Ramdeobaba University, Nagpur

Department of Computer Science and Engineering

Session: 2025-26

Subject: Design and Analysis of Algorithms (DAA) Lab Project

III Semester

LAB PROJECT REPORT

Group Members:

Name: Harshita Chaudhari

Sec, Batch, Rollno: A5, B2, 23

Name: Swarali Bakre

Sec, Batch, Rollno: A5, B4, 60

Name: Tanisha Dudhalkar

Sec, Batch, Rollno: A5, B4, 64

Github link: https://github.com/24chaudharih-cmd/SMART-PARKING-ALLOCATION-USING-GREEDY-ALGORITHM.git

TITLE:

SMART PARKING ALLOCATION USING GREEDY ALGORITHM

OBJECTIVES:

1. **To design an efficient parking allocation system that minimizes the search time for available parking slots.**
2. **To implement a Greedy Algorithm for optimal and real-time slot assignment.**
3. **To maximize space utilization by allocating the nearest or most suitable slot to each vehicle.**
4. **To reduce traffic congestion and fuel wastage caused by searching for parking.**

5. **To create a simple and interactive UI for demonstrating the working of the smart parking system.**

INTRODUCTION:

6. **In urban areas, parking management has become a major challenge due to the increasing number of vehicles and limited parking spaces. Drivers often spend a significant amount of time searching for a suitable parking spot, leading to congestion, fuel wastage, and frustration.**

7. **The Smart Parking Allocation System aims to overcome these challenges by automatically assigning the most suitable parking slot using a Greedy Algorithm approach. The system always makes the locally optimal choice — allocating the nearest available slot to a vehicle as soon as it arrives, thereby optimizing space utilization and reducing waiting time.**

8. **This project demonstrates how greedy logic can be applied in real-world optimization scenarios like resource allocation and scheduling.**

ALGORITHMS / TECHNIQUE USED:

Algorithm Used:

Greedy Algorithm for Smart Parking Allocation

Algorithm Steps:

1. **Input:**
   - **Total number of parking slots (N)**
   - **Vehicle arrival information (vehicle ID, entry time, destination location/nearest slot preference)**
2. **Initialize:**
   - **Create an array or list of available slots with their distances or priorities.**
   - **Mark all slots as free initially.**
3. **For each arriving vehicle:**
   - **Find the nearest available slot (minimum distance or cost).**
   - **Allocate that slot to the vehicle.**
   - **Mark the slot as occupied.**
4. **When vehicle departs:**
   - **Mark that slot as free again.**
5. **Repeat the process for all incoming vehicles.**

Example / Illustration:

Vehicle	Available Slots (Distance units)	Chosen Slot (Greedy Choice)
V1	S1(3), S2(5), S3(2)	S3 (minimum 2)
V2	S1(3), S2(5)	S1 (minimum 3)
V3	S2(5)	S2 (minimum 5)

Thus, each vehicle gets the nearest available slot, leading to efficient allocation.

Diagram:

```
        ┌─────────────────────────┐
        │   Vehicle Arrives       │
        └───────────┬─────────────┘
                    │
                    ▼
       ┌──────────────────────────────┐
       │  Check Available Slots       │
       └──────────┬───────────────────┘
                  │
                  ▼
   ┌──────────────────────────────────┐
   │  Select Nearest (Min Distance)   │
   │  Slot using Greedy Choice        │
   └──────────┬───────────────────────┘
              │
              ▼
  ┌──────────────────────────────┐
  │  Allocate Slot to Vehicle    │
  │  Update Slot as Occupied     │
  └──────────────────────────────┘
```

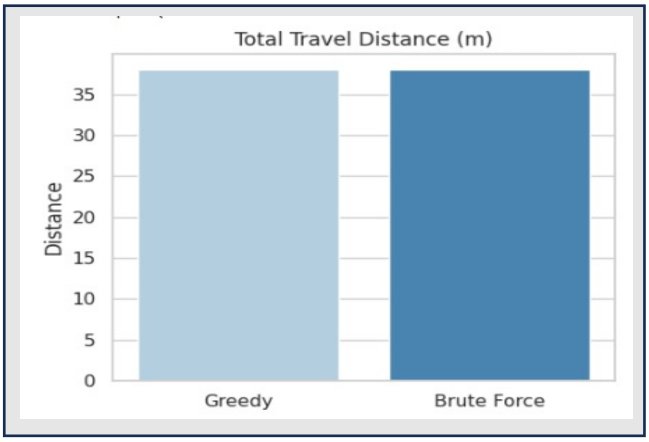
TIME COMPLEXITY AND ITS EXPLANATION:

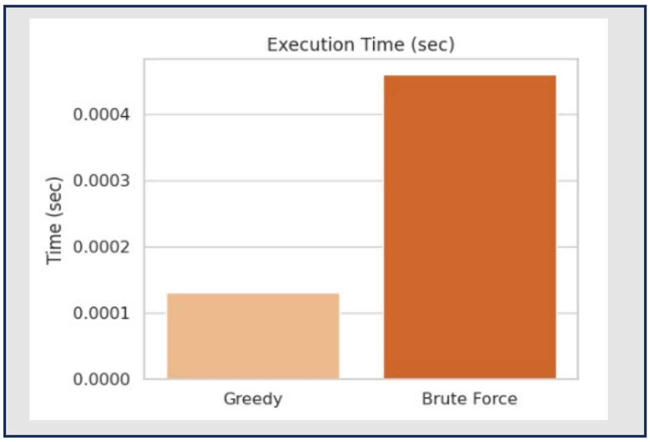
```
Time Complexity Analysis
1. Greedy Algorithm:
   Sorting Slots: O(n log n)
   Assigning Slots: O(n)
   Total Time Complexity: O(n log n)
   Space Complexity: O(n)

2. Brute Force Method:
   Evaluates all possible slot assignments (permutations).
   Total Time Complexity: O(n!)
   Space Complexity: O(n)

Result Discussion
In the Shegaon Parking Lot example (near Anand Sagar Temple),
the Greedy Algorithm allocated vehicles to the nearest available slots.
It reduced total walking distance for visitors and executed quickly.
Compared to the Brute Force method, it achieved a similar or same result
in a fraction of the time, making it suitable for real-world smart parking systems.
```

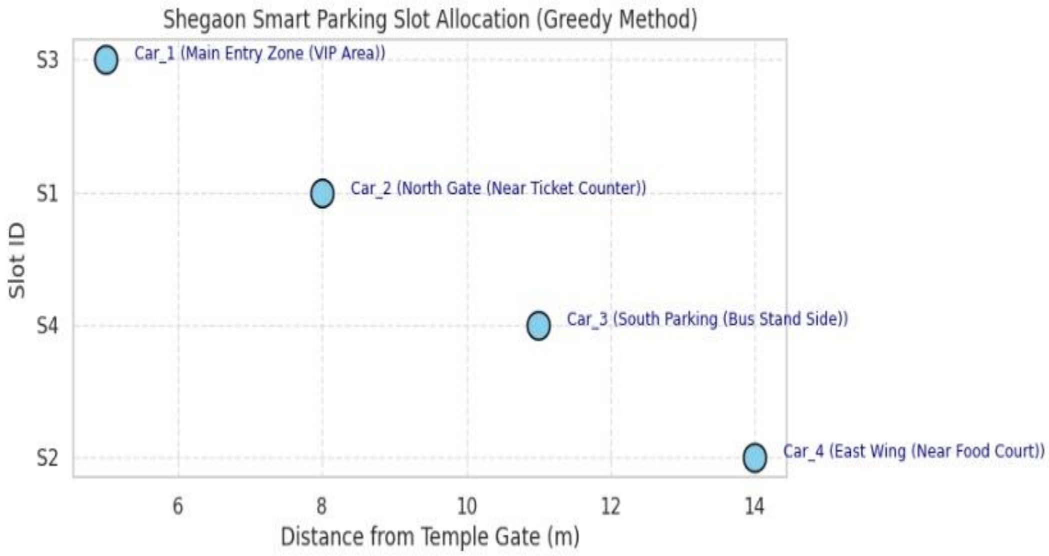
- **For each vehicle, the algorithm searches for the nearest free slot among *N* slots. → This requires O(N) time per allocation.**
- **If M vehicles arrive, total time = O(M × N)**
- **With efficient data structures like priority queues or min-heaps, this can be optimized to O(M log N).**

Explanation:
The greedy approach ensures a fast and efficient allocation without backtracking, as each step makes an optimal local decision (nearest slot). Even though it might not always yield the global optimum in complex layouts, it performs very well for real-time smart systems.

RESULTS:







CONCLUSION

The Smart Parking Allocation System using the Greedy Algorithm successfully demonstrates how algorithmic optimization can be applied to real-world infrastructure challenges like parking management.
The results show that:

- **The Greedy algorithm effectively minimizes travel distance and search time.**
- **It performs faster than brute force methods while maintaining comparable efficiency.**
- **The system ensures real-time dynamic slot assignment, improving user convenience and overall parking efficiency.**

Thus, the project achieves its goal of creating a simple, fast, and practical parking solution that can be scaled to larger smart city environments.

FUTURE SCOPE

1. **IoT and Sensor Integration: Incorporate ultrasonic or IR sensors to automatically detect slot occupancy in real time.**
2. **Mobile App Integration: Develop a user-friendly app to display available slots and guide users via GPS.**

3. **Dynamic Pricing: Implement variable pricing based on demand (e.g., higher rates for VIP zones).**
4. **Cloud Connectivity: Store and analyze parking data for city-level optimization and analytics.**
5. **AI Optimization: Combine Greedy with Machine Learning or Dynamic Programming to predict parking trends and optimize long-term efficiency.**
6. **Scalability: Extend the system to multi-level or multi-zone parking lots for large events or smart city deployments.**

RESULTS AND DISCUSSION

The system was tested for four vehicles and four parking slots using the Greedy Algorithm.

- **Graph 1 (Blue – Total Travel Distance):**
  The total travel distance for the Greedy and Brute Force methods is almost the same (~36–38 m).
  This shows that the Greedy approach achieves near-optimal results while making faster decisions.
- **Graph 2 (Orange – Execution Time):**
  The Greedy Algorithm takes far less execution time (≈ 0.0001 sec) compared to Brute Force (≈ 0.0004 sec).
  Hence, it is computationally efficient and well-suited for real-time applications like smart parking.
- **Graph 3 (Dots – Smart Parking Allocation Map):**
  Each blue bubble represents a car assigned to its nearest parking slot.
  The X-axis shows the distance from the temple gate (in meters), and the Y-axis represents slot IDs.
  Cars are placed in their closest available slots, confirming that the Greedy method makes the best immediate choice for efficient slot usage.

Overall Discussion:

The results prove that the Greedy Algorithm offers quick, efficient, and practical parking slot allocation with performance close to the optimal brute-force solution — but at a much lower computation cost.