Data Structures and Algorithms

[Risk assessment and disaster management]

Course Project Report

School of Computer Science and Engineering

2023-24

Contents

1. Course and Team Details

1.1 Course details

Course Name	Data Structures and Algorithms
Course Code	23ECSC205
Semester	III
Division	C
Year	2023-24
Instructor	Bhagya PSunag

1.2 Team Details

Si. No.	Roll No.	Name
1.	341	Sindhu
2.	359	Daneshwari H
3.	361	Sri Lakshmi G N

1.3 Report Owner

Roll No.	Name
341	Sindhu

2. Introduction

This project addresses the challenges of disaster management through intelligent planning. The goal is to work on the tasks such as event tracking, resource allocation, network understanding, and logistics management. This is important because of the increasing number and severity of natural disasters worldwide. This role is from the Guidelines for Disaster Management The guidelines also emphasized the importance of managing information effectively during a crisis and ensuring good relationships. Our work is inspired by the real-life problems outlined in the guide, which aims to make disaster response more realistic and effective.This project addresses the challenges of disaster management through intelligent planning. The goal is to streamline tasks such as event tracking, resource allocation, network understanding, and logistics management. This is important because of the increasing number and severity of natural disasters worldwide. This role is from the Guidelines for Disaster Management The guidelines also emphasized the importance of managing information effectively during a crisis and ensuring good relationships. Our work is inspired by the real-life problems outlined in the guide, which aims to make disaster response more realistic and effective.

3. Problem Statement

3.1 Domain

Our challenge addresses the problems in disaster control. We need to construct the code which could quick and efficiently deal with incident information, distribute medical resources, analyze connectivity among different reaction devices, and manage logistics at some stage in disasters. This is all inspired via the challenges cited in the guide . The primary problem we are solving is the shortage of a unified system that could smoothly deal with distinctive factors of disaster management. Our purpose is to use era to speed up responses and make certain sources, consisting of clinical help and transportation, are used successfully all through crises

3.2 Module Description

1.Hashing Functions (hfun, shfun, dhfun):

Responsible for generating hash values used in the hash table for incident data.

2.Hash Table Functions (inserthash, displayhash, search):

Storing and retrieving incident data.

3.Binary Search Tree (create, insert, fBST, initBST):

To store the resources and search the resource by its name.

4.Resource Allocation (alloctr):

Allocating resources to incidents based on user input and resource availability.

5.Merge Sorting Functions (merge, mergeSort):

Sorting based on Resource urgency.

7.Quick sort(quicksort):

Sorting based on incident name.

8.Heap sort(heapify, heapsort):

Sorting based on safety score of safer places.

9.String Matching (compute, searchr):

Knuth-Morris-Pratt algorithm for searching resources by their name.

10.Stack Functions (full, push, empty, displays, pop, peek):

Keep track of user choices and display the recent choice.

11.User Interface (welcome, feedback):

Displaying a welcome message and taking feedback from the user.

12.File Operations (loadIncidentData, loadResourceData, loadShelterData):

Loading incident, resource, and shelter data from files.

13.Graph Traversal (bfs):

Breadth-First Search for traversing incidents in a graph.

14.Queue (createQueue, enqueue, dequeue, isQueueEmpty):

Handling dynamic memory allocation for structures like the queue.

4. Functionality Selection

Si. No.	Functionality Name	Known	Unknown	Principles Applicable	Algorithms	Data Structures
1	Historical Data Representation	Data structure of historical incidents	Efficient data retrieval and display	Data visualization principles	Hashing, Linear Probing	Hash Table
2	Resource Allocation	Resource data, Incident details	Optimal resource allocation strategies	Resource allocation principles	Binary Search Tree	Arrays, Link list
3	Sorting by Resource Urgency	Resource data, Urgency values	Sorting techniques for urgency	Divide-and-conquer	Merge Sort	Array
4	Sorting by Incident Name	Incident data	Sorting incidents alphabetically	Divide-and-conquer	Quick Sort	Array
5	Sorting Shelters by Safety Score	Shelter data, Safety scores	Sorting shelters based on safety	Transform and Conquer	Heap Sort	Array
6	Searching Resource by Name	Resource data	Efficient resource search	Brute Force	KMP Algorithm	
7	Stack Operations	Stack operations	Efficient stack operations	First-in last-out	-	Stack
8	Feedback Collection	User feedback	Feedback collection principles	-	-	File I/O
9	Shelter Search	Shelter data, Location	Efficient shelter search	Space and time trade-off	Rabin Karp Algorithm	Array
10	BFS Analysis	Incident data, Adjacency matrix	BFS algorithm for incident exploration	Decrease and Conquer	Breadth-First Search	Queues,Array

5. Functionality Analysis

1. Historical References:
Workflow: Historical data is read from a file and stored in a hash table. A hash table is implemented by using hash functions. The user can then search for specific incidents by incident_id.
Efficiency analysis: The time complexity of inserting historical data into a hash table is O(N), where N is the number of historical records. Searching an event in a hash table results in a complex transaction time of O(1) due to hash table efficiencyin average case ,if there are many collision than it may extend nearly to O(n).

2. Distribution:
Workflow: Objects are stored in a binary search tree (BST) based resource name. Resource are then distributed based on users input.
Efficiency analysis: The BST process has an average time complexity of O(log N), where N is the number of objects. The allocation process is O(N log N) for the worst-case.

3. Divide and conquer:
Workflow: Sort items by urgency using the Merge Sort algorithm, a divide-and-conquer strategy.
Efficiency Analysis: Merge Sort has a time complexity constant of O(N log).

4. Schedule of Events (Quick Schedule):
Workflow: The Quick Sort algorithm is used to sort events based on event names.
Efficiency analysis: Quick sorting has, on average, a time complexity of O(N log N).

5.Safety score:
Workflow: Heap Sort algorithm is used to sort dependencies based on safety scores.
Efficiency Analysis: The time complexity of Heap Sort is O(N log N), making it suitable for sorting arrays. It does not significantly affect the input order and provides an in-place sorting solution.

6. Requirements (KMP Algorithm):
Workflow: The Knuth-Morris-Pratt (KMP) string matching algorithm is used to find the names of the Resoures.
Efficiency analysis: KMP linear time complexity is O(N + M), where N is the text length and M is the sample length.

7.Users recent choice (Stack Operation):
Performance: A stack is used to track user's update options.

Efficiency: Stack operations (push, pop) have a time complexity constant O(1), making them more efficient for recent sampling and preparation choices.

8. Feedback:
Workflow: Users are given the opportunity to provide feedback.
Efficiency analysis: Response selection always has a time complexity, and the response process is independent of the data set size.

9. Shelters (Rabin-Karp Algorithm): 1.1.
Workflow: The shelters are searched by location using Rabin-Karp string matching algorithm.
Search efficiency: The average case time complexity of the Rabin cadaver is O(N + M), where N is the text length and M is the sample length is efficient for searching for patterns in text in large quantities.

10. BFS between events:
Workflow: Breadth-first search (BFS) is used to find the shortest paths between events in a graph to represent a proximity matrix.
Efficiency analysis: The time complexity in BFS is O(V + E), where V is the number of vertices (events) and E is the number of edges (combinations) It efficiently analyzes the graph in extension-first order.

6. Conclusion

Data Structures and Algorithms:

Hash Tables: The usage of hash tables for storing information about the incidents and search through incident_id. This shows an application of hash-based data structures.

Binary Search Tree (BST): The implementation of a BST for useful resource allocation demonstrates a solid draw close of tree system.

Sorting Algorithms: Different sorting algorithms are added to the function including Merge Sort, Quick Sort, and Heap Sort. Importananceof understanding when to apply any code.

String Matching Algorithms:
Rabin-Karp Algorithm: The challenge uses the Rabin-Karp set of rules for searching shelters by place, showcasing a numerous set of algorithms for extraordinary kinds of statistics.

File Handling:
Reading and Writing Files: The code involves reading historical information from a record and updating files based on consumer input. This demonstrates talent in record coping with operations.

User Feedback: How to encourage consumer engagement and presenting a manner for customers to make contributions their thoughts.

7. References

https://www.geeksforgeeks.org/learn-data-structures-and-algorithms-dsa-tutorial/
https://leetcode.com/

~*~*~*~*~*~*~*~