UNIT V- 2 MARKS

Fourth Semester

B.E. Computer Science and Technology

23TSB202 – Analysis of Algorithm

Regulations 2023

1.Why is it difficult to solve NP-complete problems optimally?

Ans: NP-complete problems have no known polynomial-time solutions and require checking an exponential number of possibilities. As input size grows, computation time increases drastically. This makes finding optimal solutions computationally infeasible for large instances.

2.Describe the Branch and Bound strategy in solving optimization problems.

Ans: Branch and Bound systematically explores branches of a solution tree, eliminating those that can't yield better results than the current best. It uses bounds to prune unpromising paths. This helps efficiently solve combinatorial optimization problems.

3. What is an Approximation Algorithm, and why is it important in solving NP-hard problems?

Ans: An Approximation Algorithm provides near-optimal solutions in polynomial time. It's important for NP-hard problems where exact solutions are too slow or impossible for large inputs. These algorithms guarantee results within a specific factor of the optimal.

4.Differentiate between Naïve and KMP string matching algorithms.

Ans: The Naïve algorithm checks for the pattern at every position, resulting in higher time complexity in worst cases. KMP uses a prefix table to skip unnecessary checks, offering linear time performance. KMP is more efficient for large texts and patterns.

5.Define the P, NP, and NP-Complete classes with one example each.

Ans: P includes problems solvable in polynomial time (e.g., Binary Search). NP includes problems verifiable in polynomial time (e.g., Subset Sum). NP-Complete problems, like the Travelling Salesman, are both in NP and as hard as any problem in NP.

6.Explain Lower bound arguments in analyzing algorithms.

Ans: Lower bound arguments establish the minimum possible time an algorithm must take to

solve a problem. They help identify the theoretical limits of algorithm performance. For example, comparison-based sorting has a lower bound of O(n log n).

7.Write the condition for a Hamiltonian Circuit to exist in a graph.
Ans: A Hamiltonian Circuit exists if there is a cycle that visits each vertex exactly once and returns to the starting point. There is no simple necessary and sufficient condition, but Dirac's and Ore's theorems provide useful criteria. These are mainly used for undirected graphs.

8.What are limitations of algorithms in solving complex problems?
Ans: Algorithms face limitations when problems require excessive time or resources to solve optimally. Some problems, like NP-complete ones, have no known efficient solutions. In such cases, alternative strategies like heuristics or approximations are used.

9.What is a decision tree in algorithm analysis?
Ans: A decision tree is a model that shows all possible decisions or outcomes of a problem step by step. It's used to analyze the complexity of comparison-based algorithms. The height of the tree gives a lower bound on the number of operations required.

10.Explain backtracking using the n-Queens problem.
Ans: Backtracking tries to place queens on a chessboard such that no two attack each other. It incrementally builds the solution and backtracks when a conflict is found. This technique reduces the number of configurations to check.

11.What is the subset sum problem and how is backtracking applied?
Ans: The subset sum problem asks if there exists a subset of numbers that adds up to a given sum. Backtracking explores subsets recursively, pruning branches that exceed the sum. It's useful for small to medium-sized instances.

12.What is the Assignment Problem and how is it solved using Branch and Bound?
Ans: The Assignment Problem assigns tasks to agents to minimize cost or maximize efficiency. Branch and Bound is used to explore assignments while pruning non-promising branches. It helps reduce computation by using bounds to eliminate bad choices early.

13.What is the Traveling Salesman Problem (TSP) and how is it solved using Branch and Bound?
Ans: TSP seeks the shortest path visiting each city exactly once and returning to the start. Branch and Bound explores possible paths, using cost bounds to eliminate longer routes early. It reduces computation in solving this NP-hard problem.