Summary on methods, algorithms, and data structures: For assignment 1

ASSIGNMENT NUMBER : 01

NAME : Dinesh Raj Pampati

ULID : dpampat

NAME OF SECRET DIRECTORY : NAMO001

Summary on methods, algorithms, and data structures:

Methods used:

1. **check(boolean[] temp)**: This method checks if all the elements in the given boolean array are true. If all are true, it returns false; otherwise, it returns true.
2. **main(String[] args)**: The main method reads the input file, processes the data, and performs topological sorting on the given directed acyclic graph (DAG).

Algorithm:

The code implements a topological sorting algorithm which is used to order the vertices of a directed acyclic graph (DAG) such that for every directed edge (u, v), vertex u comes before v in the order. The code uses an adjacency list to represent the graph and an array to keep track of the in-degrees of the vertices. The algorithm follows these steps:

- Read the input file and create an adjacency list representation of the graph.
- Initialize an array to keep track of the in-degrees of the vertices.
- Identify vertices with in-degree 0 and print them and mark them as visited by flagging the boolean array. So that we don't visit the vertices with indegree 0.
- Now we will remove the connects (Edges) for vertices within degree as 0. Also decrease the indegree of the neighboring vertices by 1.
- Now will repeat the above steps until every vertex is reached. If we can't find a vertex within degree 0 then it is called a cyclic graph.

Data Structures used:

1. **File**: Used to read the input data.
2. **Scanner**: Used to parse the input data.
3. **ArrayList**: Used to represent the adjacency list of the graph.
4. **LinkedList**: Used within the ArrayList to store the neighbors of each vertex.
5. **Boolean Array**: Used to keep track of visited vertices.
6. **Integer Array**: Used to keep track of the in-degrees of the vertices

Time And Space Complexity:

The topological sorting algorithm processes each vertex and edge of a graph exactly once. The number of vertices is represented by V, and the number of edges is denoted by E. Therefore, the time taken to process all vertices is O(V), and for all edges, it's O(E). Combining these, the overall

time complexity of the algorithm becomes **O(V + E).** This ensures efficient performance for directed acyclic graphs.

The space complexity of the topological sorting algorithm is determined by the data structures used. The adjacency list representation of the graph requires **O(V + E)** space, where V is the number of vertices and E is the number of edges. Additionally, the in-degree array requires O(V) space for storing in-degrees of all vertices. The queue, used for storing vertices with in-degree 0, also takes up to O(V) space in the worst case. Combining these, the overall space complexity remains **O(V + E),** ensuring efficient space utilization for the algorithm.