PROJECT TITLE :

DISASTER RESPONSE AND RESCUE PATH FINDING SYSTEM

SYNOPSIS

DEPARTMENT OF CSE/IT

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

SUBMITTED BY:

TARISHE PANDITA	2401030006
MISHTHI ABROL	2401030025
ANUSHKA GARG	2401030029

FACULTY MEMBERS:

DR. DHANALEKSHMI G.

DR. VARUN SRIVASTAVA

MS. ANUPAMA PADHA

INTRODUCTION

In real-life disaster scenarios such as earthquakes, floods, and fires, emergency response teams must quickly identify affected zones, locate victims, compute the shortest and safest rescue paths, and manage limited resources. These decisions must be made rapidly using dynamic data—blocked roads, shifting victim locations, and changing priorities.

This project proposes a C++ simulation system that models a disaster-hit city as a graph and integrates multiple data structures to represent city maps, victims, and resources. It computes optimal rescue paths, prioritizes operations, and enables fast lookups and updates. The system supports real-time decision-making and can be extended with visualization or AI-based enhancements.

DATA STRUCTURES USED

- **Classes:** Uses classes like RescueTeam, Victim, and CityMap to organize data and functions cleanly.
- **Arrays:** Store availability of resources such as vehicles and supplies. Access is very fast (constant time). **– O(1).**
- **Queue:** Manages rescue requests in the order they arrive (first-in, first-out), ensuring fairness and order. **– O(1)**.
- **Stack:** Helps with backtracking when rescue routes need to be reconsidered or undone (last-in, first-out). **– O(1)**.
- **Binary Search:**Quickly finds a victim by their ID in a sorted list, speeding up lookup. – **O(log n).**
- **Sorting Algorithms:** Sort victims by urgency (like injury severity), helping teams prioritize rescues efficiently. – **O(n log n)** (e.g., Merge Sort).
- **Hashing:** Assigns and retrieves unique IDs for rescue requests for quick access (around constant time).– **O(1) average case.**
- **Linked List:** Maintains a flexible list of victim records, allowing easy additions or removals during rescue. – insertion/deletion in **O(1).**
- **Multi-linked List:** Sparse road networks – **O(n) traversal.**
- **Backtracking Algorithms:** Rat-in-a-Maze, N-Queens –

worst case exponential **O(2^n)**, pruned optimally.

- **Binary Search Tree:** Stores rescue teams' information for quick searching and updating – search/insert in **O(log n)** - avg.
- **AVL Tree:** Stores rescue team data in a balanced manner to allow fast and efficient queries and updates– O(log n) guaranteed.
- **Min-Heap:** Keeps the most urgent rescue missions at the top, so they get handled first – insert/delete-min in **O(log n)**.
- **B+ Tree:** used to efficiently store and manage rescue operation logs, enabling fast insertion, deletion, and retrieval of records in sorted order– **O(log n).**
- **Graph Representation:** Models the city: nodes are locations, edges are roads. Supports searching and pathfinding between places.
- **BFS/DFS**: Used to explore affected city zones and locate victims by traversing the road network graph systematically.
- **Hash Table:** Quickly checks if roads are blocked during rescue planning, enabling fast decision updates – **O(1)** average.
- **Suffix Tree:**Supports fast search of partial or substring matches in location or route names for quick lookup. – **O(m)** where,(m = pattern length).

MAPPING TO REAL LIFE APPLICATIONS

- **Victim urgency ranking** → Sorting & Heaps
- **Road network modeling** → Graphs & Lists
- **Rescue team deployment** → Trees & Backtracking
- **Tracking blocked roads** → Hash Tables
- **Location search & input** → Tries & Suffix Trees

EXPECTED OUTCOME

Aids emergency planners and response teams in quickly deciding where to deploy resources and how to reach victims quickest, even in chaotic disaster scenarios. Simulates real-time city disruptions as a testing ground for training, planning, and testing different rescue and allocation strategies. Aids in effective emergency resource management, optimizing rescue rates and response times. Provides a platform for extension further, such as visual dashboards, live sensor feed integration, or AI-based path optimization.