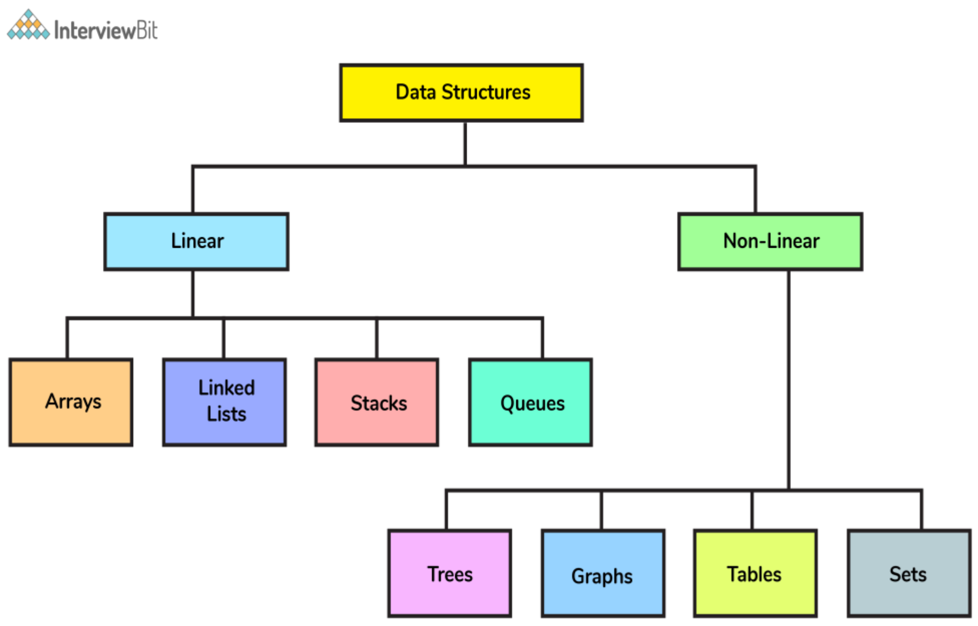
INTERVIEW QUESTIONS ON DSA

What is Data Structure?

- Data structure is a fundamental concept of any programming language, essential for algorithmic design.
- It is used for the efficient organization and modification of data.
- DS is how data and the relationship amongst different data is represented, that aids in how efficiently various functions or operations or algorithms can be applied.

Types

- There are two types of data structures:
    - Linear data structure: If the elements of a data structure result in a sequence or a linear list then it is called a linear data structure. Example: Arrays, Linked List, Stacks, Queues etc.
    - Non-linear data structure: If the elements of data structure results in a way that traversal of nodes is not done in a sequential manner, then it is a non linear data structure. Example: Trees, Graphs etc.

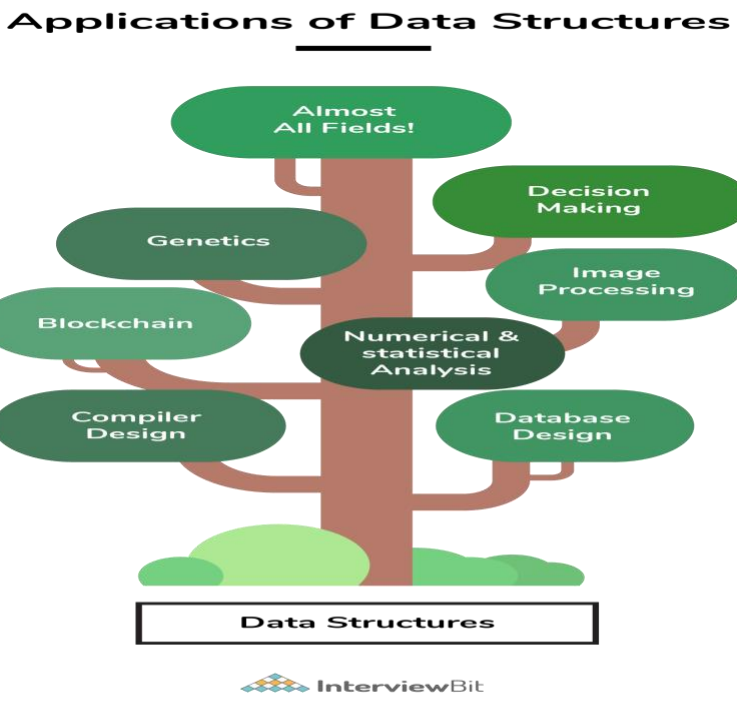


Applications

Data structures form the core foundation of software programming as any efficient algorithm to a given problem is dependent on how effectively a data is structured.

- Identifiers look ups in compiler implementations are built using hash tables.
- The B-trees data structures are suitable for the databases implementation.
- Some of the most important areas where data structures are used are as follows:

    0. Artificial intelligence
    1. Compiler design
    2. Machine learning
    3. Database design and management
    4. Blockchain
    5. Numerical and Statistical analysis
    6. Operating system development
    7. Image & Speech Processing
    8. Cryptography



Benefits of Learning Data Structures

Any given problem has constraints on how fast the problem should be solved (time) and how much less resources the problem consumes(space). That is, a problem is constrained by the space and time complexity within which it has to be solved efficiently.

- In order to do this, it is very much essential for the given problem to be represented in a proper structured format upon which efficient algorithms could be applied.
- Selection of proper data structure becomes the most important step before applying algorithm to any problem.

  Having knowledge of different kinds of data structures available helps the programmer in choosing which data structure suits the best for solving a problem efficiently. It is not just important to make a problem work, it is important how efficiently you make it work.

Data structures in C, Java

- The core concepts of data structures remains the same across all the programming languages. Only the implementation differs based on the syntax or the structure of the programming language.
  - The implementation in procedural languages like C is done with the help of structures, pointers, etc.
  - In an objected oriented language like Java, data structures are implemented by using classes and objects.
- Having sound knowledge of the concepts of each and every data structures helps you to stand apart in any interviews as selecting right data structure is the first step towards solving problem efficiently.

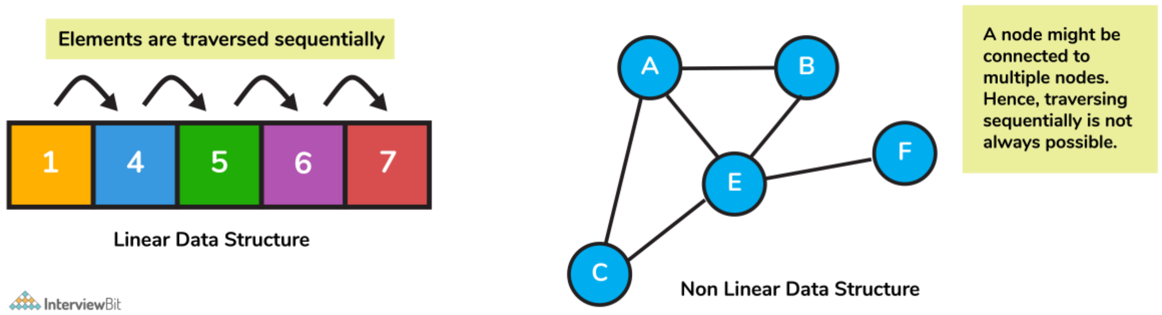
1. Can you explain the difference between file structure and storage structure?

- File Structure: Representation of data into secondary or auxiliary memory say any device such as hard disk or pen drives that stores data which remains intact until manually deleted is known as a file structure representation.
- Storage Structure: In this type, data is stored in the main memory i.e RAM, and is deleted once the function that uses this data gets completely executed.

- The difference is that storage structure has data stored in the memory of the computer system, whereas file structure has the data stored in the auxiliary memory.

2. Can you tell how linear data structures differ from non-linear data structures?

- If the elements of a data structure result in a sequence or a linear list then it is called a linear data structure. Whereas, traversal of nodes happens in a non-linear fashion in non-linear data structures.
- Lists, stacks, and queues are examples of linear data structures whereas graphs and trees are the examples of non-linear data structures.

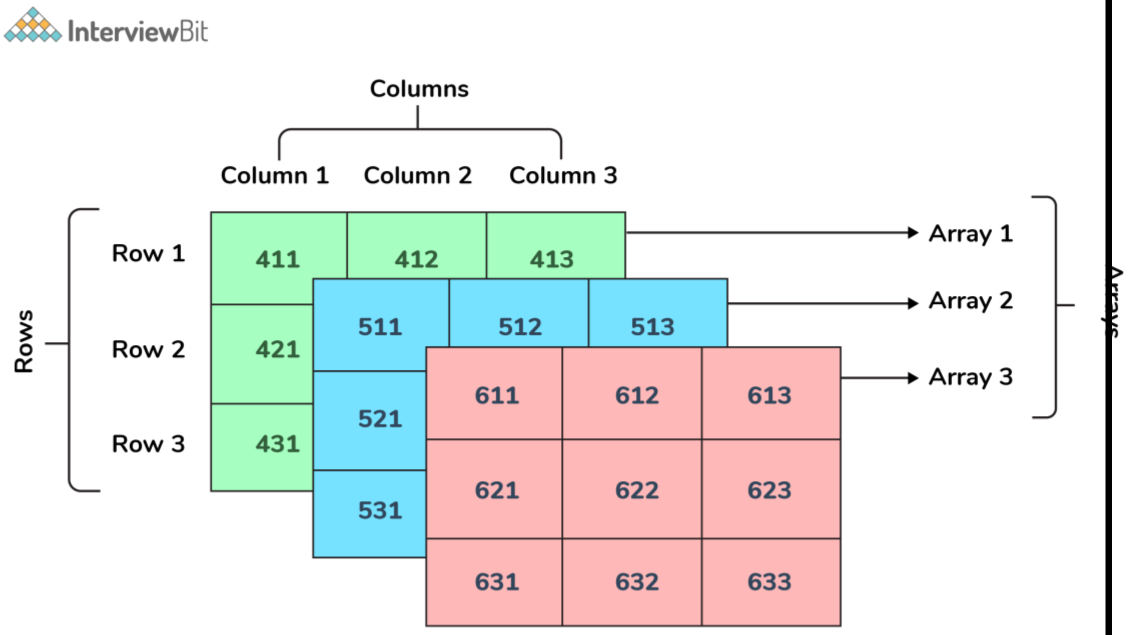

Elements are traversed sequentially
1
4
5
6
7
Linear Data Structure
InterviewBit
A
B
F
E
C
Non Linear Data Structure
A node might be connected to multiple nodes. Hence, traversing sequentially is not always possible.

3. What is an array?

- Arrays are the collection of **similar** types of data stored at **contiguous** memory locations.
- It is the simplest data structure where the data element can be accessed randomly just by using its index number.

4. What is a multidimensional array?

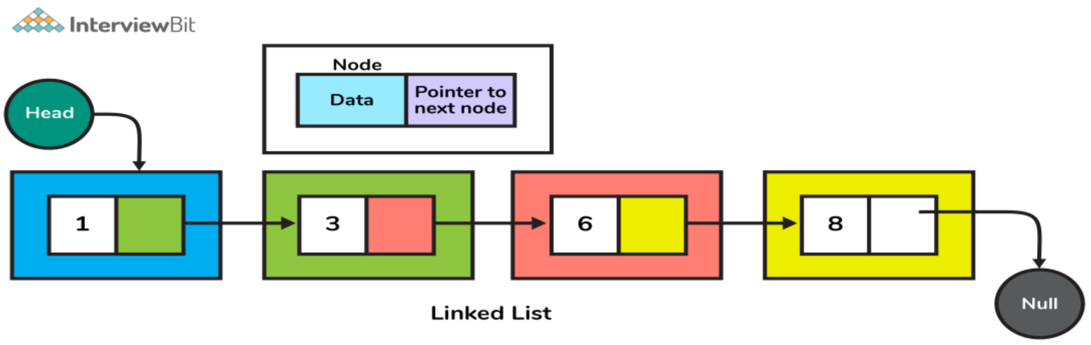
- Multi-dimensional arrays are those data structures that span across more than one dimension.
- This indicates that there will be more than one index variable for every point of storage. This type of data structure is primarily used in cases where data cannot be represented or stored using only one dimension. Most commonly used multidimensional arrays are 2D arrays.
    - 2D arrays emulates the tabular form structure which provides ease of holding the bulk of data that are accessed using row and column pointers.



5. What is a linked list?

A linked list is a data structure that has **sequence of nodes** where every node is connected to the next node by means of a reference pointer. The elements are **not stored in adjacent** memory locations. They are linked using pointers to form a chain. This forms a chain-like link for data storage.

- Each node element has two parts:
  - a data field
  - a reference (or pointer) to the next node.
- The first node in a linked list is called the head and the last node in the list has the pointer to NULL. Null in the reference field indicates that the node is the last node. When the list is empty, the head is a null reference.



6. Are linked lists of linear or non-linear type?

Linked lists can be considered both linear and non-linear data structures. This depends upon the application that they are used for.

- When linked list is used for access strategies, it is considered as a linear data-structure. When they are used for data storage, it can be considered as a non-linear data structure.

7. How are linked lists more efficient than arrays?

1. **Insertion and Deletion**
    - Insertion and deletion process is expensive in an array as the room has to be created for the new elements and existing elements must be shifted.
    - But in a linked list, the same operation is an easier process, as we only update the address present in the next pointer of a node.
2. **Dynamic Data Structure**
    - Linked list is a dynamic data structure that means there is no need to give an initial size at the time of creation as it can grow and shrink at runtime by allocating and deallocating memory.
    - Whereas, the size of an array is limited as the number of items is statically stored in the main memory.
3. **No wastage of memory**
    - As the size of a linked list can grow or shrink based on the needs of the program, there is no memory wasted because it is allocated in runtime.
    - In arrays, if we declare an array of size 10 and store only 3 elements in it, then the space for 3 elements is wasted. Hence, chances of memory wastage is more in arrays.

8. Explain the scenarios where you can use linked lists and arrays.

- Following are the scenarios where we use linked list over array:
    - When we do not know the exact number of elements beforehand.
    - When we know that there would be large number of add or remove operations.

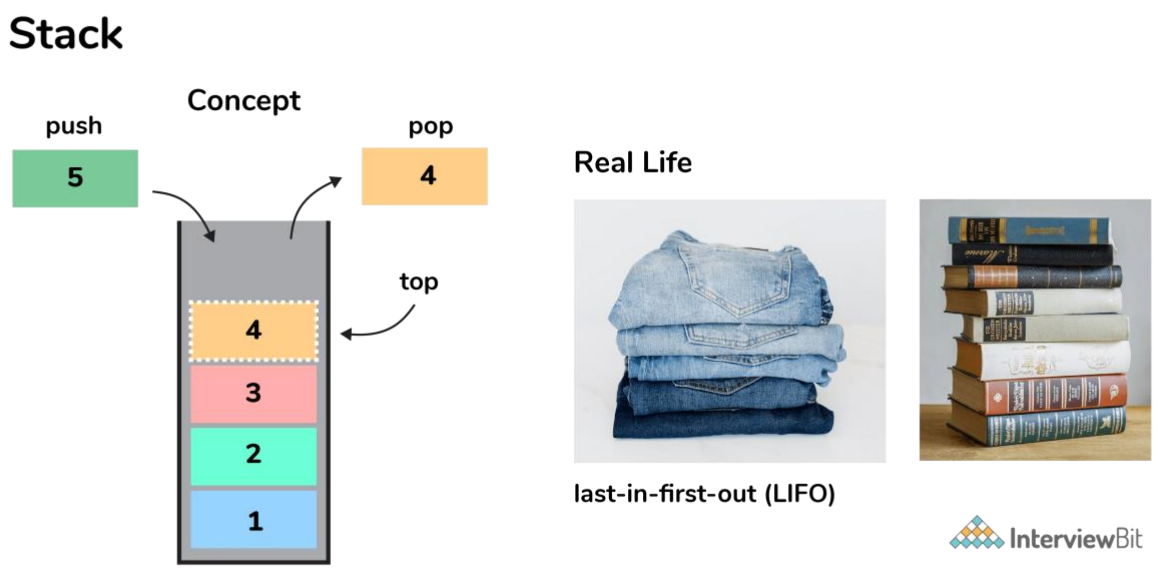
    - Less number of random access operations.
    - When we want to insert items anywhere in the middle of the list, such as when implementing a priority queue, linked list is more suitable.
- Below are the cases where we use arrays over the linked list:
    - When we need to index or randomly access elements more frequently.
    - When we know the number of elements in the array beforehand in order to allocate the right amount of memory.
    - When we need speed while iterating over the elements in the sequence.
    - When memory is a concern:
        0. Due to the nature of arrays and linked list, it is safe to say that filled arrays use less memory than linked lists.
        1. Each element in the array indicates just the data whereas each linked list node represents the data as well as one or more pointers or references to the other elements in the linked list.
- To summarize, requirements of space, time, and ease of implementation are considered while deciding which data structure has to be used over what.

9. What is a doubly-linked list (DLL)? What are its applications.

- This is a complex type of a linked list wherein a node has two references:
    - One that connects to the next node in the sequence
    - Another that connects to the previous node.
- This structure allows traversal of the data elements in both directions (left to right and vice versa).
- Applications of DLL are:
    - A music playlist with next song and previous song navigation options.
    - The browser cache with BACK-FORWARD visited pages
    - The undo and redo functionality on platforms such as word, paint etc, where you can reverse the node to get to the previous page.

10. What is a stack? What are the applications of stack?

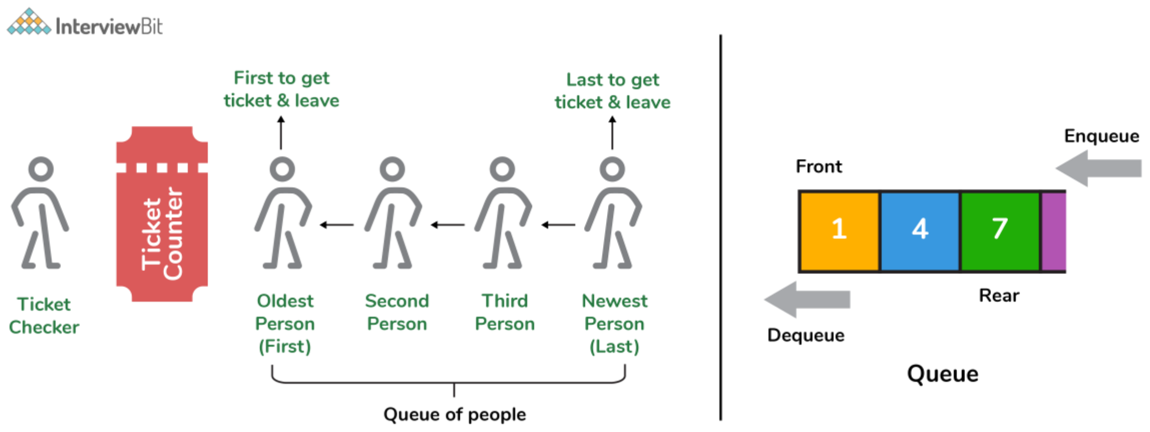
- Stack is a linear data structure that follows LIFO (Last In First Out) approach for accessing elements.
- Push, pop, and top (or peek) are the basic operations of a stack.



- Following are some of the applications of a stack:
    - Check for balanced parentheses in an expression
    - Evaluation of a postfix expression
    - Problem of Infix to postfix conversion
    - Reverse a string

11. What is a queue? What are the applications of queue?

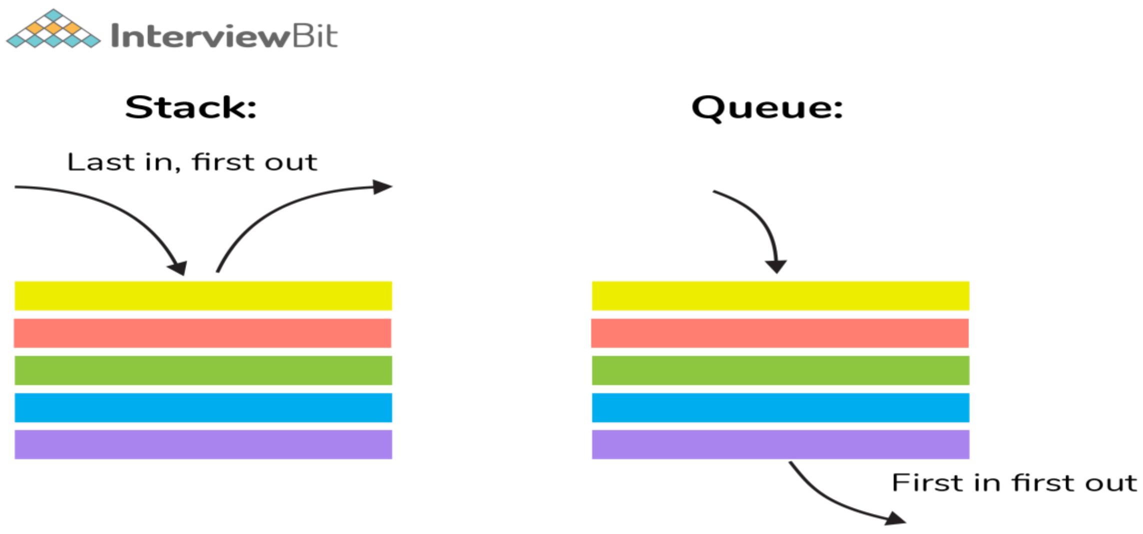
- A queue is a linear data structure that follows the FIFO (First In First Out) approach for accessing elements.
- Dequeue from the queue, enqueue element to the queue, get front element of queue, and get rear element of queue are basic operations that can be performed.



- Some of the applications of queue are:
    - CPU Task scheduling
    - BFS algorithm to find shortest distance between two nodes in a graph.
    - Website request processing
    - Used as buffers in applications like MP3 media player, CD player, etc.
    - Managing an Input stream

12. How is a stack different from a queue?

- In a stack, the item that is most recently added is removed first whereas in queue, the item least recently added is removed first.



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

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

14. How to implement a queue using stack?

- A queue can be implemented using **two stacks**. Let q be the queue andstack1 and stack2 be the 2 stacks for implementing q. We know that stack supports push, pop, peek operations and using these operations, we need to emulate the operations of queue - enqueue and dequeue. Hence, queue q can be implemented in two methods (Both the methods use auxillary space complexity of O(n)):

    0. **By making enqueue operation costly:**
        - Here, the oldest element is always at the top of stack1 which ensures dequeue operation to occur in O(1) time complexity.
        - To place element at top of stack1, stack2 is used.
        - **Pseudocode:**
            - Enqueue: Here time complexity will be O(n)

```
enqueue(q, data):
  While stack1 is not empty:
      Push everything from stack1 to stack2.
      Push data to stack1
      Push everything back to stack1.
```

- Dequeue: Here time complexity will be O(1)

```
deQueue(q):
  If stack1 is empty then error
```

```
else
    Pop an item from stack1 and return it
```

1. **By making dequeue operation costly:**

- Here, for enqueue operation, the new element is pushed at the top of stack1. Here, the enqueue operation time complexity is O(1).
- In dequeue, if stack2 is empty, all elements from stack1 are moved to stack2 and top of stack2 is the result. Basically, reversing the list by pushing to a stack and returning the first enqueued element. This operation of pushing all elements to new stack takes O(n) complexity.
- **Pseudocode:**
  - Enqueue: Time complexity: O(1)

```
enqueue(q, data):
    Push data to stack1
```

  - Dequeue: Time complexity: O(n)

```
dequeue(q):
    If both stacks are empty then raise error.
    If stack2 is empty:
        While stack1 is not empty:
            push everything from stack1 to stack2.
    Pop the element from stack2 and return it.
```

15. How do you implement stack using queues?

- A stack can be implemented using two queues. We know that a queue supports enqueue and dequeue operations. Using these operations, we need to develop push, pop operations.
- Let stack be ‘s’ and queues used to implement be ‘q1’ and ‘q2’. Then, stack ‘s’ can be implemented in two ways:

0. **By making push operation costly:**

- This method ensures that newly entered element is always at the front of ‘q1’, so that pop operation just dequeues from ‘q1’.

- 'q2' is used as auxillary queue to put every new element at front of 'q1' while ensuring pop happens in O(1) complexity.
- **Pseudocode:**
    - Push element to stack s : Here push takes O(n) time complexity.

```
push(s, data):
    Enqueue data to q2
    Dequeue elements one by one from q1 and enqueue to q2.
    Swap the names of q1 and q2
```

    - Pop element from stack s: Takes O(1) time complexity.

```
pop(s):
    dequeue from q1 and return it.
```

1. **By making pop operation costly:**

- In push operation, the element is enqueued to q1.
- In pop operation, all the elements from q1 except the last remaining element, are pushed to q2 if it is empty. That last element remaining of q1 is dequeued and returned.
- **Pseudocode:**
    - Push element to stack s : Here push takes O(1) time complexity.

```
push(s,data):
    Enqueue data to q1
```

    - Pop element from stack s: Takes O(n) time complexity.

```
pop(s):
    Step1: Dequeue every elements except the last element fro
m q1 and enqueue to q2.
    Step2: Dequeue the last item of q1, the dequeued item is st
ored in result variable.
    Step3: Swap the names of q1 and q2 (for getting updated d
ata after dequeue)
```

- Step4: Return the result.

16. What is hashmap in data structure?

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

17. What is the requirement for an object to be used as key or value in HashMap?

3. The key or value object that gets used in hashmap must implement equals() and hashcode() method.

4. The hash code is used when inserting the key object into the map and equals method is used when trying to retrieve a value from the map.

18. How does HashMap handle collisions in Java?

5. The java.util.HashMap class in Java uses the approach of chaining to handle collisions. In chaining, if the new values with same key are attempted to be pushed, then these values are stored in a linked list stored in bucket of the key as a chain along with the existing value.

6. In the worst case scenario, it can happen that all key might have the same hashcode, which will result in the hash table turning into a linked list. In this case, searching a value will take O(n) complexity as opposed to O(1) time due to the nature of the linked list. Hence, care has to be taken while selecting hashing algorithm.

19. What is the time complexity of basic operations get() and put() in HashMap class?

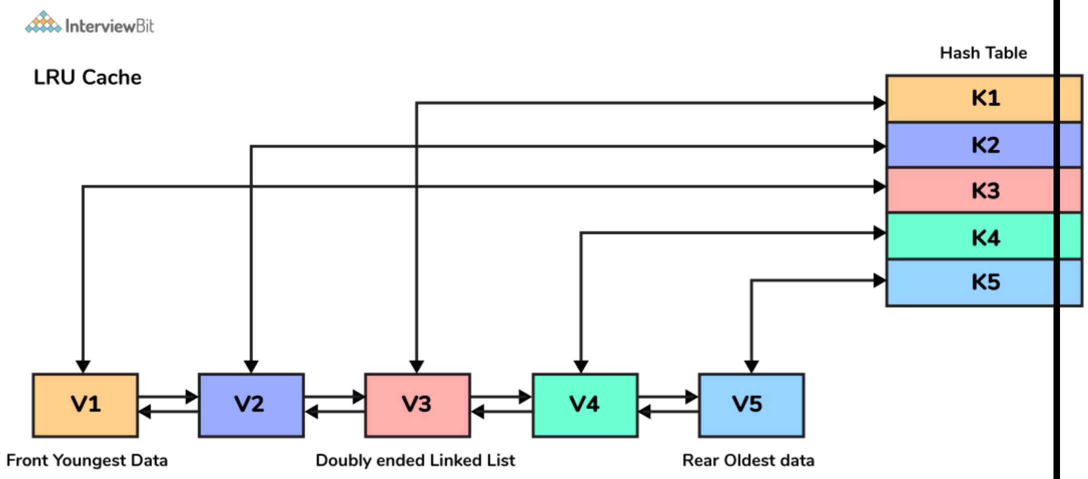
7. The time complexity is O(1) **assuming** that the hash function used in hash map distributes elements uniformly among the buckets.

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

8. 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.



21. What is a priority queue?

9. A priority queue is an abstract data type that is like a normal queue but has priority assigned to elements.
10. Elements with higher priority are processed before the elements with a lower priority.
11. In order to implement this, a minimum of two queues are required - one for the data and the other to store the priority.

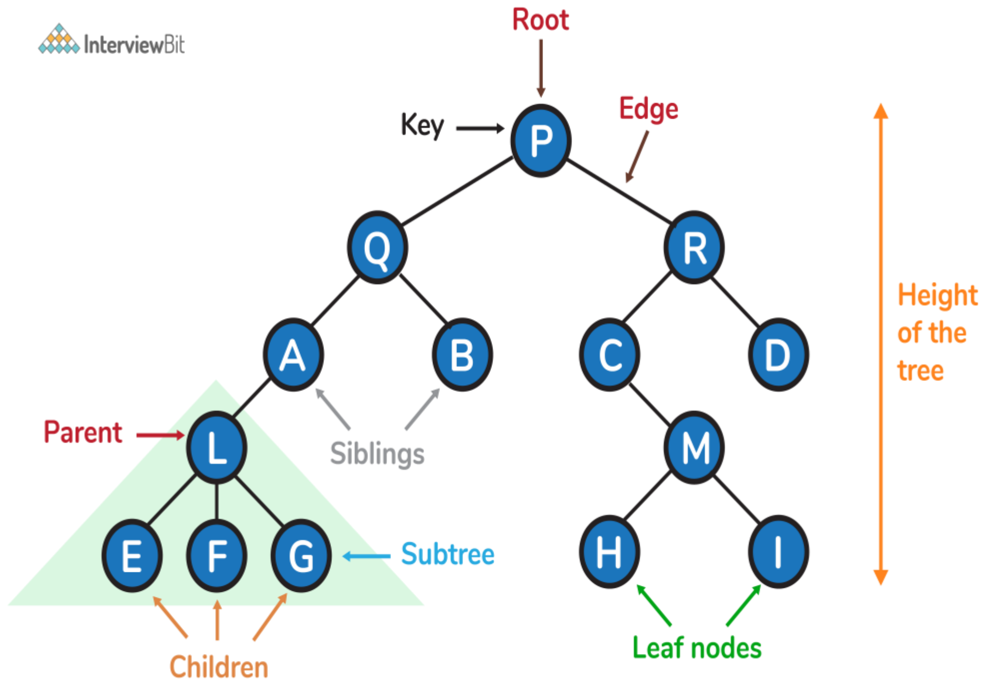
22. Can we store a duplicate key in HashMap?

12. **No**, duplicate keys cannot be inserted in HashMap. If you try to insert any entry with an existing key, then the old value would be overridden with the new value. Doing this will not change the size of HashMap.

    - This is why the keySet() method returns all keys as a SET in Java since it doesn't allow duplicates.

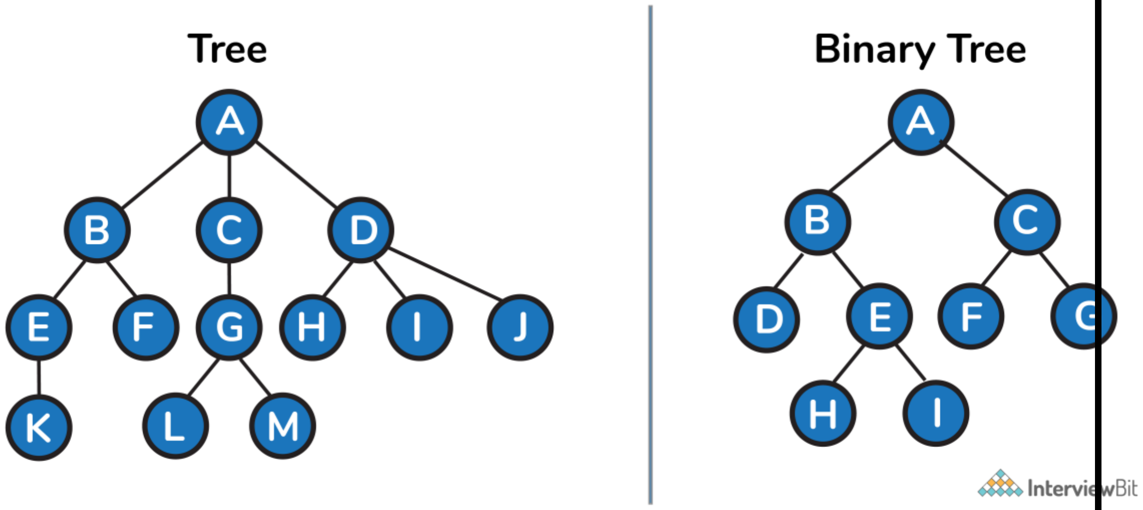
23. What is a tree data structure?

13. Tree is a recursive, non-linear data structure consisting of the set of one or more data nodes where one node is designated as the root and the remaining nodes are called as the children of the root.
14. Tree organizes data into hierarchial manner.
15. The most commonly used tree data structure is a binary tree and its variants.
16. Some of the applications of trees are:
    - **Filesystems** —files inside folders that are inturn inside other folders.
    - **Comments on social media** — comments, replies to comments, replies to replies etc form a tree representation.
    - **Family trees** — parents, grandparents, children, and grandchildren etc that represents the family hierarchy.



24. What are Binary trees?

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



25. What is the maximum number of nodes in a binary tree of height k?

- The maximum nodes are : $2^{k+1}-1$ where k >= 1

28. What are tree traversals?

- Tree traversal is a process of visiting all the nodes of a tree. Since root (head) is the first node and all nodes are connected via edges (or links) we always start with that node. There are three ways which we use to traverse a tree −
  - **Inorder Traversal:**
    - Algorithm:
      - Step 1. Traverse the left subtree, i.e., call Inorder(root.left)
      - Step 2. Visit the root.
      - Step 3. Traverse the right subtree, i.e., call Inorder(root.right)
    - Inorder traversal in Java:

```
// Print inorder traversal of given tree.
void printInorderTraversal(Node root)
```

```
    {
        if (root == null)
            return;

        //first traverse to the left subtree
        printInorderTraversal(root.left);

        //then print the data of node
        System.out.print(root.data + " ");

        //then traverse to the right subtree
        printInorderTraversal(root.right);
    }
```

- Uses: In binary search trees (BST), inorder traversal gives nodes in ascending order.

1. **Preorder Traversal:**
    - Algorithm:
        - Step 1. Visit the root.
        - Step 2. Traverse the left subtree, i.e., call Preorder(root.left)
        - Step 3. Traverse the right subtree, i.e., call Preorder(root.right)
    - Preorder traversal in Java:

```
    // Print preorder traversal of given tree.
    void printPreorderTraversal(Node root)
    {
        if (root == null)
            return;
        //first print the data of node
        System.out.print(root.data + " ");

        //then traverse to the left subtree
        printPreorderTraversal(root.left);

        //then traverse to the right subtree
        printPreorderTraversal(root.right);
    }
```

- Uses:
  - Preorder traversal is commonly used to create a copy of the tree.
  - It is also used to get prefix expression of an expression tree.

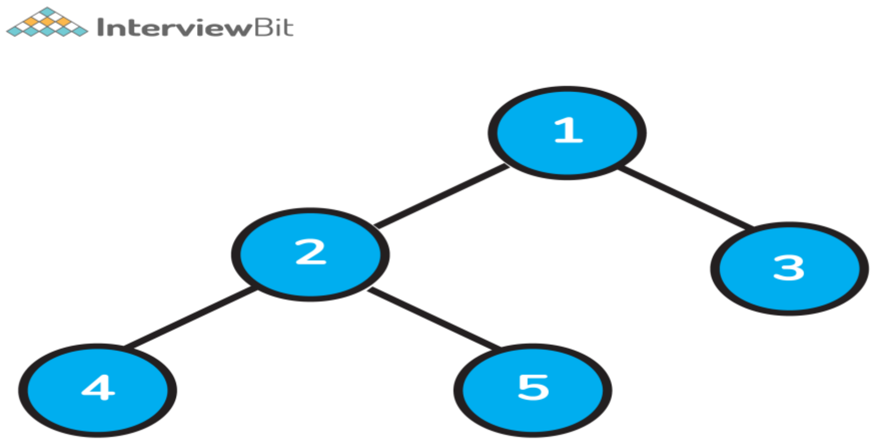
2. **Postorder Traversal:**

- Algorithm:
  - Step 1. Traverse the left subtree, i.e., call Postorder(root.left)
  - Step 2. Traverse the right subtree, i.e., call Postorder(root.right)
  - Step 3. Visit the root.
- Postorder traversal in Java:

```
// Print postorder traversal of given tree.
void printPostorderTraversal(Node root)
{
  if (root == null)
      return;

  //first traverse to the left subtree
  printPostorderTraversal(root.left);
•
•     //then traverse to the right subtree
•     printPostorderTraversal(root.right);
•
•     //then print the data of node
•     System.out.print(root.data + " ");
•
•   }
```

- Uses:
  - Postorder traversal is commonly used to delete the tree.
  - It is also useful to get the postfix expression of an expression tree.

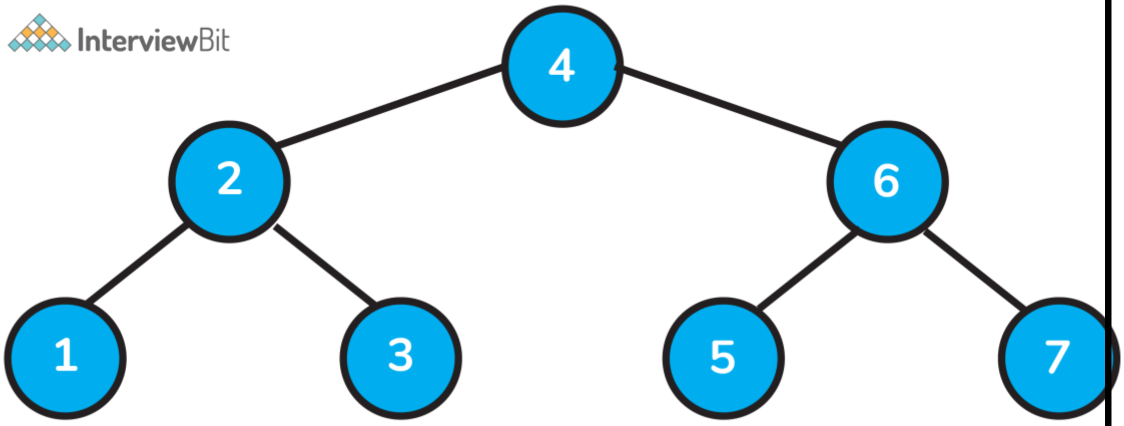
- Consider the following tree as an example, then:



- Inorder Traversal => Left, Root, Right : [4, 2, 5, 1, 3]
- Preorder Traversal => Root, Left, Right : [1, 2, 4, 5, 3]
- Postorder Traversal => Left, Right, Root : [4, 5, 2, 3, 1]

29. What is a Binary Search Tree?

- A binary search tree (BST) is a variant of binary tree data structure that stores data in a very efficient manner such that the values of the nodes in the left sub-tree are less than the value of the root node, and the values of the nodes on the right of the root node are correspondingly higher than the root.
- Also, individually the left and right sub-trees are their own binary search trees at all instances of time.



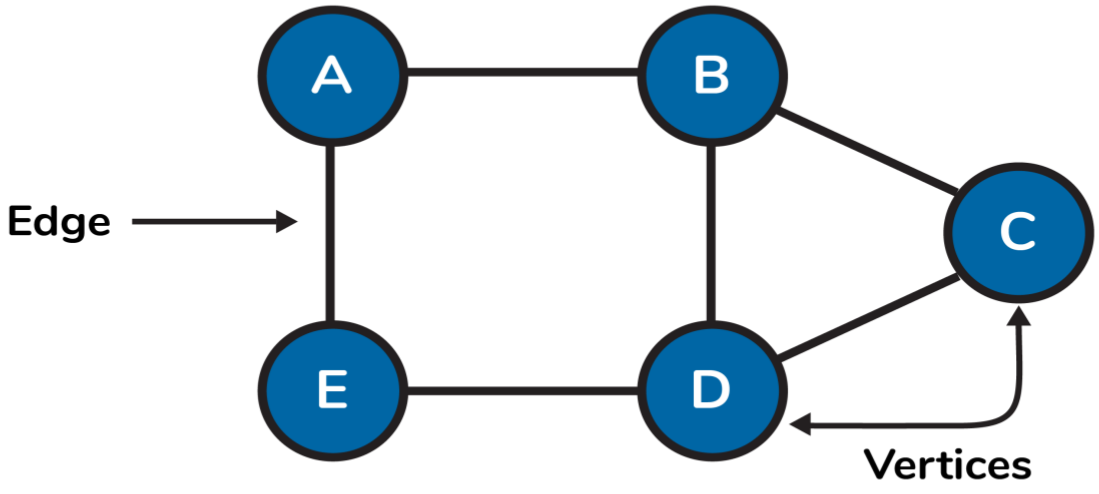
Example of Binary Search Tree

30. What is an AVL Tree?

- AVL trees are **height balancing** BST. AVL tree checks the height of left and right sub-trees and assures that the difference is **not more than 1**. This difference is called Balance Factor and is calculates as. BalanceFactor = height(left subtree) − height(right subtree)
-

32. What is a graph data structure?

Graph is a type of non-linear data structure that consists of vertices or nodes connected by edges or links for storing data. Edges connecting the nodes may be directed or undirected.



33. What are the applications of graph data structure?

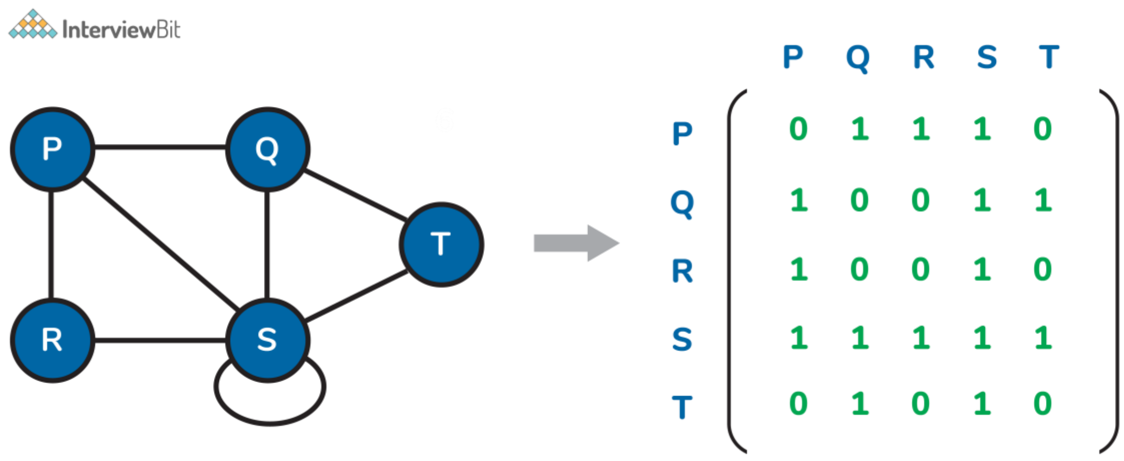
Graphs are used in wide varieties of applications. Some of them are as follows:

25. Social network graphs to determine the flow of information in social networking websites like facebook, linkedin etc.
26. Neural networks graphs where nodes represent neurons and edge represent the synapses between them
27. Transport grids where stations are the nodes and routes are the edges of the graph.
28. Power or water utility graphs where vertices are connection points and edge the wires or pipes connecting them.
29. Shortest distance between two end points algorithms.

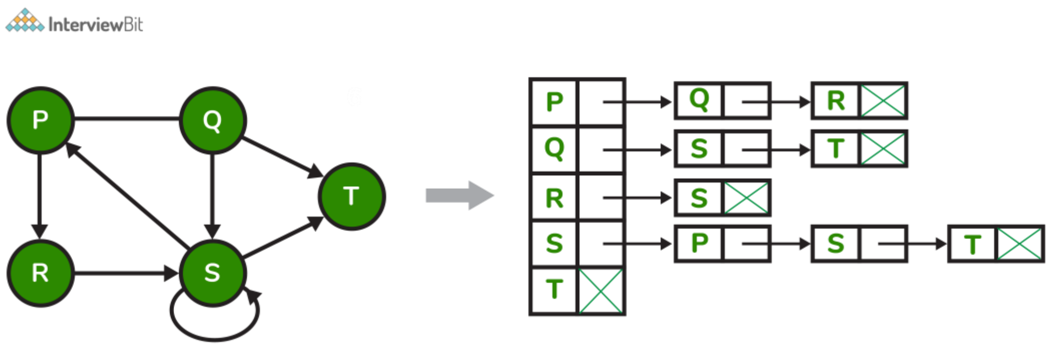
34. How do you represent a graph?

We can represent a graph in 2 ways:

- Adjacency matrix: Used for sequential data representation



- Adjacency list: Used to represent linked data



35. What is the difference between tree and graph data structure?

32. Tree and graph are differentiated by the fact that a tree structure must be connected and can never have loops whereas in the graph there are no restrictions.
33. Tree provides insights on relationship between nodes in a hierarchical manner and graph follows a network model.

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

34. BFS and DFS both are the traversing methods for a graph. Graph traversal is nothing but the process of visiting all the nodes of the graph.
35. The main difference between BFS and DFS is that BFS traverses level by level whereas DFS follows first a path from the starting to the end node, then another path from the start to end, and so on until all nodes are visited.
36. Furthermore, BFS uses queue data structure for storing the nodes whereas DFS uses the stack for traversal of the nodes for implementation.
37. DFS yields deeper solutions that are not optimal, but it works well when the solution is dense whereas the solutions of BFS are optimal.
38. You can learn more about BFS here: Breadth First Search and DFS here: Depth First Search.

37. How do you know when to use DFS over BFS?

- The usage of DFS heavily depends on the structure of the search tree/graph and the number and location of solutions needed. Following are the best cases where we can use DFS:
    - If it is known that the solution is not far from the root of the tree, a breadth first search (BFS) might be better.
    - If the tree is very deep and solutions are rare, depth first search (DFS) might take an extremely long time, but BFS could be faster.
    - If the tree is very wide, a BFS might need too much memory, so it might be completely impractical. We go for DFS in such cases.
    - If solutions are frequent but located deep in the tree we opt for DFS.

38. What is topological sorting in a graph?

- Topological sorting is a linear ordering of vertices such that for every directed edge ij, vertex i comes before j in the ordering.
- Topological sorting is only possible for Directed Acyclic Graph (DAG).
- Applications:
    0. jobs scheduling from the given dependencies among jobs.

1. ordering of formula cell evaluation in spreadsheets
2. ordering of compilation tasks to be performed in make files,
3. data serialization
4. resolving symbol dependencies in linkers.

40. What is a heap data structure?

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 towards as left as possible. Heaps are of two types:

43. Max-Heap:
    - In a Max-Heap the data element present at the root node must be greatest among all the data elements present in the tree.
    - This property should be recursively true for all sub-trees of that binary tree.
44. 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.

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.

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.

1) What is Data Structure? Explain.

The data structure is a way that specifies how to organize and manipulate the data. It also defines the relationship between them. Some examples of Data Structures are arrays, Linked List, Stack, Queue, etc. Data Structures are the central part of many computer science algorithms as they enable the programmers to handle the data in an efficient way

2) Describe the types of Data Structures?

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) List the area of applications of Data Structure.

Data structures are applied extensively in the following areas of computer science:

- Compiler Design,
- Operating System,
- Database Management System,
- Statistical analysis package,
- Numerical Analysis,
- Graphics,
- Artificial Intelligence,
- Simulation

4) What is the difference between file structure and storage structure?

Difference between file structure and storage structure:

The main difference between file structure and storage structure is based on memory area that is being accessed.

Storage structure: It is the representation of the data structure in the computer memory.

File structure: It is the representation of the storage structure in the auxiliary memory.

5) List the data structures which are used in RDBMS, Network Data Modal, and Hierarchical Data Model.

- RDBMS uses Array data structure
- Network data model uses Graph
- Hierarchal data model uses Trees

6) Which data structure is used to perform recursion?

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

7) What is a Stack?

Stack is an ordered list in which, insertion and deletion can be performed only at one end that is called the top. It is a recursive data structure having pointer to its top element. The stack is sometimes called as Last-In-First-Out (LIFO) list i.e. the element which is inserted first in the stack will be deleted last from the stack.

8) List the area of applications where stack data structure can be used?

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

9) What are the operations that can be performed on a stack?

- Push Operations
- Pop Operations
- Peek Operations

10) Write the stack overflow condition.

Overflow occurs when **top = Maxsize -1**

11) What is the difference between PUSH and POP?

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.

12) Write the steps involved in the insertion and deletion of an element in the stack.

Push:

- Increment the variable top so that it can refer to the next memory allocation
- Copy the item to the at the array index value equal to the top
- Repeat step 1 and 2 until stack overflows

Pop:

- Store the topmost element into the an another variable

- Decrement the value of the top
- Return the topmost element

13) What is a postfix expression?

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.

14)Write the postfix form of the expression: (A + B) * (C - D)

AB+CD-*

15) Which notations are used in Evaluation of Arithmetic Expressions using prefix and postfix forms?

Polish and Reverse Polish notations.

16)What is an array?

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.

17) How to reference all the elements in a one-dimension array?

It can be done by using an indexed loop such that the counter runs from 0 to the array size minus one. In this manner, you can reference all the elements in sequence by using the loop counter as the array subscript.

18) What is a multidimensional array?

The multidimensional array can be defined as the array of arrays in which, the data is stored in tabular form consists of rows and columns. 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

19) How are the elements of a 2D array are stored in the memory?

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.

20) Calculate the address of a random element present in a 2D array, given base address as BA.

Row-Major Order: If array is declared as a[m][n] where m is the number of rows while n is the number of columns, then address of an element a[i][j] of the array stored in row major order is calculated as,

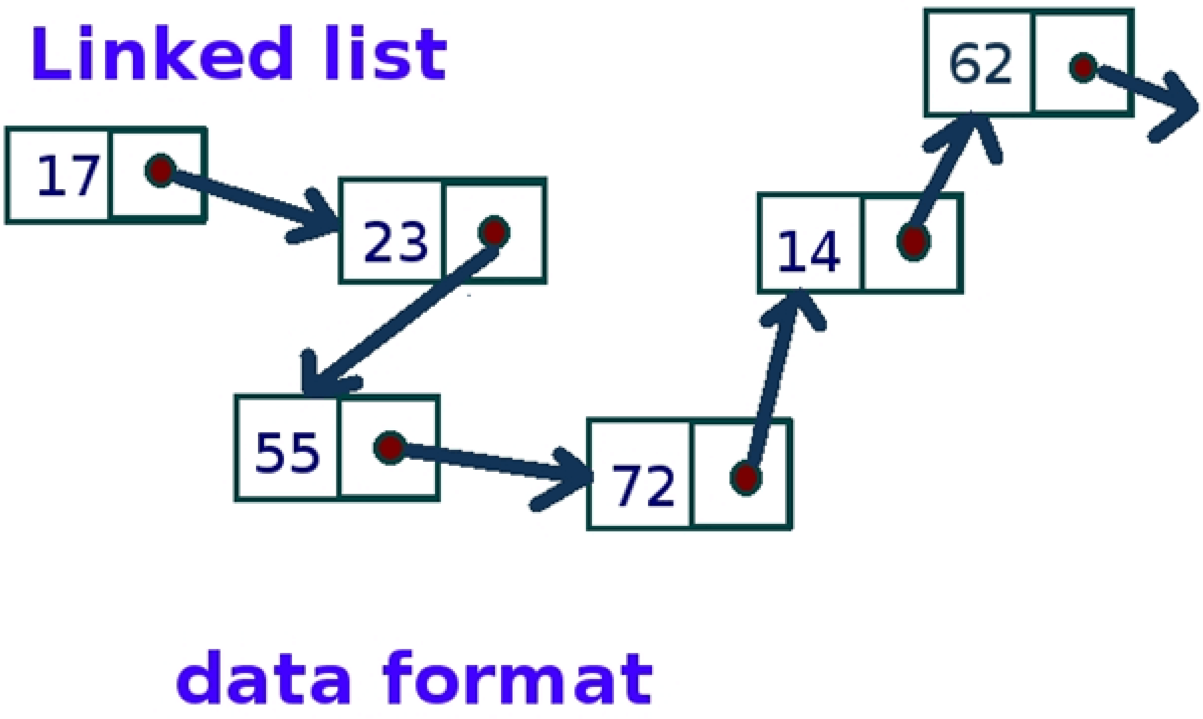
Address(a[i][j]) = B. A. + (i * n + j) * size

Column-Major Order: If array is declared as a[m][n] where m is the number of rows while n is the number of columns, then address of an element a[i][j] of the array stored in column major order is calculated as

Address(a[i][j]) = ((j*m)+i)*Size + BA.

21) Define Linked List Data structure.

Linked List is the collection of randomly stored data objects called nodes. In Linked List, each node is linked to its adjacent node through a pointer. A node contains two fields, i.e. Data Field and Link Field.



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

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.

23) What are the advantages of Linked List over an array?

- 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.

24) Write the syntax in C to create a node in the singly linked list.

```
struct node
{
    int data;
    struct node *next;
};
struct node *head, *ptr;
ptr = (struct node *)malloc(sizeof(struct node));
```

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

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.

26) What is doubly linked list?

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).

28) Define the queue data structure.

A queue can be defined as an ordered list which enables insert operations to be performed at one end called REAR and delete operations to be performed at another end called FRONT.

29) List some applications of queue data structure.

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.

30) What are the drawbacks of array implementation of Queue?

- **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.

31) What are the scenarios in which an element can be inserted into the circular queue?

- If (rear + 1)%maxsize = front, the queue is full. In that case, overflow occurs and therefore, insertion can not be performed in the queue.
- If rear != max - 1, the rear will be incremented to the mod(maxsize) and the new value will be inserted at the rear end of the queue.
- If front != 0 and rear = max - 1, it means that queue is not full therefore, set the value of rear to 0 and insert the new element there.

32) What is a dequeue?

Dequeue (also known as double-ended queue) can be defined as an ordered set of elements in which the insertion and deletion can be performed at both the ends, i.e. front and rear.

33) What is the minimum number of queues that can be used to implement a priority queue?

Two queues are needed. One queue is used to store the data elements, and another is used for storing priorities.

34) Define the tree data structure.

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.

35) List the types of tree.

There are six types of tree given as follows.

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

36) What are Binary trees?

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.

37) Write the C code to perform in-order traversal on a binary tree.

```
void in-order(struct treenode *tree)
  {
    if(tree != NULL)
    {
      in-order(tree→ left);
      printf("%d",tree→ root);
      in-order(tree→ right);
    }
  }
```

38) What is the maximum number of nodes in a binary tree of height k?

2^{k+1}-1 where k >= 1

39) Which data structure suits the most in the tree construction?

Queue data structure

40) Which data structure suits the most in the tree construction?

Queue data structure

41) Write the recursive C function to count the number of nodes present in a binary tree.

```
int count (struct node* t)
{
  if(t)
  {
    int l, r;
    l = count(t->left);
    r=count(t->right);
    return (1+l+r);
  }
```

```
    else
    {
        return 0;
    }
}

```

42) Write a recursive C function to calculate the height of a binary tree.

```
int countHeight(struct node* t)
{
    int l,r;
    if(!t)
        return 0;
    if((!(t->left)) && (!(t->right)))
        return 0;
    l=countHeight(t->left);
    r=countHeight(t->right);
    return (1+((l>r)?l:r));
}
```

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

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.

44) State the properties of B Tree.

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.

45) What are the differences between B tree and B+ tree?

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 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.

46) List some applications of Tree-data structure?

Applications of Tree- data structure:

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

47) Define the graph data structure?

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.

48) Differentiate among cycle, path, and circuit?

- **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.

49) Mention the data structures which are used in graph implementation.

For the graph implementation, following data structures are used.

- In sequential representation, Adjacency matrix is used.
- In Linked representation, Adjacency list is used.

50) Which data structures are used in BFS and DFS algorithm?

- In BFS algorithm, Queue data structure is used.
- In DFS algorithm, Stack data structure is used.

51) What are the applications of Graph data structure?

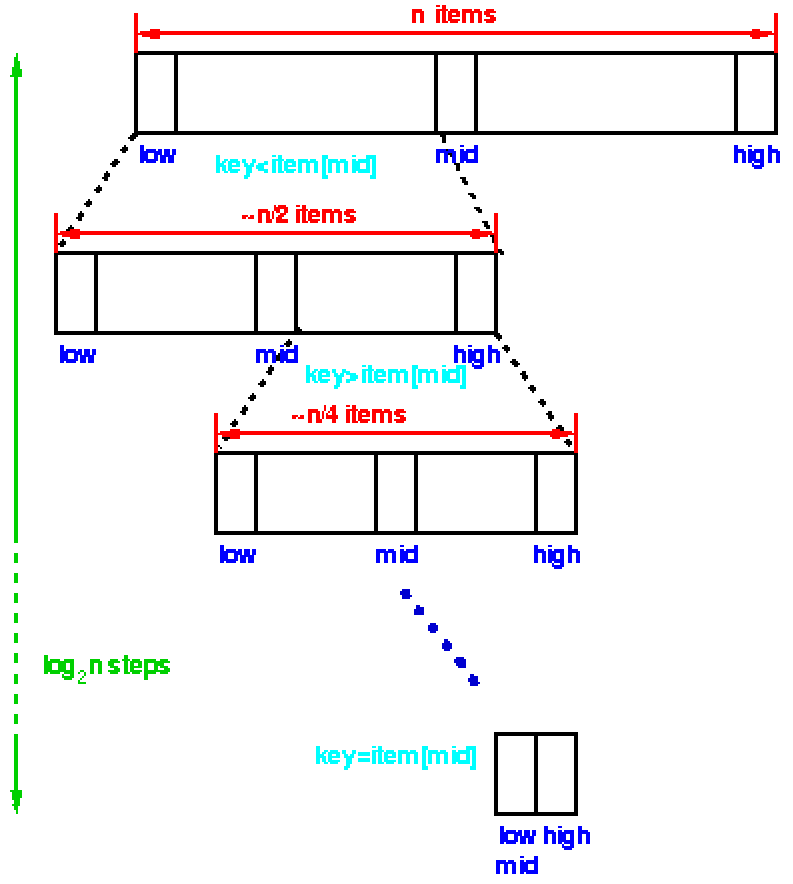
The graph has the following applications:

- 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.

54) In what scenario, Binary Search can be used?

Binary Search algorithm is used to search an already sorted list. The algorithm follows divide and conqer approach

Example:



52) What are the advantages of Binary search over linear search?

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.

53) What are the advantages of Selecetion Sort?

- 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.

55) List Some Applications of Multilinked Structures?

- Sparse matrix,
- Index generation.

56) What is the difference between NULL and VOID?

- 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.

1) What is data structure?

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

2) Differentiate between file and structure storage structure.

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

3) When is a binary search best applied?

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

4) What is a linked list?

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

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

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

6) In what areas do data structures are applied?

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

7) What is LIFO?

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

8) What is a queue?

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

9) What are binary trees?

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

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

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

11) What is a stack?

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

12) Explain Binary Search Tree

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

13) What are multidimensional arrays?

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

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

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

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

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

16) What is FIFO?

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

17) What is an ordered list?

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

18) What is merge sort?

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

19) Differentiate NULL and VOID

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

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

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

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

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

22) What is a linear search?

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

23) How does variable declaration affect memory allocation?

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

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

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

25) What is a postfix expression?

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

26) What is Data abstraction?

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

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

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

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

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

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

29) How do signed and unsigned numbers affect memory?

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

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

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

31) What are dynamic data structures?

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

32) In what data structures are pointers applied?

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

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

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

34) What are ARRAYs?

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

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

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

36) Which sorting algorithm is considered the fastest?

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

37) Differentiate STACK from ARRAY.

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

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

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

39) What is a dequeue?

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

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

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

41) What are the parts of a linked list?

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

42) How does selection sort work?

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

43) What is a graph?

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

44) Differentiate linear from a nonlinear data structure.

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

45) What is an AVL tree?

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

46) What are doubly linked lists?

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

47) What is Huffman's algorithm?

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

48) What is Fibonacci search?

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

49) Briefly explain recursive algorithm.

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

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

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

1. What are data structures?

A data structure is a way of storing and organization of data values for further use in an application. Examples include Graph, Trees, Arrays, Linked List, etc.

2. Name different types of data structures?

Data structures are of two types:

Linear	Array, Stack, Queue, Linked-List
Non-Linear	Graphs, Trees

3. What is the use of dynamic Data Structures?

Dynamic data structures are flexible and size changes can occur during insertion or deletion operations. Dynamic data structures play a key role in programming because they provide the programmer with the flexibility to adjust the memory consumption of programs.

4. Name various operations that can be performed in DSA.

- Insert: Here, we add a new data item in a data structure.
- Search: In this, we find an element in the data structure.

- Sort: Simple sorting like arranging values in ascending or descending order. For example, arranging values in an array.
- Delete: Here, we delete unwanted data points in a data structure.
- Traversal: We access each data item exactly once so that it can be processed for further use.

5. What are Infix, prefix, Postfix notations?

- Infix: We write expressions in infix notation, e.g. x+y - z, where operators are used in-between operands. For humans, it goes well, but for computing devices, it's not preferred.
- Prefix: Here, the operator is prefixed to operands, i.e. operator is written ahead of operands. For example, +xy instead of x+y.
- Postfix: Here, we use the operator after the operands. For example, X*Y is written as XY*.

6. What are the types of searching used in Data Structures?

- Linear Search.
- Binary Search.
- Jump Search.
- Interpolation Search.
- Exponential Search.

7. What is an array?

An array is a collection of data points of the same type stored at contiguous memory locations.

For example, an array of integers like 1,2,3,4,5. This array has 5 elements.

The index of an array usually starts with 0.

8. What are dynamic arrays?

A dynamic array is an array with a modification that is automatic resizing. This means that the array expands when we add more elements. This helps to provide flexibility as size is not fixed. So, there's no need to know the size in advance.

9. **Name some characteristics of Array Data Structure.**

- Array elements are stored in contiguous memory blocks in the primary memory.
- Array name represents its base address.
- The base address is the address of the first element of the array.
- Array's index starts with 0 and ends with size-1.

10. What is a linked list?

This is similar to an array but it consists of a node that has two fields:

- Data Field: For storing data values.
- Reference Field: For storing address.

A linked list is used in the next feature of a music player.

11. What are the different types of linked lists?

Ans: The different types of linked lists are:

- Singly Linked list
- Doubly Linked list
- Circular Linked list
- Doubly Circular Linked list

12. What are trees in DSA?

A collection of nodes is a tree. A node is a data point connected with other points with the help of edges. The top of a tree is the root node. Applications of trees include indexing in databases.

13. What are graphs and their uses?

A graph is a collection of nodes connected to one another via edges. It forms a network of nodes like in the case of a journey from one source to a destination.

Uses:

- Google Maps
- Linkedin and Facebook etc..

14. What's the difference between the data structure Tree and Graph?

Ans: A tree structure is connected and can never have loops whereas in the graph there are no such constraints. A tree provides information about the relationship between nodes in a hierarchical manner and the graph follows a network model.

15. What is the LIFO and FIFO principle?

LIFO: (Last-In-First-Out) Last inserted element is removed first in the LIFO principle. Example: Stack follows LIFO.

FIFO: (First-In-First-Out) First Element (first to be inserted) is taken out first.

Example: Queue follows FIFO

16. What is a queue?

A queue in English is a line. Similarly, in DSA, it is a line of data values that follows the FIFO (First in First out) principle. Insertion is done at the rear end and deletion at the front end.

17. What is a priority queue?

A priority queue is like a normal queue of elements but here each element has some priority. The elements in the priority queue occur in this order only.

Say, we have some values like 1, 2, 7, 8, 14, 31 inserted in a priority queue with an ordering based on least values to the greatest value. Therefore, the 1 number will have the highest priority while 31 will have the lowest priority.

18. What is a stack?

A stack is a linear data structure having values that are inserted as per the LIFO principle. To make the point more clear, imagine a pile of chairs. You want to add another chair, for that, you will have to add the chair at the top. That’s a stack. Insertion or deletion takes place at the top of the stack of data values.

19. State the difference between stack and queue

Stack	Queue
Insertion and deletion in stacks take place only from one end of the list called the top.	Insertion and deletion in queues take place from the opposite ends of the list. The insertion takes place at the rear end of the queue and the deletion takes place from the front end.
Insert operation is called push in a stack.	Insert operation is called enqueue operation.
Delete operation is called pop in a stack.	Delete operation is called dequeue operation.

20. What is a heap in DSA?

A heap data structure is a complete binary tree that follows a specific order. Heaps are of two types:

1. Max Heap
2. Min Heap

21. What is a binary heap?

- A binary heap is a complete binary tree that satisfies the heap ordering property.
- A Binary Heap can either be Min Heap or Max Heap.
- It’s a complete tree, thus it is suitable for being stored in an array.

22. What is the meaning of the AVL tree?

An AVL tree is a type of binary search tree. It is named after its inventors Adelson, Velskii, and Landis. An AVL tree is a balanced binary search tree where the height of the two subtrees of a node differs by a maximum of one unit.

23. What do you mean by Hash Table?

A Hash table is a data structure used to store data points in an associative manner. The values are stored in an array format. Hash tables are used to store keys/value pairs

24. What is the complexity of a Hash Table?

Hash tables provide constant-time O(1) lookup on average, irrespective of the number of items(n) in the table.

25. What Data Structures make use of pointers?

- Stack
- Queue
- Linked List
- Binary Tree.

26. What is a dequeue?

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

27. What is the meaning of the stack overflow condition?

When a stack is completely full and we try to insert more elements onto the stack then this condition is called stack overflow condition. Here, top=maxsize-1, and no further elements can be inserted.

28. What is the postfix form of (X + Y) * (Z - C)

The postfix form of the given expression is XY+ZC-*

29. What is a Balanced Tree and why is that important?

A tree is perfectly height-balanced if the left and right subtrees of any node are of the same height. We can also say that a tree is height-balanced if the heights of the left and right subtrees of each node differ by a maximum of one unit.

30. What are the Data Structures that are used to represent graphs?

Adjacency matrix , Adjacency list And adjacency set.

31. What operations can be performed on stacks?

- push() - Adds an element at the top of the stack.
- pop() - Deletes an element from the top of the stack.
- peek()- Displays the topmost element.

32. Why use queues?

Queues are important in computer science. They are useful in transport, and operations research. They are also useful in a case where resource sharing takes place among a myriad of consumers. For example, FCFS scheduling.

33. What is linear search?

Linear search is a technique in which we traverse a list in a sequential manner to find an element. When we find the element that is required, we return the index or position of that element.

34. What is binary search?

In this type of search technique, we divide the sorted array or list into two halves. This is done to save time in searching. Here, we consider arrays in sorted form only.

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

Linear Search-O(N) Binary Search- O(log 2 N)

36. Tell me about tree traversal.

Tree traversal is a process that goes through the entire tree in a particular manner. Depending upon the order of traversal, we have different types:

- Inorder Traversal (Left, Root, Right)
- Preorder Traversal (Root, Left, Right)
- Postorder Traversal (Left, Right, Root)

37. How does Kruskal's algorithm work?

Kruskal algorithm treats a graph as a forest and every node as an individual tree. A tree connects to another only and only if it has the least cost among all available choices without violating Minimum Spanning Tree (MST) properties.

38. How does Prim's algorithm find a spanning tree?

Prim's algorithm treats each node as a single tree and continues to add new nodes to the spanning tree from the given graph.

39. What is a minimum spanning tree (MST)?

It is a spanning tree that has the minimum weight among all the spanning trees of the same graph.

40. Explain the Tower of Hanoi Problem.

1. The Tower of Hanoi is a problem that comprises three rods and multiple disks.
2. At the start, all the disks are placed on one rod, one over the other in increasing order.
3. The aim of this problem is to move the stack of disks from the starting rod to another rod, following these rules as below:

- A disk cannot be placed on top of a smaller disk
- No disk can be placed on top of the smaller disk.

41. What are recursive algorithms?

Recursion in general is a function calling itself. Extending the same logic for Algorithms, we can say that an algorithm that calls itself is a recursive algorithm. One good example of their use would be searching through a file system. Usually, they are used when the iterative approach is not useful for complex problems.

42. Explain why stack is a recursive data structure?

A stack is a recursive data structure because:

- A stack can either be empty or
- It will have a top pointer and the rest part apart from the top is also a stack by itself, thus it’s recursive.

43. What is merge sort time complexity?

The time complexity of MergeSort is O(n*log n)

44. What is shell sort?

Shell sort is a sorting algorithm based on the insertion sort algorithm.

This algorithm is highly efficient in sorting. This algorithm tries to avoid large shifts as in the case of insertion sort if the smaller value is to the far right and has to be moved to the far left.

45. What is quicksort time complexity?

Worst-case time complexity is O(n^2)

46. What is a Red-Black Tree?

A red-black tree is a binary tree that has nodes represented by two colors: red and black. The tree follows specific properties.

These include:

1. The root node of the tree is always black.
2. Every path from the root to any of the leaf nodes should have the same number of black nodes.
3. No two red nodes can be adjacent to each other.

47. When does the worst case of QuickSort occur?

It occurs when the picked pivot is an extreme (smallest or largest) element. Usually, when the input array is sorted or reverse sorted, it also leads to the worst case.

48. **Which data structures are used for the BFS and DFS of a graph?**

- For BFS - Queue Data Structure
- For DFS - Stack Data Structure

49. What do you mean by BFS and DFS?

BFS Algorithm stands for Breadth-First Search. It is a vertex-based technique for finding the shortest path in the graph. For BFS, we use a Queue data structure.

DFS Algorithm stands for Depth First Search. It is an edge-based technique. In DFS, traversal can be started from any node. For DFS, we use a stack data structure.

50. What is the maximum number of nodes in a binary tree of height k?

Ans: The maximum nodes in a binary tree are: 2k+1-1 where k >= 1.

1. What is the structure of the data?

A data structure is a technique to organize the data to make efficient use of the data. Various types of data structures are suitable for various types of applications, some being extremely specialized. B-trees are, for example, especially ideal for database implementation, but compiler implementations often utilize hash tables to look for identifiers.

2. What is a linked list?

A list linked is a sequence of nodes where every node is linked to the node after it. This establishes a connection for data storage like a chain.

3. Describe the Data Structures types?

In particular, data structures are divided into two types:

Linear Data Structure: If all its elements are arranged in a sequential sequence, a data structure is termed linear. The items are saved non-hierarchically in linear data structures, with each item having successors and predecessors other than the first and last element.

Non-Linear Data Structure: The non-linear data structure is not a sequence, i.e. each item or element in a non-linear arrangement is associated with two or more other factors. In the sequence structure, the data elements are not ordered.

4. How different from the array is the Linked List?

Array-linked list differences are –

1) Arrays are index data structures in which an index is linked to each element. On the other hand, the list of linked elements relies on references where the data and references of the previous and next elements are included in each node.

2) Arrays have their size fixed, lists linked have size dynamic.

3) Randomly accessed by index in array elements while in sequence accessible items in the linked list.

4) Insert and delete operations in the array are costly, but in the linked list they are readily done.

5) The linked list requires extra memory space for the next node storage address.

6) During the compile-time in the array, the item placement should be assigned while during run time in the linked list.

5. What is the difference between a linear data structure and a nonlinear data structure?

If a sequence or a linear list is formed with data structure elements, then a linear data structure is termed. Non-linear data structures, on the other hand, are ones in which non-linear transversal of nodes occurs.

Arrays, linked lists, ports and queues are examples of linear data structures and nonlinear data structure graphs and trees.

6. What Is the Difference Between a Stack and An Array?

STACK:

i) Stack is a collection of articles arranged.

ii) Stack is a dynamic object whose size changes steadily when things are pressed and popped.

(iii) Stack may have many sorts of data.

iv) Stack is specified as a structure with an array that holds the stack element, and an integer that specifies the stack top in the array.

ARRAY:

i) Array is a group of articles ordered.

ii) Array is an object static, i.e. no item is fixed, and the array statement assigns it.

iii) It includes the same kinds of data.

iv) A pile array, i.e. a pile array can be declared large enough to have the maximum stack size.

7. Can we get a sorted Linked list using the binary search data structure algorithm?

No, the binary data structure algorithm cannot be used for a sorted linked list since it is impossible to determine an index of the middle member.

8. What is an asymptotic analysis of a data structure algorithm?

The mathematical binding/framing of its run-time functioning refers to the asymptotic analysis of a data structure algorithm. We can very easily infer the best, average and worst-case case scenario of a data structure algorithm for the data structure with asymptotic analysis.

9. Explain several Linked List kinds.

Types of Linked List:

1. Singly Linked List: Each node saves address or reference to the next node in the list with the following node or reference as NULL in this form of linked list. For instance, 1->2->3->4-> NULL
2. Doubly Linked List: Here are two references for each node, one for the next and one for the prior nodes. The other for the next. For instance, zero. <-1<->2<->3-> NULL
3. Circular Linked List: Circular list linked is a list with links that connect all the nodes into a circle. No NULL at the end. At the end. A round linked list may be a single circular linked list or a round linked list. For example, 1->2->3->1. [The final node's next pointer points to the first]

10. What operations on data structures can be performed?

After operations can be carried out –

Insertion: Adding a new data item.

Deletion: Deleting the existing data item.

Traversal: Accessing each data item.

Searching: Finding a particular data item.

Sorting: Arranging the data item in a particular sequence.

11. What is a queue?

A queue is a data structure capable of simulating a list or data stream. New elements at one end are put in this structure and existing elements at the other.

12. What does a binary search mean to you?

A binary search is a data structure algorithm for the data structure that starts from the central search. If the middle element is not the target element, it checks further if the lower half of the higher half is to continue to be searched. The procedure continues till the end of the item.

Do you know how dynamic data management supports the allocation of memory?

Simple structured data types are stored with Dynamic Memory attribution. In addition, it may combine the structured blocks allotted individually to build composite structures which, when needed, shrink and grow.

13. Which data structure is appropriate for recurrence and why?

Stack is the perfect way to operate recursion. It recalls the components and their location, mainly because of its LIFO (Last in First Out) characteristics, so it knows precisely which one to return when a function is called.

14. What instances are there of data structure algorithms divided and conquered?

The following issues can be solved using the procedure of division and data structure −

- Merge Sort
- Quick Sort
- Binary Search
- Strassen’s Matrix Multiplication
- Closest pair (points)

15. How does depth-first traversal work?

Depth First Data Structure Algorithm traverses a graph in a deep movement and utilizes a stack to remember when a dead end occurs in every iteration, the next vertex to start a search.

1) Explain what is an algorithm in computing?

An algorithm is a well-defined computational procedure that take some value as input and generate some value as output. In simple words, it’s a sequence of computational steps that converts input into the output.

2) Explain what is Quick Sort algorithm?

Quick Sort algorithm has the ability to sort list or queries quickly. It is based on the principle of partition exchange sort or Divide and conquer. This type of algorithm occupies less space, and it segregates the list into three main parts.

- Elements less than the Pivot element
- Pivot element
- Elements greater than the Pivot element

3) Explain what is time complexity of Algorithm?

Time complexity of an algorithm indicates the total time needed by the program to run to completion. It is usually expressed by using the **big O notation.**

4) Mention what are the types of Notation used for Time Complexity?

The types of Notations used for Time Complexity includes

- **Big Oh:** It indicates “fewer than or the same as” <expression>iterations
- **Big Omega**: It indicates “more than or same as” <expression>iterations
- **Big Theta:** It indicates “the same as”<expression>iterations

- **Little Oh:** It indicates “fewer than” <expression>iterations
- **Little Omega:** It indicates “more than” <expression>iterations

5) Explain how binary search works?

In binary search, we compare the key with the item in the middle position of the array. If the key is less than the item searched then it must lie in the lower half of the array, if the key is greater than the item searched than it should be in upper half of the array.

6) Explain whether it is possible to use binary search for linked lists?

Since random access is not acceptable in linked list, it is impossible to reach the middle element of O(1) time. Thus, binary search is not possible for linked list.

7) Explain what is heap sort?

Heap-sort can be defined as a comparison based sorting algorithm. It divides its input into the unsorted and sorted region, until it shrinks the unsorted region by eliminating the smallest element and moving that to the sorted region.

8) Explain what is Skip list?

Skip list the method for data structuring, where it allows the algorithm to search, delete and insert elements in a symbol table or dictionary. In a skip list, each element is represented by a node. The search function returns the content of the value related to key. The insert operation associates a specified key with a new value, while the delete function deletes the specified key.

9) Explain what is Space complexity of insertion sort algorithm?

Insertion sort is an in-place sorting algorithm which means that it requires no extra or little. storage. For insertion sort, it requires only single list elements to be stored out-side the initial data, making the space-complexity 0(1).

10) Explain what a “Hash Algorithm” is and what are they used for?

“Hash Algorithm” is a hash function that takes a string of any length and decreases it to a unique fixed length string. It is used for password validity, message & data integrity and for many other cryptographic systems.

11) Explain how to find whether the linked list has a loop?

To know whether the linked list has a loop, we will take two pointer approach. If we maintain two pointers, and we increase one pointer after processing two nodes and other after processing every node, we are likely to encounter a situation where both the pointer will be pointing to the same node. This will only occur if linked list has a loop.

12) Explain how encryption algorithm works?

Encryption is the process of converting plaintext into a secret code format referred as “Ciphertext”. To convert the text, algorithm uses a string of bits referred as “keys” for calculations. The larger the key, the greater the number of potential patterns for creating cipher text. Most encryption algorithm use codes fixed blocks of input that have length about 64 to 128 bits, while some uses stream method.

13) List out some of the commonly used cryptographic algorithms?

Some of the commonly used cryptographic algorithms are

- 3-way
- Blowfish
- CAST
- CMEA
- GOST
- DES and Triple DES
- IDEA
- LOKI and so on

14) Explain what is the difference between best case scenario and worst case scenario of an algorithm?

- **Best case scenario:**Best case scenario for an algorithm is explained as the arrangement of data for which the algorithm performs best. For example, we take a binary search, for which the best case scenario would be if the target value is at the very center of the data you are searching. The best case time complexity would be 0 (1)
- **Worst case scenario:** It is referred for the worst set of input for a given algorithm. For example quicksort, which can perform worst if you select the largest or smallest element of a sublist for the pivot value. It will cause quicksort to degenerate to O (n2).

15) Explain what is Radix Sort algorithm?

Radix sort puts the element in order by comparing the digits of the numbers. It is one of the linear sorting algorithms for integers.

16) Explain what is a recursive algorithm?

Recursive algorithm is a method of solving a complicated problem by breaking a problem down into smaller and smaller sub-problems until you get the problem small enough that it can be solved easily. Usually, it involves a function **calling itself**.

17) Mention what are the three laws of recursion algorithm?

All recursive algorithm must follow three laws

- It should have a base case
- A recursive algorithm must call itself

- A recursive algorithm must change its state and move towards the base case

18) Explain what is bubble sort algorithm?

Bubble sort algorithm is also referred as sinking sort. In this type of sorting, the list to be sorted out compares the pair of adjacent items. If they are organized in the wrong order, it will swap the values and arrange them in the correct order.

1. How can we compare between two algorithms written for the same problem?

The complexity of an algorithm is a technique that is used to categorise how efficient it is in comparison to other algorithms. It focuses on how the size of the data set to be processed affects execution time. In computing, the algorithm's computational complexity is critical. It is a good idea to categorise algorithms according to how much time or space they take up and to describe how much time or space they take up as a function of input size.

- **Complexity of Time:** The running time of a program as a function of the size of the input is known as time complexity.
- **Complexity of Space:** Space complexity examines algorithms based on how much space they require to fulfil their tasks. In the early days of computers, space complexity analysis was crucial (when storage space on the computer was limited).

Note: Nowadays, a lack of space is rarely an issue because computer storage is plentiful. Therefore, it is mostly the Time Complexity that is given more importance while evaluating an Algorithm.

2. What do you understand by the best case, worst case and average case scenario of an algorithm?

The mathematical foundation/framing of an algorithm's run time performance is defined by asymptotic analysis. We can easily determine the best case, average case, and worst-case scenarios of an algorithm using asymptotic analysis.

- **Best Case Scenario of an Algorithm:** The best-case scenario for an algorithm is defined as the data arrangement in which the algorithm performs the best. Take a binary search, for example, where the best-case scenario is if the target value is in the very centre of the data we are looking for. The best-case scenario for binary search would have a time complexity of O(1) or constant time complexity.
- **Worst Case Scenario of an Algorithm:** The worst collection of input for a given algorithm is referred to as the worst-case scenario of an Algorithm. For example, quicksort can perform poorly if the pivot value is set to the largest or smallest element of a sublist. Quicksort will degenerate into an algorithm with a time complexity of O(n^2), where n is the size of the list to be sorted.
- **Average Case Scenario of an Algorithm:** The average-case complexity of an algorithm is the amount of some computational resource (usually time) used by the process, averaged over all possible inputs, according to computational

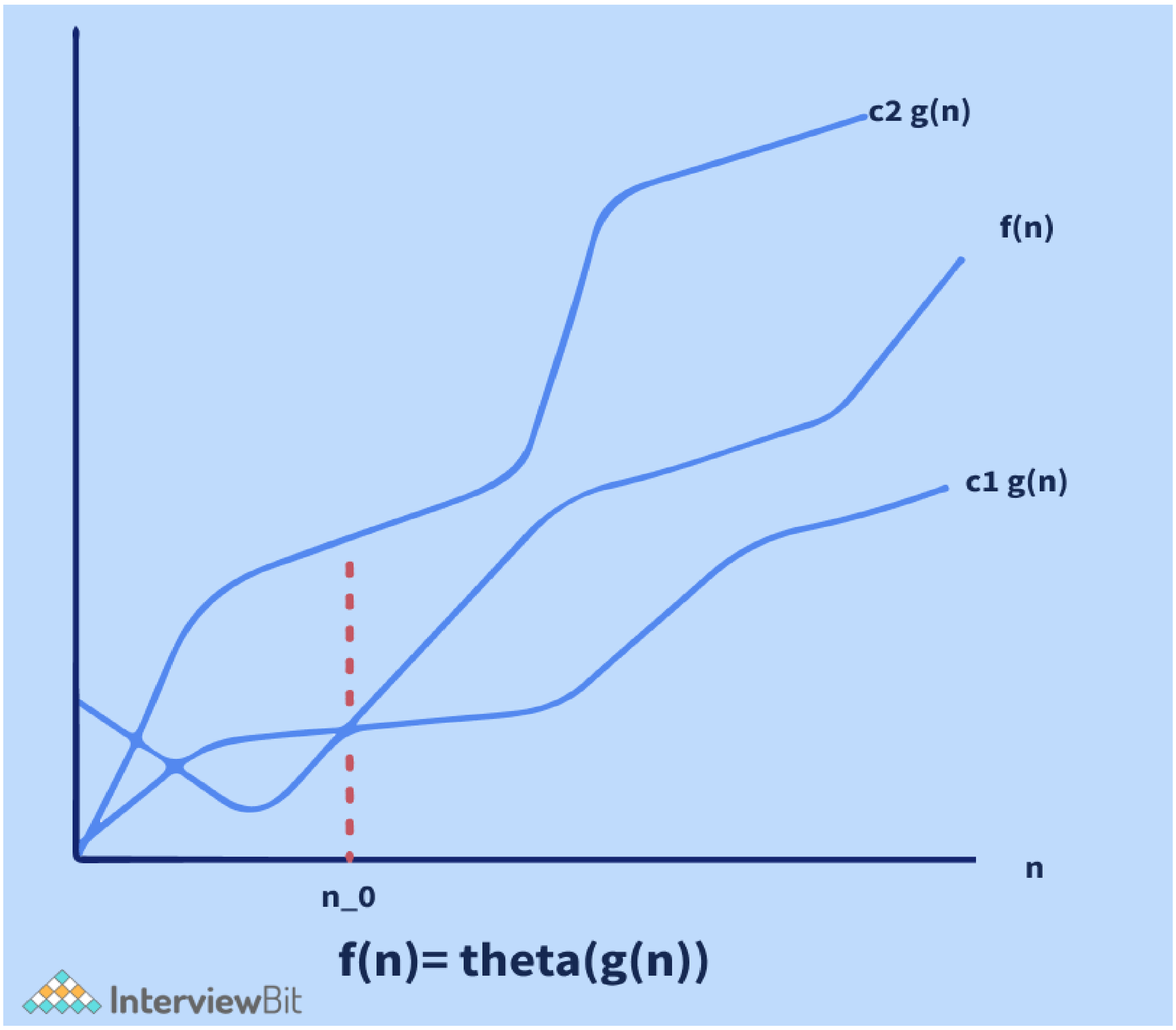
complexity theory. For example, the average-case complexity of the randomised quicksort algorithm is O(n*log(n)), where n is the size of the list to be sorted.

3. What do you understand by the Asymptotic Notations?

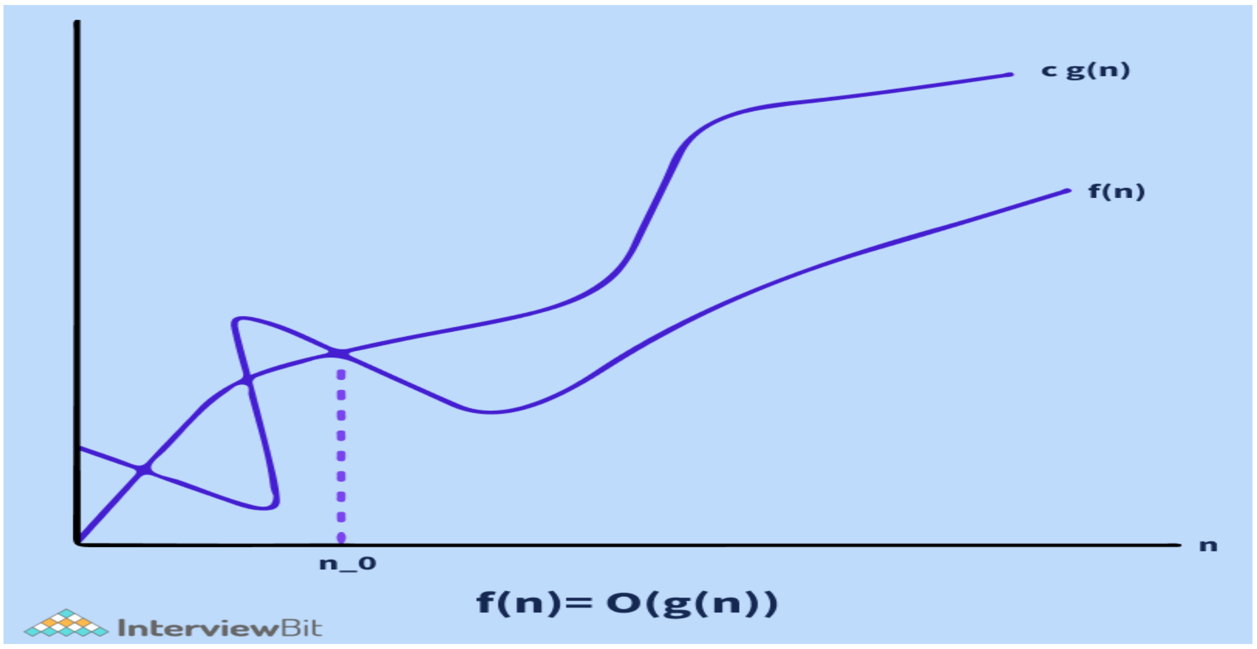
Asymptotic analysis is a technique that is used for determining the efficiency of an algorithm that does not rely on machine-specific constants and avoids the algorithm from comparing itself to the time-consuming approach. For asymptotic analysis, asymptotic notation is a mathematical technique that is used to indicate the temporal complexity of algorithms.

The following are the three most common asymptotic notations.

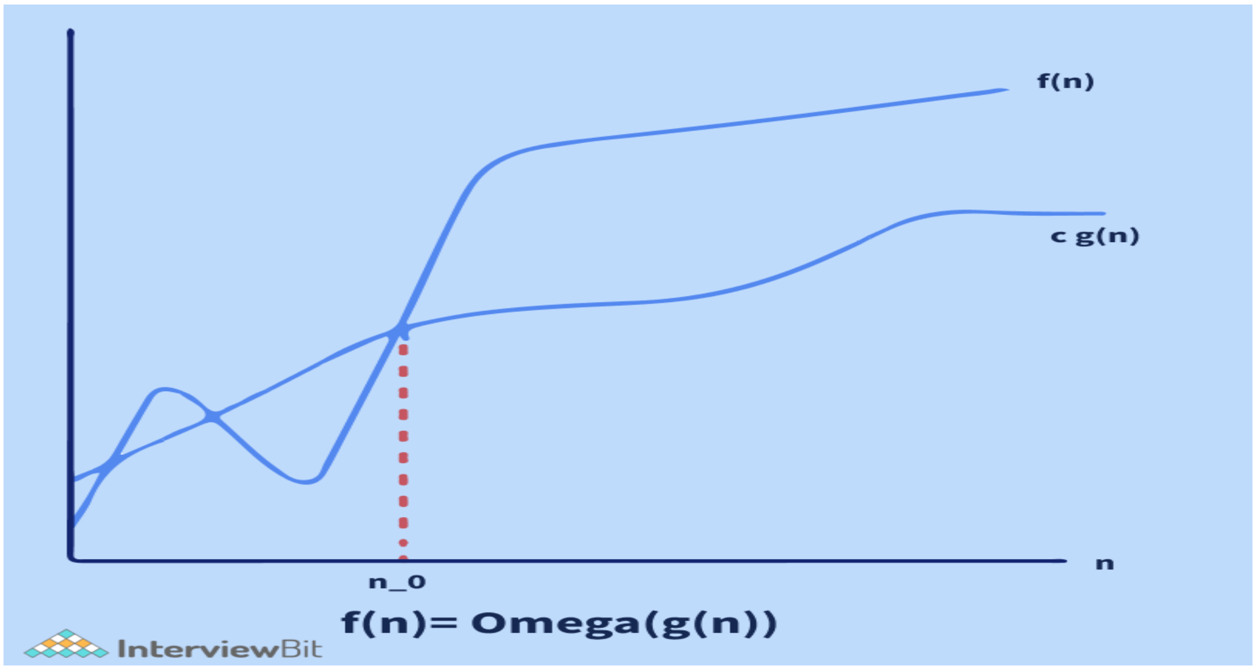
- **Big Theta Notation: (θ Notation)**
  The exact asymptotic behaviour is defined using the theta (θ) Notation. It binds functions from above and below to define behaviour. Dropping low order terms and ignoring leading constants is a convenient approach to get Theta notation for an expression.



- **Big O Notation:**
  The Big O notation defines an upper bound for an algorithm by bounding a function from above. Consider the situation of insertion sort: in the best case scenario, it takes linear time, and in the worst case, it takes quadratic time. Insertion sort has a time complexity O(n^2). It is useful when we just have an upper constraint on an algorithm's time complexity.



- **Big Omega (Ω) Notation:**
  The Ω Notation provides an asymptotic lower bound on a function, just like Big O notation does. It is useful when we have a lower bound on an algorithm's time complexity.



4. Write an algorithm to swap two given numbers in Java without using a temporary variable.

It is a trick question that is frequently asked in the interviews of various companies. This problem can be solved in a variety of ways. However, while solving the problem, we must solve it without using a temporary variable, which is an essential condition. For this problem, if we can consider the possibility of integer overflow in our solution while coming up with an approach to solving it, we can make a great impression on interviewers.

Let us say that we have two integers a and b, with a's value equal to 5 and a's value equal to 6, and we want to swap them without needing a third variable. We will need to use Java programming constructs to solve this problem. Mathematical procedures such as addition, subtraction, multiplication, and division can be used to swap numbers. However, it is possible that it will cause an integer overflow problem.

Let us take a look at two approaches to solve this problem:

Using Addition and subtraction:

```
a = a + b;

b = a - b; // this will act like (a+b) - b, and now b equals a.

a = a - b; // this will act like (a+b) - a, and now an equals b.
```

It is a clever trick. However, if the addition exceeds the maximum value of the int primitive type as defined by Integer.MAX_VALUE in Java, or if the subtraction is less than the minimum value of the int primitive type as defined by Integer.MIN_VALUE in Java, there will be an integer overflow.

Using the XOR method:

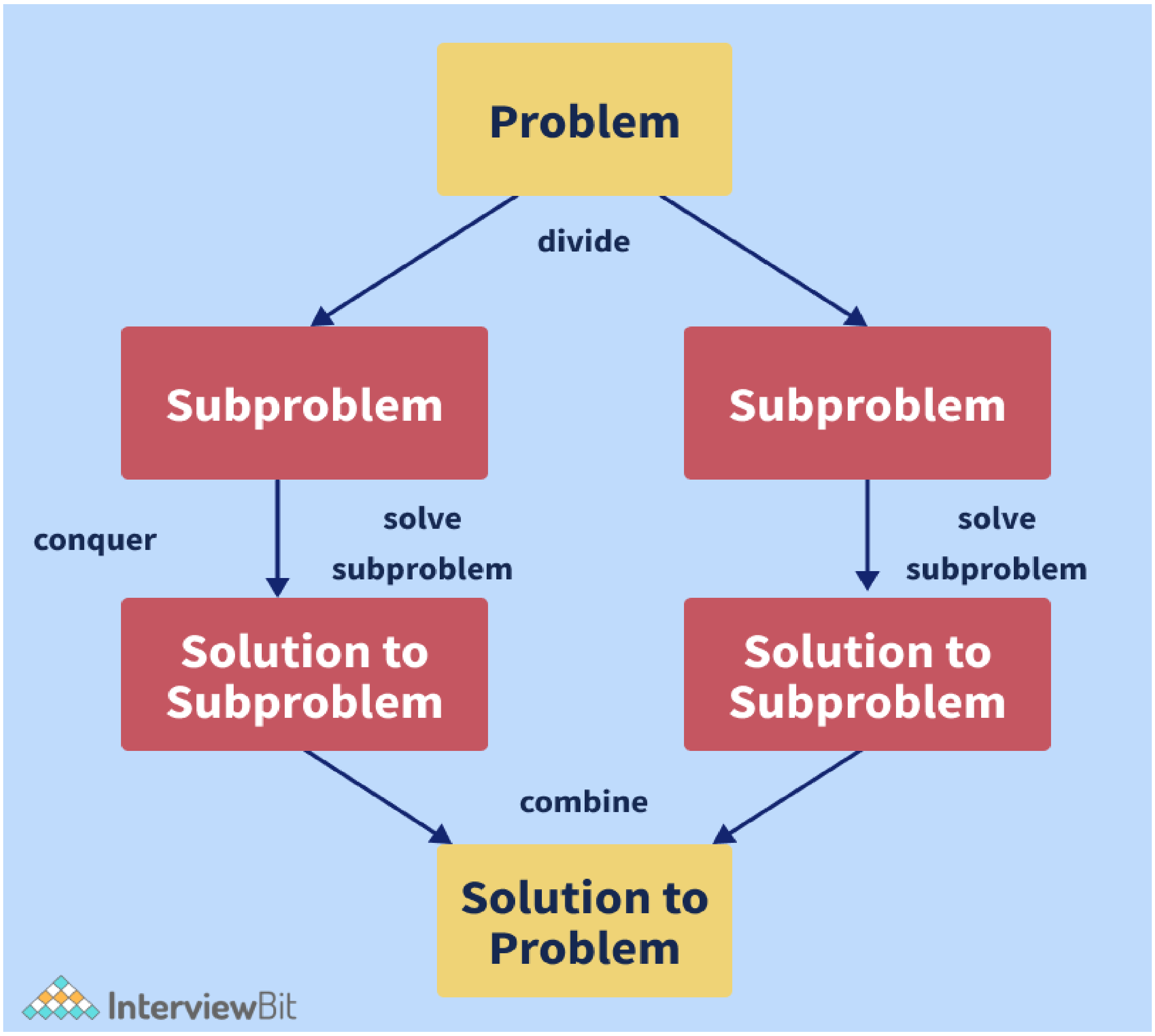
Another way to swap two integers without needing a third variable (temporary variable) is using the XOR method. This is often regarded as the best approach because it works in languages that do not handle integer overflows, such as Java, C, and C++. Java has a number of bitwise operators. XOR (denoted by ^) is one of them.

```
x = x ^ y;

y = x ^ y;

x = x ^ y;
```

5. Explain the Divide and Conquer Algorithmic Paradigm. Also list a few algorithms which use this paradigm.

Divide and Conquer is an algorithm paradigm, not an algorithm itself. It is set up in such a way that it can handle a large amount of data, split it down into smaller chunks, and determine the solution to the problem for each of the smaller chunks. It combines all of the piecewise solutions of the smaller chunks to form a single global solution. This is known as the divide and conquer technique. The Divide and Conquer algorithmic paradigm employ the steps given below:

- **Divide:** The algorithm separates the original problem into a set of subproblems in this step.
- **Conquer:** The algorithm solves each subproblem individually in this step.
- **Combine:** In this step, the algorithm combines the solutions to the subproblems to obtain the overall solution.



Some of the algorithms which use the Divide and Conquer Algorithmic paradigm are as follows:

- Binary Search
- Merge Sort
- Strassen's Matrix Multiplication
- Quick Sort
- Closest pair of points.

6. What do you understand about greedy algorithms? List a few examples of greedy algorithms.

A greedy algorithm is an algorithmic method that aims to choose the best optimal decision at each sub-step, eventually leading to a globally optimal solution. This means that the algorithm chooses the best answer available at the time, regardless of the consequences. In other words, when looking for an answer, an algorithm always selects the best immediate, or local, option. Greedy algorithms may identify less than perfect answers for some cases of other problems while finding the overall, ideal solution for some idealistic problems.

The Greedy algorithm is used in the following algorithms to find their solutions:

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

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

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:

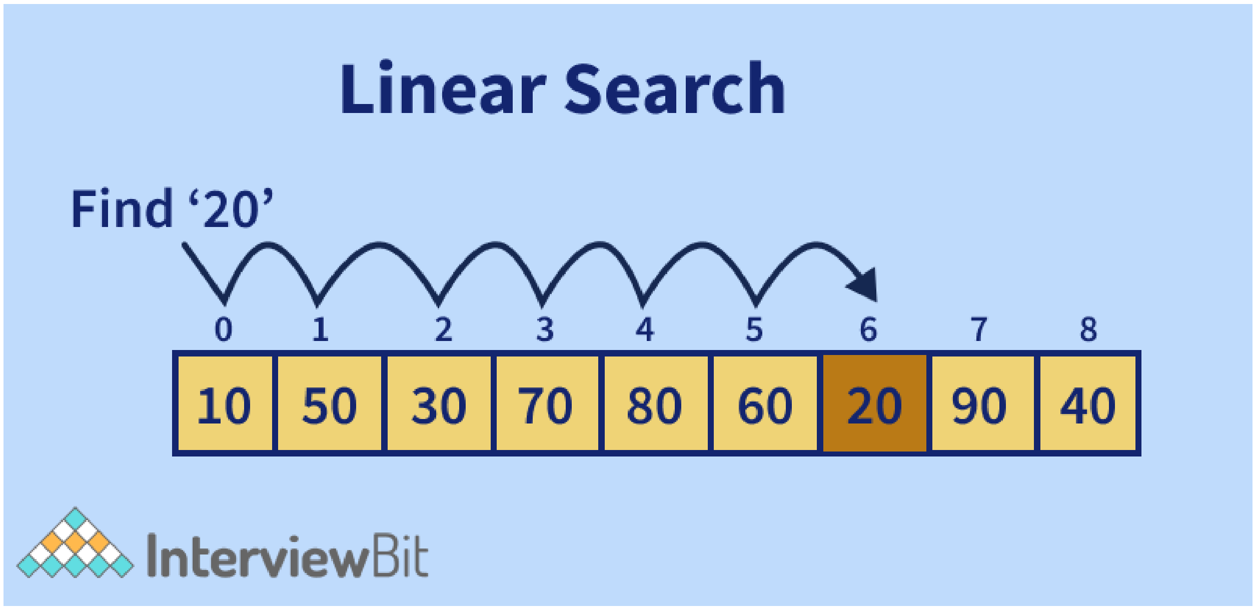
- **Sequential Search:** This method traverses the list of elements consecutively, checking each element and reporting if the element to be searched is found. Linear Search is an example of a Sequential Search Algorithm.
- **Interval Search:** These algorithms were created specifically for searching sorted data structures. Because they continually target the centre 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.

8. Describe the Linear Search Algorithm.

To find an element in a group of elements, the linear search can be used. It works by traversing the list of elements from the beginning to the end and inspecting the properties of all the elements encountered along the way. Let us consider the case of an array containing some integer elements. We want to find out and print all of the elements' positions that match a particular value (also known as the "key" for the linear search). The linear search works in a flow here, matching each element with the number from the beginning to the end of the list, and then printing the element's location if the element at that position is equal to the key.

Given below is an algorithm describing Linear Search:

- Step 1: Using a loop, traverse the list of elements given.
- Step 2: In each iteration, compare the target value (or key-value) to the list's current value.
- Step 3: If the values match, print the array's current index.
- Step 4: Move on to the next array element if the values do not match.
- Step 5: Repeat Steps 1 to 4 till the end of the list of elements is reached.



The time complexity of the Linear Search Algorithm is O(n) where n is the size of the list of elements and its space complexity is constant, that is, O(1).

9. Describe the Binary Search Algorithm.

To apply binary search on a list of elements, the prerequisite is that the list of elements should be sorted. It is based on the Divide and Conquers Algorithmic paradigm. In the Binary Search Algorithm, we divide the search interval in half periodically to search the sorted list. We begin by creating an interval that spans the entire list. If the search key's value is less than the item in the interval's midpoint, the interval should be narrowed to the lower half. Otherwise, we limit it to the upper half of the page. We check for the value until it is discovered or the interval is empty. Given below is an algorithm describing Binary Search: (Let us assume that the element to be searched is x and the array of elements is sorted in ascending order)

- Step 1: x should be firstly compared to the middle element.
- Step 2: We return the middle element's index if x matches the middle element.

- Step 3: Else If x is greater than the middle element, x can only be found after the middle element in the right half subarray since the array is sorted in the ascending order. As a result, we repeat the process for the right half.
- Step 4: Otherwise, we repeat for the left half (x is smaller).
- Step 5: If the interval is empty, we terminate the binary search.

The time complexity of the Binary Search Algorithm is O(log(n)) where n is the size of the list of elements and its space complexity is constant, that is, O(1).

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

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.

11. Write an algorithm for counting the number of leaf nodes in a binary tree.

An algorithm for counting the number of leaf nodes in a binary tree is given below:

- Step 1: If the current node is null, return a value 0.
- Step 2: If a leaf node is encountered, that is, if the current node's left and right nodes are both null, then return 1.
- Step 3: Calculate the number of leaf nodes recursively by adding the number of leaf nodes in the left subtree by the number of leaf nodes in the right subtree.

12. What do you understand about the Dynamic Programming (DP) Algorithmic Paradigm? List a few problems which can be solved using the same.

Dynamic Programming is primarily a recursion optimization. We can use Dynamic Programming to optimise any recursive solution that involves repeated calls for the same inputs. The goal is to simply save the results of subproblems so that we do not have to recalculate them later. The time complexity of this simple optimization is reduced from exponential to polynomial. For example, if we create a simple recursive solution for Fibonacci Numbers, the time complexity is exponential, but if we optimise it by storing subproblem answers using Dynamic Programming, the time complexity is linear.

The following codes illustrate the same:

With Recursion (no DP): The time complexity of the given code will be exponential.

```
/*Sample C++ code for finding nth fibonacci number without DP*/
int nFibonacci(int n){
   if(n == 0 || n == 1) return n;
   else return nFibonacci(n - 1) + nFibonacci(n - 2);
}
```

With DP: The time complexity of the given code will be linear because of Dynamic Programming.

```
/*Sample C++ code for finding nth fibonacci number with DP*/
int nFibonacci(int n){
   vector<int> fib(n + 1);
    fib[0] = 0;
    fib[1] = 1;
    for(int i = 2;i <= n;i ++){
         fib[i] = fib[i - 1] + fib[i - 2];
    }
    return fib[n];
}
```

A few problems which can be solved using the Dynamic Programming (DP) Algorithmic Paradigm are as follows:

- Finding the nth Fibonacci number
- Finding the Longest Common Subsequence between two strings.
- Finding the Longest Palindromic Substring in a string.
- The discrete (or 0-1) Knapsack Problem.
- Shortest Path between any two nodes in a graph (Floyd Warshall Algorithm)

13. Write down a string reversal algorithm. If the given string is "kitiR," for example, the output should be "Ritik".

An algorithm for string reversal is as follows:

- Step 1: Start.
- Step 2: We take two variables l and r.
- Step 3: We set the values of l as 0 and r as (length of the string - 1).
- Step 4: We interchange the values of the characters at positions l and r in the string.
- Step 5: We increment the value of l by one.
- Step 6: We decrement the value of r by one.
- Step 7: If the value of r is greater than the value of l, we go to step 4
- Step 8: Stop.

14. What do you understand about the BFS (Breadth First Search) algorithm.

BFS or Breadth-First Search is a graph traversal technique. It begins by traversing the graph from the root node and explores all of the nodes in the immediate vicinity. It chooses the closest node and then visits all of the nodes that have yet to be visited. Until it reaches the objective node, the algorithm repeats the same method for each of the closest nodes.

The BFS Algorithm is given below:

- Step 1: Set status = 1 as the first step for all the nodes(ready state).
- Step 2: Set the status of the initial node A to 2, that is, waiting state.
- Step 3: Repeat steps 4 and 5 until the queue is not empty.
- Step 4: Dequeue and process node N from the queue, setting its status to 3, that is, the processed state.
- Step 5: Put all of N's neighbours in the ready state (status = 1) in the queue and set their status to 2 (waiting state)
- Step 6: Exit.

15. What do you understand about the DFS (Depth First Search) algorithm.

Depth First Search or DFS is a technique for traversing or exploring data structures such as trees and graphs. The algorithm starts at the root node (in the case of a graph, any random node can be used as the root node) and examines each branch as far as feasible before retracing. So the basic idea is to start at the root or any arbitrary node and mark it, then advance to the next unmarked node and repeat until there are no more unmarked nodes. After that, go back and check for any more unmarked nodes to cross. Finally, print the path's nodes. The DFS algorithm is given below:

- Step1: Create a recursive function that takes the node's index and a visited array as input.
- Step 2: Make the current node a visited node and print it.
- Step 3: Call the recursive function with the index of the adjacent node after traversing all nearby and unmarked nodes.

Algorithm Interview Questions for Experienced

16. How do the encryption algorithms work?

e process of transforming plaintext into a secret code format known as "Ciphertext" is known as encryption. For calculations, this technique uses a string of bits known as "keys" to convert the text. The larger the key, the more potential patterns for producing ciphertext there are. The majority of encryption algorithms use fixed blocks of input with lengths ranging from 64 to 128 bits, while others use the stream technique.

17. What are few of the most widely used cryptographic algorithms?

A few of the most widely used cryptographic algorithms are as follows:

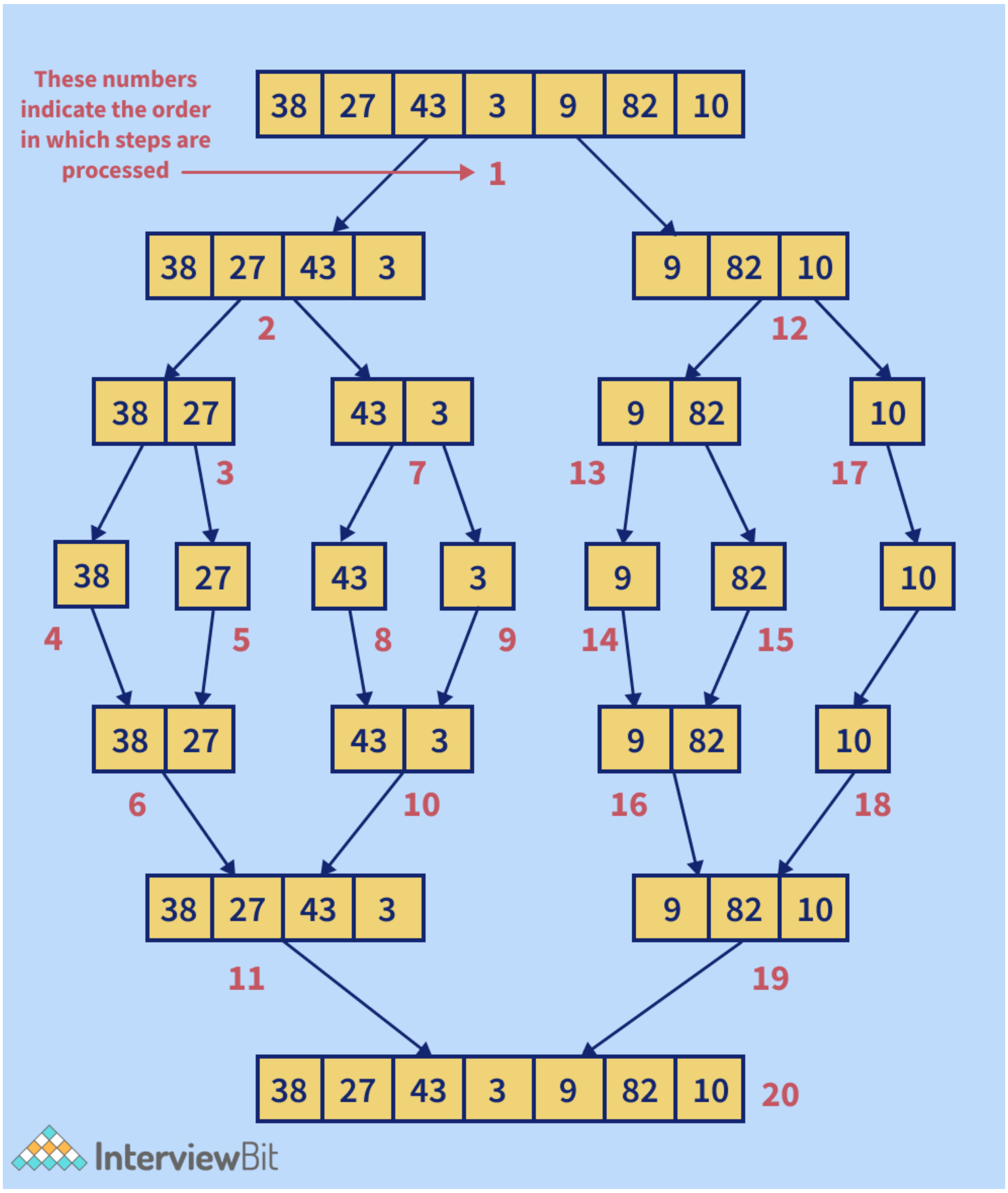
- IDEA
- CAST
- CMEA
- 3-way
- Blowfish
- GOST
- LOKI
- DES and Triple DES.

18. Describe the merge sort algorithm.

Merge sort (also known as mergesort) is a general-purpose, comparison-based sorting algorithm developed in computer science. The majority of its implementations result in a stable sort, which indicates that the order of equal elements in the input and output is the same. In 1945, John von Neumann devised the merge sort method, which is a divide and conquer algorithm. The following is how a merge sort works conceptually:

- Separate the unsorted list into n sublists, each with one element (a list of one element is considered sorted).
- Merge sublists repeatedly to create new sorted sublists until only one sublist remains. The sorted list will be displayed then.

The time complexity of the Merge Sort Algorithm is O(nlog(n)) where n is the size of the list of the elements to be sorted while the space complexity of the Merge Sort Algorithm is O(n), that is, linear space complexity.



19. Describe the quick sort algorithm.

Quicksort is a sorting algorithm that is in place (in-place algorithm is an algorithm that transforms input using no auxiliary data structure). It was created by the British computer scientist Tony Hoare in 1959 and was published in 1961, and it is still a popular sorting algorithm. It can be somewhat quicker than merge sort and two or three times faster than heapsort when properly done.

Quicksort is based on the divide and conquer algorithmic paradigm. It operates by picking a 'pivot' element from the array and separating the other elements into two subarrays based on whether they are greater or less than the pivot. As a result, it is also known as partition exchange sort. The subarrays are then recursively sorted. This can be done in place, with only a little amount of additional RAM (Random Access Memory) required for sorting.

Quicksort is a comparison sorting algorithm, which means it can sort objects of any type that have a "less-than" relation (technically, a total order) declared for them. Quicksort is not a stable sort, which means that the relative order of equal sort items is not retained in efficient implementations. Quicksort (like the partition method) must be written in such a way that it can be called for a range within a bigger array, even if the end purpose is to sort the entire array, due to its recursive nature.

The following are the steps for in-place quicksort:

- If there are less than two elements in the range, return immediately because there is nothing else to do. A special-purpose sorting algorithm may be used for other very small lengths, and the rest of these stages may be avoided.
- Otherwise, choose a pivot value, which is a value that occurs in the range (the precise manner of choice depends on the partition routine, and can involve randomness).
- Partition the range by reordering its elements while determining a point of division so that all elements with values less than the pivot appear before the division and all elements with values greater than the pivot appear after it; elements with values equal to the pivot can appear in either direction. Most partition procedures ensure that the value that ends up at the point of division is equal to the pivot, and is now in its ultimate location because at least one instance of the pivot is present (but termination of quicksort does not depend on this, as long as sub-ranges strictly smaller than the original are produced).
- Apply the quicksort recursively to the sub-range up to the point of division and the sub-range after it, optionally removing the element equal to the pivot at the point of division from both ranges. (If the partition creates a potentially bigger sub-range near the boundary with all elements known to be equal to the pivot, these can also be omitted.)

Quicksort's mathematical analysis reveals that, on average, it takes O(nlog (n) time complexity to sort n items. In the worst-case scenario, it performs in time complexity of O(n^2).

Note: The algorithm's performance can be influenced by the partition routine (including the pivot selection) and other details not fully defined above, possibly to a large extent for specific input arrays. It is therefore crucial to define these alternatives before discussing quicksort's efficiency.

20. Describe the bubble sort algorithm with the help of an example.

Bubble sort, also known as sinking sort, is a basic sorting algorithm that iterates through a list, comparing neighbouring elements and swapping them if they are out of order. The list is sent through again and again until it is sorted. The comparison sort method is named from the manner that smaller or larger components "bubble" to the top of the list. This simplistic method performs badly in real-world situations and is mostly used as a teaching aid. Let us take an example to understand how bubble sort works:

Let us assume that the array to be sorted is (50 10 40 20 80). The various passes or rounds of bubble sort are given below:

- **First Pass:**
  - (50 10 40 20 80) –> (10 50 40 20 80), Since 50 > 10, the algorithm compares the first two elements and swaps them.
  - (10 50 40 20 80) –> (10 40 50 20 80), Since 50 > 40, the algorithm swaps the values at the second and third positions.
  - (10 40 50 20 80) –> (10 40 20 50 80), Since 50 > 3, the algorithm swaps the third and fourth elements.
  - (10 40 20 50 80) -> (10 40 20 50 80), The method does not swap the fourth and fifth elements because they are already in order (80 > 50).
- **Second Pass:**
  - (10 40 20 50 80) –> (10 40 20 50 80) , Elements at first and second position are in order so now swapping.
  - (10 40 20 50 80) –> (10 20 40 50 80), Since 40 > 20, the algorithm swaps the values at the second and third positions.
  - (10 20 40 50 80) –> (10 20 40 50 80), Elements at the third and fourth position are in order so now swapping.
  - (10 20 40 50 80) –> (10 20 40 50 80), Elements at fourth and fifth position are in order so now swapping.

The array is now sorted, but our algorithm is unsure whether it is complete. To know if the algorithm is sorted, it must complete one complete pass without any swaps.

- **Third Pass:**
  - (10 20 40 50 80) –> (10 20 40 50 80), Elements at the first and second position are in order so now swapping.
  - (10 20 40 50 80) –> (10 20 40 50 80), Elements at the second and third position are in order so now swapping.
  - (10 20 40 50 80) –> (10 20 40 50 80), Elements at the third and fourth position are in order so now swapping.

- (10 20 40 50 80) –> (10 20 40 5 80), Elements at the fourth and fifth position are in order so now swapping.

21. Write an algorithm to find the maximum subarray sum for a given array. In other words, find the maximum sum that can be achieved by taking contiguous elements from a given array of integers.

Kadane's algorithm can be used to find the maximum subarray sum for a given array. From left to right, Kadane's algorithm searches the provided array. It then computes the subarray with the largest sum ending at position j in the jth step, and this sum is stored in the variable "currentSum". Furthermore, it computes the subarray with the biggest sum anywhere in the subarray starting from the first position to the jth position, that is, in A[1...j], and stores it in the variable "bestSum". This is done by taking the maximum value of the variable "currentSum" till now and then storing it in the variable "bestSum". In the end, the value of "bestSum" is returned as the final answer to our problem.

Formally, Kadane's algorithm can be stated as follows:

- Step 1: Initialize the following variables:
    - bestSum = INT_MIN
    - currentSum = 0 // for empty subarray, it is initialized as value 0
- Step 2: Loop for each element of the array A
    - (a) currentSum = currentSum + A[i]
    - (b) if(bestSum < currentSum)
      bestSum = currentSum
    - (c) if(currentSum < 0)
      currentSum = 0
- Step 3: return bestSum

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

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.

23. Can we use the binary search algorithm for linked lists? Justify your answer.

No, we cannot use the binary search algorithm for linked lists.
Explanation: Because random access is not allowed in linked lists, reaching the middle element in constant or O(1) time is impossible. As a result, the usage of a binary search algorithm on a linked list is not possible.

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

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.

25. Devise an algorithm to insert a node in a Binary Search Tree.

An algorithm to insert a node in a Binary Search Tree is given below:

- Assign the current node to the root node.
- If the root node's value is greater than the value that has to be added:
    - If the root node has a left child, go to the left.
    - Insert node here if it does not have a left child.
- If the root node's value is less than the value that has to be added:
    - If the root node has a right child, go to the right.
    - Insert node here if it does not have the right child.

26. Define insertion sort and selection sort.

- **Insertion sort:** Insertion sort separates the list into sorted and unsorted sub-lists. It inserts one element at a time into the proper spot in the sorted sub-list. After insertion, the output is a sorted sub-list. It iteratively works on all the elements of an unsorted sub-list and inserts them into a sorted sub-list in order.
- **Selection sort:** Selection sort is an in-place sorting technique. It separates the data collection into sorted and unsorted sub-lists. The minimum element from the unsorted sub-list is then selected and placed in the sorted list. This loops

until all of the elements in the unsorted sub-list have been consumed by the sorted sub-list.

Note: Both sorting strategies keep two sub-lists, sorted and unsorted, and place one element at a time into the sorted sub-list. Insertion sort takes the currently selected element and places it in the sorted array at the right point while keeping the insertion sort attributes. Selection sort, on the other hand, looks for the smallest element in an unsorted sub-list and replaces it with the current element.

27. Define tree traversal and list some of the algorithms to traverse a binary tree.

The process of visiting all the nodes of a tree is known as tree traversal.

Some of the algorithms to traverse a binary tree are as follows:

- Pre-order Traversal.
- In order Traversal.
- Post order Traversal.
- Breadth First Search
- ZigZag Traversal.

28. Describe the heap sort algorithm.

Heap sort is a comparison-based sorting algorithm. Heapsort is similar to selection sort in that it separates its input into a sorted and an unsorted region, then successively decreases the unsorted part by taking the largest element from it and putting it into the sorted region. Unlike selection sort, heapsort does not waste time scanning the unsorted region in linear time; instead, heap sort keeps the unsorted region in a heap data structure to identify the largest element in each step more rapidly. Let us take a look at the heap sort algorithm:

The Heapsort algorithm starts by converting the list to a max heap. The algorithm then swaps the first and last values in the list, reducing the range of values considered in the heap operation by one, and filters the new first value into its heap place. This process is repeated until the range of values considered is only one value long.

- On the list, use the buildMaxHeap() function. This function, also known as heapify(), creates a heap from a list in O(n) operations.
- Change the order of the list's first and last elements. Reduce the list's considered range by one.
- To sift the new initial member to its appropriate index in the heap, use the siftDown() function on the list.
- Unless the list's considered range is one element, proceed to step 2.

Note: The buildMaxHeap() operation runs only one time with a linear time complexity or O(n) time complexity. The siftDown() function works in O(log n) time complexity, and is called n times. Therefore, the overall time complexity of the heap sort algorithm is O(n + n log (n)) = O(n log n).

29. What is the space complexity of the insertion sort algorithm?

Insertion sort is an in-place sorting method, which implies it does not require any additional or minimal data storage. In insertion sort, only a single list element must be stored outside of the starting data, resulting in a constant space complexity or O(1) space complexity.

30. What is the space complexity of the selection sort algorithm?

Selection sort is an in place sorting method, which implies it does not require any additional or minimal data storage. Therefore, the selection sort algorithm has a constant space complexity or O(1) space complexity.

1) What is an algorithm? What is the need for an algorithm?

An algorithm is a well-defined computational procedure that takes some values or the set of values, as an input and produces a set of values or some values, as an output.

Need for Algorithm

The algorithm provides the basic idea of the problem and an approach to solve it. Some reasons to use an algorithm are as follows.

- The algorithm improves the efficiency of an existing technique.
- To compare the performance of the algorithm with respect to other techniques.
- The algorithm gives a strong description of requirements and goal of the problems to the designer.
- The algorithm provides a reasonable understanding of the flow of the program.
- The algorithm measures the performance of the methods in different cases (Best cases, worst cases, average cases).
- The algorithm identifies the resources (input/output, memory) cycles required by the algorithm.
- With the help of an algorithm, we can measure and analyze the complexity time and space of the problems.
- The algorithm also reduces the cost of design.

2) What is the Complexity of Algorithm?

The complexity of the algorithm is a way to classify how efficient an algorithm is compared to alternative ones. Its focus is on how execution time increases with the data set to be processed. The computational complexity of the algorithm is important in computing.

It is very suitable to classify algorithm based on the relative amount of time or relative amount of space they required and specify the growth of time/ space requirement as a function of input size.

Time complexity

Time complexity is a Running time of a program as a function of the size of the input.

Space complexity

Space complexity analyzes the algorithms, based on how much space an algorithm needs to complete its task. Space complexity analysis was critical in the early days of computing (when storage space on the computer was limited).

Nowadays, the problem of space rarely occurs because space on the computer is broadly enough.

We achieve the following types of analysis for complexity

Worst-case: f(n)

It is defined by the maximum number of steps taken on any instance of size n.

Best-case: f(n)

It is defined by the minimum number of steps taken on any instance of size n.

Average-case: f(n)

It is defined by the average number of steps taken on any instance of size n.

3) Write an algorithm to reverse a string. For example, if my string is "uhsnamiH" then my result will be "Himanshu".

Algorithm to reverse a string.

Step1: start

Step2: Take two variable i and j

Step3: do length (string)-1, to set J at last position

Step4: do string [0], to set i on the first character.

Step5: string [i] is interchanged with string[j]

Step6: Increment i by 1

Step7: Increment j by 1

Step8: if i>j then go to step3

Step9: Stop

4) Write an algorithm to insert a node in a sorted linked list.

Algorithm to insert a node in a sorted linked list.

Case1:

Check if the linked list is empty then set the node as head and return it.

1. New_node-> Next= head;
2. Head=New_node

Case2:

Insert the new node in middle

1. While(P!= insert position)
2. {
3. P= p-> Next;
4. }
5. Store_next=p->Next;
6. P->Next= New_node;
7. New_Node->Next = Store_next;

Case3:

Insert a node at the end

1. While (P->next!= **null**)
2. {
3. P= P->Next;
4. }
5. P->Next = New_Node;
6. New_Node->Next = **null**;

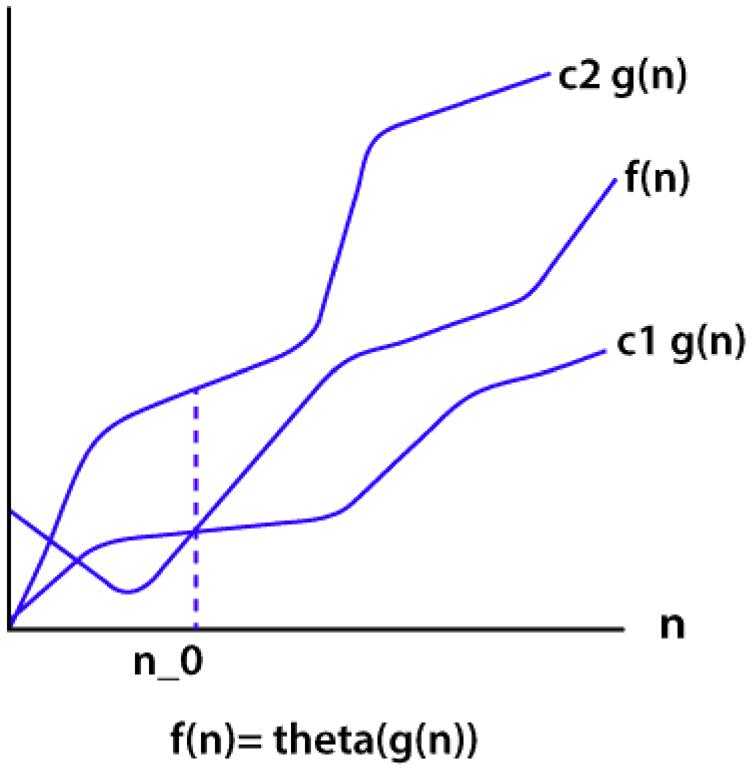
5) What are the Asymptotic Notations?

Asymptotic analysis is used to measure the efficiency of an algorithm that doesn't depend on machine-specific constants and prevents the algorithm from comparing the time taking algorithm. Asymptotic notation is a mathematical tool that is used to represent the time complexity of algorithms for asymptotic analysis.

The three most used asymptotic notation is as follows.

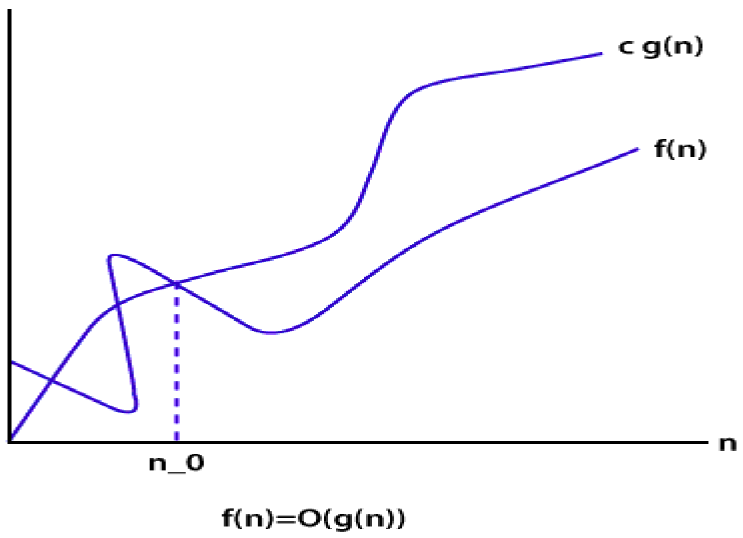
θ Notation

θ Notation defines the exact asymptotic behavior. To define a behavior, it bounds functions from above and below. A convenient way to get Theta notation of an expression is to drop low order terms and ignore leading constants.



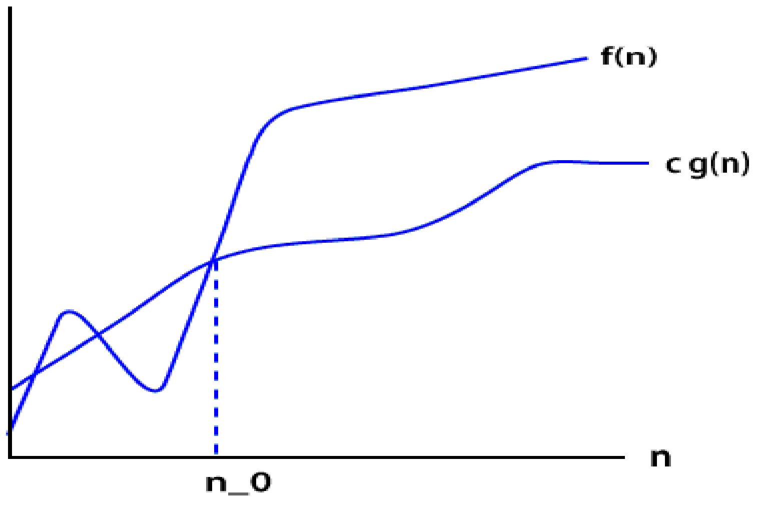
Big O Notation

The Big O notation bounds a function from above, it defines an upper bound of an algorithm. Let's consider the case of insertion sort; it takes linear time in the best case and quadratic time in the worst case. The time complexity of insertion sort is O(n^2). It is useful when we only have upper bound on time complexity of an algorithm.



Ω Notation

Just like Big O notation provides an asymptotic upper bound, the **Ω Notation** provides an asymptotic lower bound on a function. It is useful when we have lower bound on time complexity of an algorithm.



f(n)= Omega(g(n))

8) What is a Hash Table? How can we use this structure to find all anagrams in a dictionary?

A Hash table is a data structure for storing values to keys of arbitrary type. The Hash table consists of an index into an array by using a Hash function. Indexes are used to store the elements. We assign each possible element to a bucket by using a hash function. Multiple keys can be assigned to the same bucket, so all the key and value pairs are stored in lists within their respective buckets. Right hashing function has a great impact on performance.

To find all anagrams in a dictionary, we have to group all words that contain the same set of letters in them. So, if we map words to strings representing their sorted letters, then we could group words into lists by using their sorted letters as a key.

```
1. FUNCTION find_anagrams(words)
2.     word_groups = HashTable<String, List>
3.     FOR word IN words
4.         word_groups.get_or_default(sort(word), []).push(word)
5.     END FOR
6.     anagrams = List
7.     FOR key, value IN word_groups
```

```
8.        anagrams.push(value)
9.     END FOR
10.    RETURN anagrams
```

The hash table contains lists mapped to strings. For each word, we add it to the list at the suitable key, or create a new list and add it to it.

9) What is Divide and Conquer algorithms?

Divide and Conquer is not an algorithm; it's a pattern for the algorithm. It is designed in a way as to take dispute on a huge input, break the input into minor pieces, and decide the problem for each of the small pieces. Now merge all of the piecewise solutions into a global solution. This strategy is called divide and conquer.

Divide and conquer uses the following steps to make a dispute on an algorithm.

Divide: In this section, the algorithm divides the original problem into a set of subproblems.

Conquer: In this section, the algorithm solves every subproblem individually.

Combine: In this section, the algorithm puts together the solutions of the subproblems to get the solution to the whole problem.

10) Explain the BFS algorithm?

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

11) What is Dijkstra's shortest path algorithm?

Dijkstra's algorithm is an algorithm for finding the shortest path from a starting node to the target node in a weighted graph. The algorithm makes a tree of shortest paths from the starting vertex and source vertex to all other nodes in the graph.

Suppose you want to go from home to office in the shortest possible way. You know some roads are heavily congested and challenging to use this, means these edges have a large weight. In Dijkstra's algorithm, the shortest path tree found by the algorithm will try to avoid edges with larger weights.

12) Give some examples of Divide and Conquer algorithm?

Some problems that use Divide and conquer algorithm to find their solution are listed below.

- Merge Sort
- Quick Sort
- Binary Search
- Strassen's Matrix Multiplication
- Closest pair (points)

13) What are Greedy algorithms? Give some example of it?

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

14) What is a linear search?

Linear search is used on a group of items. It relies on the technique of traversing a list from start to end by visiting properties of all the elements that are found on the way.

For example, suppose an array of with some integer elements. You should find and print the position of all the elements with their value. Here, the linear search acts in a flow like matching each element from the beginning of the list to the end of the list with the integer, and if the condition is `True then printing the position of the element.'

Implementing Linear Search

Below steps are required to implement the linear search.

Step1: Traverse the array using **for loop**.

Step2: In every iteration, compare the target value with the current value of the array

Step3: If the values match, return the current index of the array

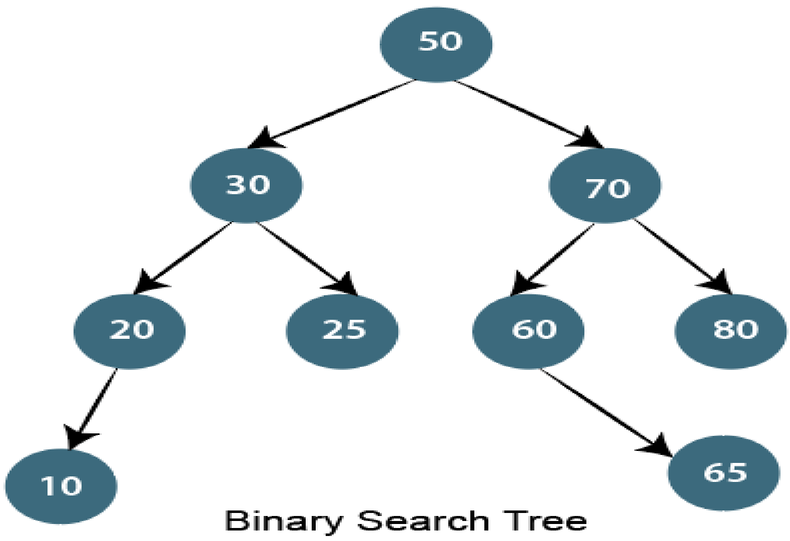
Step4: If the values do not match, shift on to the next array element.

Step5: If no match is found, return -1

15) What is a Binary Search Tree?

The binary search tree is a special type of data structure which has the following properties.

- Nodes which are less than root will be in the left subtree.
- Nodes which are greater than root (i.e., contains more value) will be right subtree.
- A binary search tree should not have duplicate nodes.
- Both sides subtree (i.e., left and right) also should be a binary search tree.



Binary Search Tree

16) Write an algorithm to insert a node in the Binary search tree?

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.

17) How to count leaf nodes of the binary tree?

Algorithm-

Steps for counting the number of leaf nodes are:

- If the node is null (contains null values) then return 0.
- If encountered leaf node. Left is null and node Right is null then return 1.
- Recursively calculate the number of leaf nodes using

No. of leaf nodes= no of leaf nodes in left subtree + number of leaf nodes in the right subtree.

18) How to find all possible words in a board of characters (Boggle game)?

In the given dictionary, a process to do a lookup in the dictionary and an M x N board where every cell has a single character. Identify all possible words that can be formed by order of adjacent characters. Consider that we can move to any of the available 8 adjacent characters, but a word should not have multiple instances of the same cell.

Example:

```
dictionary[] = {"Java", "Point","Quiz"};
Array[][]    = {{'J', 'T', 'P',},
          {'U', 'A', 'A'},
          {'Q', 'S', 'V'}};
```

5. isWord(str): returns **true if** str is present in dictionary
6. **else false**.

Output:

Following words of the dictionary are present

JAVA

19) Write an algorithm to insert a node in a link list?

Algorithm

- Check If the Linked list does not have any value then make the node as head and return it
- Check if the value of the node to be inserted is less than the value of the head node, then insert the node at the start and make it head.
- In a loop, find the appropriate node after which the input node is to be inserted. To find the just node start from the head, keep forwarding until you reach a node whose value is greater than the input node. The node just before is the appropriate node.
- Insert the node after the proper node found in step 3.

20) How to delete a node in a given link list? Write an algorithm and a program?

Write a function to delete a given node from a Singly Linked List. The function must follow the following constraints:

- The function must accept a pointer to the start node as the first argument and node to be deleted as the second argument, i.e., a pointer to head node is not global.
- The function should not return a pointer to the head node.
- The function should not accept pointer to pointer to head node.

What is data-structure?

- Data structure is a way of defining, storing & retriving of data in a structural & systemetic way. A data structure may contain different type of data items.

What are various data-structures available?

- Data structure availability may vary by programming languages. Commonly available data structures are list, arrays, stack, queues, graph, tree etc.

Why we need to do algorithm analysis?

A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.

What are the criteria of algorithm analysis?

An algorithm are generally analyzed on two factors − time and space. That is, how much execution time and how much extra space required by the algorithm.

What is asymptotic analysis of an algorithm?

Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.

What are asymptotic notations?

Asymptotic analysis can provide three levels of mathematical binding of execution time of an algorithm −

Best case is represented by Ω(n) notation.

Worst case is represented by O(n) notation.

Average case is represented by Θ(n) notation.

What is linear data structure?

A linear data-structure has sequentially arranged data items. The next time can be located in the next memory address. It is stored and accessed in a sequential manner. Array and list are example of linear data structure.

What are common operations that can be performed on a data-structure?

The following operations are commonly performed on any data-structure −

Insertion − adding a data item

Deletion − removing a data item

Traversal − accessing and/or printing all data items

Searching − finding a particular data item

Sorting − arranging data items in a pre-defined sequence

Briefly explain the approaches to develop algorithms.

There are three commonly used approaches to develop algorithms −

Greedy Approach − finding solution by choosing next best option

Divide and Conquer − diving the problem to a minimum possible sub-problem and solving them independently

Dynamic Programming − diving the problem to a minimum possible sub-problem and solving them combinedly

Give some examples greedy algorithms.

The below given problems find their solution using greedy algorithm approach −

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

What are some examples of divide and conquer algorithms?

The below given problems find their solution using divide and conquer algorithm approach −

Merge Sort

Quick Sort

Binary Search

Strassen's Matrix Multiplication

Closest pair (points)

What are some examples of dynamic programming algorithms?

The below given problems find their solution using divide and conquer algorithm approach −

Fibonacci number series

Knapsack problem

Tower of Hanoi

All pair shortest path by Floyd-Warshall

Shortest path by Dijkstra

Project scheduling

What is a linked-list?

A linked-list is a list of data-items connected with links i.e. pointers or references. Most modern high-level programming language does not provide the feature of directly accessing memory location, therefore, linked-list are not supported in them or available in form of inbuilt functions.

What is stack?

In data-structure, stack is an Abstract Data Type (ADT) used to store and retrieve values in Last In First Out method.

Why do we use stacks?

Stacks follows LIFO method and addition and retrieval of a data item takes only O(n) time. Stacks are used where we need to access data in the reverse order or their arrival. Stacks are used commonly in recursive function calls, expression parsing, depth first traversal of graphs etc.

What operations can be performed on stacks?

The below operations can be performed on a stack −

push() − adds an item to stack

pop() − removes the top stack item

peek() − gives value of top item without removing it

isempty() − checks if stack is empty

isfull() − checks if stack is full

What is a queue in data-structure?

Queue is an abstract data structure, somewhat similar to stack. In contrast to stack, queue is opened at both end. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

Why do we use queues?

As queues follows FIFO method, they are used when we need to work on data-items in exact sequence of their arrival. Every operating system maintains queues of various processes. Priority queues and breadth first traversal of graphs are some examples of queues.

What operations can be performed on Queues?

The below operations can be performed on a stack −

enqueue() − adds an item to rear of the queue

dequeue() − removes the item from front of the queue

peek() − gives value of front item without removing it

isempty() − checks if stack is empty

isfull() − checks if stack is full

What is linear searching?

Linear search tries to find an item in a sequentially arranged data type. These sequentially arranged data items known as array or list, are accessible in incrementing memory location. Linear search compares expected data item with each of data items in list or array. The average case time complexity of linear search is O(n) and worst case complexity is O(n2). Data in target arrays/lists need not to be sorted.

What is binary search?

A binary search works only on sorted lists or arrays. This search selects the middle which splits the entire list into two parts. First the middle is compared.

This search first compares the target value to the mid of the list. If it is not found, then it takes decision on whether.

What is bubble sort and how bubble sort works?

Bubble sort is comparison based algorithm in which each pair of adjacent elements is compared and elements are swapped if they are not in order. Because the time complexity is O(n2), it is not suitable for large set of data.

Tell me something about 'insertion sort'?

Insertion sort divides the list into two sub-list, sorted and unsorted. It takes one element at time and finds it appropriate location in sorted sub-list and insert there. The output after insertion is a sorted sub-list. It iteratively works on all the elements of unsorted sub-list and inserts them to sorted sub-list in order.

What is selection sort?

Selection sort is in-place sorting technique. It divides the data set into two sub-lists: sorted and unsorted. Then it selects the minimum element from unsorted sub-list and places it into the sorted list. This iterates unless all the elements from unsorted sub-list are consumed into sorted sub-list.

How insertion sort and selection sorts are different?

Both sorting techniques maintains two sub-lists, sorted and unsorted and both take one element at a time and places it into sorted sub-list. Insertion sort works on the current element in hand and places it in the sorted array at appropriate location maintaining the properties of insertion sort. Whereas, selection sort searches the minimum from the unsorted sub-list and replaces it with the current element in hand.

What is merge sort and how it works?

Merge sort is sorting algorithm based on divide and conquer programming approach. It keeps on dividing the list into smaller sub-list until all sub-list has only 1 element. And then it merges them in a sorted way until all sub-lists are consumed. It has run-time complexity of O(n log n) and it needs O(n) auxiliary space.

What is shell sort?

Shell sort can be said a variant of insertion sort. Shell sort divides the list into smaller sublist based on some gap variable and then each sub-list is sorted using insertion sort. In best cases, it can perform upto O(n log n).

How quick sort works?

Quick sort uses divide and conquer approach. It divides the list in smaller 'partitions' using 'pivot'. The values which are smaller than the pivot are arranged in the left partition and greater values are arranged in the right partition. Each partition is recursively sorted using quick sort.

What is a graph?

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

How depth first traversal works?

Depth First Search algorithm(DFS) traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search when a dead end occurs in any iteration.

How breadth first traversal works?

Breadth First Search algorithm(BFS) traverses a graph in a breadthwards motion and uses a queue to remember to get the next vertex to start a search when a dead end occurs in any iteration.

What is a tree?

A tree is a minimally connected graph having no loops and circuits.

What is a binary tree?

A binary tree has a special condition that each node can have two children at maximum.

What is a binary search tree?

A binary search tree is a binary tree with a special provision where a node's left child must have value less than its parent's value and node's right child must have value greater than it's parent value.

What is tree traversal?

Tree traversal is a process to visit all the nodes of a tree. Because, all nodes are connected via edges (links) we always start from the root (head) node. There are three ways which we use to traverse a tree −

In-order Traversal

Pre-order Traversal

Post-order Traversal

What is an AVL Tree?

AVL trees are height balancing binary search tree. AVL tree checks the height of left and right sub-trees and assures that the difference is not more than 1. This difference is called Balance Factor.

BalanceFactor = height(left-sutree) − height(right-sutree)

What is a spanning tree?

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. A spanning tree does not have cycles and it can not be disconnected.

How many spanning trees can a graph has?

It depends on how connected the graph is. A complete undirected graph can have maximum nn-1 number of spanning trees, where n is number of nodes.

How Kruskal's algorithm works?

This algorithm treats the graph as a forest and every node it as an individual tree. A tree connects to another only and only if it has least cost among all available options and does not violate MST properties.

How Prim's algorithm finds spanning tree?

Prim's algorithm treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

What is a minimum spanning tree (MST)?

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight that all other spanning trees of the same graph.

What is a heap in data structure?

Heap is a special balanced binary tree data structure where root-node key is compared with its children and arranged accordingly. A min-heap, a parent node has key value less than its childs and a max-heap parent node has value greater than its childs.

What is a recursive function?

A recursive function is one which calls itself, directly or calls a function that in turn calls it. Every recursive function follows the recursive properties − base criteria where functions stops calling itself and progressive approach where the functions tries to meet the base criteria in each iteration.

What is tower of hanoi?

Tower of Hanoi, is a mathematical puzzle which consists of three tower (pegs) and more than one rings. All rings are of different size and stacked upon each other where the large disk is always below the small disk. The aim is to move the tower of disk from one peg to another, without breaking its properties.

What is fibonacci series?

Fibonacci Series generates subsequent number by adding two previous numbers. For example − 0 1 1 2 3 5 8 13.

What is hashing?

Hashing is a technique to convert a range of key values into a range of indexes of an array. By using hash tables, we can create an associative data storage where data index can be find by providing its key values.

What is interpolation search technique?

Interpolation search is an improved variant of binary search. This search algorithm works on the probing position of required value.

What is the prefix and post fix notation of (a + b) * (c + d) ?

Prefix Notation − * + a b + c d

Postfix Notation − a b + c d + *