Data Structures and Algorithms

Smart Traffic Management System

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	B
Year	2023-24
Instructor	DR. PRIYANKA GAWADE

1.2 Team Details

Si. No.	Roll No.	Name
1.	62	SONALI JADHAV
2.	67	STUTI HUNACHAGI
3.	42	SARVESH NIRMALKAR
4.	03	AKHILESH JOSHI

1.3 Report Owner

Roll No.	Name
62	Sonali Jadhav

2. Introduction

The Smart Traffic Management project is a comprehensive software system designed to address challenges in urban traffic congestion and parking management. The project incorporates various modules, focusing on efficient traffic analysis, sorting algorithms such as heap sort, and a parking reservation system. The primary goal is to enhance traffic flow, reduce congestion, and optimize parking space utilization in urban areas. The implementation involves modules such as heap sort to analyse and prioritize traffic data, reserve parking slots for users, display parking details, and search for reserved user information. These functionalities contribute to an intelligent and user-friendly system that promotes efficient traffic management and seamless parking experiences in urban areas. The project aligns with the broader goals of building sustainable and technology-driven solutions for urban challenges.

3. Problem Statement

3.1 Domain

The problem addressed by the Smart Traffic Management project revolves around optimizing urban traffic flow, efficient parking space utilization, and providing a seamless experience for users. Urban areas face challenges such as traffic congestion and inadequate parking facilities, leading to increased pollution, fuel consumption, and overall inconvenience for commuters. The need for efficient algorithms like heap sort for traffic analysis and reserve parking slot management emerged as a key aspect to streamline traffic patterns and enhance parking experiences. By automating these processes, the project aims to create a solution that not only improves the efficiency of urban traffic but also contributes to a more sustainable and user-friendly urban environment.

3.2 Module Description

In the Smart Traffic Management project, my individual contribution involves working on the modules related to heap sort, reserve parking slot management, display functionalities, and the search functionality for reserved users using the brute force string matching algorithm.

4. Functionality Selection

Si. No.	Functionality Name	Known	Unknown	Principles applicable	Algorithms	Data Structures
	Name the functionality within the module	What information do you already know about the module?	What are the pain points? What information needs to be explored and	What are the supporting principles and design	List all the algorithms you will use	What are the supporting data structures?

		What kind of data you already have? How much of process information is known?	understood? What are challenges?	techniques ?		
1	A function to perform heap sort on an array of cities based on their traffic indices.	PURPOSE- The purpose of the heapsort module in the provided code is to facilitate the sorting of cities based on either their traffic indices or their names in alphabetical order Available Data: cities which are assigned traffic index and adjacency matrix representing the connectivity between them.	Array indexing in heap operations can be confusing. Heap sort involves multiple steps, and bugs can be challenging to trace. Ensuring that the algorithm works correctly for all input cases is vital. **Challenges:** **1.Algorithmic Complexity .** **2.Comparisons and swaps .**	Implementing heapify operations to maintain the heap property during construction and sorting. Recursive implementation of heapify operations to efficiently organize elements in the heap. Heap sort follows the divide-and-conquer principle.	1.Heapfy Operation: To maintain the heap property in a binary heap. 2.Heapify Sorting Algorithm: Sorting elements in ascending order based on a specified criterion (traffic index or alphabetical order).	Array , Array of structures and file usage to store the details
2.	Function to store the details of the user who reserve the parking slots for particular	The function prompts the user for input, validates the slot number, and checks if the slot is	Challenges includes input validation,data integrity in module. So Explore different scenarios	Implement distinct functions for each functionality (reserve, display, search) to enhance	Linear search- Iterate through the parking slots linearly to find an	Array or linked list of parking slots. Sorted array or binary search tree of

	time and display their information using. Reserve parking slot, Display parking slots.	already reserved. Allocates memory for a new reservation, initializes its details, and links it in the parkingSlots array. **2.** In display function it Iterates through the parkingSlots array, printing details for each slot. Checks if a slot is available or reserved, and prints relevant information accordingly.	and edge cases to ensure the system handles invalid inputs gracefully. Explore ways to validate the consistency of the parkingSlots data structure and its synchronization with other parts of the system.	maintainability and ease of future modifications. Use conditional statements and error handling to validate user inputs, preventing unexpected behaviour or security vulnerabilities.	available slot for reservation. Purpose: Find the first available parking slot. Linear display Iterate through the parking slots linearly to display all slot information	parking slots.
3	Function to find reserved user name using Brute force string matching algorithm	An array or linked list of reserved parking slots, where each slot contains information about the user who reserved it. The function utilizes a Brute-force string matching algorithm to find a reserved user name within	The current implementation may stop after finding the first match. If there are multiple reservations for the entered username, it might not display all of them.The database size, user requirement must be	Selecting or optimizing the string matching algorithm to improve efficiency, Error handling where handling error gracefully manage scenarios where the entered username	Brute force string matching algorithm	Array of strings ,indexes and pointers

		the parking slots.	explored. Challenges include partial matching.	is not found.		
4						
…						
…						

5. Functionality Analysis

1. Sorting Cities by Traffic Index –

Workflow:

1. Input: List of cities with their traffic indices.
2. Process:

- Sorting is performed based on the user's choice (alphabetical order or traffic index) using heap sort.

3. Output: Sorted list of cities.

Efficiency Analysis:

- Time Complexity: O(N * log(N)) for heap sort, where N is the number of cities.
- Space Complexity: O(N) for the sortedCities array.

2] Reserve Parking Slot

Workflow:

1. Input: User's username, slot number, and hours to reserve.
2. Process:

- Check if the slot is available.
- Reserve the slot if available.
- Update the parkingSlots array.

3. Output: Confirmation message.

Efficiency Analysis:

- Time Complexity: O(1) for reserving a slot.
- Space Complexity: O(1) for each reservation.

3] Display Parking Slots

Workflow:

1. Input: None.
2. Process:

- Iterate through parkingSlots array to display the status of each slot.

3. Output: Display of parking slot status.

Efficiency Analysis:

- Time Complexity: O(N), where N is the number of parking slots.
- Space Complexity: O(1).

4] Search Reserved User

Workflow:

1. Input: User's username to search.

2. Process:
- Brute-force string matching is applied to find the username in reserved slots.
- Display information if found.

3. Output: Display search results.

Efficiency Analysis:

- Time Complexity: O(M * N) in the worst case, where M is the length of the reserved username and N is the total number of reserved slots.
- Space Complexity: O(1).

6. Conclusion

1] Algorithm and Data Structure Implementation:

Gained experience in implementing fundamental algorithms and data structures such as Dijkstra's algorithm, heap sort, breadth-first search (BFS), and others.

2] Efficiency Analysis:

Learned to analyze the efficiency of algorithms and understand their time and space complexities, considering factors like input size and data characteristics.

3] Error Handling and Debugging:

Enhanced skills in error handling and debugging by addressing issues.

4] Continuous Learning:

Recognized the importance of continuous learning in the rapidly evolving field of software development, staying updated on best practices and emerging technologies.

7. References

1.Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, Fourth Edition, The MIT Press, 2022.

2. Anany V. Levitin, Introduction to the Design and Analysis of Algorithms. Addison-Wesley Longman Publishing Co, 2012.

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