**CHUKWU DANIEL NONSO**

**BHU/20/04/05/0010**

**CMP 418 - Algorithm and Complexity Analysis**

**Questions**

1. Why are computing problems considered as problems
2. Discuss the different algorithmic methods and the examples of problem each method can solve

**ANSWER 1**

Computing problems are considered problems because they involve challenges that need to be solved using computational techniques. These challenges can range from simple tasks, such as sorting a list of numbers, to more complex problems.

Some of the reasons why computing problems are considered problems are as follows:

1. **Complexity:** Many computing problems are complex and require careful analysis and planning to solve. They may involve multiple steps, dependencies, and constraints that must be considered.
2. **Limitations or Constraints on Resource:** Computing problems often have constraints on resources, constraints such as time, memory, or processing power. Solutions must be efficient and effective within these constraints in other to ensure a great solution that runs on not just powerful computers, but also less powerful ones.
3. **Variety:** Computing problems come in a wide variety of types and can be spotted in a lot of fields, including mathematics, science, engineering, finance, and more. Each of these computing problems do require a unique approach and solution to ensure an efficient and effective solution.
4. **Innovation and Creativity:** Solving computing problems often requires innovative and creative thinking. Developers must come up with new algorithms, techniques, or approaches to solve complex problems efficiently.

**ANSWER 2**

1. **Sorting**

Sorting algorithms rearrange a collection of items into a specified order.

**Examples:**

1. **Bubble Sort:** Sorting a list of numbers in ascending or descending order.
2. **Merge Sort:** Sorting a large dataset efficiently by dividing it into smaller sub-arrays and merging them in order.
3. **Quick Sort:** Sorting a list of elements by partitioning it into smaller sub-arrays based on a pivot element.
4. **Heap Sort:** Sorting elements by building a binary heap and repeatedly extracting the minimum (or maximum) element.

**Applications:** Sorting is used in various applications such as organizing data in databases, generating sorted reports, and implementing search algorithms efficiently.

2. **Searching**

Searching algorithms locate a target element within a collection of items such as an array, list, tuple, set etc.

**Examples:**

1. **Linear Search:** Searching for a target element by iterating through each element in a collection sequentially.
2. **Binary Search:** Searching for a target element in a sorted collection by repeatedly dividing the search interval in half.
3. **Depth-First Search (DFS):** Searching for a target element in a graph by traversing its vertices in depth-first order.
4. **Breadth-First Search (BFS):** Searching for a target element in a graph by exploring all neighbors of a vertex before moving on to the next level.

**Applications:** Searching is used in various applications such as finding an item in a database, locating files on a computer, and pathfinding in maps or networks.

3. **String Processing**

String processing algorithms manipulate strings of characters to perform operations such as searching, matching, and transforming.

**Examples:**

1. **Pattern Matching:** Finding occurrences of a pattern within a text string using algorithms like the Knuth-Morris-Pratt (KMP) algorithm or the Boyer-Moore algorithm.
2. **String Matching:** Determining whether two strings are identical or similar using algorithms like the Levenshtein distance or the Longest Common Subsequence (LCS) algorithm.
3. **Text Compression:** Reducing the size of a text string by encoding it using algorithms like Huffman coding or Lempel-Ziv-Welch (LZW) compression.

**Applications:** String processing is used in various applications such as text editing, search engines, DNA sequencing, and data compression.

4. **Graph Problems**

Graph algorithms operate on graphs, which collections of vertices (nodes) are connected by edges (links).

**Examples:**

1. **Shortest Path:** Finding the shortest path between two vertices in a weighted graph using algorithms like Dijkstra's algorithm or Bellman-Ford algorithm.
2. **Minimum Spanning Tree:** Finding a subset of edges that form a tree connecting all vertices with the minimum possible total edge weight using algorithms like Prim's algorithm or Kruskal's algorithm.
3. **Network Flow:** Determining the maximum flow that can be sent through a network from a source to a sink using algorithms like Ford-Fulkerson algorithm or Edmonds-Karp algorithm.

**Applications:** Graph problems have applications in various domains such as transportation networks, social networks, computer networks, and project planning.

5. **Numerical Problems**

Numerical algorithms solve mathematical problems involving numerical data.

**Examples:**

1. **Root Finding:** Finding the roots of a polynomial equation or the solution of a nonlinear equation using algorithms like Newton-Raphson method or bisection method.
2. **Matrix Operations:** Performing operations on matrices such as addition, multiplication, and inversion using algorithms like Gaussian elimination or Strassen's algorithm.
3. **Numerical Integration:** Approximating the definite integral of a function over a specified interval using algorithms like Simpson's rule or Trapezoidal rule.

**Applications:** Numerical problems are used in scientific computing, engineering simulations, financial modeling, and many other fields that involve mathematical calculations.