# Optimizing Parcel Delivery Routes

Group 11


Muhammad Syafi (214321)
Ariff Ameer Aizad (214037)
Muhammad Fazrul Hafizi (214271)
Muhammad Arif Zahiruddin (214264)

## OVERVIEW OF THE ISSUE

Courier services today often plan delivery routes manually, which leads to longer delivery times, higher fuel costs, and poor adaptation to real-time changes like new orders or traffic conditions.

- It doesn’t always find the shortest or fastest route.
- It can’t quickly adapt to changes during the day.
- It causes delays, customer dissatisfaction, and wasted resources.

# PROBLEM STATEMENTS

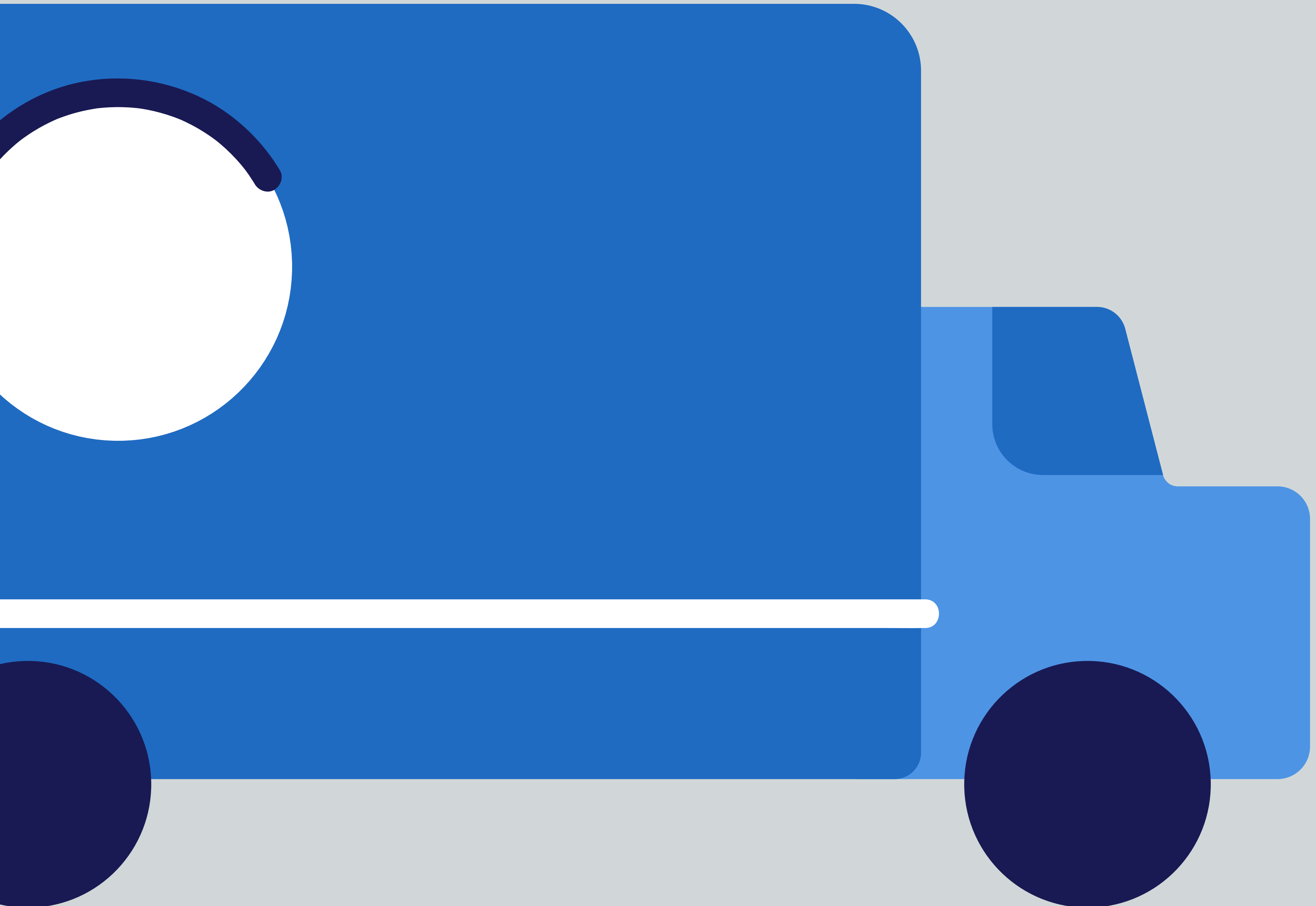
Goal:

Determine the shortest route for a courier to visit all required destinations once and return to the starting point

- Courier takes too long to delivery parcel
- Current parcel delivery routes are manually planned
- do not adapt well to changing delivery locations

# OBJECTIVE

A smart, fast, and scalable parcel delivery ecosystem.



- To minimize time taken to deliver parcel
- To automate delivery route plan
- To adapt to changing delivery location

## WHY IT'S IMPORTANT

- leads to significant cost reductions for delivery companies.
- vastly improves customer satisfaction
- contribute to environmental sustainability

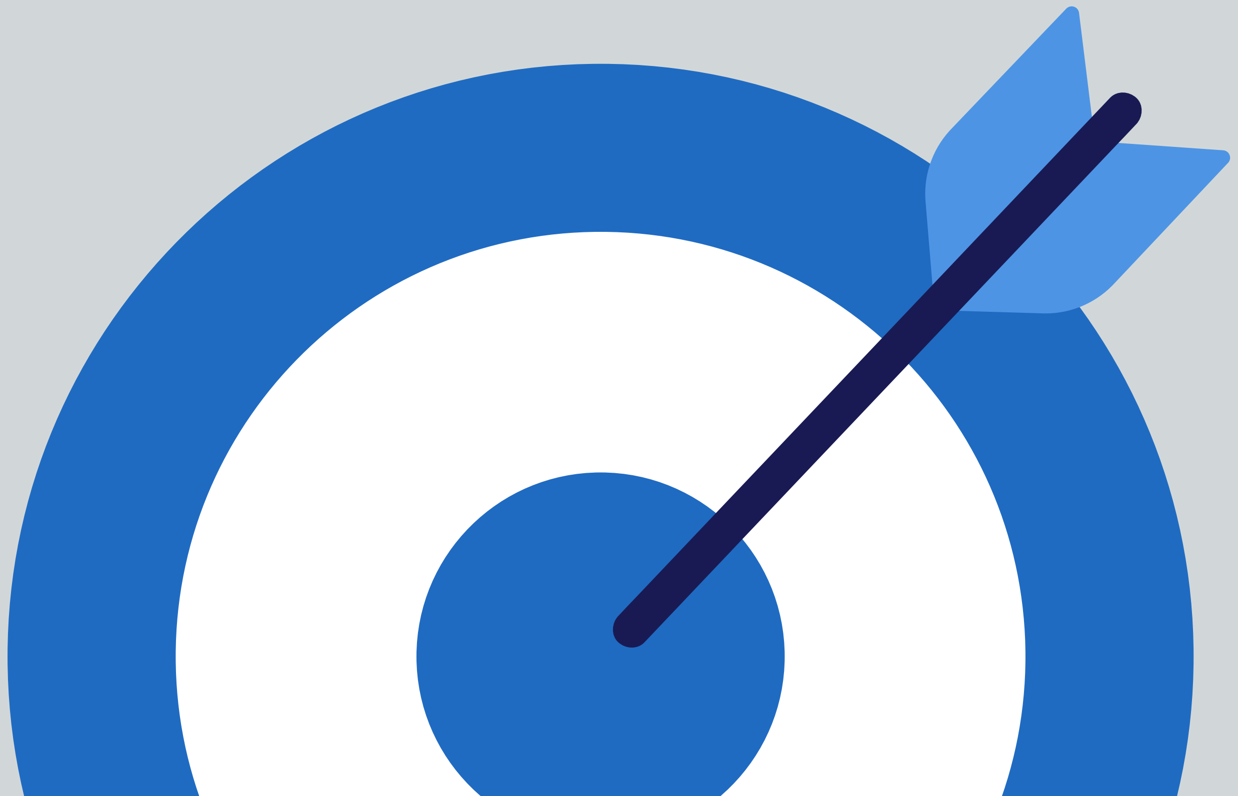
EXPECTED OUTPUT

Minimize delivery distance

Dynamic re-routing capability

Shorter delivery time

Automated route planning module



# ALGORITHM OVERVIEW

- Helps solve hard real-world problems efficiently.
- The right algorithm for solving problems faster
- Hybrid methods can achieve balanced between performance and precision
- Specific algorithm for specific problems

# ALGORITHM DESIGN

Flexible and intelligent delivery solutions.

- Greedy Algorithm
- Dynamic Programming
- Divide and Conquer
- Sorting Algorithm
- Graph Algorithm

# COMPARISON OF ALGORITHM

| | Greedy Algorithm | Dynamic Programming | Graph | Divide and Conquer | Sorting Algorithm |
|---|---|---|---|---|---|
| **Strength** | Fast and Simple | Guarantees Optimal Solution | Excellent for Shortest path | Great for sorting and structured data processing | Faster search operations |
| **Weakness** | May lead to suboptimal solutions | High time and space complexity | Cannot solve multi-node route problems alone | Not suitable for problems requiring global path awareness | Extra computational overhead |
| **Time Complexity** | O(n log n) | $O(n^2)$ to $O(n^3)$ | varies | O(n log n) | O(n log n) |

CHOSEN ALGORITHM OVERVIEW

## Dynamic Programming

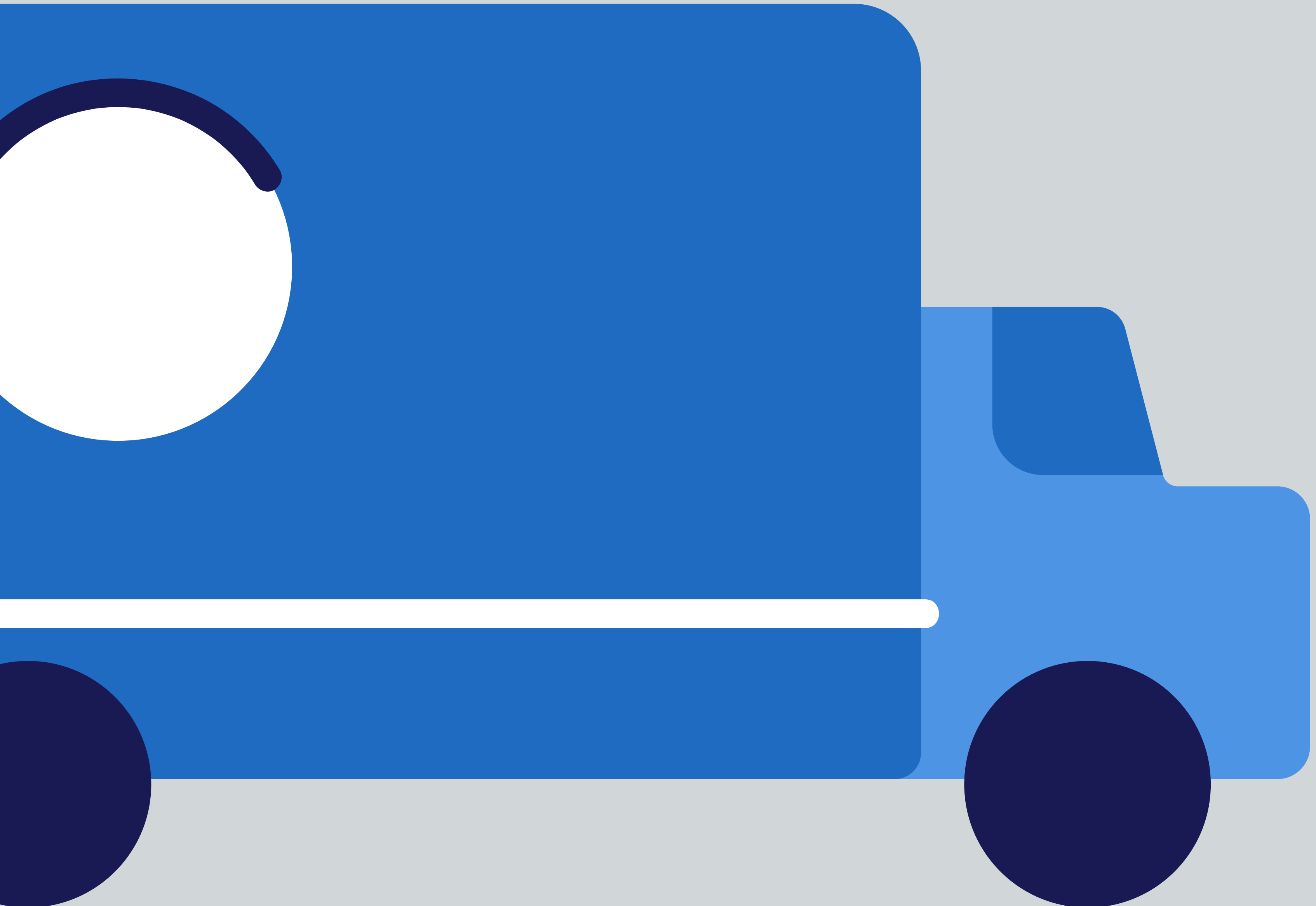
DP is chosen because it guarantees the shortest possible route, which is crucial for reducing fuel costs, travel time and ensuring efficient delivery in courier operations.

## Greedy Algorithm

Greedy algorithm is fast, simple, and effective for real time delivery routing by always choosing the nearest unvisited location. While not always optimal, it scales well and works best when combined with heuristics to quickly approximate efficient routes.

## DATA TYPE

- Visited Nodes (Boolean)
- Matrix (Array)
- Path (List)



# OBJECTIVE FUNCTION

## MINIMIZE DELIVERY TIME

- This objective aims for the fastest possible parcel delivery.
- The system will create routes that cut down total travel time.
- It also reduces idle time for drivers.
- The goal is to get parcels to customers quickly.
- This considers factors like distance, speed, and time spent at each stop.

## AUTOMATE ROUTE PLANNING

- This objective removes manual effort from route planning.
- Technology will generate the best delivery routes.
- Algorithms and data will calculate efficient stop sequences.
- The system will also assign parcels to specific vehicles.
- This automation boosts consistency and cuts down on human errors.
- It also allows for quick re-planning when things change.

## ADAPT TO CHANGING LOCATION

- This objective addresses real-world delivery changes.
- The automated system must be flexible.
- It needs to re-optimize routes fast, almost instantly.
- This applies to new orders, cancellations, or traffic shifts.
- It ensures the delivery plan stays efficient.
- The system can then handle an ever-changing environment.

## THE CONSTRAINT

**Time:** The courier must complete all deliveries with the shortest possible travel distance in order to reduce fuel costs, improves delivery speed and ensures customer satisfaction.

**Visit-once constraint:** Each delivery location must be visited exactly once to avoid backtracking to prevent inefficiency and ensures fair services for all destinations.

# PSEUDOCODE GREEDY

```
function NearestNeighbor(matrix, start):
    current ← start
    repeat until all nodes are visited:
        find nearest unvisited node to current
        mark it visited
        add to path
        current ← nearest
    return path
```

# PSEUDOCODE DP

RECURRENCE ->

```
function TSP_DP(dist):
    for each subset mask:
        for each current node u in mask:
            for each unvisited node v:
                update dp[nextMask][v] with min cost
                update parent for path reconstruction

    find endNode with minimum cost from fullMask

    path = reconstruct path from parent table
    return minCost
```

# ALGORITHM ANALYSIS

## Algorithm Correctness

- Partial Correctness:

  The algorithm must return a valid delivery route that visits all locations exactly once and returns to the start.

- Termination:

  The DP and Greedy algorithms must stop after computing the optimal (or shortest) route.

# DYNAMIC PROGRAMMING

## Algorithm Correctness

Correctness Goal:
Find the optimal full delivery route that visits all locations exactly once and returns to the start

Partial Correctness:
Always returns the minimum total route distance for small input sizes

Termination:
Terminates after processing all subproblems (bitmask states)

Duplicate Visit:
No duplicates (each location visited once by design)

# GREEDY ALGORITHM

## Algorithm Correctness

Correctness Algorithm:
Chooses the next closest unvisited location at each step to build a delivery path

Partial Correctness:
Returns a valid route, but not guaranteed to be globally optimal

Termination:
Stops when all delivery points are visited or constraints are met

Duplicate Visit:
Avoids revisiting nodes in simple cases, but may loop in complex route planning like TSP without safeguards

# GREEDY ALGORITHM COMPLEXITY

$O(n^2)$

- n = Number of Delivery points

### Best Case

$O(n^2)$

- Even if the distances are ideally laid out the algorithm still checks all unvisited nodes at each step

### Average Case

$O(n^2)$

- the algorithm performs the same number of comparisons: scanning all unvisited nodes to find the nearest one.

### Best Case

$O(n^2)$

- the algorithm still checks all nodes, but may also produce a poor-quality route.

# DP ALGORITHM COMPLEXITY

$O(n^2 \times 2^n)$

- n = Number of Delivery Points

**Best Case**

$O(n^2 \times 2^n)$

- most optimistic performance under ideal input conditions.

**Average Case**

$O(n^2 \times 2^n)$

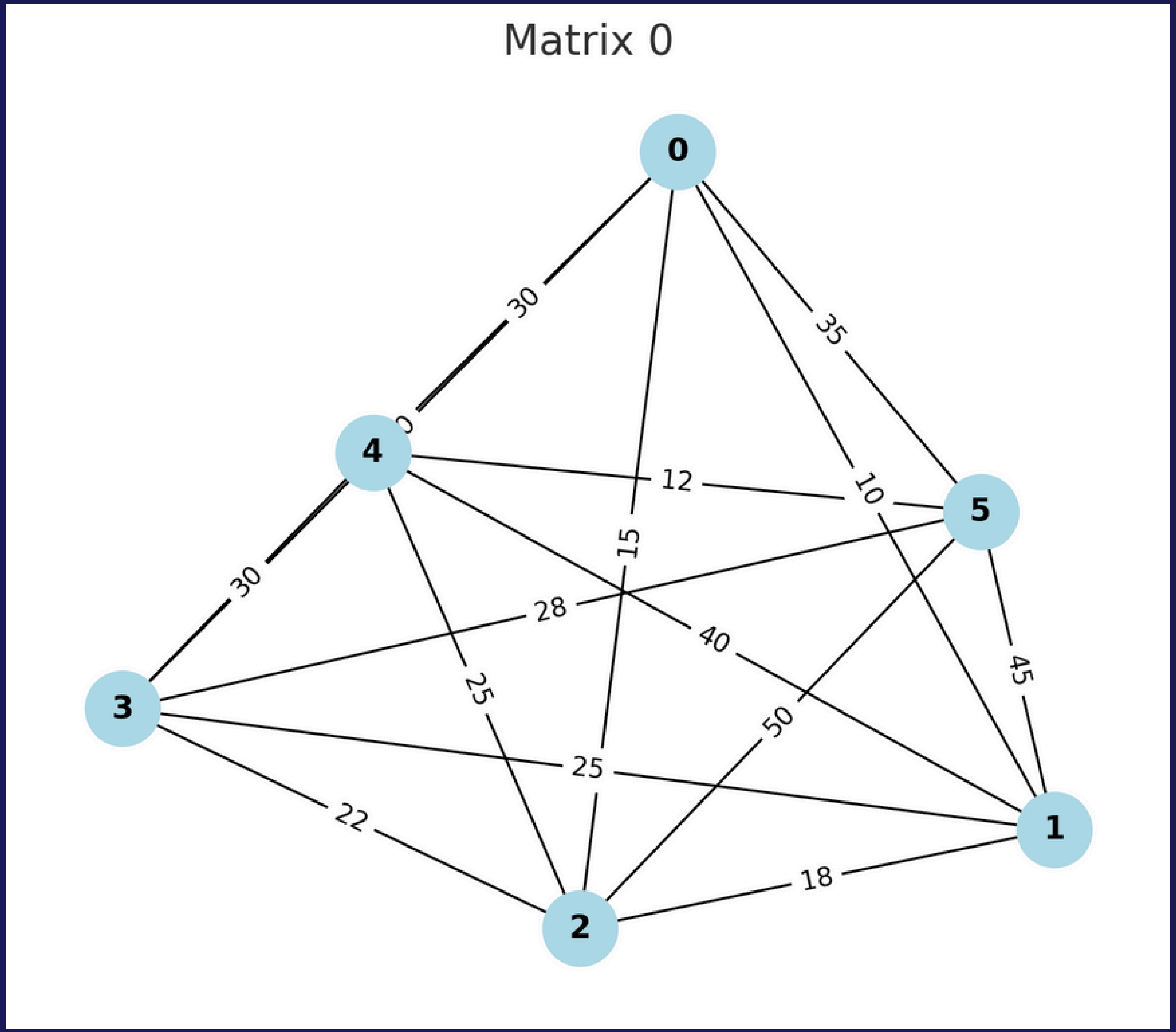
- most realistic performance expectation for typical inputs.

**Best Case**

$O(n^2 \times 2^n)$

- provides a crucial upper bound guarantee on performance, regardless of input.

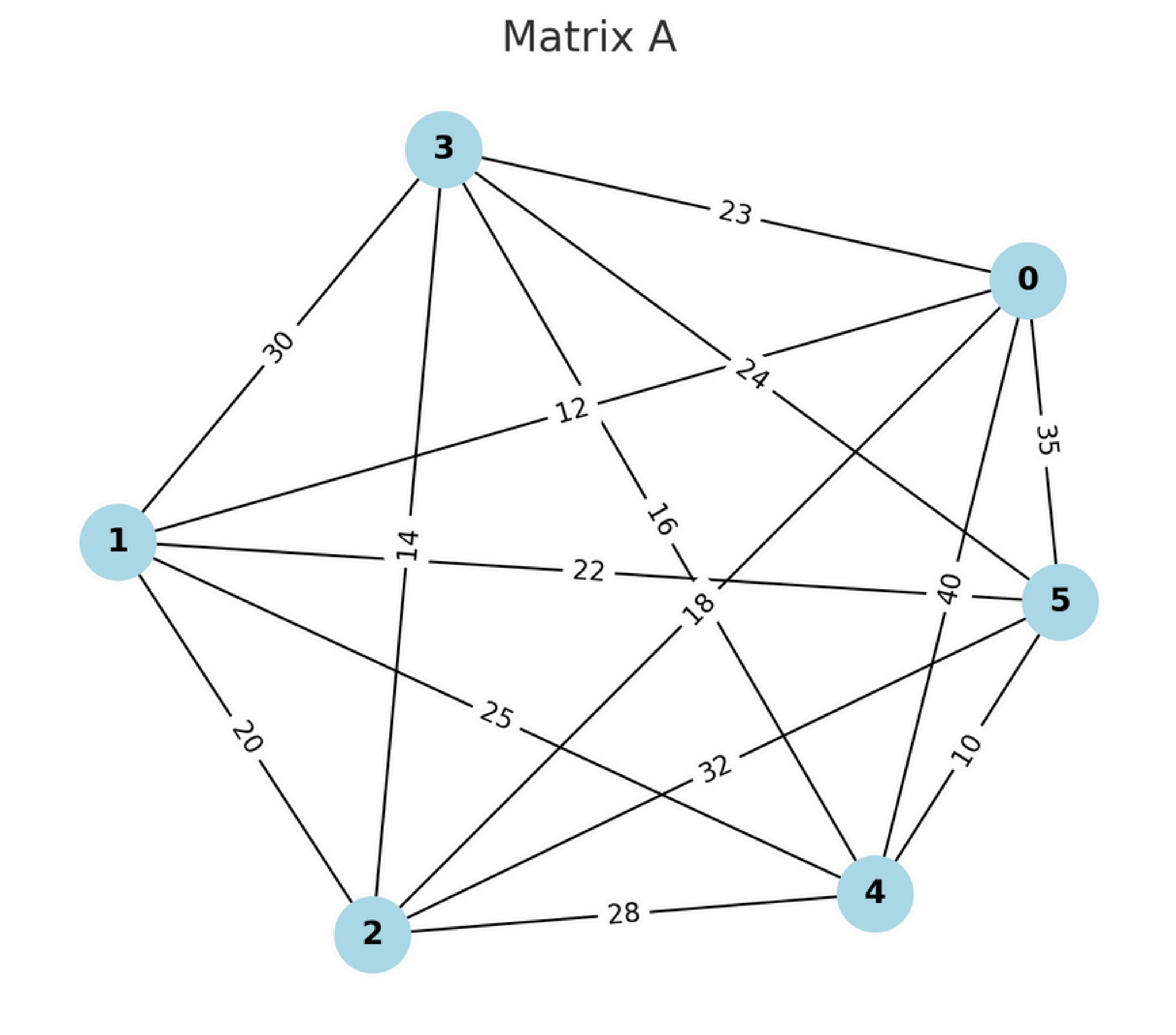
RESULT




```
{ 0, 10, 15, 20, 30, 35 },
{ 10, 0, 18, 25, 40, 45 },
{ 15, 18, 0, 22, 25, 50 },
{ 20, 25, 22, 0, 30, 28 },
{ 30, 40, 25, 30, 0, 12 },
{ 35, 45, 50, 28, 12, 0 }
```

```
Greedy Route: 0 -> 1 -> 2 -> 3 -> 5 -> 4 -> 0
Greedy Route Cost: 120
DP Route: 0 -> 0 -> 3 -> 5 -> 4 -> 2 -> 1 -> 0
DP Route Cost: 113
```

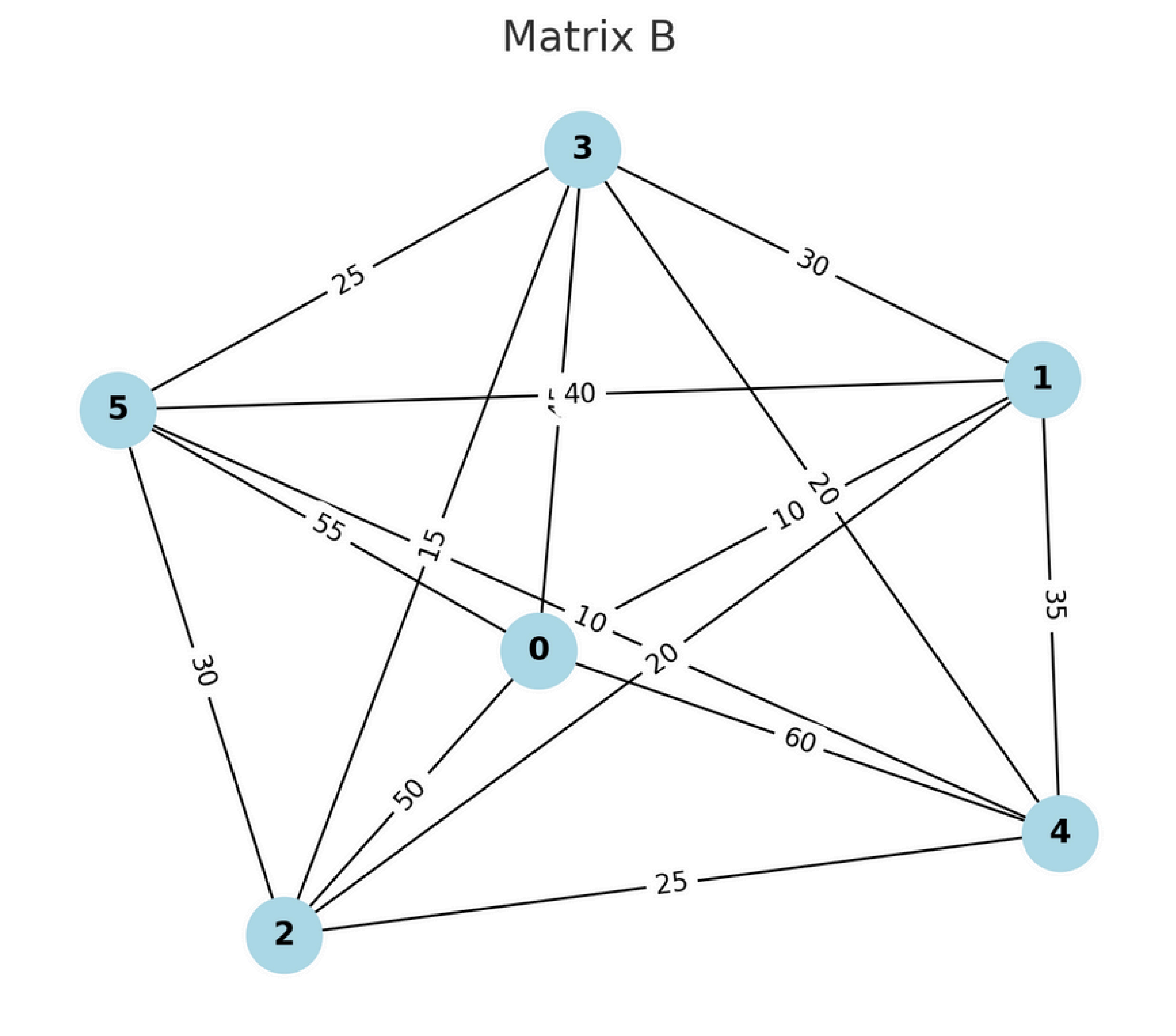
# RESULT




```
{ 0, 12, 18, 23, 40, 35 },
{ 12, 0, 20, 30, 25, 22 },
{ 18, 20, 0, 14, 28, 32 },
{ 23, 30, 14, 0, 16, 24 },
{ 40, 25, 28, 16, 0, 10 },
{ 35, 22, 32, 24, 10, 0 }
```

```
Greedy Route: 0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 0
Greedy Route Cost: 107
DP Route: 0 -> 0 -> 2 -> 3 -> 4 -> 5 -> 1 -> 0
DP Route Cost: 92
```

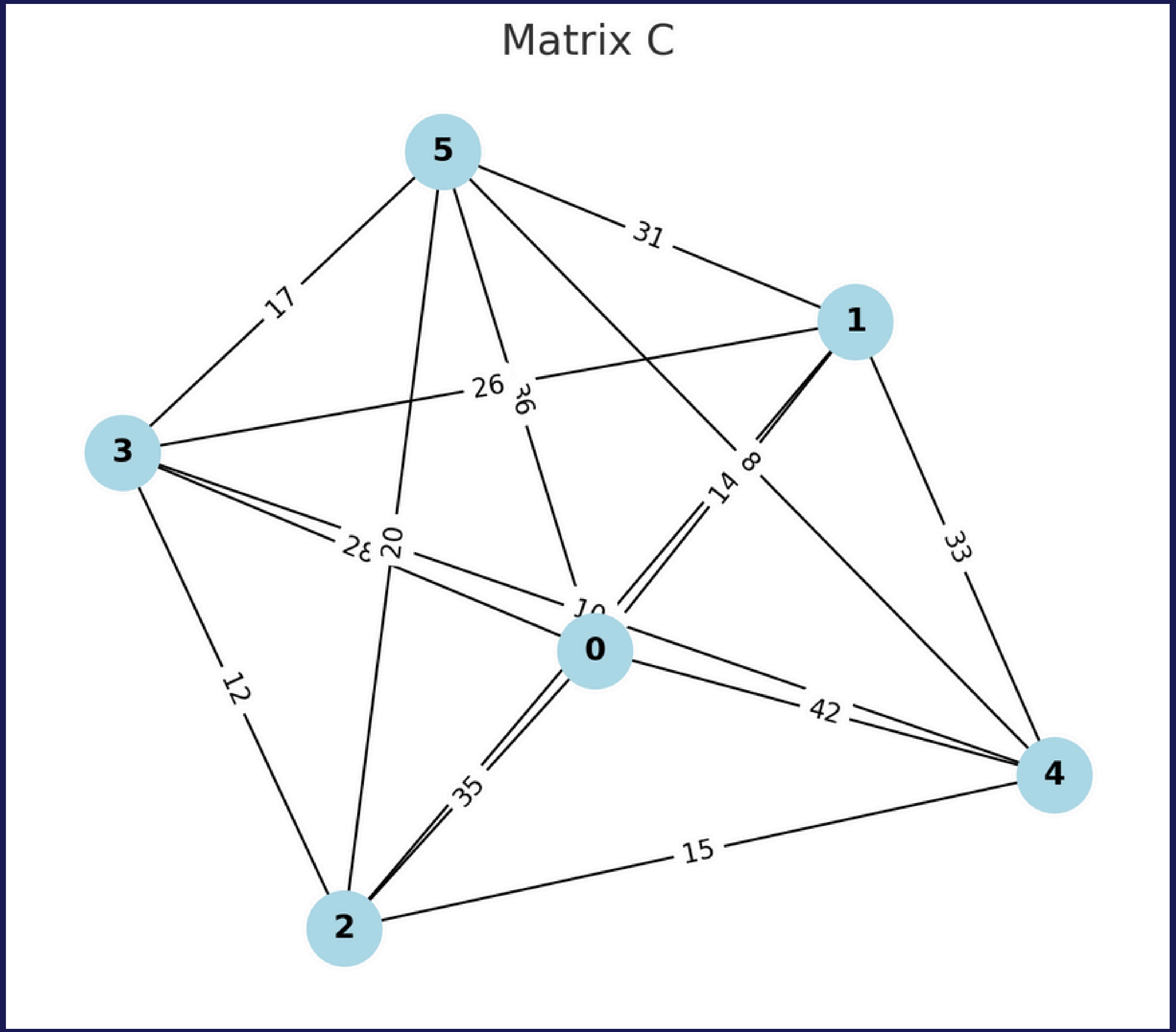
RESULT




```
{ 0, 10, 50, 45, 60, 55 },
{ 10, 0, 20, 30, 35, 40 },
{ 50, 20, 0, 15, 25, 30 },
{ 45, 30, 15, 0, 20, 25 },
{ 60, 35, 25, 20, 0, 10 },
{ 55, 40, 30, 25, 10, 0 }
```

```
Greedy Route: 0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 0
Greedy Route Cost: 130
DP Route: 0 -> 0 -> 5 -> 4 -> 3 -> 2 -> 1 -> 0
DP Route Cost: 130
```

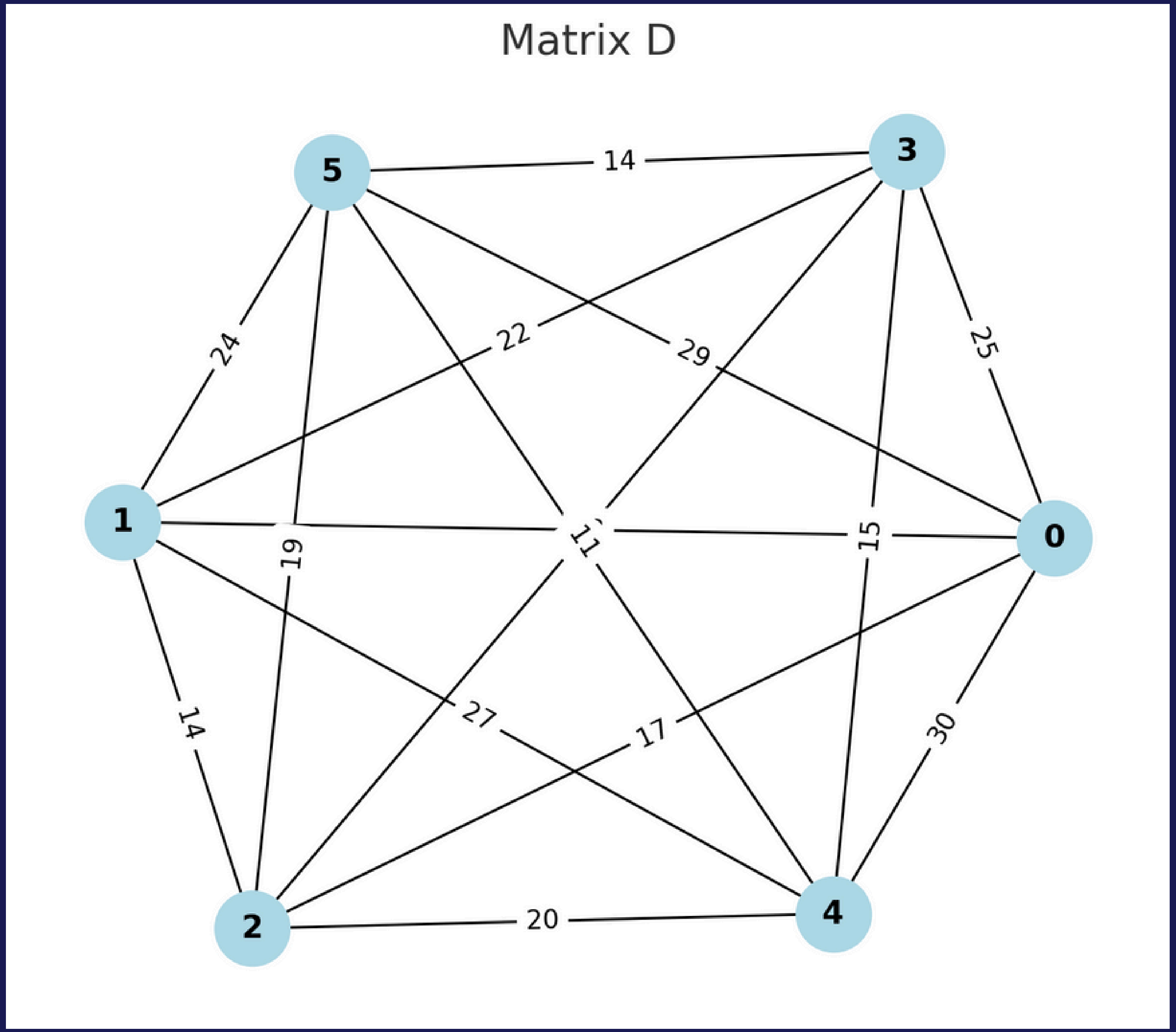
# RESULT




```
{ 0, 14, 35, 28, 42, 36 },
{ 14, 0, 18, 26, 33, 31 },
{ 35, 18, 0, 12, 15, 20 },
{ 28, 26, 12, 0, 10, 17 },
{ 42, 33, 15, 10, 0, 8 },
{ 36, 31, 20, 17, 8, 0 }
```

```
Greedy Route: 0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 0
Greedy Route Cost: 98
DP Route: 0 -> 0 -> 5 -> 4 -> 3 -> 2 -> 1 -> 0
DP Route Cost: 98
```

RESULT




```
{ 0, 13, 17, 25, 30, 29 },
{ 13, 0, 14, 22, 27, 24 },
{ 17, 14, 0, 10, 20, 19 },
{ 25, 22, 10, 0, 15, 14 },
{ 30, 27, 20, 15, 0, 11 },
{ 29, 24, 19, 14, 11, 0 }
```

```
Greedy Route: 0 -> 1 -> 2 -> 3 -> 5 -> 4 -> 0
Greedy Route Cost: 92
DP Route: 0 -> 0 -> 2 -> 3 -> 4 -> 5 -> 1 -> 0
DP Route Cost: 90
```

OBSERVATION

# DP PERFORMS BETTER

- Out of 5 graphs, DP goes through shorter paths in 3 of them, while the other 2 graphs, have tied with Greedy algorithm

## CONCLUSION

This project evaluated Greedy and Dynamic Programming (DP) algorithms for optimizing courier delivery routes. DP outperformed Greedy in 3 out of 5 test cases, offering shorter and more efficient routes, while Greedy provided faster results with less accuracy.

Automating route planning significantly helps to reduce delivery time, lower costs, and improve overall service quality, leading to a more efficient and scalable delivery system.

# Thank you

Join us in delivering the future, one parcel at a time.