**MCQ:-**

**What is the time complexity of the swapping operation in bubble sort?**Options: O, O(n log n), **O(n²)**, O(n³)

**How many passes are required for insertion sort?**  
Options: N, **N-1**, N+1, None

**What type of search is classified as direct search?**Options: Binary Search, Hashing, **Linear Search**, None

**Which sorting algorithm utilizes the concept of a pivot element?**  
Options: Selection Sort, Insertion Sort, **Quick Sort**, Merge Sort

**What is the primary technique used in merge sort?**Options: **Divide and Conquer**, Greedy Algorithm, Dynamic Programming, Branch and Bound

**How many sub-arrays does the quick sort algorithm divide the main array into during each iteration?**  
Options: One, **Two**, Three, Four

**Which data structure implementation resembles insertion sort?**  
Options: Binary Heap, **Insertion Sort**, Quick Sort, Merge Sort

**What is the average number of inversions in an array of N distinct numbers?**Options: **N(N-1)/4**, N(N-1)/2, N(N-1)/3

**What is the running time of insertion sort if the input array is already sorted?**  
Options: **O**, O(n log n), O(n²), O(n³)

**After the second pass of insertion sort, how are the array elements arranged for [34, 8, 64, 51, 32, 21]?**  
Options: **8, 21, 32, 34, 51, 64**, 8, 32, 34, 51, 64, 21; 8, 34, 51, 64, 32, 21; 8, 34, 64, 51, 32, 21

**What is an external sorting algorithm?**  
Options: **Algorithm using tape/disk**, Algorithm using main memory, Algorithm with swapping, In-place algorithm

**What is an internal sorting algorithm?**Options: Algorithm using tape/disk, **Algorithm using main memory**, Algorithm with swapping, In-place algorithm

**What is the worst-case time complexity of bubble sort?**Options: O, O(n log n), **O(n²)**, O(n³)

**What is the best-case complexity of the improved bubble sort?**Options: **O**, O(n log n), O(n²), O(n³)

**What is the difference between external and internal sorting algorithms?**Options: **Medium of storage**, Execution speed, Algorithm complexity, Purpose of use

**How does selection sort determine the next element to swap?**  
Options: **By finding the smallest/largest**, Using a pivot, By comparing neighbours, Using heap property

**What sorting algorithm is most efficient for small datasets?**  
Options: Merge Sort, Quick Sort, Bubble Sort, **Insertion Sort**

**Why is quick sort preferred over bubble sort?**  
Options: **Lower average complexity**, Stability, Simplicity, Space usage

**What are the three key steps in the divide-and-conquer approach?**  
Options: **Divide, Combine, Conquer**, Divide, Swap, Sort; Split, Sort, Merge; None of these

**What happens during the merging step in merge sort?**Options: Elements are split, **Elements are compared and combined**, Elements are swapped, Elements are deleted

**What is the primary advantage of heap sort?**  
Options: Efficient memory usage, Stability, **Fast for large datasets**, Simplicity

**How is stability defined in sorting algorithms?**  
Options: **Preserves order of equal elements**, Uses less memory, Maintains original array, Faster execution

**What type of data does radix sort work best with?**  
Options: **Numeric data**, Text data, Complex objects, Linked lists

**Which sorting algorithm has the best average-case time complexity?**  
Options: Merge Sort, **Quick Sort**, Bubble Sort, Selection Sort

**How does the space complexity of merge sort compare to quick sort?**  
Options: **More space**, Less space, Equal space, None of these

**What is the minimum number of swaps needed for selection sort?**  
Options: N, **N-1**, N/2, None

**Why is bubble sort inefficient for large datasets?**Options: **High time complexity**, Space requirement, Unstable behaviour, Difficult implementation

**What type of sorting algorithm is insertion sort?**Options: Divide and Conquer, **Iterative**, Recursive, Comparison-based

**When does the performance of insertion sort match its best case?**Options: **Sorted input**, Reversed input, Random input, Large datasets

**What is the pivot's role in quick sort?**Options: **Dividing the array**, Merging sub-arrays, Reducing time complexity, None of these

**Which sorting algorithm uses divide and conquer and compares halves of the array?**  
Options: **Merge Sort**, Quick Sort, Bubble Sort, Heap Sort

**Why does merge sort require additional memory during execution?**  
Options: For merging, For recursion, For swapping, None of these

**How is insertion sort different from selection sort?**  
Options: Direct swapping, Stable sorting, Divides into sub-arrays, Incremental sorting

**What is the primary use case of bubble sort?**  
Options: Large datasets, **Educational purposes**, High-speed sorting, None of these

**Which algorithm is commonly used for external sorting tasks?**  
Options: Quick Sort, **Merge Sort**, Heap Sort, Bubble Sort

**How does quick sort choose the pivot element?**  
Options: First element, Middle element, **Randomly or predefined rule**, Last element

**What is the difference between stable and unstable sorting algorithms?**  
Options: Execution speed, Memory usage, **Order preservation**, Sorting type

**Why is insertion sort suitable for small or nearly sorted arrays?**  
Options: Simple logic, Low space complexity, **Low average time complexity**, Faster for large datasets

**What is the result of sorting [5, 3, 8, 6, 2] using selection sort after the first pass?**  
Options: **[2, 3, 8, 6, 5]**, [5, 3, 6, 8, 2], [2, 6, 3, 8, 5], [5, 6, 3, 8, 2]

**Which sorting technique is fastest when the input size is large?**  
Options: Merge Sort, **Quick Sort**, Insertion Sort, Bubble Sort

**What is the worst-case time complexity of quick sort?**  
Options: O, O(n log n), **O(n²)**, O(n³)

**How does heap sort arrange data?**  
Options: Using a pivot, In sorted order directly, **Using a binary heap structure**, Splitting into sub-arrays

**Which sorting algorithm maintains the relative order of equal elements?**  
Options: Bubble Sort, Selection Sort, **Insertion Sort**, Heap Sort

**Why is quick sort preferred for arrays over linked lists?**  
Options: **Cache performance**, Stability, Memory requirements, Recursive nature

**How many comparisons are required in selection sort for an array of size N?**  
Options: N²/2, N(N-1)/2, **(N-1)/2**, None

**What is the main disadvantage of using merge sort?**  
Options: Slow execution, Unstable behaviour, **High space complexity**, Not suitable for small data

**What is the key operation in insertion sort for placing elements?**  
Options: Swapping, **Shifting**, Recursion, Partitioning

**Why is bubble sort not suitable for large datasets?**  
Options: **High time complexity**, Memory usage, Lack of stability, Difficult implementation

**What is the best-case scenario for quick sort?**  
Options: Pivot at the first position, Pivot at the middle, **Balanced partitioning**, None

**What is the advantage of using insertion sort for partially sorted arrays?**  
Options: **Reduced comparisons**, Low memory usage, Divide-and-conquer approach, High speed

**When can a deletion operation be performed inside a circular queue?**

Options: **Front = Front + 1,** Rear = Rear – 1, Rear = Rear + 1, Front = Front – 1

**What is the empty condition (underflow) of a circular queue?**

**Options: Front == -1** , Front == Rear, Front > Rear, Rear == -1

**What is the full condition (overflow) of a circular queue?**

Options: Front == (Rear - 1) % Size, Front == (Front + 1) % Size, **Front == (Rear + 1) % Size,** Rear == (Front + 1) % Size

**What is the complexity of Linear Search and Binary Search?**

Options: O(n²), O(n log n), O, **O(log n)**

**What method does FIFO use?**

Option: **Stack,** Hash Table, Linked List, Queue

**What is the correct C code to create a new node?**

*ptr = (NODE)malloc(NODE);* ptr = (NODE)malloc(sizeof(NODE)); ptr = (NODE\*)malloc(sizeof(NODE\*)); ptr = (NODE\*)malloc(NODE);

**What is the time complexity to insert a node at the beginning of a singly linked list, when the pointer is initially pointing to the head?**

Options**: O,** O(log n), O(n log n), O(1)

**Which of the following is not suitable for the implementation of a linked list?**

Options: Polynomial manipulation, Binary Search, **Radix sort,** Polynomial

**What happens during an underflow condition when attempting to delete a non-existent element in a linked list?**

Options: delete an existing element in the list, **insert a new node in the empty list,** delete a non-existent element in the list, delete a node in an empty list

**In which variation of the linked list can the concatenation of two lists be performed in O(1) time?**

Options: Doubly linked list, Circular doubly linked list, **Array implementation of list,** Singly linked list

**What does the following code snippet output for the list 1->2->3->4->5->6?**

Options: 1 2 3 4 5 6, **1 3 5 5 3 1,** 1 3 5 1 3 5, 2 4 6 1 3 5

**How many pointers need to be modified to insert an element at the middle of a linked list?**

Options: **2,** 1, 3, 4

**In the worst case, how many comparisons are needed to search a singly linked list of length n for a given element?**

Options: O(log n), **O**, O(1), None of the above

**Which algorithm is not related to linked lists?**

Options: Linear search, Merge sort, Insertion sort, **Binary sort**

**What is the value of “sum” after the following code snippet terminates?**

**Options: 15,** 20, 1, 5

**Which type of linked list stores the address of the head node in the next pointer of the last node?**

Options: Singly linked list, Doubly linked list, Hashed list, **Circular linked list**

**What is the time complexity of reversing a linked list?**

Options: **O,** O(1), O(n²) , O(logn)

**What is the time complexity of concatenating two linked lists?**

Options: O, O(1), **O(1) if we have the address of the last node of one of the lists,** None of these

**What does the following code snippet do?**

Deletes the given node from the linked list

Deletes the head of the list

Deletes the last node of the list

**None of these**

**What is the space complexity for deleting a linked list?**

**O(1)**

O

O(n²)

None of these

**What information is stored by linked lists used to implement binary trees?**

Value of current node

Pointer to left child

Pointer to right child

**All of the above**

**What is the time complexity of the swapping operation in bubble sort?**

O

O(n log n)

**O(n²)**

O(n³)

**What is the number of passes an insertion sort algorithm consists of?**

N

**N-1**

N+1

None of them

**What search technique uses a direct approach?**

Binary Search

Hashing

None of these

Linear Search

**Which sorting technique uses the term ‘pivot’?**

Selection Sort

Insertion Sort

**Quick Sort**

Merge Sort

**Which technique is used in Merge Sort?**

**Divide and conquer**

Greedy algorithm

Dynamic programming

Branch and bound

**How many sub-arrays does the Quick Sort algorithm divide the entire array into?**

**2**

3

4

1

**Which algorithm implementation is similar to that of insertion sort?**

**Binary heap**

Insertion Sort

Quick Sort

Merge Sort

**What is the average number of inversions in an array of N distinct numbers?**

**N(N-1)/4**

N(N+1)/2

N(N-1)/2

N(N-1)/3

**What is the running time of an insertion sort algorithm if the input is pre-sorted?**

**O**

O(n log n)

O(n²)

O(n³)

**What is the arrangement of the array elements after the second pass using insertion sort for [34, 8, 64, 51, 32, 21]?**

8, 21, 32, 34, 51, 64

8, 32, 34, 51, 64, 21

8, 34, 51, 64, 32, 21

**8, 34, 64, 51, 32, 21**

**Which of the following is the best option for an external sorting algorithm?**

**Algorithm that uses tape or disk during the sort**

Algorithm that uses main memory during the sort

Algorithm that involves swapping

Algorithm that is considered ‘in-place’

**Which of the following is the best option for an internal sorting algorithm?**

Algorithm that uses tape or disk during the sort

**Algorithm that uses main memory during the sort**

Algorithm that involves swapping

Algorithm that is considered ‘in-place’

**What is the worst-case complexity of bubble sort?**

O

O(n log n)

**O(n²)**

O(n³)

**What is the best-case complexity of the improved bubble sort?**

**O**

O(n log n)

O(n²)

O(n³)

**What is the main disadvantage of selection sort?**

It requires auxiliary memory

**It is not scalable**

It can be used for small keys

It takes linear time to sort the elements

**When is linear search used?**

When the list has only a few elements

When performing a single search in an unordered list

Used all the time

**When the list has only a few elements and when performing a single search in an unordered list**

**What is the output of the following code for the array [1, 2, 3, 6, 8, 10]?**

**Print the duplicate elements in the array**

Print the element with maximum frequency

Print the unique elements in the array

Prints the element with minimum frequency

**What is the space consumed by the recursive version of linear search compared to the iterative version?**

**Recursive algorithm consumes more space**

Recursive algorithm consumes less space

Same

Can’t be said

**What is the correct recurrence relation for the linear search recursive algorithm?**

T(n-2) + c

2T(n-1) + c

**T(n-1) + c**

T(n+1) + c

**What is the mid-value in the first and second levels of recursion for the binary search using the array [45, 77, 89, 90, 94, 99, 100] and key = 99?**

**90 and 99**

90 and 94

89 and 99

89 and 94

**What is the average-case time complexity of binary search using recursion?**

O(n log n)

**O(log n)**

O

O(n²)

**What is not an application of binary search?**

To find the lower/upper bound in an ordered sequence

Union of intervals

Debugging

**To search in an unordered list**

**Which algorithm is based on heap sort?**

Fibonacci heap

Binary tree

**Priority queue**

FIFO

**What is the time complexity to build a binary heap?**

**O**

O(N log N)

O(log N)

O(N²)

**At which position in the array does heap sort contain data?**

-1

**0**

1

Anywhere in the array

**What happens to the array after deleting the last minimum element in heap sort?**

* Increasing sorting order
* Tree inorder
* Tree preorder
* **Decreasing sorting order**

**What is the typical running time of the heap sort algorithm?**

* O(N)
* **O(N log N)**
* O(log N)
* O(N²)

**Which algorithm is used in the top tree data structure?**

* Backtracking
* **Divide and Conquer**
* Branch
* Greedy

**What is the number of edges from the root to the node in a tree called?**

* Height
* **Depth**
* Length
* Width

**What is the number of edges from the node to the deepest leaf in a tree called?**

* **Height**
* Depth
* Length
* Width

**What is the correct option for a full binary tree?**

* Each node has exactly zero or two children
* Each node has exactly two children
* All the leaves are at the same level
* Each node has exactly one or two children

**What defines a complete binary tree?**

* Each node has exactly zero or two children
* A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from right to left
* **A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right**
* A tree in which all nodes have degree 2

**What is the average case time complexity for finding the height of a binary tree?**

* O(loglogn)
* O(nlogn)
* **O(n)**
* O(log n)

**What is the relationship between the number of leaves (L) and internal nodes (I) in a full binary tree?**

* L = 2\*I
* **L = I + 1**
* L = I - 1
* L = 2\*I - 1

**How many nodes (N) are in a full binary tree if the number of internal nodes is I?**

* N = 2\*I
* N = I + 1
* N = I - 1
* **N = 2\*I + 1**

**What is the total number of nodes (N) in a full binary tree if there are L leaves?**

* N = 2\*L
* N = L + 1
* N = L - 1
* **N = 2\*L - 1**

**Which of the following statements is incorrect with respect to binary trees?**

* **Let T be a binary tree with N nodes. Then the number of levels is at least floor(log (N + 1))**
* Let T be a binary tree. For every k ≥ 0, there are no more than 2k nodes in level k
* Let T be a binary tree with λ levels. Then T has no more than 2λ – 1 nodes
* Let T be a binary tree with N nodes. Then the number of levels is at least ceil(log (N + 1))

**What is the disadvantage of implementing a tree using normal arrays?**

* Difficulty in knowing children nodes of a node
* Difficult in finding the parent of a node
* **Have to know the maximum number of nodes possible before creation of trees**
* Difficult to implement

**What are the children of node ‘w’ in a complete-binary tree in an array representation?**

* **2w and 2w+1**
* 2+w and 2-w
* w+1/2 and w/2
* w-1/2 and w+1/2

**What is the parent for a node ‘w’ in a complete binary tree in an array representation when w is not 0?**

* ceil(w-1/2)
* **floor(w-1/2)**
* w-1/2
* w/2

**Can a tree stored in an array using one of the inorder, postorder, or preorder traversals be reformed?**

* Yes, just traverse through the array and form the tree
* No, we need one more traversal to form a tree
* No, in case of sparse trees
* **Yes, by using both inorder and array elements**

**Which of the following is not a tree traversal algorithm?**

* Postorder
* Preorder
* Inorder
* **Randomized**

**Which traversal strategy is used in a binary tree?**

* Depth-first traversal
* **Breadth-first traversal**
* Random traversal
* Priority traversal

**What formula can be used to locate a left child of a node with index i in an array representation of a binary tree?**

* **2i+1**
* 2i+2
* 2i
* 4i

**What formula can be used to locate a right child of a node with index i in an array representation of a binary tree?**

* 2i+1
* **2i+2**
* 2i
* 4i

**What formula is used to find a parent node of a binary tree when represented in an array?**

* (i+1)/2
* **(i-1)/2**
* i/2
* 2i/2

**Which property is obeyed by all three tree traversals?**

* **Left subtrees are visited before right subtrees**
* Right subtrees are visited before left subtrees
* Root node is visited before left subtree
* Root node is visited before right subtree

**What traversal is used to obtain a prefix expression?**

* Level-order traversal
* **Post-order traversal**
* Post-order traversal
* In-order traversal

**Given the preorder traversal of a binary tree as A, B, E, C, D and inorder traversal as B, E, A, D, C, what is the level order sequence of the tree?**

* A, C, D, B, E
* A, B, C, D, E
* **A, B, C, E, D**
* D, B, E, A, C

**Given the following sequences S1: N, M, P, O, Q; S2: N, P, Q, O, M; and S3: M, N, O, P, Q, what is the correct identification of the sequences as preorder, inorder, and postorder?**

* S1 is preorder, S2 is inorder, S3 is postorder
* S1 is inorder, S2 is preorder, S3 is postorder
* **S1 is inorder, S2 is postorder, S3 is preorder**
* S1 is postorder, S2 is inorder, S3 is preorder

**What is the possible number of binary trees that can be created with 3 nodes, giving the sequence N, M, L when traversed in post-order?**

* 15
* **5**
* 20
* 10

**If the post-order traversal of a binary tree is O, P, Q, R, S, T, what is the pre-order traversal?**

* T Q R S O P
* T O Q R P S
* **T Q O P S R**
* T Q O S P R

**Given a binary search tree with values 7, 8, 13, 26, 35, 40, 70, 75, and a pre-order sequence as 35, 13, 7, 8, 26, 70, 40, 75, what is the valid post-order sequence?**

* 7, 8, 26, 13, 75, 40, 70, 35
* 26, 13, 7, 8, 70, 75, 40, 35
* 7, 8, 13, 26, 35, 40, 70, 75
* **8, 7, 26, 13, 40, 75, 70, 35**

**What is the depth-first search (DFS) traversal equivalent to?**

* **Pre-order traversal of binary tree**
* Pre-order traversal of binary tree
* In-order traversal of binary tree
* Level-order traversal of binary tree

**What is the time complexity of DFS (V – number of vertices, E – number of edges)?**

* **O(V + E)**
* O(V)
* O(E)
* O(V\*E)

**Which data structure is used in the standard implementation of Breadth First Search?**

* Queue
* Tree
* Graph
* **Stack**

**Which data structure is used in the standard implementation of Depth First Search?**

* List
* Tree
* **Queue**
* Stack

**What is the result of a Depth First Search traversal of a graph?**

* Linked List
* **Tree**
* Graph with back edges
* Array

**How many times is a node visited in a Depth First Search traversal?**

* Once
* Twice
* **Equivalent to the number of indegree of the node**
* Thrice

**What is an AVL tree?**

* **A tree which is balanced and is a height balanced tree**
* A tree which is unbalanced and is a height balanced tree
* A tree with three children
* A tree with at most 3 children

**What is the maximum height of an AVL tree with p nodes?**

* p
* **log(p)**
* log(p)/2
* P/2

**What is the maximum difference in heights between the leaves of an AVL tree?**

* **log(n) where n is the number of nodes**
* n where n is the number of nodes
* 0 or 1
* At most 1

**Given an empty AVL tree, what is the correct way to construct the AVL tree when a set of numbers are given, without performing any rotations?**

* Just build the tree with the given input
* **Find the median of the set of elements given, make it as root and construct the tree**
* Use trial and error
* Use dynamic programming to build the tree

**What is the worst-case time complexity of the linear search algorithm?**

* **O(n)**
* O(logn)
* O(nlogn)
* None

**What is the location of a given item in a collection of items called?**

* Finding
* Searching
* Mining
* **Find**

**What operation processes each element in a list?**

* **Inserting**
* Merging
* Sorting
* Insert

**What are the two main measures for the efficiency of an algorithm?**

* Complexity and capacity
* **Time and space**
* Data and space
* Complexity and capac

**What is an algorithm defined as?**

* Only jump
* One step
* None of these
* **Only j**

**Which case does not exist in complexity theory?**

* Worst case
* Average case
* **Null case**
* Best case

**What is an algorithm that calls itself directly or indirectly called?**

* Sub algorithm
* **Recursion**
* Traversal Algorithm
* None of these

**Which of the following is a LIFO method?**

* Queue
* **Stack**
* Hash Table
* Linked List

**Which of the following is a FIFO method?**

* **Queue**
* Stack
* Hash Table
* Linked List

**What term is used to describe the process of adding an element to a data structure?**

* Deletion
* Traversal
* **Insertion**
* Analysis

**Which asymptotic notation should be used to describe the best-case time complexity of an algorithm?**

* O-notation
* **Ω-notation**
* Θ-notation
* Big-Oh notation

**Which asymptotic notation represents the worst-case time complexity of an algorithm?**

* **O(n)**
* Ω(n)
* Θ(n)
* O(1)

**What asymptotic notation describes an upper bound on the running time of an algorithm?**

* O(n)
* Ω(n)
* Θ(n)
* **O(1)**

**What is the purpose of asymptotic notations in computer science and algorithm analysis?**

* To precisely measure the execution time of algorithms
* To provide a lower bound on the running time of algorithms
* **To classify algorithms based on their efficiency with respect to input size**
* To define algorithms in a way that minimizes their time complexity

**What typically happens to the execution time in a time-space trade-off if you optimize for lower space usage?**

* Execution time decreases
* **Execution time increases**
* Execution time remains unchanged
* It depends on the specific algorithm

**What typically happens to the space usage in a time-space trade-off if you optimize for faster execution time?**

* **Space usage increases**
* Space usage decreases
* Space usage remains unchanged
* It depends on the specific algorithm

**What does O(n) time complexity mean for an algorithm?**

* The algorithm runs in constant time.
* The algorithm runs in logarithmic time.
* **The algorithm runs in linear time.**
* The algorithm runs in exponential time.

**Which asymptotic notation describes a lower bound on the running time of an algorithm?**

* O(n)
* **Ω(n)**
* Θ(n)
* O(1)

**What is an algorithm?**

* A type of computer program
* **A step-by-step procedure to solve a problem**
* A data structure
* A programming language

**Which characteristic is NOT typical of a good algorithm?**

* Efficiency
* Clarity
* Simplicity
* **Inefficiency**

**What is the correct statement about algorithms, data structures, and programs?**

* **Algorithm + Data Structure = Program**
* Algorithm + Program = Data Structure
* Program + Data Structure = Algorithm
* None

**Which of the following is an incorrect syntax for a 1D array?**

* int a[10]
* int A[10]
* float 10[12]
* **int 10[var]**

**What is the formula to determine the size of an array?**

* Size = [(upper bound - lower bound) - 1]
* **Size = [(upper bound - lower bound) + 1]**
* Size = [(upper bound - lower bound)]
* None of these

**How do you calculate the size of an array in bytes (1 byte = 8 bits) where float = 4 bytes, integer = 2 bytes, and character = 1 byte for the array float BCSE[10]?**

* **Size of array = 10 \* size of (float)**
* Size of array = float \* size of (10)
* Size of array = 10 \* size of (int)
* None of these

**What is the correct syntax to allocate memory dynamically for an integer array of size 10?**

* \**ptr = (int ) malloc (10 \* size of (int))*
* ptr = (int \*) malloc (int \* size of (10))
* ptr = (float \*) malloc (10 \* size of (int))
* None of these

**Which pair is correct for reallocating memory in C?**

* **(a, b) & (c, d)**
* (b, c) & (a, d)
* (b, d) & (a, c)
* None of these

**Which of the following is the correct sequence for comparing time complexities?**

* O(1) < O(log n) < O(n²) < O(n log n)
* O(1) < O(log n) < O(n) < O(n log n)
* **O(1) < O(log n) < O(n) < O(n log n) < O(n²) < O(n³) < O(2n) < O(3n)**
* O(n log n) < O(n²) < O(n) < O(2n)

**Which match correctly associates asymptotic notations with their definitions?**

* **a-II, b-I, c-III, d-IV, e-V**
* a-I, b-II, c-III, d-IV, e-V
* a-III, b-II, c-II, d-V, e-IV
* a-IV, b-V, c-III, d-II, e-I

**Which statement is incorrect about a stack?**

* Non-primitive linear data structure.
* Working principle is LIFO (Last in First out) and can be implemented in two ways: static (using an array) and dynamic (using a linked list).
* Underflow is associated with Pop() operation and Overflow is associated with Push().
* **Push() operation == deletion and Pop() operation == insertion.**

**What is the correct sequence of operations on a stack?**

* After every pop operation, the stack is incremented by one.
* **After every push operation, the top is incremented by one.**
* After every push operation, the top is decremented by one.
* **After every pop operation, the top is decremented by one.**

**When is the queue considered empty?**

* Font == -1, Rear = 1
* **Font == -1, Rear == -1**
* Rear < Font
* Rear >= Font

**Which is not true about a linked list?**

* Linked list is a dynamic data structure that can grow or shrink during the execution of a program, and the START or Head pointer holds the address of the 1st node.
* Memory is not pre-allocated; memory is allocated whenever required, and NULL pointer indicates the end of the linked list held by the last node.
* It is deallocated when not needed, and a Linked List can be used to implement data structures such as Stack and Queue. When START = NULL, there is no list.
* **Searching in a Linked List is not done using linear search.**

**Which condition indicates a DEQUE overflow when inserting at the right side?**

* **Both a) & b) are correct.**
* a) is correct but b) is incorrect.
* Both a) & b) are incorrect.
* a) is incorrect but b) is correct.

**Which condition indicates a DEQUE underflow when deleting an element from the right side?**

* Both a) & b) are incorrect.
* a) is correct but b) is incorrect.
* **Both a) & b) are correct.**
* a) is incorrect but b) is correct.

**Which statement is incorrect about Linked Lists?**

* Time required to search inside a Linked List is O(n).
* Worst case time complexity to search inside a Sorted Linked List is O(n).
* **In a Circularly Linked List, concatenation using two lists can be done in O(1) time.**
* None of these

**How are elements accessed in an array?**

* Using a pointer
* Using the dot operator (.)
* **Using an index**
* Using a loop

**How are elements typically arranged in an array in programming?**

* Alphabetically
* Randomly
* **Sequentially**
* Reversely

**What is the purpose of an array index?**

* To store the element's value
* To specify the data type of the element
* **To represent the element's position in the array**
* To limit the size of the array

**From which value do most programming languages start array indices?**

* **0**
* 1
* -1
* The length of the array

**What operation allows changing the value of an element in an array?**

* Append
* **Modify**
* Sort
* Resize

**Which programming language does not support arrays?**

* **Python**
* C++
* Java
* JavaScript

**What is the primary advantage of using arrays in programming?**

* Efficient memory usage
* **Simplicity in data access**
* Enhanced security
* Dynamic resizing

**Which real-world analogy represents a stack?**

* **Stack of plates**
* Line of people
* Bucket of water
* Random assortment of items

**What does the "Peek" operation do in a stack?**

* **Get the top item without removing it.**
* Removes the top item
* Inserts an item at the bottom
* Reverses the stack

**What is an application of a stack?**

* **Expression conversion**
* Sorting numbers
* Audio playback
* Video editing

**What does expression conversion mean in the context of stacks?**

* **Changing the format of an expression**
* Evaluating an expression
* Converting numbers to text
* Creating a new expression

**What is an advantage of postfix notation in expressions?**

* **Eliminates the need for parentheses**
* Requires more memory
* Slower evaluation
* Complex to read

**What does expression evaluation mean in the context of stacks?**

* **Calculating the result of an expression**
* Converting an expression
* Changing variable names
* Printing the expression

**What is the primary drawback of using a linked list to implement a stack?**

* It has a fixed size.
* It is not efficient for pop operations.
* **It consumes more memory.**
* It cannot be used for expression evaluation.

**In postfix notation, in which order are operators applied to operands?**

* From left to right
* From right to left
* In the reverse order of appearance
* **According to operator precedence**

**What is the process of finding an element in a data structure?**

* Insertion
* Deletion
* **Searching**
* Sorting

**What term describes a data structure's ability to store multiple elements of the same type?**

* **Homogeneity**
* Abstraction
* Encapsulation
* Polymorphism

**What is the time complexity of the pop operation in a stack implemented using an array?**

* **O(1)**
* O(log n)
* O(n)
* O(n log n)

**Which data structure allows elements to be accessed in O(1) time using an index?**

* Linked list
* **Static array**
* Dynamic array
* None of these

**What operation is used to add an element to the end of a dynamic array when it reaches its current capacity?**

* **Appending**
* Inserting
* Deleting
* None of these

**What type of matrix has most of its elements as zero?**

* Identity Matrix
* **Sparse Matrix**
* Unit Matrix
* None of these

**Where are local variables stored?**

* **Stack**
* Free memory
* Permanent storage area
* Heap

**What header file must be included to use dynamic memory allocation functions?**

* **stdlib.h**
* stdio.h
* memory.h
* dos.h

**What is an advantage of using linked lists over arrays?**

* **Insertion and deletion of an element can be done at any position in a linked list**
* The size of a linked list is fixed
* Linked list is an example of linear data structure
* None of these

**Which operator has the highest order of precedence?**

* **\***
* #ERROR!
* None of these

**Which type of memory representation is used for a stack where the size can dynamically change as elements are added or removed?**

* **Dynamic Memory**
* Static Memory
* Heap Memory
* Cache Memory

**What type of queue inserts elements at the rear end and removes them from the front end?**

* Simple queue
* Linear Queue
* Circular Queue
* **Both a and b**

**What type of queue has elements with different priorities, where higher priority elements are dequeued first?**

* **Priority Queue**
* Simple queue
* Circular Queue
* Both a and b

**What type of queue has elements with different priorities, where lower priority elements are dequeued first?**

* **Priority Queue**
* Simple queue
* Circular Queue
* Both a and b

**What is the correct statement regarding postfix operators?**

* Postfix operators use value to their right
* **Postfix operators use value to their left**
* Prefix operators use value to their right
* In postfix expression, operands are followed by operators

**What is the result of the following postfix expression "ab*cd*+" where a=2, b=2, c=3, d=4?**

* **16**
* 12
* 14
* 1

**What is the value of the postfix expression "ab+cd/-" where a=5, b=4, c=9, d=3?**

* **6**
* 23
* 5
* 3

**What is the result of pushing 2, 3, and 4 onto an empty stack and then popping two elements?**

* **4, 3**
* 2, 3
* 3, 4
* 3, 2

**In a normal queue, where is insertion performed and where is deletion performed?**

* Front, Front
* Rear, Rear
* Front, Rear
* **Rear, Front**

**In a priority queue, where are elements with higher priority removed from?**

* Inserted first
* **Removed first**
* Inserted last
* Removed last

**What is the main disadvantage of a linear queue?**

* Slow insertion
* Slow deletion
* **Wastage of space due to fixed size**
* Overflow condition

**Which of the following is the most efficient way to store a sparse matrix?**

* 1D array
* 2D array
* Linked list
* **Compressed representation (like CSR or COO)**

**Which of the following matrices can be represented as a sparse matrix efficiently?**

* **Identity matrix**
* Dense matrix
* Matrix with more non-zero elements
* All matrices

**Which statement is not true about arrays in Data Structure?**

* **Arrays are passed by value to functions**
* Arrays can have multiple dimensions
* Arrays can store elements of the same type
* Arrays can be initialized when they are declared

**When is a dynamic array more suitable than a static array?**

* When the number of elements is fixed
* **When frequent resizing is required**
* When the array size is small
* When elements need to be accessed by index

**Which tree is height-balanced, where the heights of two child subtrees of any node differ by at most one?**

* Binary Tree
* **AVL Tree**
* Threaded Binary Tree
* B+ Tree

**Which tree allows both sequential access and efficient searching by adding extra links between nodes?**

* Binary Tree
* AVL Tree
* **Threaded Binary Tree**
* B Tree

**Which graph traversal algorithm explores all vertices at the present depth level before moving on to the next depth level?**

* Depth First Search (DFS)
* **Breadth First Search (BFS)**
* Dijkstra's Algorithm
* Kruskal’s Algorithm

**Which tree has at most two children per node?**

* B Tree
* AVL Tree
* **Binary Tree**
* Threaded Binary Tree

**Which tree maintains balance by performing rotations during insertions and deletions?**

* B+ Tree
* **AVL Tree**
* Threaded Binary Tree
* Binary Search Tree

**Which tree maintains pointers to the next node in traversal order to speed up in-order traversal?**

* Binary Tree
* AVL Tree
* B+ Tree
* **Threaded Binary Tree**

**Which graph traversal algorithm uses a queue data structure?**

* DFS
* Dijkstra's Algorithm
* **BFS**
* Prim's Algorithm

**Which search tree ensures all leaf nodes are at the same level and internal nodes have varying numbers of children?**

* Binary Tree
* AVL Tree
* **B Tree**
* Binary Search Tree

**Which traversal algorithm explores as far as possible along each branch before backtracking?**

* **DFS**
* BFS
* Prim’s Algorithm
* Kruskal’s Algorithm

**Which tree type allows each node to have at most two children, and all values in the left subtree are smaller than the node's value?**

* **Binary Search Tree**
* AVL Tree
* Threaded Binary Tree
* B+ Tree

**Which tree structure maintains a balanced property by allowing rotations during insertions and deletions?**

* Binary Tree
* **AVL Tree**
* Binary Search Tree
* B Tree

**Which tree structure has nodes linked in such a way that allows traversal in both forward and backward directions?**

* Binary Search Tree
* AVL Tree
* **Threaded Binary Tree**
* B+ Tree

**Which graph search algorithm uses a stack (explicitly or recursively) to explore nodes?**

* Breadth First Search (BFS)
* **Depth First Search (DFS)**
* Dijkstra's Algorithm
* Prim’s Algorithm

**Which traversal method visits nodes level by level starting from the root node?**

* Inorder
* Preorder
* Depth First Search
* **Breadth First Search**

**What happens when inserting an element into an AVL Tree where the balance factor exceeds 1 or -1?**

* No change
* **Tree is rotated**
* Tree becomes a Binary Search Tree
* Tree remains balanced

**What is the next node visited in a Depth First Search starting from the root node of a binary tree?**

* **Left child**
* Right child
* Parent node
* Last visited node

**Which tree does not guarantee balanced height but ensures that data is stored in a sorted manner?**

* AVL Tree
* **Binary Search Tree**
* B+ Tree
* B Tree

**Which algorithm continues exploring deeper paths before backtracking to explore other branches of the graph?**

* Breadth First Search
* **Depth First Search**
* Kruskal's Algorithm
* Dijkstra’s Algorithm

**Short Answer Question:**

1. What is the balancing factor in an AVL tree and how does it maintain the tree's balance?
2. What properties distinguish an AVL tree from a regular binary search tree (BST)?
3. How does the balancing of AVL trees occur after insertion or deletion, and why is this important?
4. What are the time complexities of common operations (insertion, deletion, and search) in an AVL tree compared to a regular BST?
5. What is the concept of a binary search tree (BST) and its key properties?
6. How do you construct a binary search tree from the following numbers: [10, 5, 15, 3, 7, 12, 18]?
7. How do you construct a binary search tree from the following numbers: [9, 4, 14, 2, 6, 11, 17]?
8. What is a graph and what are its main components?
9. What is a path in a graph? What is the difference between a simple path and a cycle?
10. What distinguishes a connected graph from a disconnected graph? Provide examples.
11. What is the primary purpose of a B-tree in database systems?
12. What is the difference between a B+ tree and a B-tree? What are their advantages for searching and range queries?
13. How is a new value inserted into a B-tree?
14. What is the difference between a directed graph (digraph) and an undirected graph? Provide examples.
15. What is the time complexity of the following nested loop?

for (int i=0; i<=n; i++) {

for (int j=0; j<=n; j++) {

Statement 1;

Statement 2;

...

Statement n;

}

}

1. What is the output of the following pseudo code when n=5?

int Factorial(int n) {

if (n == 0) return 1;

else return n \* factorial(n - 1);

}

1. Find F[9] in the Fibonacci sequence given F[0]=0, F[1]=1.
2. Predict the address of an element in a 1D array given the formula:

B = 2000 (Base Address), W = 4 (Size of each element), K = 5 (Element index)

Address of element A[K] = B + W \* K

1. What is the concept of a push operation in a stack?
2. What is the concept of a pop operation in a stack?
3. What is the EnQueue algorithm in a queue?
4. What is the DeQueue algorithm in a queue?
5. How do you check whether a circular queue is empty or full?
6. What is the difference between a circular queue and a regular queue?
7. What are the key advantages of using a linked list to implement a queue over an array?
8. What are the operations performed on stacks and queues?
9. What is the significance of preorder, postorder, level-order, or inorder tree traversals to check whether a binary tree is full?
10. Given the data, identify the preorder, inorder, and postorder sequences.

* S1: N, M, P, O, Q
* S2: N, P, Q, O, M
* S3: M, N, O, P, Q

1. What is the difference between BFS and DFS for solving a maze problem?
2. What are the applications of stack and queue in real-world problems?
3. What are the different types of queues and their uses?
4. How do you convert the infix expression A + B \* (C - D) to postfix notation using a stack?
5. Evaluate the postfix expression "5 3 \* 10 2 / +" using a stack.
6. Construct the postfix expression for "A \* B + C / D".
7. What is a sparse matrix and how is it represented? Provide an example.
8. What is the difference between a 1D and 2D array in data structures?
9. What are the key differences between a sparse matrix and a regular matrix?
10. Define asymptotic notations: Big-O, Omega, and Theta.
11. What is time complexity and space complexity?
12. Interpret the running time complexity for the following pseudo code:

for (int i=0; i<=n; i++) {

if (i % 2 == 0) {

i = i \* 2;

} else {

i = i - 2;

}

}

1. **What are self-balancing and non-self-balancing nonlinear data structures?**
2. **Why are self-balancing properties important in search trees?**
3. **What are the differences between balanced and unbalanced nonlinear data structures?**
4. **What is the significance of balancing in search trees, and how does it affect efficiency?**
5. **Why is the B-tree suitable for scenarios like file systems and databases?**
6. **Explain a threaded binary tree with an example.**
7. **What are the advantages of nonlinear data structures in computation?**
8. **What is a graph strongly connected or not, and how do you determine it?**
9. **How do you solve the maze problem using BFS or DFS?**
10. **What are the applications of stack and queue in various systems?**
11. **How is the BFS sequence deduced from a graph?**
12. **How is the DFS sequence deduced from a graph?**
13. **What is the sequence of operations when performing a series of pushes and pops on a stack?**

**Long Answer Question:**

* **What is AVL tree insertion and deletion with an example?**
* **What are the tree traversal techniques with an example?**
* **What is the process of balancing AVL trees after insertion or deletion?**
* **Why is balancing necessary in AVL trees?**
* **What is a binary search tree and how is it structured?**
* **What is the difference between a binary tree and a binary search tree (BST)?**
* **What is the time complexity of common operations in an AVL tree compared to a BST?**
* **What is the difference between AVL trees and binary search trees in terms of insertion, deletion, and searching?**
* **What is a graph and what are its main components?**
* **What is a path in a graph, and how is a simple path different from a cycle?**
* **What is the difference between a connected and a disconnected graph?**
* **What is a strongly connected graph?**
* **What is the difference between a directed graph (digraph) and an undirected graph?**
* **What is the purpose of the "front" and "rear" pointers in a queue?**
* **What are the EnQueue and DeQueue operations in a queue?**
* **What is the concept of a push and pop algorithm in a stack?**
* **What are the advantages of using a linked list to implement a queue over an array?**
* **What is the difference between a queue and a stack?**
* **What is the purpose of a priority queue and how does it differ from a regular queue?**
* **What is a circular queue and how is it different from a regular queue?**
* **What is a double-ended queue (deque) and when is it used?**
* **What is Big-O notation?**
* **What is the difference between time complexity and space complexity?**
* **What is Theta notation and when is it used?**
* **What is Omega notation with an example?**
* **What are different types of asymptotic notations with examples?**
* **What is the pseudo code for insertion into a stack?**
* **What is the pseudo code for popping from a stack?**
* **What is the pseudo code for adding an element to a queue?**
* **What is the pseudo code for inserting an element at the beginning of a linked list?**
* **What is the pseudo code for deleting an element from a linke-d list at any position?**
* **What is the pseudo code for inserting an element into a circular queue?**
* **What is the difference between an array and a linked list?**
* **What are the key advantages of using a linked list to implement a queue over an array?**
* **What is the concept of a circular queue and how does it differ from a regular queue?**
* **What are the different types of tree traversals, and when are they used?**
* **What is the difference between BFS and DFS?**
* **What is the algorithm for BFS using an example?**
* **What is the algorithm for DFS using an example?**
* **What is the difference between a binary tree and a binary search tree (BST)?**
* **What is the time complexity of searching, insertion, and deletion in a BST and AVL tree?**
* **What is the significance of balancing in search trees and how does it affect efficiency?**
* **What is the importance of self-balancing properties in search trees?**
* **What is the significance of balance in search trees?**
* **What is the primary purpose of a B-tree in database systems?**
* **What is the difference between a B-tree and a B+ tree?**
* **What is the purpose of threaded binary trees?**
* Explain the term "queue capacity" and its significance in queue data structures.
* Explain is a blocking queue? Provide an example of when it might be used.
* Explain the need for a double-ended queue (deque) and its usage scenarios.
* Construct a binary search tree for the following numbers start from an empty binary search tree. 45,26,10,60,70,30,40. Delete keys 10,60 and 45 one after the other and show the trees at each stage.
* Compute address in Row-major fashion using the formula.

Address of an element defined as:

Address of A [ Row] [ Col] = Base address + Size of each element [ No of columns (Row – Lower Bound of Row) + (Col- Lower bound of Col)]

where,

n= No of columns =U2 – L2+1,

L1 =Lower Bound of Row,

L2 = Lower Bound of Col,

U1 =Upper Bound of Row,

U2 =Lower Bound of Row

B=Base Address

W=Size of element

Given, Base Address = 100, a 2D array A [4…7, -1...3] requires 2 bytes of storage for each element, compute the address of element at location A [6,2]?

Hints: A [ Row] [ Col] = Base address + Size of each element [ No of columns (Row – Lower Bound of Row) + (Col- Lower bound of Col)]

Or A [ Row] [ Col] = B + W [n (Row – L1) + (Col- L2)]

* Compute address in Column-major fashion using the formula.

Address of an element defined as:

Address of A [ Row] [ Col] = Base address + Size of each element [ No of Rows (Col – Lower Bound of Col) + (Row- Lower Bound of Row)]

where,

m= No of rows =U1 – L1+1,

L1 =Lower Bound of Row,

L2 = Lower Bound of Col,

U1 =Upper Bound of Row,

U2 =Lower Bound of Row

B=Base Address

W=Size of element

Given, Base Address = 500, a 2D array A [-20…20, 10...35] requires 1 bytes of storage for each element, compute the address of element at location A [0,3]?

Hints: A [ Row] [ Col] = Base address + Size of each element [ No of columns (Col – Lower Bound of Col) + (Row- Lower bound of Row)]

Or A [Row] [ Col] = B + W [m (Col – L2) + (Row- L1)]

* Develop the pseudo code.

# define Max 5

int Stack [Max];

Top = -1 // When No elements exist inside the Stack

Void Push ( )

{

int item;

If (Top < Max -1 or Top == -1)

{

Top=……?… ;

Stack [?] = item;

}

Else:

---------?----------; // Hints: Either it may be Overflow or Underflow

}

* Develop the pseudo code.

# define Max 5

int Stack [Max];

Top = -1 // When No elements exist inside the Stack

Void Pop ( )

{

int item;

If (Top == -1)

{

……?.......;// Hints: Either it may be Overflow or Underflow

}

Else:

item =Stack [ …..?..... ];

Top = ……?.......;

}

* Develop the pseudo code for adding element inside the queue or Develop the partially completed pseudo code.

# define Queue [5];

int Font = -1;

int Rear = -1;

void Queue ()

{

int item;

If (Rear < Max -1)

{

printf (“Enter the Number :);

scanf (“%d”,& item);

If (Font == -1)

{

Font =……?.....;

Rear = ……?....;

}

Else

{

Rear =…...?.....; // Hints: either increment or decrement, choose the correct one

}

Queue [….?] =item //Hints: either Font or Rear, choose the correct one

}

Else

Printf (……?........) //Hints : either Over Flow or Under Flow, choose the correct one

}

* Design the pseudo code to insert an element at the beginning or develop the partially completed pseudo code.

struct node

{

int num;

struct node \*next;

} start =NULL;

void insert\_at\_begining (int item)

{

typedef struct node Node;

Node \*p;

P=(Node \*) malloc (sizeof (Node));

Pà….?.... =item;

If (start == NULL)

P ànext ==…?…;

Else

Pà next ==…?.....;

Start =….?...;

}