46 MOST ASKED DATA STRUCTURES

INTERVIEW QUESTIONS

@DISHA MUKHERJEE

DSA Interview Questions

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?

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.

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 question! 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.

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 data structure interview question, 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?

It 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

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?

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?

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?

It 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?

- 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.