UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, DEHRADUN

Optimal Parking Spot Allocation

Final Report of the (Minor Project - 1) in

Semester V Prepared by:

S. No	Students Name	Roll Number	Sap Id
1.	Akshay Mohpal	R2142210078	500088177
2	Diya Khandelwal	R2142210293	500090939
3	Gaurav Bhandari	R2142210311	500090993

BACHELORs OF TECHNOLOGY, COMPUTER SCIENCE ENGINEERING

With specialization in DevOps

Under the guidance of

Mr. Sandeep Pratap Singh

Division of Computer Science (SOCS),UPES

Bidholi Campus, Energy Acres, Dehradun – 248007

INDEX

Optimal Parking Spot Allocation

Introduction

In the evolving urban landscape, efficient parking allocation is a pressing challenge. The "Optimal Parking Spot Allocation System" project addresses this issue by integrating advanced algorithms, realtime updates, and user-friendly interaction. This synopsis report offers an overview of the project's objectives, key components, and anticipated benefits.

As cities grow and vehicles multiply, finding parking spots becomes increasingly time-consuming. This project aims to transform this experience by employing algorithms like shortest path techniques and binary search. By dynamically calculating optimal routes and efficiently selecting suitable parking spots based on vehicle size, the system aims to minimize search times and enhance user satisfaction.

Central to the project is its intelligent interplay of data structures, algorithmic principles, and user interface design. Through a command-line interface, drivers receive real-time parking spot suggestions, visual paths, and navigation guidance. The project's integration of technology and user experience aims to redefine the parking experience in urban environments.

This synopsis will delve into the project's core elements, exploring data representation, algorithmic implementation, real-time adaptability, and user interface design. It aims to display how the "Optimal Parking Spot Allocation System" project aspires to streamline parking allocation, saving time for drivers and contributing to more organized urban spaces.

Background Information

In today's urban landscape, the escalating number of vehicles and limited parking spaces have led to congestion, wasted time, and environmental concerns. Traditional parking allocation methods often fall short, resulting in inefficient space utilization and prolonged search times for drivers.

The "Optimal Parking Spot Allocation System" project emerges as a response to these challenges. By leveraging advanced algorithms, real-time data updates, and user-friendly design, the project seeks to revolutionize parking allocation. The goal is to minimize search times, enhance user satisfaction, and contribute to more organized urban spaces.

Motivated by the pressing need for efficient parking solutions, the project envisions a future where technology streamlines the allocation process. By optimizing routes and selecting suitable parking spots, the project aims to transform the parking experience, saving time for drivers and fostering sustainable urban mobility.

Motivation

Designing this system enables users to create a solution that streamlines the process of finding available parking spots.

It could significantly reduce the time users spend circling around parking lots searching for empty slots.

Related work

Efforts to optimize parking allocation and enhance urban mobility have spurred a range of related research and technological developments. The following represents a snapshot of key areas in related work:

- Smart Parking Solutions: Numerous smart parking systems have emerged, integrating real-time data from sensors and mobile apps to guide drivers to available parking spots. These systems utilize occupancy sensors, data analytics, and user interfaces to streamline parking allocation.
- Pathfinding Algorithms in Navigation: Pathfinding algorithms, such as Dijkstra's, and FloydWarshall, have been extensively used in navigation and robotics. These algorithms efficiently compute routes from point A to B, a concept directly applicable to guiding drivers to optimal parking spots.
- Parking Guidance Systems: Parking guidance systems, both indoor and outdoor, use various technologies like cameras and sensors to monitor parking spot availability. These systems offer real-time information to drivers, improving parking efficiency.
- Sustainable Urban Planning: Urban planning research addresses the need for sustainable transportation solutions. Integrating efficient parking allocation with urban planning strategies contributes to reduced traffic congestion and improved environmental outcomes.

Problem Statement

Develop an "Optimal Parking Spot Allocation System" that employs advanced algorithms to efficiently guide vehicles to suitable parking spot, reducing search times and enhancing urban mobility, user experience, and environmental sustainability.

Objectives

1. **Size-based Allocation:** Create an allocation system that considers the size of the user's vehicle and suggests spots that match the vehicle's dimensions

2. **Real-time Updates**: Design the system to handle real-time updates to the parking lot's occupancy status and dynamically adjust path-finding and allocation recommendations accordingly to calculate the most efficient routes from entry points to vacant parking spots within the parking lot.

3. **User-friendly Interface:** The interface should provide clear instructions, visual representations of parking spot availability, and an easy way for users to input their vehicle size and receive recommendations for the nearest available parking spot. The objective is to enhance the overall user experience and make the system accessible.

4. **Analytical View:** Provide an analytical view of the parking lot management system's performance by implementing and comparing different path-finding algorithms, such as Dijkstra's, to assess their efficiency in finding the most suitable routes from entry points to available parking spots within the parking lot.

Problems and Challenges in the model:

- **Grid Complexity**: As the parking lot grid grows in size, the graph representation and pathfinding algorithms become more complex. This can result in longer preprocessing times and slower path calculations.
- **Dynamic Update**: If the parking lot occupancy changes frequently, you need to handle real-time updates to the graph representation and shortest path data structures. This can be challenging to maintain efficiently.
- **Implementation Complexity:** Integrating different algorithms (Dijkstra's, binary search) and managing their interactions can lead to complex code and potential bugs.
- **Real-time Constraints:** The entire process needs to be fast enough for real-time parking spot allocation. Balancing computational speed with accuracy is critical.
- **Edge Cases and User Experience**: Handling edge cases like oversized vehicles, limited-size spots, or crowded areas can be challenging. Ensuring a smooth user experience in these cases is important.

Example illustration

In the bustling urban landscape characterized by a complex parking lot featuring parking spots of varying sizes, the deployment of the "Optimal Parking Spot Allocation System" is aimed at refining parking allocation processes and elevating the user experience. The system's methodology unfolds in a sequence of steps, as outlined below:

Step 1: Data Representation

The project commences with the establishment of a structured data representation of the parking lot. A 2D grid configuration is employed to map out individual parking spots indicating whether each spot is occupied or vacant, alongside denoting the size of each parking spot.

Step 2: Preprocessing

A pivotal facet of this preprocessing entails creating an innovative graph representation of the parking lot. In this conceptual graph, the vacant parking spots are translated into nodes, with edges interlinking adjacent spots, symbolizing pathways. The magnitude of the edges' weights corresponds to the distances between the respective parking spots.

Additionally, the vacant parking spots are subjected to size-based sorting, furnishing a foundation for streamlined binary search functionality in subsequent steps.

Step 3: Allocation Algorithm

3.1: Finding Shortest Path

When a vehicle seeks parking, a judicious amalgamation of shortest path algorithms serves to identify the most fitting parking spot. Depending on the parking lot's dimensions, distinct algorithms are engaged to optimize pathfinding:

- For smaller parking lots, the project employs algorithms such as Dijkstra's algorithm or the Floyd Marshall algorithm. These techniques ascertain the shortest route from the entry point to all available parking spots.

- For more expansive parking lots, the A* search algorithm comes to the fore. This heuristic-driven approach expeditiously determines an efficient path to the nearest vacant parking spot, thereby curtailing computation times.

3.2: Binary Search for Spot Size

Upon the identification of potential parking spots, the system harnesses the power of binary search. This search is executed on the sorted list of vacant parking spots, which has been organized based on spot sizes. This aligns the allocations process with the right size spots and narrows down the searching space.

Having settled on an appropriate parking spot, the project undertakes a further refinement of the allocation. In essence, the proposed methodology systematically navigates the complexities of parking allocation by combining data ate representation, preprocessing strategies, and algorithmic finesse. The orchestrated sequence of steps reflects a holistic endeavor to usher in a parking allocation system that thrives on efficiency and elevates the parking experience for users.

Proposed Method

The comparative analysis of pathfinding algorithms within the "Optimal Parking Spot Allocation System" command-line project aims to evaluate the efficiency and accuracy of various algorithms in the context of real-time parking spot allocation. The following approach outlines the implementation of multiple pathfinding algorithms, their execution, comparative analysis, and the selection of the most suitable algorithm based on performance metrics.

- **Data Initialization:** A representative parking lot grid is set up, encompassing diverse sizes of parking spots and varying occupancies to create realistic test scenarios. Entry points are designated, and predefined vehicle arrival scenarios are defined for consistent comparisons.

- **Algorithm Implementation:** Dijkstra's, Floydd Warshall and potentially additional pathfinding algorithms are implemented as separate modules within the project. Each algorithm module calculates the shortest path from parking lot entry points to vacant parking spots using its specific methodology.

- **Execution and Timing:** The execution time for each pathfinding algorithm is measured. For each algorithm, the system allocates parking spots for the predefined scenarios and records the execution time.

- **Comparative Analysis:** Collected execution times are analyzed to assess the efficiency of each algorithm under different parking lot conditions. Average execution times are calculated and compared to quantify algorithm performance.

- **Algorithm Selection:** The algorithm demonstrating the shortest average execution time consistently across scenarios is identified. Performance considerations encompass accuracy, path quality, and execution time.

- **Performance Visualization:** Text-based or graphical visualizations are included within the command-line interface to present the comparative analysis outcomes.

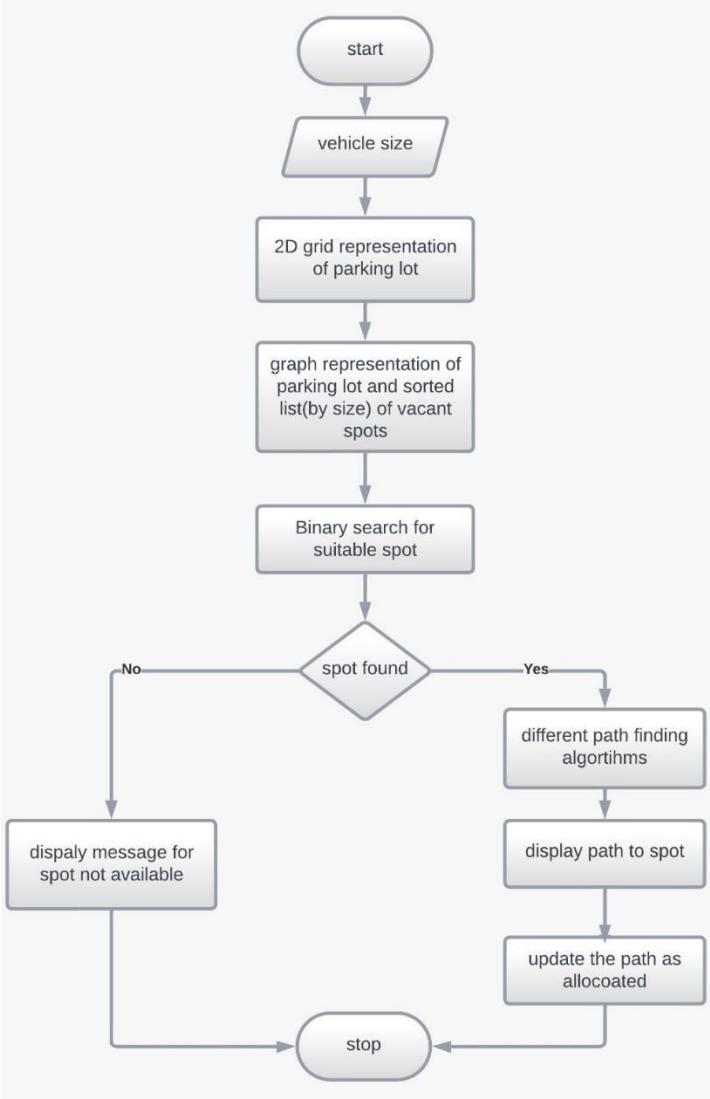
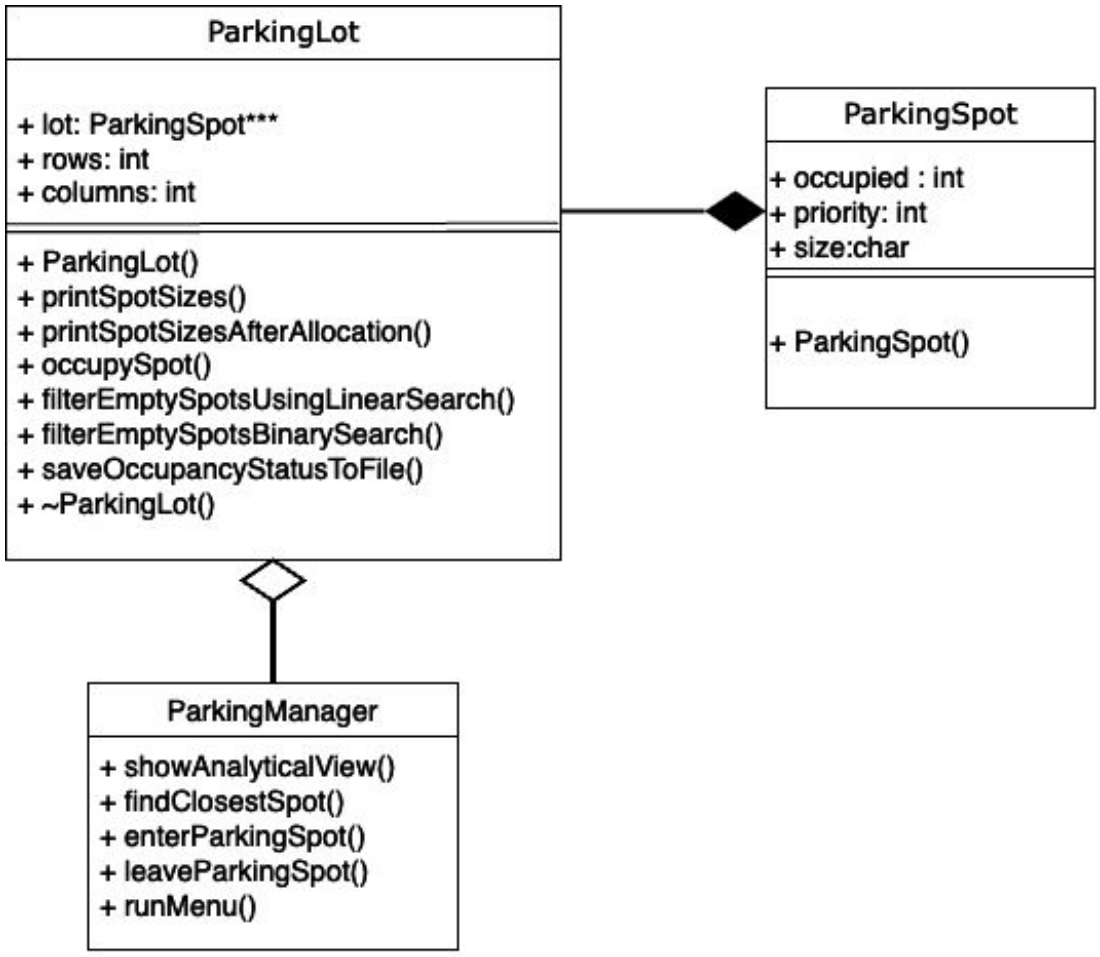


Fig: Proposed Flowchart

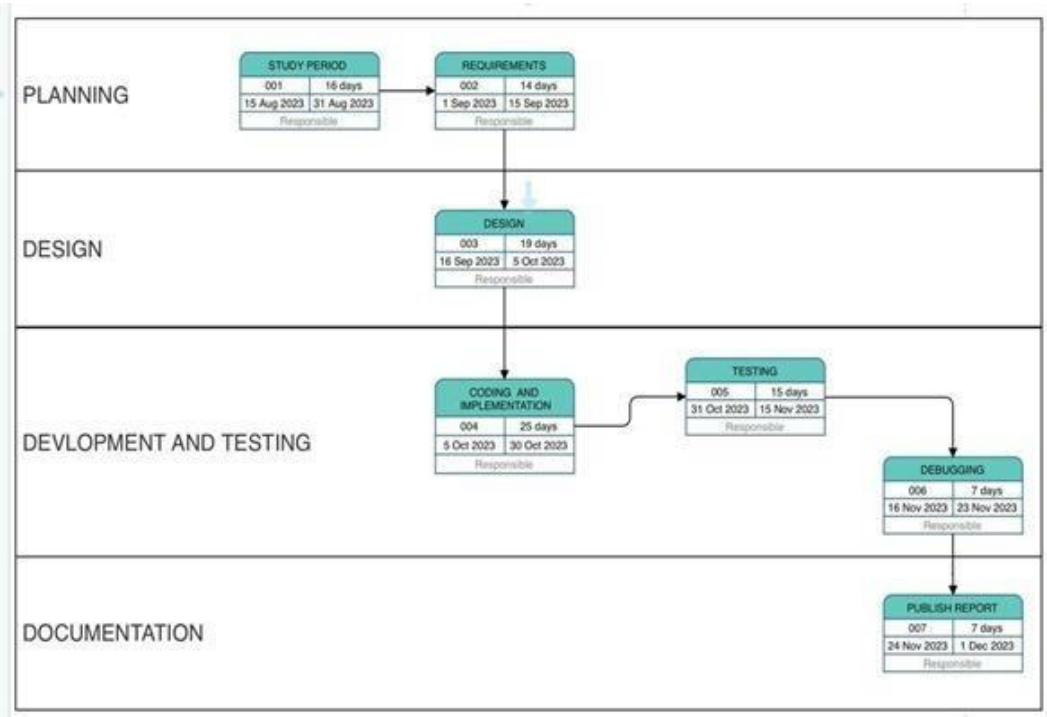
Users gain insights into the advantages of the selected algorithm in terms of performance metrics.

By following this approach, the "Optimal Parking Spot Allocation System" project systematically compares and evaluates pathfinding algorithms, ultimately enabling the informed selection of the most efficient algorithm for real-time parking spot allocation. This contributes to enhancing the overall user experience by optimizing the allocation process.

CLASS DIAGRAM :



Pert Chart



HOW THE PROJECT WORKS

Initialization:

The program starts by opening a file ("occupancy.txt") to read the occupancy status of the parking lot.

It reads the occupancy status from the file and initializes a 2D matrix (array of ParkingSpot objects) representing the parking lot.

The size of the parking lot (number of rows and columns) is set based on the data in the file.

Matrix Representation:

The parking lot is represented as a 2D matrix (ParkingSpot*** lot) of ParkingSpot objects.

Each element of the matrix represents a parking spot with attributes like its size (S for small, M for medium, L for large), occupancy status (1 for occupied, 0 for unoccupied), and priority.

User Interaction:

The program interacts with the user by asking them to input their vehicle size (S, M, or L) in the main function.

It also gives the user the option to choose a combination of algorithms and approaches for finding the nearest parking spot.

Algorithm Selection:

Based on the user's choice of algorithm and approach, the program uses one of the following methods to find the nearest empty parking spot: findNearestSpotUsingDijkastras with Dijkstra's algorithm and either linear or binary search. findNearestSpotBellmanFord with Bellman-Ford algorithm and either linear or binary search.

Finding the Nearest Spot:

The selected algorithm is applied to calculate the nearest parking spot based on the chosen approach.

The algorithm calculates distances and finds the parking spot with the minimum distance from the entrance (first row and first column in the matrix).

Display Results:

The program displays the results, including the chosen algorithm, approach, and the time taken to find the nearest spot.

It also displays the coordinates (row and column) of the nearest parking spot.

Matrix Update:

After finding and displaying the nearest spot, the program updates the occupancy status in the matrix to mark the spot as occupied (sets occupied to 1).

File Update:

The program saves the updated occupancy status to the "occupancy.txt" file, which reflects the current state of the parking lot.

Display Parking Lot Status:

It prints the current occupancy status of the parking lot, displaying each parking spot as a cell in a grid.

Unoccupied spots are shown in green, and the nearest spot is highlighted in a different color.

Error Handling:

The program handles errors, such as when there are no available parking spots for the given vehicle size or when the user provides an invalid option.

Memory Cleanup:

The program cleans up memory through the destructor when it's done. This involves deallocating memory for the 2D matrix of ParkingSpot objects.

Code Summary:

1. **Classes:**

- **ParkingSpot:** A single parking place with features such size ('S' for small, 'M' for medium, and 'L' for large), occupancy status (1 for occupied, 0 for vacant), and priority based on size.

- **ParkingLot:** Controls the parking lot, which is a two-dimensional array of ParkingSpot objects. It reads and initialises the parking lot from a file depending on occupancy state, provides ways for filtering and finding empty spaces using linear or binary search, and handles tasks such as occupying a place and writing occupancy status to a file.
- **ParkingLotManager:** Orchestrates the parking system's basic logic. It communicates with the user, locates the nearest available parking spot using various algorithms, and provides an analytical perspective of the parking lot. It also has functions for entering and leaving parking spots.

2. **Algorithms:**
   To discover the nearest open parking spot, the code applies several techniques, including Dijkstra's algorithm (both linear and binary search versions), the Bellman-Ford algorithm, and the A* algorithm. These algorithms take distance from the entrance into account and prioritize slots based on size and occupancy status.

3. **User Interaction:**
   The ParkingLotManager class provides the user with a menu-driven interface. Users have the option of entering a parking spot (specifying vehicle size), leaving a parking spot, or exiting the programme.

4. **File Handling:**
   The code reads and writes the status of occupancy to a file (occupancy.txt). After a parking spot is occupied, the file is used to initialize the parking lot and preserve the occupancy state.

5. **Visualization:**
   The code provides routines for printing the current status of the parking lot as well as a visual representation of the parking lot when a vehicle has been parked, emphasizing the closest spot.

6. **Handling Exceptions:**
   Exception handling is included in the code to solve potential issues such as invalid user input or when no parking spaces are available.

7. **Optimization:**
   The optimization part involves efficiently locating the nearest available parking spot using various search algorithms and minimizing the search space based on the vehicle's size.

In summary, the code represents a programme for an optimized parking system including user interaction, algorithmic approaches to parking spot detection, and efficient data management for parking lot status.

Conclusion:

This Software Requirements Specification outlines the requirements for the Parking Spot Management System, including the implementation of pathfinding algorithms and searching techniques for finding the nearest available parking spot. The system aims to conduct a comparative study to evaluate the performance of these algorithms and techniques.

The "Optimal Parking Spot Allocation System" project uses this approach to compare and assess pathfinding algorithms, allowing for the informed selection of the most efficient algorithm for real-time parking spot distribution. This improves the overall user experience by optimizing the allocation pro

References:

[1] A Balanced Algorithm for In-City Parking Allocation: A Case Study of Al Madinah City
Mohammad A. R. Abdeen,* Ibrahim A. Nemer,and Tarek R. Sheltami
Giovanni Pau, Academic Editor

[2] An Improved Vehicle Parking Mechanism to reduce Parking Space Searching Time using Firefly Algorithm and Feed Forward Back Propagation Method by Ruby Singh1*, Chiranjit Dutta, Niraj Singhal, Tanupriya Choudhury

[3] Smart-parking management algorithms in smart city Mahdi Jemmali, Loai Kayed B. Melhim1*, MafawezT.Alharbi, Abdullah Bajahzar & Mohamed Nazih Omri

[4] Automated parking system using graph algorithm by Ashim Md. Ahsan Fahim and Shahrin Chowdhury