Algorithms Lab

Project Evaluation Sheet

Name: Anirudh Inamdar

SRN: 01fe23bcs218

Roll no: 341

Implementation Analysis

| Algorithm/Data Structure | Used? (Yes/No) | How and where? | Space Efficiency | Time Efficiency |
| --- | --- | --- | --- | --- |
| Arrays | Yes | Used in graph representation (adjacency list, matrix), product IDs, priorities. | O(n) | O(n) for storage and access |
| Structures | Yes | Used Edge struct for Kruskal’s algorithm. | O(1) | O(1) per operation |
| List | Yes | Graph represented as adjacency list | O(n) | O(n) for each edge insertion |
| Stack | No | - | - | - |
| Queue | Yes | Used in BFS and priority queue in Dijkstra/Prim. | O(n) | O(n) for queue operations |
| Binary Tree | No | - | - | - |
| Binary Search Tree | No | - | - | - |
| AVL Tree | No | - | - | - |
| 2-3 Tree | No | - | - | - |
| Red-Black Tree | No | - | - | - |
| Trie | No | - | - | - |
| Heap | Yes | Used in Dijkstra, Prim (min-heap for MST and shortest path). | O(n) | O(log n) for insertion and extraction |
| Lookup Table | No | - | - | - |
| Sparse Table | No | - | - | - |
| Fenwick Tree | No | - | - | - |
| Segment Tree | No | - | - | - |
| Skip List | No | - | - | - |
| Union-Find | Yes | Used in Kruskal’s algorithm for cycle detection. | O(α(n)) | O(α(n)) for each find/unite operation |
| Hashing | No | - | - | - |
| DFS | Yes | Depth-First Search implemented for graph traversal (warehouse layout). | O(V + E) | O(V + E) for graph traversal |
| BFS | Yes | Breadth-First Search for the shortest route in the graph. | O(V + E) | O(V + E) for graph traversal |
| Bubble Sort | No | - | - | - |
| Selection Sort | Yes | Used for sorting customer requests. | O(1) | O(n^2) overall |
| Insertion Sort | No | - | - | - |
| Quick Sort | Yes | Used for sorting product IDs. | O(log n) | O(n log n) on average |
| Merge Sort | Yes | Used for sorting customer requests in one example | O(n) | O(n log n) |
| Brute Force String Search | No | - | - | - |
| Rabin Karp | No | - | - | - |
| Boyer-Moore | No | - | - | - |
| Knuth-Morris-Pratt | No | - | - | - |
| Heap Sort | Yes | Used for product priorities. | O(n) | O(n log n) |
| Kruskal | Yes | Used for MST with union-find and edge sorting. | O(E) | O(E log E) |
| Prim | Yes | Used for MST with priority queue (min-heap). | O(V + E) | O(E log V) |
| Dijkstra | Yes | Used for the shortest path algorithm with priority queue (min-heap). | O(V + E) | O(E log V) |
| Floyd | Yes | Used for All Pairs Shortest Path with adjacency matrix. | O(V²) | O(V^3) |
| Warshall | No | - | - | - |
| Bellman-Ford | Yes | Used for single-source shortest path with negative weights. | O(V + E) | O(VE) |
| Any Other | No | - | - | - |

**Other Parameters:**

Number of Lines of Code Written: above 500 lines

Number of Functions:19

Design Techniques and Principles used:

* Divide and Conquer (Merge Sort, Quick Sort)
* Greedy Algorithm (Prim’s and Kruskal’s algorithms)
* Dynamic Programming (Floyd-Warshall)
* Backtracking (DFS)
* Union-Find for cycle detection in Kruskal’s algorithm

SDG connect: SDG 11

Reflections:

The project covered a variety of classical graph algorithms (Dijkstra, Kruskal, Prim), sorting algorithms (Quick Sort, Merge Sort), and other algorithms like Bellman-Ford, DFS, BFS.

The focus on data structure usage (like heap, union-find) and their applications in real-world network optimization (Uber routes, ISP network, warehouse optimization) made the implementation both challenging and educational.