1. Distinguish between divide-and-conquer and dynamic programming with suitable examples.

2. What do you mean by backtracking and why is it required? Why is it so called?

3. If \( f(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_0 \) is a polynomial of degree \( m \), then prove that \( f(n) = \Theta(n^m) \).

4. Differentiate Big-O, Big-\( \Omega \), and Big-\( \Theta \) notations.

5. What do you mean by cost optimality of a parallel algorithm?

6. Consider the evaluation of the product of \( n \) matrices \( M_1 \times M_2 \times \ldots \times M_n \).
   Assuming multiplication of \( p \times q \) matrix and \( q \times r \) matrix requires \( pqr \) operations, write an algorithm for ordering the above multiplication.

7. Under what circumstances the condition “P=NP” is true?

(7x4)

2. What is an algorithm?

3. a) What are the differences between heuristic and approximation algorithms?
   b) Design an approximation algorithm for colouring a planar graph.
   c) Design a heuristic algorithm for chromatic partitioning of a simple, connected and undirected graph.

(4+7+7)

4. a) Write an algorithm to compute the k-th smallest element of a list of \( n \) numbers, where \( k \leq n \).
   Determine the number of comparisons required to compute it.
   b) Write the Kruskal's algorithm for computing a minimum spanning tree of a simple, connected, undirected graph \( G \).
   Trace this algorithm to compute a minimum spanning tree for such a graph \( G \) that contains at least 9 vertices and 13 weighted edges.

(9+9)

5. a) What are the differences between the min-heap property and the binary search tree property?—Exemplify. Can the min-heap property be used to print out the keys of a binary tree of \( n \) vertices in sorted order in \( O(n) \) time?—Justify.
   b) Devise a \( O(n+m) \) time algorithm for computing a component graph of a directed graph \( G=(V,E) \), where \(|V|=n\) and \(|E|=m\).
   Make sure that your algorithm produces at most one edge between any pair of vertices in the component graph.

(10+8)
6.  
   a) Is dynamic programming a Top-Down or Bottom-Up technique? Why? Explain with an example.  
   b) What is amortized analysis? Give the usefulness of it. Give the various types of it using examples.  
      (9+9)  

7.  
   a) Differentiate between BFS and DFS.  
   b) What is polynomial time reducibility? Give example(s).  
   c) Differentiate among P, NP, NP-complete, and NP-hard class of problems with suitable examples.  
      (3+5+10)