One Line Questions

1. Define the types of algorithms commonly used in computer science
2. What is an algorithm, and what are its primary properties?
3. How do you determine the lower, upper, and tight bounds of an algorithm?
4. What are the key parameters to consider when writing an algorithm analysis?
5. In the Fractional Knapsack problem, how does the Greedy algorithm choose items?
6. What are the key parameters to consider when writing an algorithm analysis?
7. How do you analyze control statements and loop invariants to ensure the correctness of an algorithm
8. How does the master theorem apply to solve divide-and-conquer recurrences of the form $T(n)=aT(n/b)+f(n)$)?
9. Which of the following problems is typically solved using a greedy algorithm? a) Traveling Salesman Problem b) Knapsack Problem (0/1) c) Minimum Spanning Tree d) N-Queens Problem
10. What is the primary use of Huffman coding? a) Sorting data b) Searching data c) Data compression d) Data encryption
11. What does "stable sorting algorithm" mean? a) The algorithm never crashes b) The relative order of equal elements is preserved c) The performance is consistent across all inputs d) The algorithm uses minimal memory
12. What is the time complexity of Kruskal's algorithm for finding the Minimum Spanning Tree? a) $O(E \log V)$ b) $O(V^2)$ c) $O(E + V \log V)$ d) $O(V + E)$
13. In the Fractional Knapsack problem, how does the Greedy algorithm choose items? a) By weight b) By value c) By value-to-weight ratio d) Both a and b
14. Which sorting algorithm follows the "divide and conquer" paradigm?
15. Which type of complexity provides the best-case scenario for an algorithm's performance?
16. Which is the following is a characteristic of divide and conquer algorithms?
17. Consider a Binary Search algorithm that searches for an element in a sorted array of size n. If the element is not found, what is the maximum number of comparisons required?
18. Which algorithm is used to find the shortest path from a single source vertex to all other vertices in a weighted graph with non-negative edge weights?
19. Suppose you have coins of denominations 1,3 and 4. You use a greedy algorithm, in which you choose the largest denomination coin which is not greater than the remaining
20. Consider a weighted undirected graph with 6 vertices and 9 edges. Which of the following problems cannot be directly solved using a Greedy Strategy?

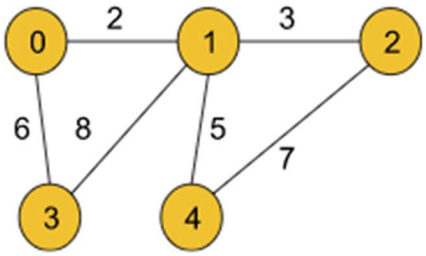
Fill in the blanks

1. The Max-Min problem aims to find the maximum and minimum elements in an array with the minimum number of _____.
2. Greedy algorithms make the _____ choice at each step with the hope of finding the _____ solution.

3. The _____ case represents the least time taken by an algorithm, while the _____ case represents the maximum time.
4. The greedy solution to the fractional knapsack problem involves taking items in order of their _____ value per _____.
5. Big Oh (O) notation describes the _____ bound of an algorithm's time complexity.
6. Huffman coding is a greedy algorithm used for lossless _____ compression.
7. The primary properties of an algorithm include _____, _____, finiteness, definiteness, and effectiveness.
8. Dijkstra's algorithm finds the shortest path from a starting node to all other nodes in a graph with _____ edge weights.
9. Strassen's matrix multiplication is applicable to ----------
10. The recurrence relation for the time complexity of the Merge Sort algorithm is T(n)=2T(n/2)+---------
11. Quick sort selects a _____ element, partitions the array, and recursively sorts the subarrays.
12. The notation O(n) is used to describe the _____ case time complexity of an algorithm.
13. The time required by an algorithm to complete as a function of the size of the input is referred to as its _____ complexity.
14. _____ profiling is a dynamic program analysis that measures where a program spends its time or which functions consume the most resources
15. Benchmarking involves running a program or algorithm on a set of _____ tasks or inputs to evaluate its performance.
16. The conquer step in the Max-Min problem involves finding the maximum and minimum elements in each half of the -----
17. Huffman coding is a method of _____ data compression that uses variable-length codes to represent symbols with shorter codes for more frequent symbols.
18. The Knapsack Problem involves selecting a subset of items to maximize the total _____, while not exceeding a given weight constraint.
19. Dijkstra's algorithm is particularly useful for finding the shortest path in a _____ graph.
20. In the Activity Selection Problem, a greedy algorithm selects activities based on their _____ time

Q2. 3 mark questions

1. Describe the main idea behind the Divide and-Conquer strategy and give an example of an algorithm that uses this approach.
2. What are the different types of algorithms? Provide examples for each type.
3. Explain the parameters used in algorithm analysis. How do they impact the evaluation of an algorithm's efficiency?
4. Explain the recursion tree method with a simple example.
5. Explain what is a minimum spanning tree. Use Prim’s algorithm to compute the minimum spanning tree for the following graph

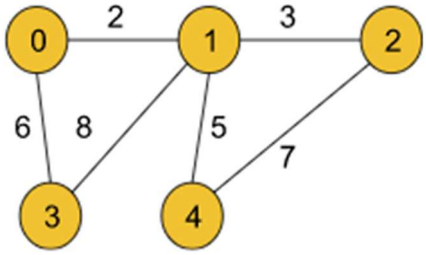


6. Describe the Knapsack Problem and the Greedy approach to solving it. What is the limitation of the Greedy approach in this context?
7. In a candy store, we have N different kinds of candies each of which has a particular price. For every candy bought from the store, K other candies (of different types) are given for free Write a Python program to compute: a) Minimum amount of money required to buy all N different candies b) Maximum amount of money required to buy all N different candies c) Return both the values as a tuple
   Input: K = 2, candies = [3 2 1 4]
   Output: 3,7

   NOTE: In both the cases you must utilize the offer i.e. you buy one candy and get K other candies for free.
8. Explain the Insertion Sort algorithm and its time complexity in the worst-case and best case scenarios.
9. There is one meeting room in a firm. There are n meetings in the form of (start[i], end[i]) where start[i] is start time of meeting i and end[i] is finish time of meeting i. Write a Python program to determine what is the maximum number of meetings that can be accommodated in the meeting room when only one meeting can be held in the meeting room at a particular time?
   Input: start = [1,3,0,8,5,5], end = [2,4,6,9,7,9]
   Output:4
   Note: Start time of one chosen meeting can't be equal to the end time of the other chosen meeting.
10. What are the ways of finding a pivot element for a quick sort algorithm? Define a function find_pivot() which takes the input array, start, end and choice as the parameters and returns the appropriate pivot as per the choice made among the options defined in the first part of the question.
11. What is the purpose of Strassen's multiplication? Describe in detail the formula for the algorithm and calculate the total number of matrix multiplications and additions involved. Derive the recurrence relation step by step. Use master's theorem to calculate the time complexity. Compare and contrast it with the traditional matrix multiplication method
12. Derive the recurrence relation for Binary Search and solve it using Master's theorem. Explain what are the situations wrt the actual position of the target element in the given array in which the best case, worst case occurs
13. Can you explain the O(n) part of the merge operation?
14. Discuss the key characteristics of problems that are well-suited for solving using greedy algorithms.
15. Evaluate the feasibility of using a quadratic time complexity algorithm for a dataset with one million elements.

Q3. 4 Mark questions

1. Analyze the time complexity and space complexity of the Strassen Multiplication algorithm for matrix multiplication. Compare it with the traditional matrix multiplication algorithm in terms of efficiency and practical implementation.
2. Compare the Quick Sort algorithm with Merge Sort in terms of worst-case and average-case time complexities. Explain how Quick Sort's performance can be improved with different pivot selection strategies.
3. Illustrate the process of Huffman Coding for data compression. Given a set of character frequencies, construct the Huffman tree and generate the Huffman codes for each character. Analyze the efficiency of Huffman Coding compared to fixed-length encoding.
4. Explain what is a minimum spanning tree. Use Kruskal's algorithm to compute the minimum spanning tree for the following graph



5. Explain the Activity Selection Problem and describe the greedy approach to solve it. Why does the greedy strategy work for this problem?
6. Huffman Coding

   For the word, "mississippi",

   1. Create a frequency table
   2. Build a huffman tree
   3. Create an encoding and decoding map using the huffman tree
   4. Calculate the compression ratio for the word after huffman encoding

7. Huffman Coding

The characters a to h have the set of frequencies based on the first 8 Fibonacci numbers as follows: a : 1, b : 1, c : 2, d : 3, e : 5, f : 8, g : 13, h : 21

A Huffman code is used to represent the characters. What is the sequence of characters corresponding to the following code: 110111100111010

NOTE: You must create the frequency map, min heap and the huffman tree before showing the decoding

8. Explain the concept of algorithmic complexity in simple terms. Provide an example to illustrate how algorithmic complexity affects the performance of an algorithm.
9. Explain the concept of growth of functions in relation to algorithm analysis. Provide a comparison between linear, logarithmic, and exponential growth functions, highlighting their implications for algorithm performance.
10. A man has 10 varities of fish: fish1, fish2,----fish10 namely, in his aquarium. The number of g=fishes from fish1,fish2--fish10 are as follows: 48,56,24,96,68,59,71,84,99 and 35. Apply divide and conquer methodology to find the name of the fish which is maximum and minimum in the aquarium.

11. Sort the list E,X,A,M,P,L,E in alphabetical order using the quick sort algorithm.
12. Differentiate between Kruskal's and Prim's algorithms for finding minimum spanning trees.
13. Why is finding a minimum spanning tree important in network design and optimization?
14. Describe a real-world scenario where a greedy algorithm might be a good choice for solving a problem.
15. Define Activity Selection Problem with proper example..

Q4. 5mark questions

1. Outline the steps involved in the Divide and-Conquer approach. Illustrate with the Merge Sort algorithm.
2. Explain Kruskal's Algorithm for finding the Minimum Spanning Tree. Include a brief example of its application.
3. Arrange the functions in ascending order of growth (from slowest-growing to fastest growing)
   $g_1(n) = n \log(\log n)$
   $g_2(n) = 2^{(\sqrt{n})}$
   $g_3(n) = n^{(1/3)}$
   $g_4(n) = \log(n)^n$
   $g_5(n) = n^2 \log n$
   $g_6(n) = n^{(n^{(1/2)})}$
   $g_7(n) = (\log n)!$
   $g_8(n) = n / \log n$
4. Explain the Activity Selection Problem and describe the greedy approach to solve it. Why does the greedy strategy work for this problem?
5. Derive the time complexity of the following recurrence relation using the Master Method: $T(n) = 4T(n/2) + O(n^2)$. Explain each step in detail and justify the solution.
6. Arrange the functions in ascending order of growth (from slowest-growing to fastest-growing).
   $f_1(n) = 3n^2 + 2n \log n + 100$
   $f_2(n) = n^3 / (\log n)^2$
   $f_3(n) = 2^{(\log_2 n)}$
   $f_4(n) = n (2^n)$
   $f_5(n) = n^{(\log_2 n)}$
   $f_6(n) = n!$
   $f_7(n) = \sqrt{n}$
   $f_8(n) = \log(n!)$
7. List the four sorting algorithms mentioned and briefly describe the primary approach used in each.
8. Critically evaluate the performance of Shell sort, Quick sort, Merge sort, and Heap sort algorithms in parallel computing environments. Discuss the challenges and opportunities
9. Develop pseudocode for implementing the Quick sort algorithm with three-way partitioning. Explain how this modification enhances the performance of Quick sort, particularly for arrays with many duplicate elements.
10. How does parallelism improve the efficiency of divide-and-conquer algorithms?
11. What kinds of problem can be solved using Divide and Conquer method? Why we use divide and conquer method?

12. List some characteristics that are often present in problems that are solvable by greedy algorithms.
13. Illustrate the process of solving the Knapsack Problem using both dynamic programming and greedy strategies. Compare the solutions obtained and discuss the advantages and disadvantages of each approach.
14. Explain how Huffman coding utilizes a greedy approach to create an optimal prefix code for a set of characters and their frequencies. Briefly describe the steps involved in the algorithm.
15. Explain the concept of 'backtracking' in the context of greedy algorithms and its implications for finding optimal solutions