Campus Navigator

Programming Data Structures and Algorithms - 1

Higher National Diploma in Software Engineering

Damian Kenny – COHNDSE242F-089

Chanul Liyanage – COHNDSE242F-023

School of Computing and Engineering
National Institute of Business Management
Colombo-7

1. Introduction

Navigation and route optimization are among the key challenges in computer science, which have found their applications in logistics, travel planning, and network design. Our application, Campus Navigator, would be such an aid that would help users find out the most efficient way to get from one building to another or any two locations within a campus.

This is a project that practically demonstrates the use of graph-based algorithms and various data structures to handle a real-world problem. The fundamental shortest path is found by implementing Dijkstra’s Algorithm, whereas Kruskal’s Algorithm is used for the creation of a Minimum Spanning Tree (MST), thereby, achieving network optimization. BFS and DFS also illustrate different traversal techniques, however, Queue (FIFO) and Stack (LIFO) and Binary Search Tree (BST) act as the basic structures that support these operations.

2. Functional Requirements

2.1 Input Requirements

- Start location (node).
- Destination location (node).
- Graph of campus layout (nodes = buildings, edges = path with weights such as distance)

2.2 Process Requirements

- Represents the campus as a graph
- Apply Dijkstra's Algorithm to calculate the shortest weighted path
- Apply Kruskal's Algorithm to build an MST (optimum wiring/connection layout)
- Use BSF and DFS for traversing the graph
- Use Queue (FIFO) and Stack (LIFO) internally for BFS and DFS implementations
- Store locations in a Binary Search Tree (BST) and use search to quickly locate nodes

2.3 Output Requirements

- Shortest path between two chosen nodes with distance/cost
- Alternative traversal paths using BFS/DFS
- Display of minimum Spanning Tree using Kruskal.
- Provide location search results from the BST.

3. Data Structure Used and Justification

Data Structure	**Justification**
Graph	Represents campus buildings and paths, allows shortest path and MST calculations.
Queue (FIFO)	Ensures nodes are processed in correct level order for breadth-first traversal.
Stack (LIFO)	Enables backtracking and deep exploration of paths in DFS
Tree (Binary Search Tree)	Stores and retrieves campus locations efficiently in sorted order. Also demonstrates hierarchical data representation.