AI-project (Group-4)

• Transport Problem •

Problem description:-

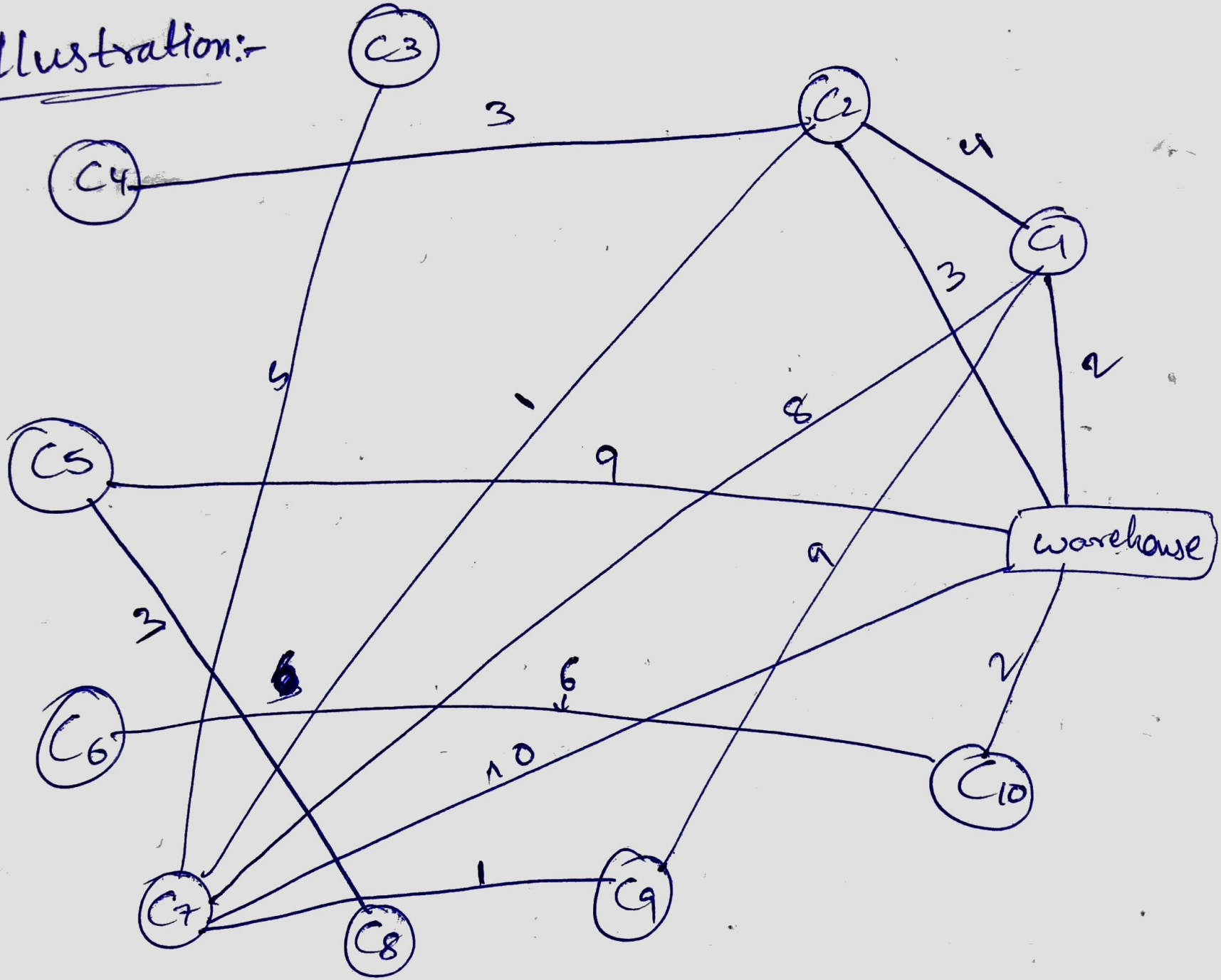
The transport problem models a delivery logistics scenario using a graph-based approach. It creates a network graph with a central 'warehouse' node and several consumer nodes. The edges between nodes represent possible delivery routes, with randomly assigned weights indicating distances. The code offers 2 methods for order delivery :- a Constraint Satisfaction problem (CSP) solver and a heuristic approach. The CSP solver ensures that delivery constraint, such as truck capacity are satisfied. The heuristic method, optimizes delivery routes based on immediate considerations. Overall, the code provides a versatile simulation for exploring

and solving delivery challenges in a graph
-based context.

(*) Algorithm Used:-

* we have used Dijkstra algorithm in this code, the aim is to find the optimal path from the current location to the target location in the graph.
* Returns both cost and path.

Illustration:-



*) Now lets explain this problem using dijkstra algorithm.

1) Starting point and Destination :-

The dijkstra's algorithm is applied to find the shortest path from the 'warehouse' (or) from consumer to another.

2) Nodes Explored :-

The nodes (or consumers) explored during the Dijkstra's algorithm execution are the nodes in the graph, including the 'warehouse' and the consumer nodes. The algorithm considers these nodes at various stages of the search.

Now Let us take these nodes as consumer nodes.

3) Number of possible Nodes:-

The number of nodes that the algorithm may explore during its search is equal to the total number of nodes in graph. (10 consumer nodes have been taken and 1 warehouse).

4) Shortest path from the consumer to another consumer & Priority for 1st consumer

The shortest path is found individually for each consumer in the order queue. The algorithm determines the path of the consumer who ordered first (priority) and with minimum cost, it travels to that consumer from warehouse (or) from the consumer who ordered first before the other consumer, ordered to get his item.

5) Total cost of the path:-

The total cost of the shortest path is

Calculated based on the sum of weights between consecutive nodes in the path.

* We have used a module named as "dijkstra_path", which implements dijkstra algorithm in the code, we have written it or included it in our code:-

* The steps or) algorithm for the above module is shown below:-

i) Initialization:-

⇒ Initialize a priority queue ('queue') with a tuple containing the cost (initialized to 0), the starting node and an empty path.

⇒ Create an empty set ('visited') to keep track of visited nodes.

ii) Main loops:-

⇒ Entering a while loop as long as the priority queue is not empty.

⇒ Each iteration, sort the queue based on the cost, and pop the node with the lowest cost.

⇒ Extract the cost, node, and path from the popped tuple.

iii) Node processing:-

⇒ If the current node has not been visited:-

- Add the node to the set of visited nodes.
- Append the node to the current path
- check if the current node is the destination ('end'). If so, return the cost & path.

iv) Neighbouring Nodes :-

⇒ Iterate over the neighbours of the current node & their corresponding weights.

⇒ For each neighbour, calculate the total cost to reach that neighbour

from the starting node.

⇒ Add a new tuple to the priority queue with the updated cost, the neighbor as the new node, and the updated path.

5) Termination:-

⇒ If the destination node is not reached after exploring all possible paths, return ∞ (infinity) for the cost and an empty path.

(*) Summary:- The dijkstra's algorithm systematically explores nodes in the graph, updating the cost & path at each step from warehouse (or) from consumer to consumer. It terminates when the destination node is reached or when all possible paths are explored. The final result is the shortest path and with prioritized (customer (or) consumer, calculates the cost of the path and prints it.