Unit 03: Algorithms and Problem Solving

Exercise Short Answer Questions.

1. **Differentiate between well-defined and ill-defined problems within the realm of computational problem-solving.**

   Well-defined vs. ill-defined Problems

   Problems can also be categorized based on how clearly they are defined:

   • **Well-defined Problems:** These problems have clear goals, inputs, processes, and outputs. For instance, the problem of determining if a number is even, is a well- defined problem because it has a clear goal (determine if the number is even), clear input (a single integer), a clear process (check if the number is divisible by 2), and a clear output (even or odd).

   • **Ill-defined Problems:** These problems lack clear definitions or may have ambiguous goals and requirements. For instance, consider a project aimed at "improving education in Pakistan". This goal is vague and broad. The input might include resources such as funding, teachers, and educational materials, but without clear specifications on how much funding is needed, what kind of teachers are required, or which educational materials are most effective.

2. **Outline the main steps involved in the Generate and Test algorithm.**

   The Generate and Test algorithm involves two main steps:

   1. **Generate**: In this step, the algorithm generates possible solutions to the problem. These solutions can be generated randomly, systematically, or based on some heuristic.
   2. **Test**: After generating a potential solution, the algorithm tests it against the problem's requirements or constraints. If the solution satisfies all the conditions, it is considered valid. If not, the algorithm generates a new solution and repeats the process.

3. **Compare tractable and intractable problems in the context of computational complexity.**

   Tractable Problems: A problem is considered tractable if it can be solved in polynomial time, denoted as P. Polynomial time means that the time taken to solve the problem increases at a manageable rate (as a polynomial function) relative to the size of the input. Tractable problems are considered "efficiently solvable."

   Example: Sorting a list of numbers using algorithms like Merge Sort or Quick Sort is a tractable problem because these algorithms have a polynomial time complexity of O (n log n), where n is the number of elements in the list.

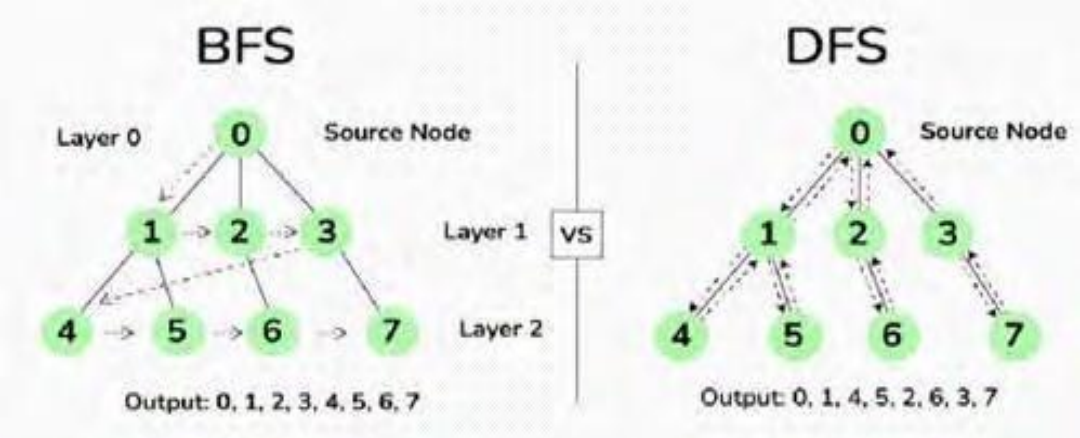
   Intractable Problems: Intractable problems are those that require super-polynomial time to solve, often growing exponentially with the size of the input. These problems are impractical to solve for large inputs because the time required becomes unmanageable.

4. **Summarize the key idea behind Greedy Algorithms.**

   Greedy algorithms work by making a sequence of choices, each of which is locally optimal, with the hope that these choices will lead to a globally optimal solution. The greedy approach is often used when a problem has an optimal substructure, meaning that the optimal solution to the problem can be constructed from optimal solutions to its sub problems. A greedy algorithm chooses the best option at each step, hoping it leads to the best overall result. It does not go back and change decisions.

5. **Discuss the advantages of using Dynamic Programming.**
   Dynamic Programming (DP) is an optimization technique used to solve problems by breaking them down into simpler subproblems and storing the results of these subproblems to avoid redundant calculations. DP is particularly useful for problems with overlapping subproblems and optimal substructure. Saves time by storing results (memoization/tabulation). Avoids repeating work.
   Example:
   Great for problems like Fibonacci.

6. **Compare the advantages of Breadth-First Search (BFS) with Depth-First Search (DFS) in graph traversal.**
   Breadth-First Search (BFS)
   Breadth-First Search (BFS) is a graph traversal algorithm that explores all the nodes of a graph level by level, starting from a given node (often called the root). It uses a queue to keep track of the nodes that need to be explored. In search Strategy it explores level by level.
   Depth-First Search (DFS)
   Depth-First Search (DFS) is another graph traversal algorithm that explores as far down a branch as possible before backtracking to explore other branches. It uses a stack to manage the nodes to be explored. Explores deep first.



7. **Explain the importance of breaking down a problem into smaller components in algorithmic thinking.**
   - Make complex problems easier to solve.
   - Helps in reusing code (modularity).
   - Make the solution clearer and organized.
   - Easier to test and debug each part.
   - Like building a house: foundation → walls → roof.

8. **Identify the key factors used to evaluate the performance of an algorithm.**
   Efficiency:

   Algorithm efficiency refers to how well an algorithm uses resources like time and space (memory) to solve a problem. The efficiency of an algorithm is typically evaluated based on its time complexity (how the run time

increases as the input size increases) and space complexity (how the memory usage grows as the input size increases).

Scalability:

Scalability refers to an algorithm's ability to maintain its efficiency as the size of the input data grows. An algorithm that scales well will perform efficiently even when the input data is significantly larger than what it was initially designed for.

Suitability for Specific Problems:

Not all algorithms are suitable for all types of problems. The choice of algorithm depends on the problem's specific requirements, such as the need for speed, memory usage, or the nature of the data.