1. Linear Search

1. **What is the best-case time complexity of Linear Search?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n^2)
   Answer: A
2. **What is the worst-case time complexity of Linear Search?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n^2)
   Answer: B
3. **Linear Search works on:**
   A. Sorted arrays only
   B. Unsorted arrays only
   C. Both sorted and unsorted arrays
   D. None of the above
   Answer: C
4. **Which of the following is a limitation of Linear Search?**
   A. Requires a sorted array
   B. Inefficient for large datasets
   C. Cannot find duplicate elements
   D. None of the above
   Answer: B
5. **What is the space complexity of Linear Search?**
   A. O(n)
   B. O(log n)
   C. O(1)
   D. O(n^2)
   Answer: C
6. **Which of the following is true about Linear Search?**
   A. It is a greedy algorithm
   B. It works by dividing the array
   C. It sequentially checks each element
   D. It requires additional memory
   Answer: C
7. **What is the average-case time complexity of Linear Search?**
   A. O(n)
   B. O(n/2)
   C. O(log n)

D. O(1)
Answer: A

8. **In Linear Search, how many comparisons are needed in the worst case for an array of size 10?**
   A. 1
   B. 5
   C. 10
   D. 20
   Answer: C
9. **Linear Search is not preferred for large datasets because:**
   A. It has high time complexity
   B. It requires additional space
   C. It works only for integers
   D. None of the above
   Answer: A
10. **Linear Search is also known as:**
    A. Binary Search
    B. Sequential Search
    C. Divide and Conquer Search
    D. Greedy Search
    Answer: B

2. Binary Search

1. **What is the prerequisite for Binary Search?**
   A. The array must be unsorted
   B. The array must be sorted
   C. The array must contain only unique elements
   D. None of the above
   Answer: B
2. **What is the time complexity of Binary Search in the worst case?**
   A. O(1)
   B. O(log n)
   C. O(n)
   D. O(n log n)
   Answer: B
3. **What is the space complexity of Binary Search using recursion?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n^2)
   Answer: C

4. **What happens when Binary Search is applied to an unsorted array?**
   A. It works fine
   B. It gives incorrect results
   C. It may work sometimes
   D. None of the above
   Answer: B
5. **What is the best-case time complexity of Binary Search?**
   A. $O(1)$
   B. $O(\log n)$
   C. $O(n)$
   D. $O(n \log n)$
   Answer: A
6. **Which of the following is true about Binary Search?**
   A. It uses a Greedy Algorithm
   B. It uses Divide and Conquer
   C. It works on unsorted data
   D. It has $O(n)$ time complexity in the worst case
   Answer: B
7. **How many comparisons does Binary Search need in the worst case for an array of size 16?**
   A. 4
   B. 8
   C. 16
   D. 2
   Answer: A
8. **What is the recurrence relation for Binary Search?**
   A. $T(n) = T(n/2) + O(1)$
   B. $T(n) = T(n/2) + O(\log n)$
   C. $T(n) = T(n-1) + O(1)$
   D. $T(n) = T(n/2) + O(n)$
   Answer: A
9. **If the array size is doubled, the number of steps in Binary Search:**
   A. Doubles
   B. Increases linearly
   C. Increases logarithmically
   D. Remains the same
   Answer: C
10. **Binary Search is faster than Linear Search for large datasets because:**
    A. It scans the entire array
    B. It reduces the search space by half in each step
    C. It uses dynamic programming
    D. None of the above
    Answer: B

3. Ternary Search

1. **What is the key difference between Binary Search and Ternary Search?**
   A. Binary Search divides into 3 parts, Ternary Search into 2
   B. Ternary Search divides into 3 parts, Binary Search into 2
   C. Ternary Search uses dynamic programming
   D. None of the above
   Answer: B
2. **What is the time complexity of Ternary Search in the worst case?**
   A. O(1)
   B. O(log n base 2)
   C. O(log n base 3)
   D. O(n)
   Answer: C
3. **What is the best-case time complexity of Ternary Search?**
   A. O(1)
   B. O(log n base 2)
   C. O(log n base 3)
   D. O(n)
   Answer: A
4. **Ternary Search is most suitable for:**
   A. Unsorted arrays
   B. Functions with a single peak or valley
   C. Large unsorted datasets
   D. None of the above
   Answer: B
5. **The recurrence relation for Ternary Search is:**
   A. T(n) = T(n/3) + O(1)
   B. T(n) = T(2n/3) + O(1)
   C. T(n) = T(n/3) + O(n)
   D. T(n) = T(n/2) + O(1)
   Answer: A
6. **What is the space complexity of Ternary Search?**
   A. O(1)
   B. O(log n base 3)
   C. O(n)
   D. O(log n base 2)
   Answer: A
7. **Ternary Search divides the array into:**
   A. Two equal parts
   B. Three equal parts
   C. Four equal parts

D. None of the above
Answer: B

8. **Ternary Search is primarily used for:**
   A. Binary trees
   B. Optimizing unimodal functions
   C. Sorting algorithms
   D. Large-scale database queries
   Answer: B
9. **If the array size is 81, how many comparisons are needed in the worst case using Ternary Search?**
   A. 4
   B. 3
   C. 5
   D. 6
   Answer: C
10. **Ternary Search is slower than Binary Search because:**
    A. It has higher overhead in splitting the array
    B. It processes fewer elements per step
    C. It requires sorting
    D. None of the above
    Answer: A

4. Merge Sort

1. **Merge Sort is based on which algorithmic technique?**
   A. Greedy
   B. Divide and Conquer
   C. Dynamic Programming
   D. Backtracking
   Answer: B
2. **What is the best-case time complexity of Merge Sort?**
   A. O(n)
   B. O(n log n)
   C. O(log n)
   D. O(n^2)
   Answer: B
3. **What is the worst-case time complexity of Merge Sort?**
   A. O(n)
   B. O(n log n)
   C. O(log n)
   D. O(n^2)
   Answer: B
4. **What is the space complexity of Merge Sort?**
   A. O(1)

B. O(n)
C. O(log n)
D. O(n log n)
Answer: B

5. **What is the main advantage of Merge Sort over Quick Sort?**
   A. Better average-case complexity
   B. Works well with small datasets
   C. Stable sorting
   D. Less memory usage
   Answer: C
6. **Merge Sort is more efficient than Bubble Sort for:**
   A. Small datasets
   B. Sorted datasets
   C. Large datasets
   D. None of the above
   Answer: C
7. **What is the recurrence relation for Merge Sort?**
   A. T(n) = 2T(n/2) + O(n)
   B. T(n) = T(n/2) + O(n)
   C. T(n) = T(n-1) + O(1)
   D. T(n) = T(n) + O(n)
   Answer: A
8. **In Merge Sort, merging two sorted arrays requires:**
   A. O(1) time
   B. O(log n) time
   C. O(n) time
   D. O(n^2) time
   Answer: C
9. **What type of problems is Merge Sort most suited for?**
   A. In-place sorting
   B. Sorting linked lists
   C. Sorting small datasets
   D. Finding duplicates
   Answer: B
10. **If the input size is 32, how many levels will the recursion tree of Merge Sort have?**
    A. 4
    B. 5
    C. 6
    D. 8
    Answer: B

5. Quick Sort

1. **Quick Sort is based on which algorithmic technique?**
   A. Dynamic Programming
   B. Divide and Conquer
   C. Backtracking
   D. Greedy
   Answer: B
2. **What is the best-case time complexity of Quick Sort?**
   A. O(n)
   B. O(n log n)
   C. O(n^2)
   D. O(log n)
   Answer: B
3. **What is the worst-case time complexity of Quick Sort?**
   A. O(n log n)
   B. O(n)
   C. O(log n)
   D. O(n^2)
   Answer: D
4. **What is the average-case time complexity of Quick Sort?**
   A. O(n)
   B. O(n log n)
   C. O(log n)
   D. O(n^2)
   Answer: B
5. **What is the space complexity of Quick Sort for in-place sorting?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n log n)
   Answer: C
6. **Which element is typically chosen as the pivot in Quick Sort?**
   A. The first element
   B. The last element
   C. The middle element
   D. Any of the above
   Answer: D
7. **Quick Sort is preferred over Merge Sort when:**
   A. Memory is limited
   B. Stability is required
   C. The dataset is linked
   D. None of the above
   Answer: A

8. **What is the recurrence relation for Quick Sort in the average case?**
   A. T(n) = T(n/2) + O(n)
   B. T(n) = 2T(n/2) + O(n)
   C. T(n) = T(n/4) + O(n log n)
   D. T(n) = T(n-1) + O(n)
   Answer: A
9. **The partitioning step in Quick Sort has a time complexity of:**
   A. O(1)
   B. O(log n)
   C. O(n)
   D. O(n^2)
   Answer: C
10. **If Quick Sort is implemented with random pivot selection, the expected time complexity is:**
   A. O(n)
   B. O(n log n)
   C. O(n^2)
   D. O(log n)
   Answer: B

6. Longest Common Subsequence (LCS)

1. **What is the time complexity of LCS using dynamic programming?**
   A. O(m+n)
   B. O(m*n)
   C. O(2^n)
   D. O(m^2 * n^2)
   Answer: B
2. **What does LCS stand for?**
   A. Longest Common Substring
   B. Longest Contiguous Subsequence
   C. Longest Common Subsequence
   D. Largest Consecutive Subset
   Answer: C
3. **LCS is useful for:**
   A. Finding the shortest path
   B. Comparing two sequences
   C. Sorting an array
   D. None of the above
   Answer: B
4. **What is the base case in the DP solution for LCS?**
   A. When one string is empty
   B. When both strings are of equal length
   C. When both strings are identical

D. None of the above
Answer: A

5. **The space complexity of the iterative DP LCS implementation is:**
   A. O(m + n)
   B. O(m * n)
   C. O(max(m, n))
   D. O(1)
   Answer: B
6. **Which property does LCS rely on to compute its solution?**
   A. Divide and Conquer
   B. Overlapping Subproblems
   C. Greedy Strategy
   D. Backtracking
   Answer: B
7. **If the strings are "ABC" and "ABD," what is the length of their LCS?**
   A. 2
   B. 3
   C. 1
   D. 0
   Answer: A
8. **LCS can be used in which of the following applications?**
   A. DNA sequence alignment
   B. Text comparison
   C. File difference checking
   D. All of the above
   Answer: D
9. **The recurrence relation for LCS is:**
   A. LCS(i, j) = max(LCS(i-1, j), LCS(i, j-1))
   B. LCS(i, j) = LCS(i-1, j-1) + 1 (if x[i] == y[j])
   C. Both A and B
   D. None of the above
   Answer: C
10. **What happens when two strings have no common subsequence?**
    A. The LCS length is -1
    B. The LCS length is 0
    C. The LCS length is 1
    D. None of the above
    Answer: B

7. 0/1 Knapsack Problem

1. **What type of algorithmic technique is used to solve the 0/1 Knapsack problem optimally?**
   A. Greedy Algorithm
   B. Dynamic Programming
   C. Divide and Conquer
   D. Backtracking
   Answer: B
2. **The 0/1 Knapsack problem is called so because:**
   A. Items can be partially included
   B. Items can be completely included or excluded
   C. It requires a binary search
   D. None of the above
   Answer: B
3. **What is the time complexity of the 0/1 Knapsack problem using dynamic programming?**
   A. O(n log n)
   B. O(nW)
   C. O(2^n)
   D. O(n^2)
   Answer: B
   (where nnn is the number of items and WWW is the capacity of the knapsack)
4. **What is the space complexity of the 0/1 Knapsack problem using a 2D DP table?**
   A. O(n)
   B. O(W)
   C. O(nW)
   D. O(1)
   Answer: C
5. **What is the recurrence relation for solving the 0/1 Knapsack problem?**
   A. T(n)=T(n−1)+WT(n) = T(n-1) + WT(n)=T(n−1)+W
   B. dp[i][w]=max(dp[i−1][w],dp[i−1][w−wt[i]]+val[i])dp[i][w] = \max(dp[i-1][w], dp[i-1][w-wt[i]] + val[i])dp[i][w]=max(dp[i−1][w],dp[i−1][w−wt[i]]+val[i])
   C. dp[i][w]=dp[i−1][w]+wt[i]dp[i][w] = dp[i-1][w] + wt[i]dp[i][w]=dp[i−1][w]+wt[i]
   D. None of the above
   Answer: B
6. **Which of the following is true about the 0/1 Knapsack problem?**
   A. It can always be solved using a greedy approach
   B. It cannot be solved using recursion
   C. It can be solved using dynamic programming or recursion
   D. It is faster than Fractional Knapsack
   Answer: C
7. **In the 0/1 Knapsack problem, if the capacity W=0W = 0W=0, the maximum profit is:**
   A. Infinity

B. 0
C. Sum of item values
D. Not defined
Answer: B

8. **The solution to the 0/1 Knapsack problem involves:**
   A. Selecting items with the highest value-to-weight ratio first
   B. Considering each item for inclusion or exclusion
   C. Dividing the items into two subsets
   D. Sorting items by value
   Answer: B
9. **What is the output of the 0/1 Knapsack problem?**
   A. A list of selected items only
   B. The maximum possible profit only
   C. Both selected items and maximum profit
   D. Total weight of the selected items
   Answer: C
10. **Which of the following problems is closely related to the 0/1 Knapsack problem?**
   A. Longest Common Subsequence
   B. Subset Sum Problem
   C. Fractional Knapsack Problem
   D. Shortest Path Problem
   Answer: B
11. **Which data structure is most commonly used to implement the dynamic programming solution for the 0/1 Knapsack problem?**
   A. Stack
   B. Queue
   C. 2D Array
   D. Binary Tree
   Answer: C
12. **In the 0/1 Knapsack problem, the greedy approach fails when:**
   A. All items have the same weight
   B. All items have the same value
   C. The item with the highest value-to-weight ratio is not part of the optimal solution
   D. The knapsack capacity is greater than the sum of weights
   Answer: C
13. **The optimal substructure property of the 0/1 Knapsack problem means:**
   A. It can be solved using divide and conquer
   B. The solution of a subproblem is part of the solution of the overall problem
   C. It cannot be solved using recursion
   D. The problem has overlapping subproblems
   Answer: B
14. **How can the space complexity of the 0/1 Knapsack problem be reduced from O(nW)O(nW)O(nW)?**
   A. By using a greedy approach

B. By using a 1D DP array
C. By pre-sorting the items
D. By using binary search
Answer: B

15. **If there are 5 items and the capacity of the knapsack is 10, the size of the DP table in a dynamic programming solution will be:**
A. 5 x 10
B. 6 x 11
C. 5 x 11
D. 6 x 10
Answer: B

8. Factorial

1. **What is the time complexity of the iterative approach to calculate factorial?**
A. O(n!)
B. O(log n)
C. O(n)
D. O(1)
Answer: C
2. **Which of the following is the base case in recursive factorial implementation?**
A. n == 1
B. n == 0
C. Both A and B
D. None of the above
Answer: C
3. **Which data structure is used to maintain function calls during recursion?**
A. Stack
B. Queue
C. Linked List
D. Heap
Answer: A
4. **What happens if the factorial function is called with a negative number?**
A. It returns a positive value
B. It throws an error
C. It results in infinite recursion
D. None of the above
Answer: C
5. **Factorial of 0 is:**
A. Undefined
B. 1
C. 0
D. Infinite
Answer: B

6. **Factorial grows at which rate?**
   A. Linear
   B. Polynomial
   C. Exponential
   D. Logarithmic
   Answer: C
7. **What is the result of 5! (5 factorial)?**
   A. 100
   B. 120
   C. 150
   D. 110
   Answer: B
8. **Which of the following applications uses factorial?**
   A. Graph Traversal
   B. Permutations and Combinations
   C. Sorting
   D. Searching
   Answer: B
9. **Which is true for the space complexity of a recursive factorial function?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n^2)
   Answer: B
10. **What is the factorial of 1?**
   A. 1
   B. 0
   C. Undefined
   D. None of the above
   A**nswer: A**

9. Fibonacci

1. **The time complexity of naive recursion for Fibonacci is:**
   A. O(n)
   B. O(2^n)
   C. O(log n)
   D. O(n^2)
   Answer: B
2. **Dynamic Programming reduces Fibonacci time complexity to:**
   A. O(n)
   B. O(log n)
   C. O(2^n)

D. O(n^2)
Answer: A

3. **What is the Fibonacci number at position 0?**
A. 1
B. 0
C. Undefined
D. None of the above
Answer: B
4. **What is the Fibonacci number at position 1?**
A. 0
B. 1
C. 2
D. None of the above
Answer: B
5. **Which formula represents the Fibonacci sequence?**
A. F(n) = F(n-1) + F(n-2)
B. F(n) = F(n+1) + F(n-1)
C. F(n) = 2 * F(n-1)
D. F(n) = n * F(n-1)
Answer: A
6. **Space complexity of dynamic programming implementation for Fibonacci is:**
A. O(1)
B. O(n)
C. O(log n)
D. O(n^2)
Answer: B
7. **How can Fibonacci be optimized further than O(n)?**
A. Using matrix exponentiation
B. Using recursion
C. Using backtracking
D. It cannot be optimized further
Answer: A
8. **Fibonacci sequence grows at what rate?**
A. Linearly
B. Exponentially
C. Quadratically
D. Logarithmically
Answer: B
9. **Which approach is most efficient for large Fibonacci numbers?**
A. Iterative
B. Recursive
C. Dynamic Programming
D. Matrix Exponentiation
Answer: D

10. **The 5th Fibonacci number is:**
A. 3
B. 5
C. 8
D. 13
Answer: B

10. Fractional Knapsack

1. **Fractional Knapsack is different from 0/1 Knapsack because:**
A. Items can be taken partially
B. It is solved using dynamic programming
C. It uses the Divide and Conquer approach
D. None of the above
Answer: A
2. **The time complexity of the Fractional Knapsack algorithm is:**
A. O(n log n)
B. O(n^2)
C. O(2^n)
D. O(log n)
Answer: A
3. **Fractional Knapsack is solved using which approach?**
A. Dynamic Programming
B. Backtracking
C. Greedy Algorithm
D. Divide and Conquer
Answer: C
4. **The property used to maximize profit in Fractional Knapsack is:**
A. Maximum weight
B. Maximum profit
C. Maximum profit-to-weight ratio
D. Minimum weight
Answer: C
5. **If the capacity of the knapsack is exceeded, the item is:**
A. Ignored
B. Partially included
C. Fully included
D. None of the above
Answer: B
6. **Which of the following applications uses Fractional Knapsack?**
A. Resource Allocation
B. Task Scheduling
C. Investment Planning

D. All of the above
Answer: D

7. **Which is true about the Fractional Knapsack algorithm?**
A. It always gives an optimal solution
B. It does not always guarantee an optimal solution
C. It works only for integer weights
D. None of the above
Answer: A
8. **In Fractional Knapsack, if an item has a profit of 100 and weight of 50, its profit-to-weight ratio is:**
A. 2
B. 50
C. 0.5
D. 100
Answer: A
9. **Which sorting criterion is used in Fractional Knapsack?**
A. Profit
B. Weight
C. Profit-to-weight ratio
D. None of the above
Answer: C
10. **What happens when the knapsack is completely filled?**
A. Remaining items are ignored
B. Remaining items are partially considered
C. Remaining items are fully included
D. None of the above
Answer: A

11. Coin Change

1. **The time complexity of the Coin Change problem using dynamic programming is:**
A. O(amount * n)
B. O(2^n)
C. O(n^2)
D. O(amount log n)
Answer: A
2. **What is the minimum value returned when no combination of coins can make up the amount?**
A. -1
B. Infinity
C. 0
D. None of the above
Answer: A

3. **Which paradigm is used in the Coin Change problem?**
   A. Greedy Algorithm
   B. Divide and Conquer
   C. Dynamic Programming
   D. Backtracking
   Answer: C
4. **What is the base case in the Coin Change problem?**
   A. If amount = 0
   B. If no coins are left
   C. Both A and B
   D. None of the above
   Answer: A
5. **Which property does the DP solution for Coin Change rely on?**
   A. Overlapping Subproblems
   B. Divide and Conquer
   C. Greedy Strategy
   D. Backtracking
   Answer: A
6. **If coins are {1, 2, 5} and the amount is 11, what is the minimum number of coins needed?**
   A. 5
   B. 3
   C. 2
   D. 4
   Answer: B
7. **What is the time complexity of the Coin Change problem using a recursive solution?**
   A. O(n^2)
   B. O(2^amount)
   C. O(n * amount)
   D. O(log n)
   Answer: B
8. **The greedy algorithm for Coin Change always works when:**
   A. Coins are in any order
   B. Coins are divisible by each other
   C. The coin denominations are powers of 2
   D. Both B and C
   Answer: D
9. **In the Coin Change problem, if the denominations are {2, 3, 7} and the amount is 12, what is the minimum number of coins required?**
   A. 4
   B. 3
   C. 5

D. 2
Answer: B

10. **Which of the following problems is similar to Coin Change?**
A. 0/1 Knapsack
B. Fractional Knapsack
C. Subset Sum Problem
D. Longest Common Subsequence
Answer: C

12. Breadth-First Search (BFS)

1. **BFS is based on which data structure?**
A. Stack
B. Queue
C. Priority Queue
D. Linked List
Answer: B
2. **What is the time complexity of BFS for a graph with VVV vertices and EEE edges?**
A. O(V)
B. O(E)
C. O(V + E)
D. O(VE)
Answer: C
3. **BFS is typically used for:**
A. Finding shortest path in unweighted graphs
B. Topological sorting
C. Depth-first traversal
D. Minimum spanning tree
Answer: A
4. **Which of the following applications can use BFS?**
A. Solving mazes
B. Detecting cycles in undirected graphs
C. Finding connected components
D. All of the above
Answer: D
5. **Which of the following is NOT a property of BFS?**
A. BFS always finds the shortest path in an unweighted graph
B. BFS uses recursion
C. BFS requires a queue to track vertices
D. BFS visits all vertices at the same depth level before going deeper
Answer: B

6. **How does BFS handle disconnected graphs?**
   A. By marking visited nodes
   B. By starting BFS from each unvisited node
   C. By ignoring disconnected components
   D. By running DFS instead
   Answer: B
7. **What is the space complexity of BFS?**
   A. O(V)
   B. O(E)
   C. O(V + E)
   D. O(V^2)
   Answer: A
8. **If BFS is applied on a tree, it is also known as:**
   A. Depth-first traversal
   B. Level-order traversal
   C. Pre-order traversal
   D. Post-order traversal
   Answer: B
9. **Which of the following statements is true about BFS?**
   A. BFS works better with weighted graphs
   B. BFS requires backtracking
   C. BFS uses a FIFO queue
   D. BFS is a recursive algorithm
   Answer: C
10. **If a graph has VVV vertices and no edges, BFS will:**
   A. Visit all vertices
   B. Visit no vertices
   C. Visit only the start vertex
   D. Result in an error
   Answer: A

13. Depth-First Search (DFS)

1. **DFS is based on which data structure?**
   A. Queue
   B. Stack
   C. Priority Queue
   D. Linked List
   Answer: B
2. **What is the time complexity of DFS for a graph with VVV vertices and EEE edges?**
   A. O(V)
   B. O(E)
   C. O(V + E)

D. O(VE)
Answer: C

3. **DFS is typically used for:**
   A. Finding shortest paths in unweighted graphs
   B. Detecting cycles in a graph
   C. Finding the minimum spanning tree
   D. Breadth-first traversal
   Answer: B
4. **Which of the following problems can DFS solve efficiently?**
   A. Topological sorting
   B. Strongly connected components
   C. Path existence
   D. All of the above
   Answer: D
5. **DFS can be implemented using:**
   A. Recursion
   B. Iteration with a stack
   C. Both A and B
   D. Neither A nor B
   Answer: C
6. **What is the space complexity of DFS in its recursive form?**
   A. O(1)
   B. O(V)
   C. O(E)
   D. O(V + E)
   Answer: B
7. **If DFS is applied on a tree, it is equivalent to:**
   A. Level-order traversal
   B. Pre-order, In-order, or Post-order traversal
   C. Random-order traversal
   D. None of the above
   Answer: B
8. **In DFS, the visited nodes are typically marked to:**
   A. Avoid cycles
   B. Improve time complexity
   C. Avoid revisiting nodes
   D. All of the above
   Answer: D
9. **In a directed graph, DFS can be used to:**
   A. Detect back edges
   B. Identify articulation points
   C. Detect strongly connected components
   D. All of the above
   Answer: D

10. **If DFS is applied to a graph with no edges, the number of connected components is:**
    A. 0
    B. 1
    C. VVV
    D. Undefined
    Answer: C

14. Dijkstra's Algorithm

1. **Dijkstra’s algorithm is used to find:**
   A. Minimum Spanning Tree
   B. Shortest Path in a graph
   C. Longest Path in a graph
   D. Strongly Connected Components
   Answer: B
2. **Dijkstra's algorithm works on which type of graph?**
   A. Graphs with negative edge weights
   B. Graphs with positive edge weights only
   C. Undirected graphs
   D. Both B and C
   Answer: D
3. **The data structure commonly used to implement Dijkstra’s algorithm is:**
   A. Stack
   B. Queue
   C. Priority Queue
   D. Binary Search Tree
   Answer: C
4. **The time complexity of Dijkstra’s algorithm with a priority queue and adjacency list is:**
   A. O(V^2)
   B. O(V + E log V)
   C. O(V log E)
   D. O(E log V)
   Answer: B
5. **What is the initialization step in Dijkstra's algorithm?**
   A. All distances are set to 0
   B. All distances are set to infinity except the source
   C. All nodes are marked as visited
   D. All edges are sorted by weight
   Answer: B
6. **Which property does Dijkstra's algorithm rely on?**
   A. Greedy Strategy
   B. Dynamic Programming

C. Divide and Conquer
D. Backtracking
Answer: A

7. **Dijkstra's algorithm can fail to give the correct solution if the graph contains:**
A. Self-loops
B. Negative edge weights
C. Parallel edges
D. None of the above
Answer: B
8. **What is the stopping condition for Dijkstra's algorithm?**
A. All nodes are visited
B. All edges are processed
C. The shortest path to the destination node is found
D. Both A and C
Answer: D
9. **What is the primary difference between Dijkstra's and Bellman-Ford algorithms?**
A. Dijkstra's works with negative weights, Bellman-Ford does not
B. Bellman-Ford works with negative weights, Dijkstra's does not
C. Dijkstra's is slower than Bellman-Ford
D. Bellman-Ford uses a priority queue
Answer: B
10. **If there are V vertices and E edges, the space complexity of Dijkstra's algorithm is:**
A. O(V^2)
B. O(V + E)
C. O(V log V)
D. O(VE)
Answer: B

15. Kruskal's Algorithm

1. **Kruskal's algorithm is used to find:**
A. Shortest Path
B. Minimum Spanning Tree
C. Longest Path
D. Strongly Connected Components
Answer: B
2. **What type of graph does Kruskal's algorithm work on?**
A. Directed Graphs
B. Undirected Graphs
C. Both A and B
D. None of the above
Answer: B
3. **What is the time complexity of Kruskal's algorithm when using Union-Find?**
A. O(V^2)

B. O(E log E)
C. O(E log V)
D. O(V log E)
Answer: B

4. **Which sorting algorithm is commonly used in Kruskal's algorithm?**
A. Bubble Sort
B. Merge Sort
C. Quick Sort
D. Any efficient sorting algorithm
Answer: D
5. **The Kruskal algorithm uses which data structure to detect cycles?**
A. Priority Queue
B. Union-Find (Disjoint Set)
C. Adjacency Matrix
D. Graph Coloring
Answer: B
6. **The first step in Kruskal's algorithm is to:**
A. Sort all edges by their weight
B. Pick the heaviest edge
C. Initialize a single vertex
D. Use DFS to explore the graph
Answer: A
7. **In Kruskal's algorithm, how are edges added to the MST?**
A. Randomly
B. Based on weight (smallest first)
C. Based on vertex order
D. None of the above
Answer: B
8. **Kruskal's algorithm terminates when:**
A. All vertices are connected
B. All edges are processed
C. Cycle detection fails
D. A specific weight is reached
Answer: A
9. **What is the main difference between Kruskal's and Prim's algorithms?**
A. Kruskal's uses adjacency matrix
B. Kruskal's works edge by edge, Prim's works vertex by vertex
C. Kruskal's is for directed graphs only
D. Prim's does not guarantee an MST
Answer: B
10. **What type of approach does Kruskal's algorithm follow?**
A. Dynamic Programming
B. Greedy Algorithm
C. Divide and Conquer

D. Backtracking

Answer: B