DESIGN AND ANALYSIS OF ALGORITHM – CBS 3003

DIGITAL ASSIGNMENT – 2

1. TITLE
   Optimizing Delivery Routes for Cost effective and Efficient Package Delivery

2. TEAM MEMBERS
   Anvesha Churi - 22BBS0032
   Pratyush Dubey - 22BBS0064

3. DISTRIBUTION OF TASKS
   a. **Problem Definition and Design:**
      i. Anvesha: Led the ideation of the problem, identified key constraints, and formulated the route optimization challenge.
      ii. Pratyush: Contributed ideas, helped refine the problem scope, and collaborated on the initial design specifications.
   b. **Algorithm Development:**
      i. Anvesha: Researched relevant algorithms, implemented the main algorithm in C++ with, and optimized for efficiency.
      ii. Pratyush: Assisted with algorithm selection, implemented support functions, and tested the code on sample data for validation.
   c. **Documentation and Presentation:**
      i. Anvesha: Drafted the technical report, organized sections on problem background, and algorithmic approach.
      ii. Pratyush: Edited and enhanced the document, prepared visual aids, and created the PowerPoint

presentation.

d. **Final Review and Testing:**
    i. Both Members: Conducted final reviews, debugged code, refined documentation, and prepared for presentation delivery.

4. PROBLEM STATEMENT

In today's logistics landscape, efficiency and cost-effectiveness are critical as companies strive to meet growing demand while managing expenses. Effective route optimization plays a key role in reducing delivery times, fuel consumption, and operational costs.

This project addresses these challenges by optimizing routes for delivery vans distributing packages from multiple warehouses to various demand points. Using advanced graph-based algorithms, we model the delivery network to identify the shortest, most efficient routes, factoring in delivery time constraints and strategic refueling stops to ensure timely and economical deliveries.

5. INTRODUCTION

a. **Motivation**

The main motivation to pick this problem was to reduce operational expenses in the logistics industry and improving customer satisfaction by applying algorithms which we have learnt and use them to solve real world problems. This project aims to develop an effective route optimization model that tackles these challenges, providing a cost-effective, scalable solution for complex delivery networks.

b. **Significance**

Efficient route optimization has become a critical factor in logistics and delivery services due to the rapid growth of e-commerce. By minimizing delivery costs and times, companies can save on fuel, reduce operational costs, and improve customer satisfaction with timely deliveries. This problem is crucial in cities and dense urban areas where delivery points are scattered and the demand for quick deliveries is high.

c. **Scope & Applications**

▪ Logistics companies managing fleets of delivery vans.

▪ Food delivery services optimizing routes for quick and efficient deliveries.

▪ E-commerce companies seeking to minimize delivery times and operational costs.

▪ Supply chain management systems aiming to minimize transportation costs while meeting time-sensitive delivery windows.

▪ Mainly used by cab, food, delivery booking applications and various such platforms.

6. SOLUTIONS

a. **Floyd Warshall Algorithm**

Given the starting point of any warehouse present in the graph to the vans according to the maximum number of houses present near that warehouse so that it will deliver all the packages in minimum cost and time. For this we will use Floyd Warshal's Algorithm which gives us the shortest distances for every pair of houses and warehouses.

The Floyd Warshall Algorithm is used to precompute the shortest paths between all pairs of nodes, including warehouses, houses, and gas stations. This precomputation is valuable because it allows us to quickly find the shortest path between any two nodes during route optimization.

Time Complexity: $O(V^3)$, where V is the number of nodes. Space Complexity: $O(V^2)$, where V is the number of nodes.
Feasibility: Suitable for dense graphs, where all-pairs shortest path computation is needed.

b. **Dijkstra’s Algorithm**
Shows the shortest time/distance path to the van owner as required from each warehouse to deliver the packages to each and every house present in the list. For this we will use the Dijkstra Algorithm which gives us the shortest distances from warehouses to different houses.

There is a fixed tank capacity for each van to travel a particular distance, So each van has to visit the gas station present in the graph to refill the tank. We will use Dijkstra’s Algorithm to get the path of nearest gas station.

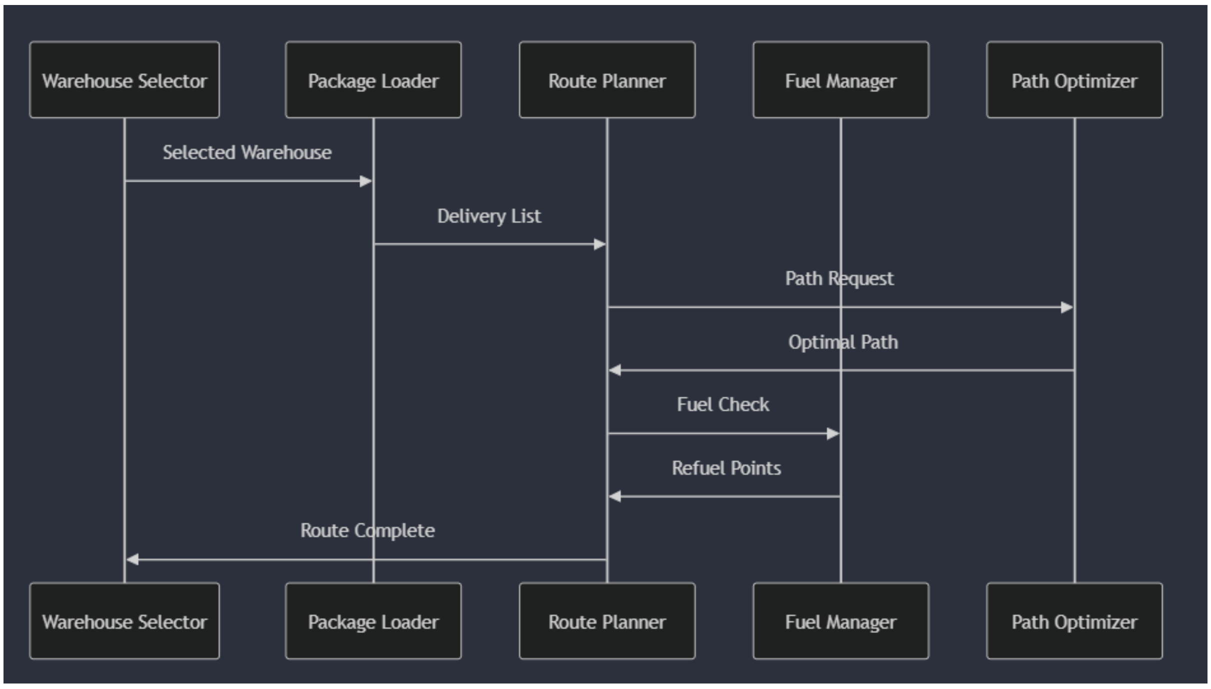
Time Complexity: $O(E\ log(V))$, where V is the number of nodes and E is the number of edges.
Space Complexity: $O(V)$, where V is the number of nodes.
Feasibility: More efficient for sparse graphs where targeted shortest paths queries are needed.

c. **Knapsack Algorithm**
To handle package selection, we use a knapsack algorithm in a greedy manner. This algorithm helps in selecting which packages to load into the delivery van based on the van's cargo capacity. Each package has a weight

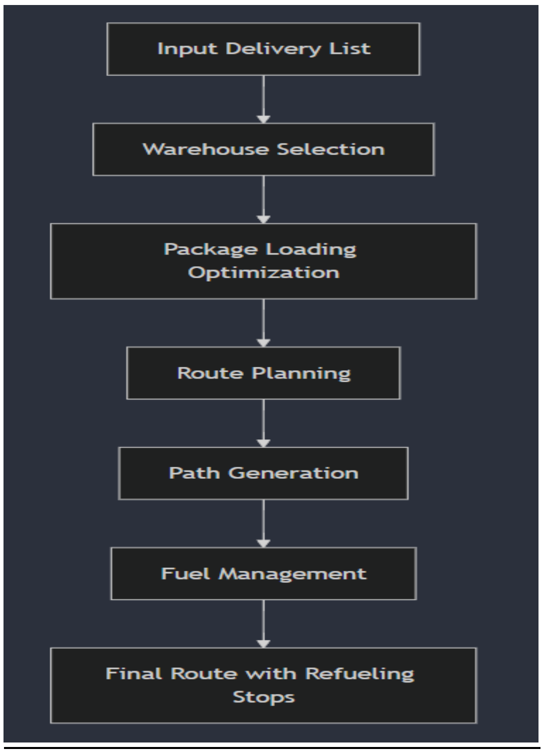
(representing the space it occupies), and the knapsack algorithm sorts these packages by weight and then selects them in order until the van's capacity is reached. The goal is to ensure that the van is fully loaded without exceeding its weight limit, maximizing the number of deliveries that can be made in a single trip.

Time Complexity: $O(N\ log(N))$, for sorting based on the weights.
Space Complexity: $O(N)$, to store various quantities of warehouses and nodes.
Feasibility: Ensures that the van's cargo capacity is used efficiently by selecting the optimal set of packages.

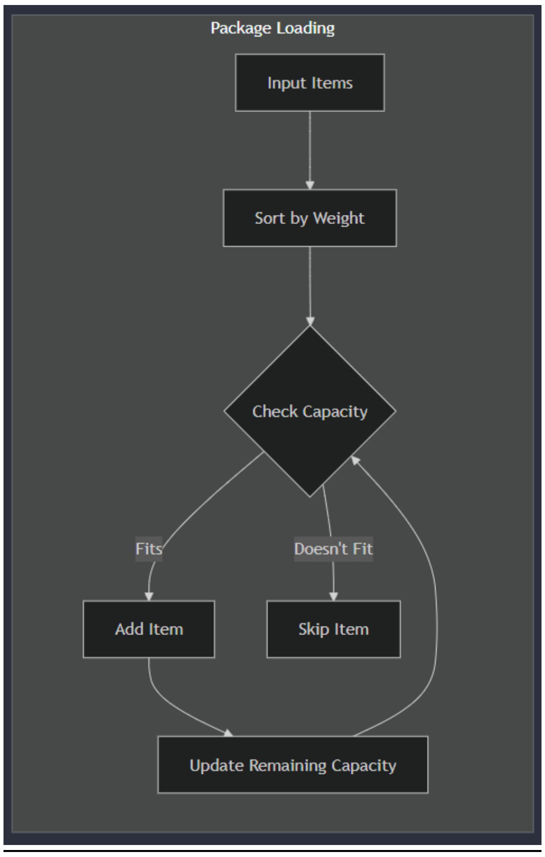
7. IMPLEMENTATION



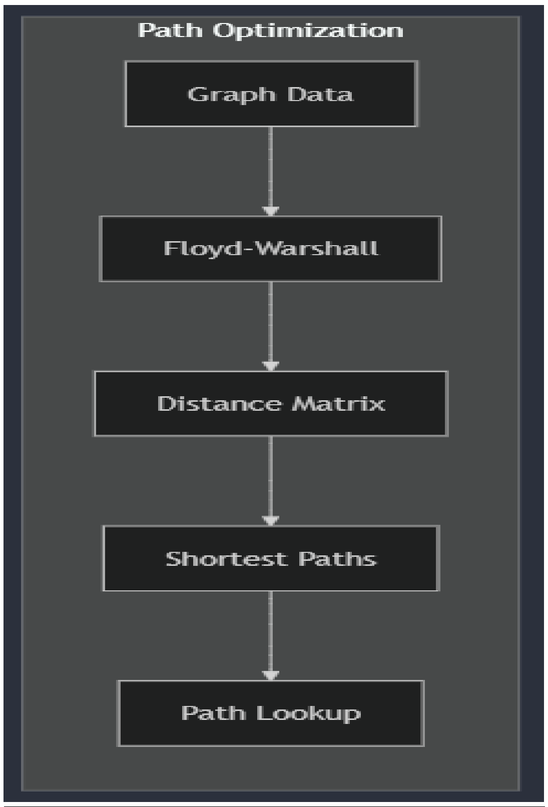
Overall Code flowchart:



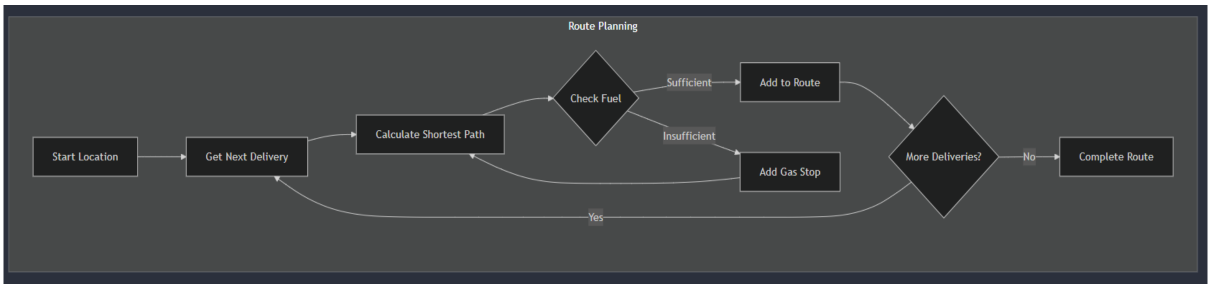
Knapsack workflow:



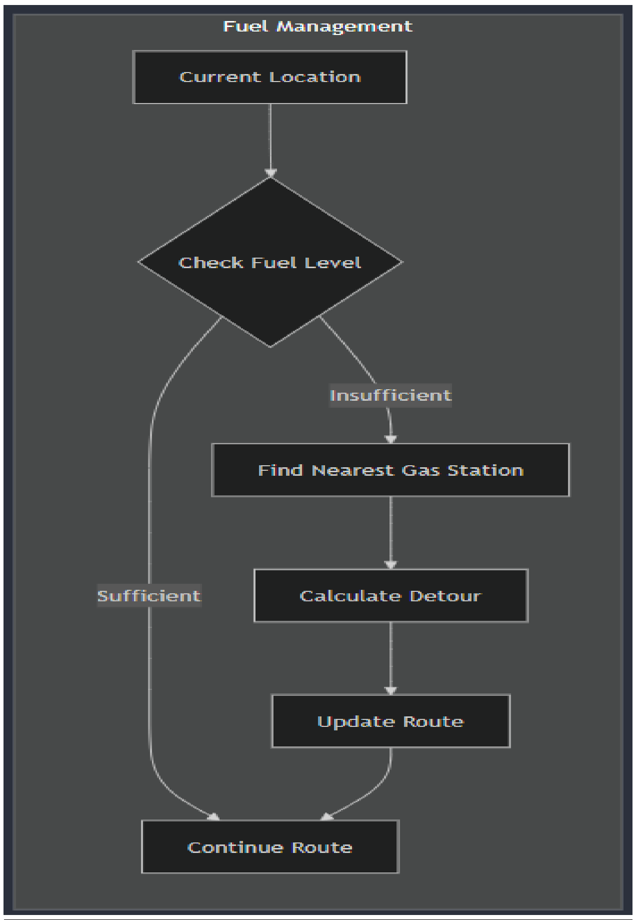
Path Optimization workflow:



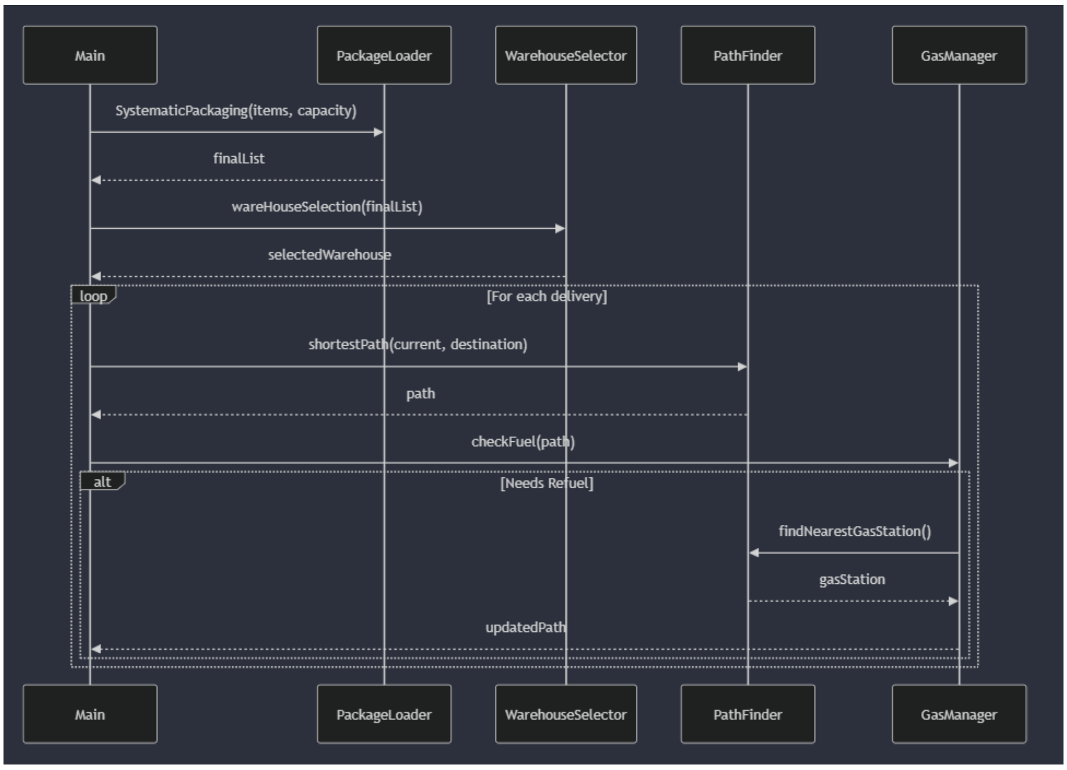
Route Planning workflow:



Fuel Management Workflow:

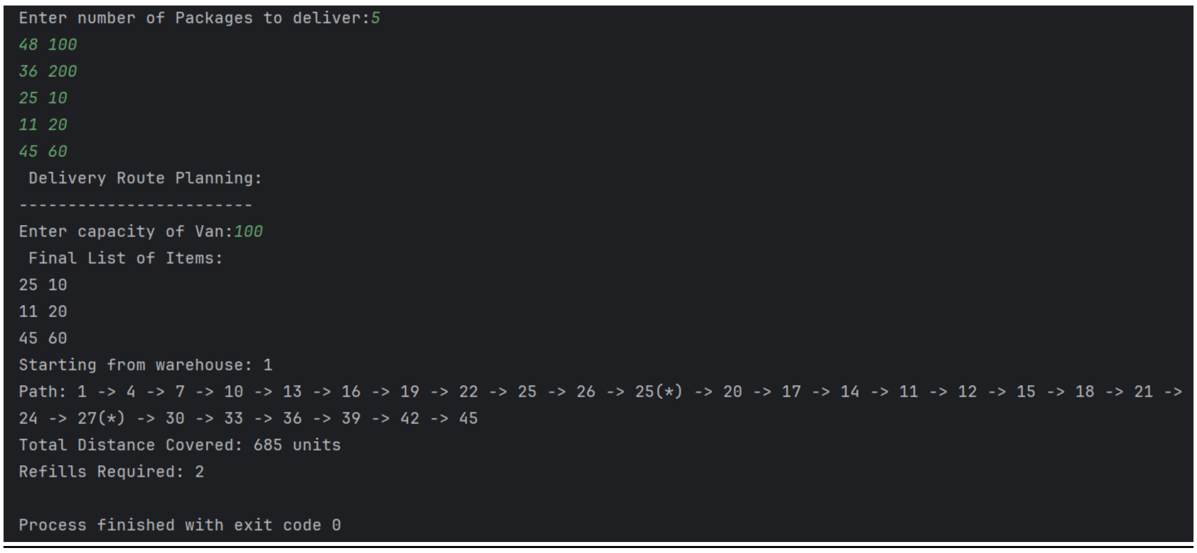


Code Flow:



8. TOY EXAMPLE

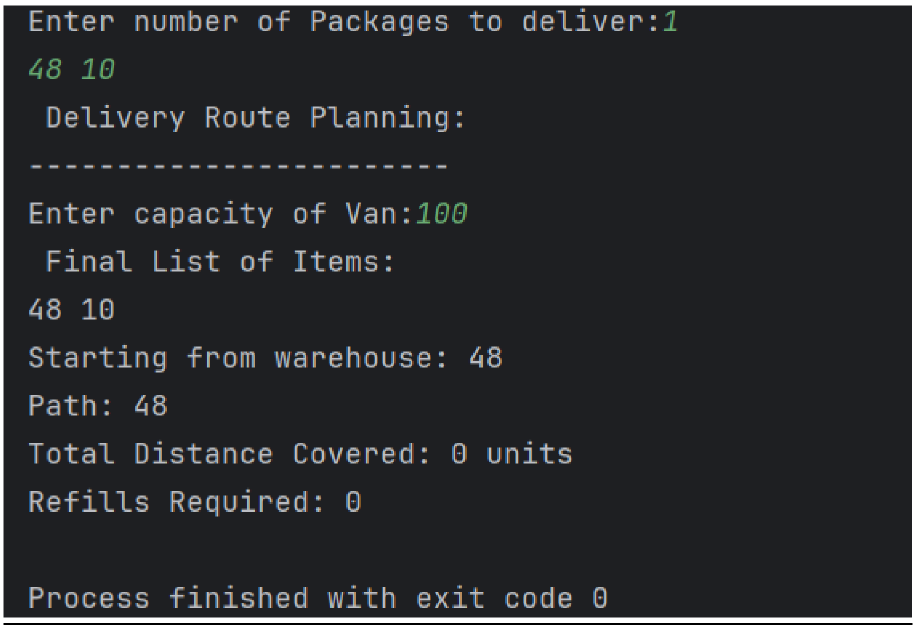
General Example:



```
Enter number of Packages to deliver:5
48 100
36 200
25 10
11 20
45 60
 Delivery Route Planning:
------------------------
Enter capacity of Van:100
 Final List of Items:
25 10
11 20
45 60
Starting from warehouse: 1
Path: 1 -> 4 -> 7 -> 10 -> 13 -> 16 -> 19 -> 22 -> 25 -> 26 -> 25(*) -> 20 -> 17 -> 14 -> 11 -> 12 -> 15 -> 18 -> 21 ->
24 -> 27(*) -> 30 -> 33 -> 36 -> 39 -> 42 -> 45
Total Distance Covered: 685 units
Refills Required: 2

Process finished with exit code 0
```

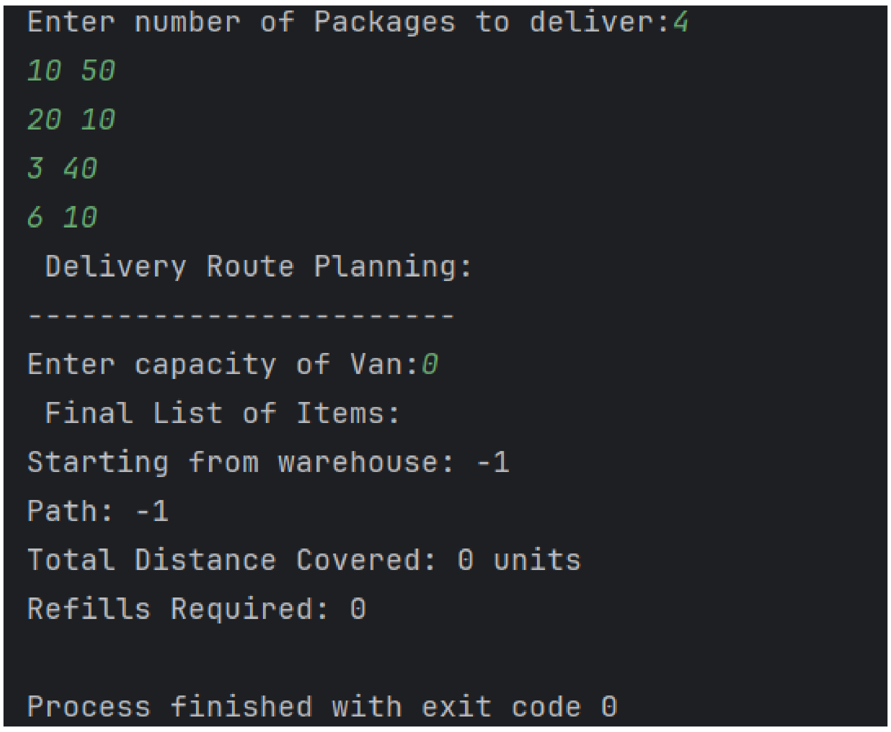
Item at warehouse:



```
Enter number of Packages to deliver:1
48 10
 Delivery Route Planning:
------------------------
Enter capacity of Van:100
 Final List of Items:
48 10
Starting from warehouse: 48
Path: 48
Total Distance Covered: 0 units
Refills Required: 0

Process finished with exit code 0
```

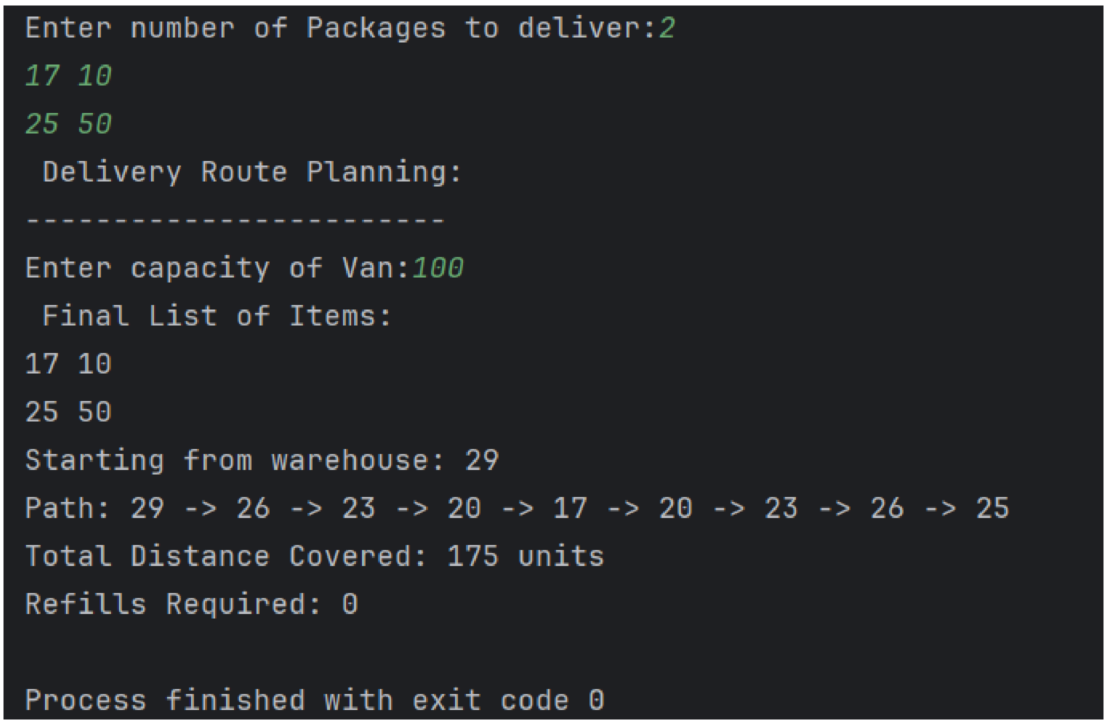
Van weight capacity = 0



```
Enter number of Packages to deliver:4
10 50
20 10
3 40
6 10
 Delivery Route Planning:
------------------------
Enter capacity of Van:0
 Final List of Items:
Starting from warehouse: -1
Path: -1
Total Distance Covered: 0 units
Refills Required: 0

Process finished with exit code 0
```

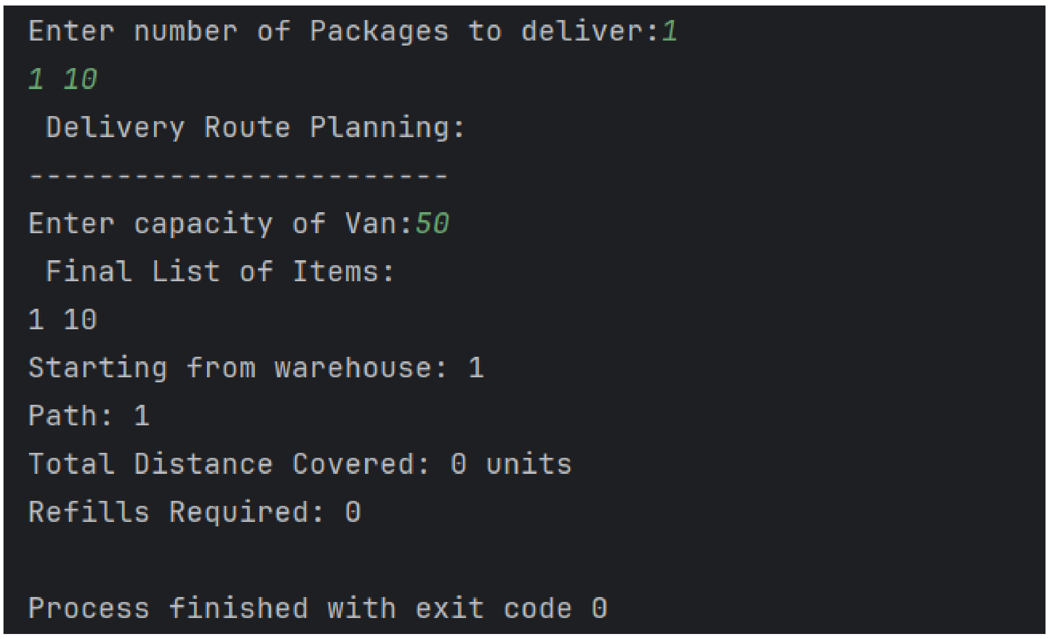
No refills required



```
Enter number of Packages to deliver:2
17 10
25 50
 Delivery Route Planning:
------------------------
Enter capacity of Van:100
 Final List of Items:
17 10
25 50
Starting from warehouse: 29
Path: 29 -> 26 -> 23 -> 20 -> 17 -> 20 -> 23 -> 26 -> 25
Total Distance Covered: 175 units
Refills Required: 0

Process finished with exit code 0
```

Source = Final Point


```
Enter number of Packages to deliver:1
1 10
 Delivery Route Planning:
------------------------
Enter capacity of Van:50
 Final List of Items:
1 10
Starting from warehouse: 1
Path: 1
Total Distance Covered: 0 units
Refills Required: 0

Process finished with exit code 0
```

9. SOURCE CODE & README

pratyush2905 (2024). *GitHub - pratyush2905/OptimizedDeliveryRouteCalculator*. [online] GitHub. Available at: Click Here!!

REFERENCES

- GeeksforGeeks (2018). Dijsktra's algorithm. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

- cp-algorithms.com. (n.d.). Floyd-Warshall Algorithm - Algorithms for Competitive Programming. [online] Available at: https://cp-algorithms.com/graph/all-pair-shortest-path-floyd-warshall.html.

- GeeksForGeeks (2012). 0-1 Knapsack Problem | DP-10. [online] GeeksforGeeks. Available at:

https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/.

- Wikipedia Contributors (2019). Greedy algorithm. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Greedy_algorithm.

********** END **********