Sea Exploration: A Graphical Approach Implemented in C++

By:

Hashir Khan Niazi 2023430

Instructor:

Engr. Junaid Ahmed

For the Partial Completion of Data Structures and Algorithms Lab Course

Project Overview

Our project focuses on solving a *real-world-inspired problem* of navigation, planning, and prioritization by developing a dynamic map system. Using core concepts from **Data Structures and Algorithms (DSA)**, the system allows users to:

- Add and manage islands dynamically.
- Determine the best route between islands.
- Sort and prioritize islands based on certain metrics (e.g., importance, resources, or distance).

The project's objective is to demonstrate the application of key DSA principles while simulating a map system inspired by a seafaring world like *One Piece*.

Core Features and Data Structures Used

1. **Linked Lists (Dynamic Island Management):**
   - **Singly Linked List** is used to dynamically manage island data.
   - Each node represents an island containing information such as its name, coordinates, and resources.

2. **Stacks and Queues (Navigation History)**:
   - A stack will maintain the navigation history (LIFO) for backtracking (like undo functionality).
   - A queue will be used to simulate BFS traversal for exploring islands level by level.

3. **Graph Algorithms (Shortest Route and Exploration):**
   - The map of islands is represented as a graph where:
     - Islands are nodes.
     - Routes are edges with weights (e.g., distance, difficulty).
   - Algorithms used:
     - Dijkstra's Algorithm: To determine the shortest path between islands.
     - DFS/BFS: To explore all reachable islands efficiently.

4. **Sorting and Searching (Island Ranking):**
   - Sorting algorithms like MergeSort or QuickSort will rank islands based on specified criteria such as resources, importance, or distance.
   - Binary Search helps retrieve island details quickly based on their rank.

5. **Dynamic Programming (Optimal Travel Plans):**
   - Dynamic programming techniques will optimize multi-route travel plans (e.g., minimizing costs or maximizing resources collected).

6. **Greedy Algorithms (Immediate Optimal Decisions):**
   - Greedy algorithms will determine immediate decisions like finding the nearest island at every step during navigation.

7. **Recursion (Complex Problem Breakdown):**
   - Recursion simplifies map traversal algorithms and hierarchical operations in the BST.

Functionalities

1. **Add New Island:**
   - Dynamically add islands with relevant data using linked lists.

2. **Find the Best Route:**
   - Determine the optimal path between two islands using **Dijkstra's Algorithm**.

3. **Rank Islands:**
   - Sort islands based on importance, distance, or resources.

4. **Explore Map (Traversal):**
   - Use BFS/DFS to explore connected islands.

5. **Undo Navigation:**
   - Use a stack to backtrack through navigation history.

6. **Plan Optimized Routes**:
   - Use **dynamic programming** to compute optimal travel plans.

7. **Search for an Island:**
   - Search island details efficiently using a BST and **binary search**.

Conclusion

This project demonstrates the practical use of data structures and algorithms to solve a real-world-inspired navigation problem. By incorporating various DSA concepts such as linked lists, stacks, queues, trees, graphs, sorting, dynamic programming, and greedy algorithms, the project delivers a robust and efficient solution.