|  |
| --- |
| **Data Structures and Algorithms** |
| [Railway Management System] |
| **Course Project Report** |

|  |
| --- |
| **School of Computer Science and Engineering**  **2023-24** |

**Contents**

|  |  |
| --- | --- |
| **Si. No.** | **Topics** |
| 1. | Course and Team Details |
| 2. | Introduction |
| 3. | Problem Definition |
| 4. | Functionality Selection |
| 5. | Functionality Analysis |
| 6. | Conclusion |
| 7. | References |

**1. Course and Team Details**

**1.1 Course details**

|  |  |
| --- | --- |
| **Course Name** | Data Structures and Algorithms |
| **Course Code** | 23ECSC205 |
| **Semester** | III |
| **Division** | A |
| **Year** | 2023-24 |
| **Instructor** | Prakash Hegade |

**1.2 Team Details**

|  |  |  |
| --- | --- | --- |
| **Si. No.** | **Roll No.** | **Name** |
| 1. | 153 | Om. R. Muddapur |
| 2. | 159 | Abhishek. B. R |
| 3. | 158 | Sumedh Kalgud |
| 4. | 101 | Anirudh Hanchinmani |

**1.3 Report Owner**

|  |  |
| --- | --- |
| **Roll No.** | **Name** |
| 153 | Om. R. Muddapur |

**2. Introduction**

Railway Management System is a program which focuses on enhancing the overall experience for passengers while travelling through trains by giving ease of things at hand tips and it also aims in improving operations carried out by the Railway men, so that they can carry out their duties easily, efficiently, and effectively.

Most of the Passengers always face problems while their journey through train like booking tickets, knowing train details, platform details, etc, so there is a need of a system which solves the passengers’ problems by giving them ease at each and every step. And the same goes for even Railway’s men as they have check and update each and every detail of trains, passengers, etc and has to maintain all the records, so here also an efficient software is necessary to carry out smoothly all these operations.

To conclude our program aims to solve all these problems faced by passengers and Railway man by giving them many new tools to use which are helpful, user-friendly and with easy-to-use interface. By providing an integrated and efficient program, the Railway Management System aims to contribute to the seamless functioning of railway operations, for giving enhanced service quality and operational effectiveness.

**3. Problem Statement**

**3.1 Domain**

Most of the people prefer travelling through trains as it cost efficient while compare to airways and roadways. The railway department is completely handled by government and some of their features are not user-friendly by which both passengers and railway working people have to face hassles while their journey. Unlike, the roadways and airways give a very good interface for passengers which makes their journey hassle free.

So, we wanted to give the same hassle-free user experience even to the people traveling through railways by making the railway system more efficient with tools like city navigation maps, parking space efficiency, etc.

**3.2 Module Description**

The module which I had worked on is the ‘Infrastructure development and Facilities’.

This module aims in achieving the development of infrastructure of the railway station in a more efficient way with modernized designs. And as of the part of facilities it improvises the user experience by giving them all necessary facilities with user-friendly interface. I have split the module into different functionalities based upon the user needs and the complete details of the functionalities within the module are described in the below section.

**4. Functionality Selection**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Si. No.** | **Functionality Name** | **Known** | **Unknown** | **Principles applicable** | **Algorithms** | **Data Structures** |
| 1 | Efficient Speakers Announcement | Announcing information to passengers about train details at respective platforms. | The information to be announced in railway stations will be different at each station and at each platform in respective stations. | Greedy Technique | Brute force string search, Bubble Sort | Arrays, Structures |
| 2 | Communication between Stations | Stations communication to provide information about train’s detail from source station to destination station. | There will be many trains travelling from one station to many other stations, so communication network should be vast and varied. | Divide and conquer | Rabin Karp Algorithm, Quick Sort | Linked list, Structure |
| 3 | Information Organization | Organization of required information to represent it to the suitable users. | The passengers are to be given required and suitable information with respect to their individual bookings. | Hierarchical data and trees or Sliding Window. | Binary Search tree | Trees, Structure |
| 4 | Multilingual Features | Information at stations should be in English, national language, local language. | The identification of home language and acquiring translated result from English should be proper. | Transform and conquer |  |  |
| 5 | Entertainment Facilities | As journeys will be long some entertainments like audio and video gadgets can be installed in trains. | Different regions will be having different kinds of entertainment plays. | Greedy Technique | Boyer Moore Search algorithm, Merge Sort | Arrays, Structure |
| 6 | Proper Lighting Facilities | Lights installed in the station and trains should be installed by following the light illumination path as graph at proper spots. | Placement of lights efficiently at different corners and edges so that continuous and max light illumination is possible. | Divide and conquer | Prims Algorithm, Kruskal Algorithm | Arrays, Structure |
| 7 | Display of city maps | City maps at stations help people to navigate their desired location easily. | Many places are connected from railway stations at respective cities. | Dynamic programming | Dijkstra's Algorithm, Heap sort | Arrays, Structure, Heap |
| 8 | Parking space efficiency at stations | As we know Indian families, to send one person many people come to just send him. So, we should make proper arrangements for car, bike, etc parking space. | It will be maintained based upon parking space allotted in the respective stations and dividing it based upon type of vehicle. | Divide and conquer | Sparse table | Arrays |
| 9 | City promotions | Providing city information like what it is famous for, etc at respective stations. | Maintaining and collecting different information for different cities. | Hierarchical data |  |  |
| 10 | Work space provisions | Ensuring the provision of adequate facilities and workspace for individuals during their waiting period for trains, as needed. | The provisions will be given based upon the dimensions of the train and for required class. | Divide and conquer |  |  |
| 11 | Ticket collection fare | To calculate the total amount collected by sold tickets and allowing range queries. | Thousands of transactions record of per day has to be maintained. | Brave and Cautious Travel | Prefix sum, Lookup table | Fenwick tree |
| 12 | Train route efficiency | It computes the efficient train route through various graph related algorithms | Sometimes route depends upon other factors importance, so algorithm application would be difficult. | Divide and conquer | Depth first Search, Breadth first search, Warshall Algorithms. | Array, Queue |

**5. Functionality Analysis**

1. **Efficient Speakers Announcement:**

* It reads a file and stores the values read in an array of structure.
* Sorts the file based upon train name using **Bubble sort**.
* Then asks user input for train name.
* Searches train name in array of structure using **Brute force string search**.
* **Time Efficiency:**

1. **Brute force string search:** The average case complexity is O(mn), where 'm' is the length of the pattern and 'n' is the length of the text.
2. **Bubble sort:** The average case complexity is O(n^2), where ‘n’ is number of trains.
3. **Communication between Stations:**

* Takes input of number of trains scheduled and their information.
* Inserts train information at the end as a node to the doubly linked list.
* Then by creating array of list it sorts data upon train name using **Quick sort**.
* Menu will be given to display, search, delete train.
* It uses **Rabin-Karp method** algorithm to check given train name in the list.
* **Time Efficiency:**

1. **Rabin-Karp method:** The average case and best-case complexity is O (m + n) and the worst-case complexity is O(mn).
2. **Quick sort:** The average case and best-case complexity of is O (nlogn) and the worst-case complexity is O(n^2).
3. **Information Organization:**

* Menu for insertion, deletion, search, display is given.
* Inserts data into **Binary search tree** based upon ticket ID.
* Deletes train data using BST deletion algorithm.
* Searches for train using ticket ID in the tree.
* Displays In-order traversal of the tree.
* **Time Efficiency:**

1. **BST insertion:** The average case complexity is O (logn).
2. **BST deletion:** The average case complexity is O (logn).
3. **Entertainment Facilities:**

* It reads a file and stores the values read in an array of structure.
* Sorts the file based upon city pin code using **Merge sort**.
* Then it asks input for city pin code, then mode of entertainment (channel, music), name of mode of entertainment.
* Searches for name in the array of structure at pin code’s index using **Boyer-Moore string search** algorithm.
* Displays result.
* **Time Efficiency:**

1. **Boyer-Moore string search:** The average case complexity is O(n/m).
2. **Merge sort:** The average case complexity is O (nlogn).
3. **Proper Lighting Facilities:**

* Takes input for number of lights to be placed.
* Menu is given for kind of light placement.
* It performs **Kruskal algorithm** by taking positions of spots and light power and displays result.
* It performs **Prim’s algorithm** by taking position matrix and displays result.
* **Time Efficiency:**

1. **Kruskal algorithm:** The average case complexity is O (E log n) and worst case complexity is O(n^2), where ‘n’ is number of vertices and ‘E’ edges.
2. **Prim’s algorithm:** The average case complexity is O(n^2), where ‘n’ is number of vertices

1. **Display of city maps:**

* It reads a file and stores the values read in an array of structure.
* Sorts the file based upon city names using **Heap sort**.
* Takes input for number of places to travel from railway station.
* Performs **Dijkstra’s algorithm** and gives shortest path.
* **Time Efficiency:**

1. **Dijkstra’s algorithm:** The average case complexity is O(|n|^2), where |n| represents the number of vertices in the graph.
2. **Heap sort**.The average case complexity is O(nlogn).
3. **Parking space efficiency in stations:**

* Prints the initial parking state matrix.
* Takes input for number of vehicles to be parked.
* Initiates **Sparse table** for parking matrix.
* Processes range minimum query for sparse table and displays output.
* **Time Efficiency:**

1. **Sparse Table:** The average case complexity is O (n ^2), where it is rows x columns.
2. **Sparse Table processing(sum):** The average case complexity is O (n ^2).
3. **Ticket collection fare:**

* Takes input for number of tickets sold with their prices.
* Menu for total amount collected up to given index and ticket sold for least price in given range.
* Performs **Fenwick tree prefix sum** to calculate amount.
* Builds **Lookup table** for range minimum query.
* **Time Efficiency:**

1. **Fenwick tree:** The average case complexity is O (n logn).
2. **Prefix sum:** The average case complexity is O (log n).
3. **Lookup table:** The average case complexity is O(n^2).
4. **Train route efficiency:**

* Takes number of stations and source station for input.
* Menu for performing 3 kinds of traversal for given stations connection matrix.
* Performs **Depth first search** traversal.
* Performs **Breadth first search** traversal.
* Performs **Warshall algorithm** to find shortest path.
* **Time Efficiency:**

1. **Depth first search:** The average case complexity is O(n^2).
2. **Breadth first search:** The average case complexity is O (n^2).
3. **Warshall algorithm:** The average case complexity is O (n^3).

**6. Conclusion**

This project was super interesting, as this was my first-time experience of building such a project. Unlike, other projects which were like writing the questions and their solutions, solve paper 2 times etc, which are very boring to do and also, we didn’t used to learn much things out there it was like just byhearting solutions. But in this project, we learnt a lot of things like practical application of all the algorithms we learnt, how to use algorithm in real world scenario, integration of different algorithms to get required results, etc was so informative and interesting to do.

By implementation of this project, I addressed real time problems occurring in railway stations of which some were personally faced by me, and found most of the effective solutions for them using various algorithms. Along with this, I also learnt the actual working of algorithms in a better way. While implementation, it gave me a feeling of being software engineer entrepreneur with development of such a system.

**7. References**

[1] lbackstrom. The Importance of Algorithms. Link: https://www.topcoder.com/community/data-science/data-science-tutorials/the-importance-of-algorithms/.

**~\*~\*~\*~\*~\*~\*~\*~**