure that stores a set of strings as a sorted tree. Each node has the same number of pointers as the number of alphabet characters. It can look up a word in the dictionary by using its prefix. Assuming that all strings are formed from the letters 'a' to 'z' in the English alphabet, each trie node can have a maximum of 26 points.

Trie is also referred to as the digital tree or the prefix tree. The key to which a node is connected is determined by its position in the Trie. Trie allows us to insert and find strings in O(L) time, where L is the length of a single word. This is clearly faster than BST. Because of how it is implemented, this is also faster than Hashing. There is no need to compute a hash function. There is no need to handle collisions (like we do in open addressing and separate chaining)
Another benefit of Trie is that we can easily print all words in alphabetical order, which is not easy with hashing. Trie can also perform prefix search (or auto-complete) efficiently.

The main disadvantage of tries is that they require a large amount of memory to store the strings. We have an excessive number of node pointers for each node
Following are some real-time applications for Trie data structure:

- Auto-Complete and Search for Search Engines
- Genome Analysis
- Data Analytics
- Browser History
- Spell Checker

50. Define Red-Black Tree and its applications

Answer:

Red Black Trees are a type of self-balancing binary search tree. Rudolf Bayer invented it in 1972 and dubbed it "symmetric binary B-trees."

A red-black tree is a Binary tree in which each node has a colour attribute, either red or black. By comparing the node colours on any simple path from the root to a leaf, red-black trees ensure that no path is more than twice as long as any other, ensuring that the tree is generally balanced.

Red-black trees are similar to binary trees in that they both store their data in two's complementary binary formats. However, red-black trees have one important advantage over binary trees: they are faster to access. Because red-black trees are so fast to access, they are often used to store large amounts of data.

Red-black trees can be used to store any type of data that can be represented as a set of values.

Every Red-Black Tree Obeys the Following Rules:

- Every node is either red or black.
- The tree's root is always black.
- There are no two red nodes that are adjacent.
- There is the same number of black nodes on every path from a node to any of its descendant's NULL nodes.
- All of the leaf nodes are black.

Following are some real-time applications for the Red-Black Tree data structure:

- The majority of self-balancing BST library functions in C++ or Java use Red-Black Trees.
- It is used to implement Linux CPU Scheduling.
- It is also used to reduce time complexity in the K-mean clustering algorithm in machine learning.
- MySQL also employs the Red-Black tree for table indexes in order to reduce searching and insertion time.

51. Which data structures are used for implementing LRU cache?

Answer:

LRU cache or Least Recently Used cache allows quick identification of an element that hasn't been put to use for the longest time by organizing items in order of use. In order to achieve this, two data structures are used:

Queue – This is implemented using a doubly-linked list. The maximum size of the queue is determined by the cache size, i.e by the total number of available frames. The least recently used pages will be near the front end of the queue whereas the most recently used pages will be towards the rear end of the queue.

Hashmap – Hashmap stores the page number as the key along with the address of the corresponding queue node as the value.

52. What is a heap data structure?

Answer:

Heap is a special tree-based non-linear data structure in which the tree is a complete binary tree. A binary tree is said to be complete if all levels are completely filled except possibly the last level and the last level has all elements as left as possible. Heaps are of two types:

Max-Heap:
In a Max-Heap the data element present at the root node must be the greatest among all the data elements present in the tree.
This property should be recursively true for all sub-trees of that binary tree.

Min-Heap:
In a Min-Heap the data element present at the root node must be the smallest (or minimum) among all the data elements present in the tree.
This property should be recursively true for all sub-trees of that binary tree.

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

Answer:

SN	B Tree	B+ Tree
1	Search keys cannot repeatedly be stored.	Redundant search keys can be present.
2	Data can be stored in leaf nodes as well as internal nodes	Data can only be stored on the leaf nodes.
3	Searching for some data is a slower process since data can be found on internal nodes as well as on the leaf nodes.	Searching is comparatively faster as data can only be found on the leaf nodes.
4	Deletion of internal nodes is so complicated and time-consuming.	Deletion will never be a complexed process since element will always be deleted from the leaf nodes.
5	Leaf nodes cannot be linked together.	Leaf nodes are linked together to make the search operations more efficient.

54. List some applications of Tree-data structure?

Answer:

Applications of Tree- data structure:

- The manipulation of Arithmetic expression,
- Symbol Table construction,
- Syntax analysis
- Hierarchal data model

55. Define the graph data structure?

Answer:

A graph G can be defined as an ordered set G(V, E) where V(G) represents the set of vertices and E(G) represents the set of edges which are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent-child relations.

56. Differentiate among cycle, path, and circuit?

Answer:

Path: A Path is the sequence of adjacent vertices connected by the edges with no restrictions.

Cycle: A Cycle can be defined as the closed path where the initial vertex is identical to the end vertex. Any vertex in the path cannot be visited twice

Circuit: A Circuit can be defined as the closed path where the intial vertex is identical to the end vertex. Any vertex may be repeated.

57. Which data structures are used in BFS and DFS algorithm?

Answer:

In BFS algorithm, Queue data structure is used.

In DFS algorithm, Stack data structure is used.

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

Answer:

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

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

Answer:

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.

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

Answer:

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.

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

Answer:

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.

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

Answer:

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

63. List some applications of multilinked structures?

Answer:

- Sparse matrix
- Index generation

64. What is dynamic memory management?

Answer:

Dynamic memory management is a technique in which storage units are allocated based on the requirements continuously. Using dynamic memory allocations, individual data structures can be either stored separately or combined to form entities called composites. These composites can be worked on when required.

65. How is a variable stored in memory when using Data Structures?

Answer:

A variable is stored based on the amount of memory that is needed. First, the required quantity of memory is assigned, and later, it is stored based on the data structure being used. Using concepts such as dynamic allocation ensures high efficiency and that the storage units can be supplied based on the requirements in real time.

66. How does Huffman's algorithm work?

Answer:

Huffman's algorithm uses a table, containing the frequency of the occurrence of every data entity on the list. This is used for creating extended binary trees, which are known to have minimum weights for the path lengths. This is considering each of the corresponding weights.

67. What are the time complexities of linear search and binary search?

Answer:

Binary search is more effective as it takes lesser comparisons to search for an element in an array. The time complexity for linear search is O(n), while it is O(log n) for binary search.

68. Where are Multi-linked Data Structures used?

Answer:

Multi-linked data structures are used in many domains. Following are the two most important use cases of multi-linked data structures:

- Generation of sparse matrices
- Index creation for data entities

69. What is the method used for inorder traversal in trees?

Answer:

Inorder traversal works in the following way:

- The tree is traversed through the left subtree.
- The root node is visited after the left visit.
- Then, the right subtree is traversed.

70. What is the working of post-order traversal in trees?

Answer:

Postorder traversal works in the following way:

- First, the left subtree is traversed through.
- The right subtree is traversed next.
- The root node is visited after the right subtree visit.

71. How can elements be inserted in the circular queue?

Answer:

There are two cases in which items can be placed in a circular queue. They are as follows:

- When front != 0 and rear = max -1. This makes it possible as the queue will not be full, and new elements can be inserted here.
- When rear != max -1. This ensures that the rear is incremented to the maximum allocation size, and values can be inserted easily to the rear end of the queue.

72. What is jagged array?

Answer:

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

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

Answer:

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.

74. What are recursive algorithms? State the important rules which every recursive algorithm must follow.

Answer:

Recursive algorithm is a way of tackling a difficult problem by breaking it down into smaller and smaller subproblems until the problem is small enough to be solved quickly. It usually involves a function that calls itself (property of recursive functions).
The three laws which must be followed by all recursive algorithms are as follows:

- There should be a base case.
- It is necessary for a recursive algorithm to call itself.
- The state of a recursive algorithm must be changed in order for it to return to the base case.

75. Explain the Dijkstra's Algorithm to find the shortest path between a given node in a graph to any other node in the graph.

Answer:

Dijkstra's algorithm is a method for determining the shortest pathways between nodes in a graph, which might be used to depict road networks. Edsger W. Dijkstra, a computer scientist, conceived it in 1956 and published it three years later. There are numerous variations of the algorithm. The original Dijkstra algorithm discovered the shortest path between two nodes, but a more frequent form fixes a single node as the "source" node and finds the shortest pathways from the source to all other nodes in the network, resulting in a shortest-path tree. Let us take a look at Dijkstra's Algorithm to find the shortest path between a given node in a graph to any other node in the graph:

Let us call the node where we are starting the process as the initial node. Let the distance from the initial node to Y be the distance of node Y. Dijkstra's algorithm will begin with unlimited distances and attempt to improve them incrementally.

Step 1: Mark all nodes that have not been visited yet. The unvisited set is a collection of all the nodes that have not been visited yet.

Step 2: Assign a tentative distance value to each node: set it to zero for our first node and infinity for all others. The length of the shortest path discovered so far between the node v and the initial node is the tentative distance of a node v. Because no other vertex other than the source (which is a path of length zero) is known at the start, all other tentative distances are set to infinity. Set the current node to the beginning node.

Step 3: Consider all of the current node's unvisited neighbours and determine their approximate distances through the current node. Compare the newly calculated tentative distance to the current assigned value and choose the one that is less. If the present node A has a distance of 5 and the edge linking it to a neighbour B has a length of 3, the distance to B through A will be 5 +3 = 8. Change B to 8 if it was previously marked with a distance greater than 8. If this is not the case, the current value will be retained.

Step 4: Mark the current node as visited and remove it from the unvisited set once we have considered all of the current node's unvisited neighbours. A node that has been visited will never be checked again.
Stop if the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance between the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and the remaining unvisited nodes). The algorithm is now complete.

Step 5: Otherwise, return to step 3 and select the unvisited node indicated with the shortest tentative distance as the new current node.
It is not required to wait until the target node is "visited" as described above while constructing a route: the algorithm can end once the destination node has the least tentative distance among all "unvisited" nodes (and thus could be selected as the next "current"). For arbitrary directed graphs with unbounded non-negative weights, Dijkstra's algorithm is asymptotically the fastest known single-source shortest path algorithm with time complexity of O(|E| + |V|log(|V|)), where |V| is the number of nodes and|E| is the number of edges in the graph.

76. Explain the BFS algorithm

Answer:

BFS (Breadth First Search) is a graph traversal algorithm. It starts traversing the graph from the root node and explores all the neighboring nodes. It selects the nearest node and visits all the unexplored nodes. The algorithm follows the same procedure for each of the closest nodes until it reaches the goal state.

Algorithm

Step1: Set status=1 (ready state)
Step2: Queue the starting node A and set its status=2, i.e. (waiting state)
Step3: Repeat steps 4 and 5 until the queue is empty.
Step4: Dequeue a node N and process it and set its status=3, i.e. (processed state)
Step5: Queue all the neighbors of N that are in the ready state (status=1) and set their status =2 (waiting state)
[Stop Loop]
Step6: Exit

77. What are Greedy algorithms? Give some example of it?

Answer:

A greedy algorithm is an algorithmic strategy which is made for the best optimal choice at each sub stage with the goal of this, eventually leading to a globally optimum solution. This means that the algorithm chooses the best solution at the moment without regard for consequences.
In other words, an algorithm that always takes the best immediate, or local, solution while finding an answer.
Greedy algorithms find the overall, ideal solution for some idealistic problems, but may discover less-than-ideal solutions for some instances of other problems.
Below is a list of algorithms that finds their solution with the use of the Greedy algorithm.

- Travelling Salesman Problem
- Prim's Minimal Spanning Tree Algorithm
- Kruskal's Minimal Spanning Tree Algorithm
- Dijkstra's Minimal Spanning Tree Algorithm
- Graph - Map Coloring
- Graph - Vertex Cover
- Knapsack Problem
- Job Scheduling Problem

78. Write an algorithm to insert a node in the Binary search tree?

Answer:

Insert node operation is a smooth operation. You need to compare it with the root node and traverse left (if smaller) or right (if greater) according to the value of the node to be inserted.

Algorithm:

- Make the root node as the current node
- If the node to be inserted < root
- If it has left child, then traverse left
- If it does not have left child, insert node here
- If the node to be inserted > root
- If it has the right child, traverse right
- If it does not have the right child, insert node here.

79. What do you understand by a searching algorithm? List a few types of searching algorithms.

Answer:

Searching Algorithms are used to look for an element or get it from a data structure (usually a list of elements). These algorithms are divided into two categories based on the type of search operation:

- Sequential Search: This method traverses the list of elements consecutively, checking each element and reporting if the element to be searched is found. Linear Search is an example of a Sequential Search Algorithm.
- Interval Search: These algorithms were created specifically for searching sorted data structures. Because they continually target the center of the search structure and divide the search space in half, these types of search algorithms are far more efficient than Sequential Search algorithms. Binary Search is an example of an Interval Search Algorithm.

80. Write down an algorithm for adding a node to a linked list sorted in ascending order(maintaining the sorting property).

Answer:

An algorithm for adding a node to a link list sorted in ascending order (maintaining the sorting property) is given below:

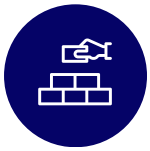
- Step 1: Check if the linked list has no value (or is empty). If yes, then set the new node as the head and return it.
- Step 2: Check if the value of the node to be inserted is smaller than the value of the head node. If yes, place it at the beginning and make it the head node.
- Step 3: Find the suitable node after which the input node should be added in a loop. To discover the required node, begin at the head and work your way forward until you reach a node whose value exceeds the input node. The preceding node is the correct node.
- Step 4: After the correct node is found in step 3, insert the node.

TalentBattle

Introducing...

Complete Placement Preparatory Masterclass

What is included?



400+ Hours Foundation of LIVE + Recorded Training on **Quant, Reasoning, Verbal, C, C++, Java, Python, DSA, OS, CN, DBMS**

Interview preparation Training along with One-to-One Mock **Technical & Personal Interviews** by experts

Latest Technologies Certification Courses to get higher packages. 500+ hours of courses on **Full stack development, AI, ML, Data Science, AWS Cloud Computing, IoT, Robotics, Tkinter, Django Data Analytics, Tableau, PowerBI and much more**

Company-specific LIVE and Recorded training for **30+ service and product-based companies.**

TCS NQT | Accenture | Capgemini | Cognizant |Infosys | Persistent | Deloitte | Mindtree | Virtusa | Goldman Sachs | Bosch | Samsung Amazon | Nalsoft | Zoho Cisco and 15+ more companies preparation.

15+ Real Time Projects based on Latest Technologies and **10+ Mini Projects** based on C, C++, Java and Python to build your Profile

Get TCS NQT Paid Exam for Free

Our Placement Reports

97.6%

Selection Ratio
of Complete Masterclass Students

4.91 / 5

★★★★★

Overall Rating
out of 5

Highest CTC
40 LPA

85% students
placed with CTC more
than 6 LPA

Average CTC
8.5 LPA

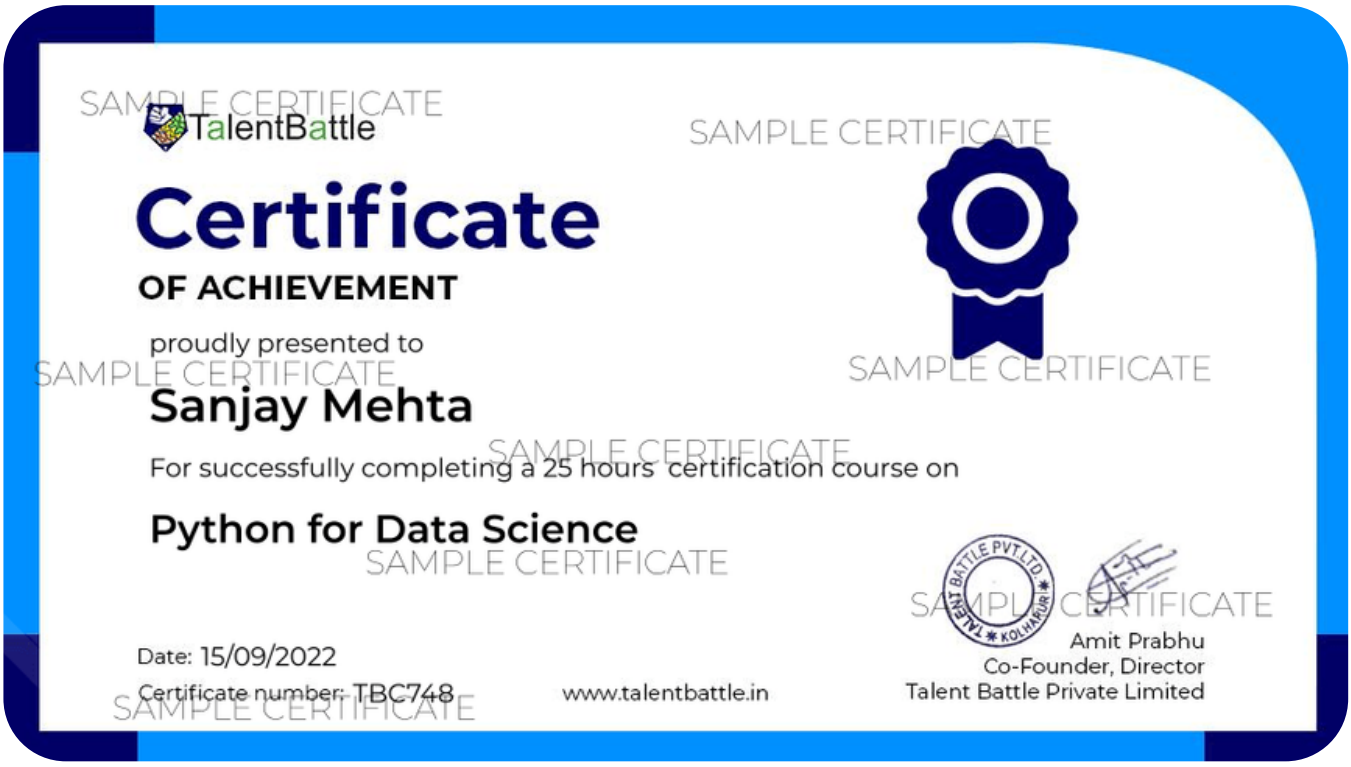
Referral Opportunities
in Top Companies & Startups
by Talent Battle

Average Offers
per student: 2.7

Students from
5000+ colleges
use Masterclass for
Placement Preparation

- **50000+ placed** students (in the last 10 years)
- Our Students placed in **600+ Companies**
- **10 Lakh+ Students** Prepare with Talent Battle Resources Every Year

Talent Battle Certifications

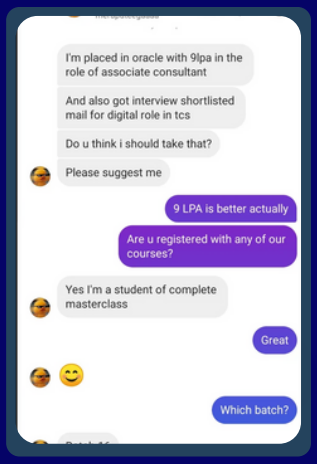


Get a Free Mentorship from experts for your Campus Placement Preparation

- Discuss your queries with experts
- Get a roadmap for your placement preparation

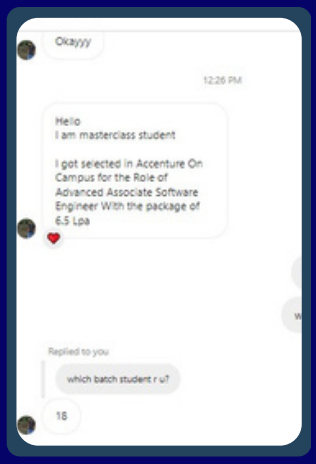
Click to know more

Some of our placed students



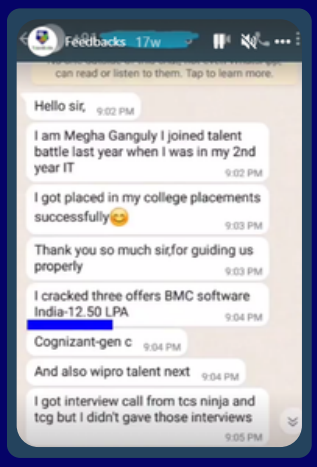
Srinija Kalluri

COMPLETE MASTERCLASS STUDENT
SELECTED AT ORACLE - 9 LPA



Rohit

COMPLETE MASTERCLASS STUDENT
SELECTED AT ACCENTURE - 6.5 LPA



Megha Ganguly

COMPLETE MASTERCLASS STUDENT
SELECTED COGNIZANT, WIPRO & BMC INDIA - 12.5 LPA

Aditya Kulsestha

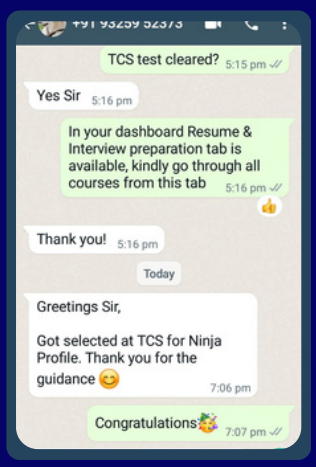
COMPLETE MASTERCLASS STUDENT
SELECTED AT COGNIZANT - 7.8 LPA

Shubham Verma

COMPLETE MASTERCLASS STUDENT
SELECTED AT CAPGEMINI- 7.5 LPA

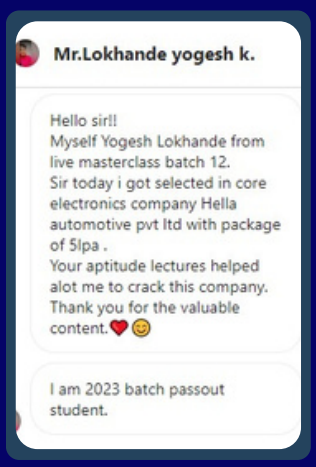
Amardeep Prajapati

COMPLETE MASTERCLASS STUDENT
SELECTED AT HAPPIEST MIND TECHNOLOGY - 5.4 LPA



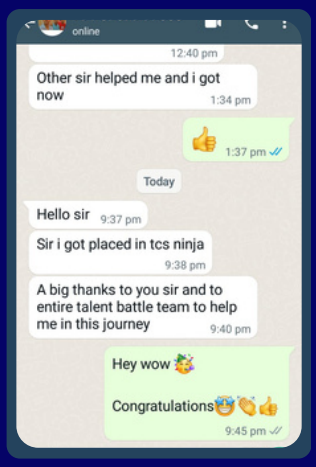
Rutuja Jangam

COMPLETE MASTERCLASS STUDENT
SELECTED AT TCS NINJA



Yogesh Lokhande

COMPLETE MASTERCLASS STUDENT
SELECTED AT HELLA ELECTRONICS -5 LPA



Vikas Varak

COMPLETE MASTERCLASS STUDENT
SELECTED AT TCS NINJA

Tools & Technologies

covered in Placement Pro!

Our Team

Amit Prabhu

Co-founder

9+ years of experience in training students for Quantitative Aptitude, Verbal & Monitoring students for Campus Placement Preparation

Ajinkya Kulkarni

Co-founder

9+ years of experience in training students for Reasoning, Interview Training & Mentoring Students for Campus Placement Preparation

Rohit Bag

Lead Technical Trainer

10+ years of experience in training students for Programming Languages & Core Computer Science Subjects along with Company Specific Training

Vaishnavi K Dharan

Technical Trainer

5+ years of experience in training students on different Programming Languages and Data Structure. Master certified trainer in Robotic Process Automation in Automation Anywhere.

Poojitha Renati

Lead Aptitude Trainer

5+ years of experience in training students for Aptitude and Logical Reasoning. Trained more than 5000+ hours in various institutes including GITAM, Parul, KITS, JNTU and more

Chand Basha

Lead Aptitude Trainer

8+ years of experience in training students Quantitative Aptitude, Logical Reasoning & Verbal Ability for Campus Placements & Competitive Exams

Jasleen Chhabra

Mentor-Training and Placement

3+ years of experience in dealing with students and their counselling related to Academic problems as well as their placements.

Samradhni Wankhede

Graphic Designer

4+ years experience as a Creativity Expert, Graphic Designer, Teacher/Trainer and a social media enthusiast

Akshay Paricharak

Customer Service Manager

8+ years of experience in Customer service, Students counselling, Business development, Project co-ordination, Strategies Implementation, Planning and Execution.

Niranjan Kulkarni

Program Manager - Training and Placement

15 years of overall multi-functional experience in Industry and Academia, in managing diverse functions such as Training, and Human Resource Management.

Ruturaj Sankpal

Placement Mentor

2 years of experience in IT industry and currently mentoring students for campus placements.

Swapnil Wankhede

Marketing & Ops.

2.5+ years of experience in compassionate and ethical marketing. Striving to ensure students get the best opportunities and are prepared to make the most of it, learning and growing with Talent Battle.

Industry Mentors

Sandip Karekar

Industry Mentor

8 years of Industry experience and currently working with Mastercard. Decent understanding of core technologies in Computer Science & Information Technology and passionate about learning Cutting-Edge technologies.

Mayuresh Diwan

Industry Mentor

Placement Mentor: **4+ years of experience** in automotive software development.

Swapnil Patil

Industry Mentor

Lead Engineer at John Deere **Over 4+ years of experience as a Software Developer.** Having expertise in developing and maintaining web and desktop applications of different domains like Telecom, Simulations and Automations, ERP and CRM

Shadab Gada

Industry Mentor

Software developer with 3+ years of experience in design, developing, testing and maintenance of web based applications. Passionate about learning new technologies, always eager to share my knowledge, love interacting with new people.

FAQs

1 What is Talent Battle?

Talent battle is a group of mentors and educators who help students to prepare for their On and Off campus placement opportunities. We have trainers with an average of 10 years of experience in training students for their Tech company drives. We train students on Aptitude, Programming, communication skills, projects, advance technologies and all other necessary skills to get placed in their dream companies. If you want to get placed in any of your dream companies, then join our complete masterclass and fulfill your dream!

2 When and how to prepare for campus placements?

The best time to start preparing for your campus is in your third year of engineering. During this time you can start preparing for your Aptitude, Verbal and Programming skills.

Most of the companies have a similar testing pattern for selecting students. There are typically tests on aptitude, programming and communication skills.

The short answer for this question is, **prepare with Talent Battle's Masterclass** as we cover all of the above mentioned topics in detail.

3 What is Complete Masterclass?

Complete Masterclass is a combination of Concept Clearing lectures for Aptitude, Coding, DSA + Company Specific Training for 30+ Companies + Interview Preparation with Mock Interviews. Foundational and Company Specific Training is conducted LIVE. Whenever companies launch their drives, we will be conducting company specific live sessions. Foundational training right from basic to advance level will also be available in recorded format.

Along with that we have **240+ hours of Full stack Development course**, either of TCS ion internship or PAID NQT (for 2 years package) and 250+ hours of Advance certification courses like AI, ML, Data Science, etc will be available free of cost.

4 Why to chose Talent Battle?

We have structured and disciplined way of preparation. You don't need to seek outside information source. All the study material will be available on Talent Battle's dashboard. We provide end to end support to our students until they get placed. Talent Battle is one stop solution to prepare for placement drives.

Dont delay your placement **preparation** anymore!!

Learn from the experts!

Check out our social media to get regular

placement related content & job drive updates

- @talentbattle.in
- @talentbattle_2023
- @talentbattle_2024
- @talentbattle_2025
- @talentbattle_2026
- WhatsApp Group
- Free Mentorship
- Talent Battle Facebook
- Talent Battle YouTube
- Talent Battle LinkedIn
- https://talentbattle.in/