Amit Srivastav

RA1911003010633

Artificial Intelligence Lab

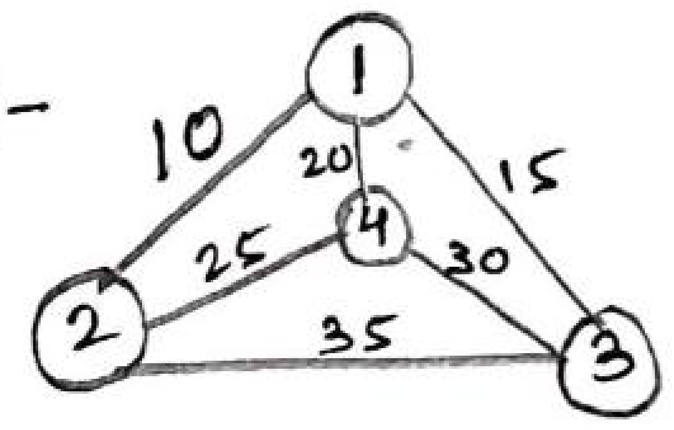
Lab-2

Aim: Developing agent programs for Real Word Problems - Travelling Salesman Problem (TSP).

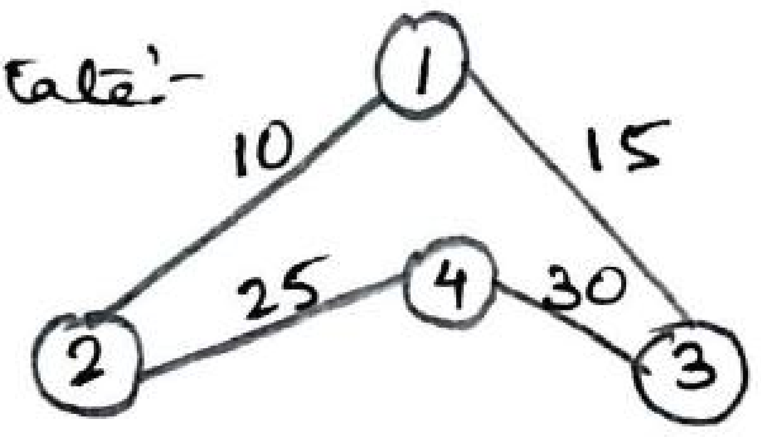
Problem Formulation:

For a given complete graph with n vertices and weight function defined on the edges, the objective is to construct a tour i.e, a circuit that passes through each vertex only once of minimum total weight.

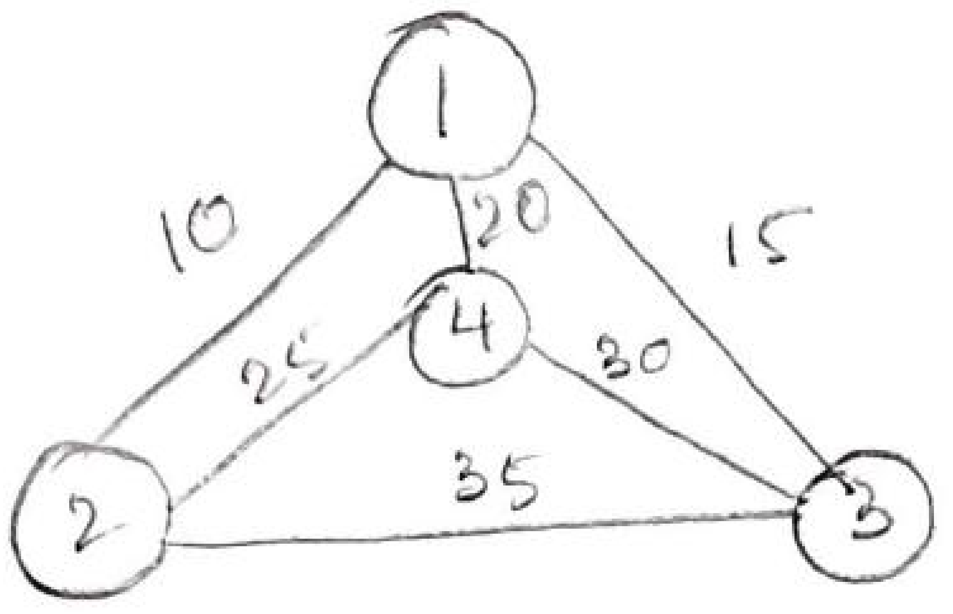
Initial State:-



Final state:-

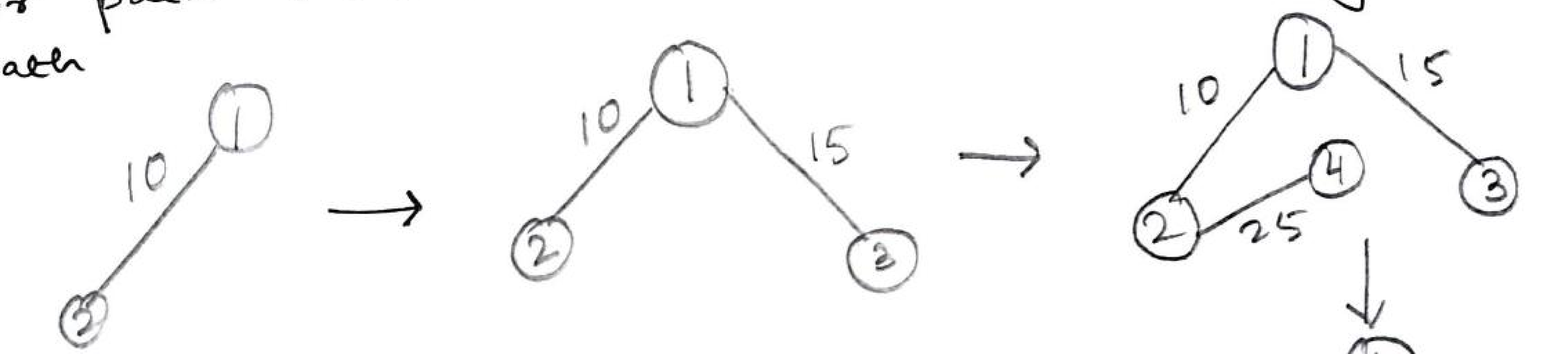


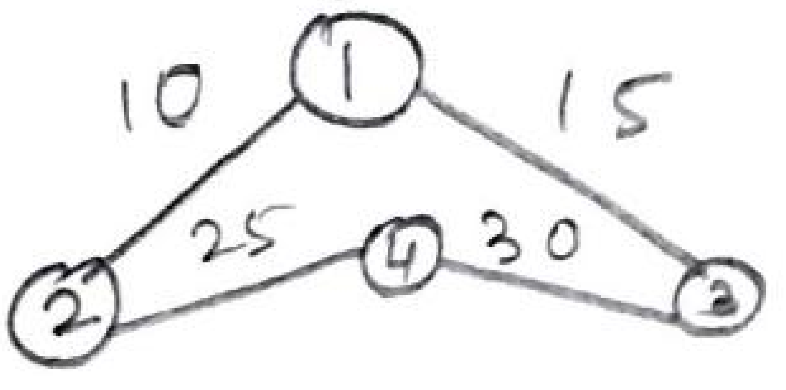
Problem solving:



We start at vertex 1 and find the minimum cost path with 1 as starting point, i as ending point and all vertices appearing exactly once.

For path 1 → 2 the minimum cost would be through direct path





It tries for various other permutations as well and 1-2-4-3-1 permutation works perfect as it provides minimum cost.

AMIT SRIVASTAV

RA1911003010633

ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO: 2

DEVELOPING AGENT PROGRAMS FOR REAL WORLD PROBLEMS (TRAVELLING SALESMAN PROBLEM)

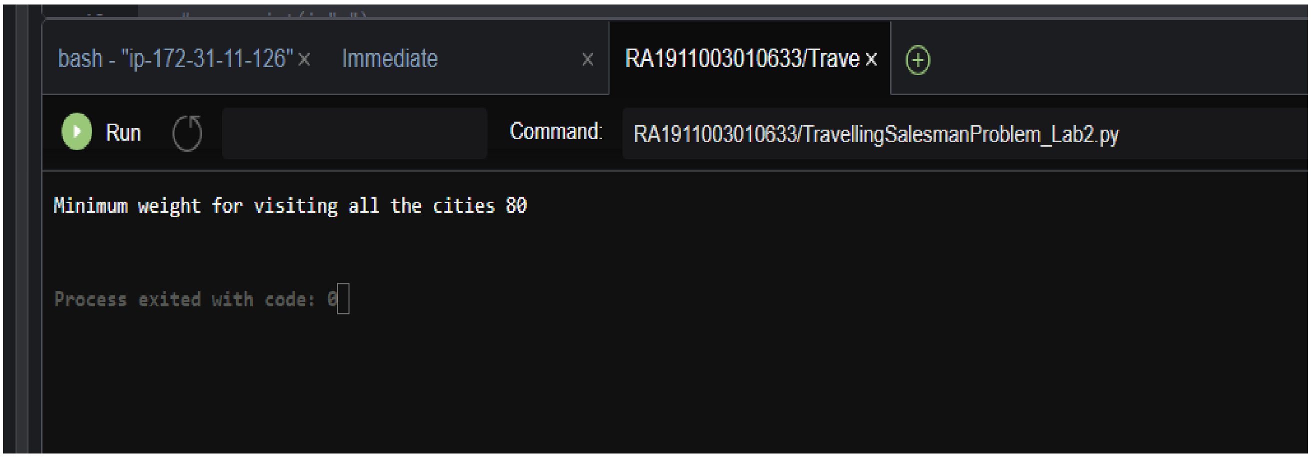
Algorithm:

Step 1: Consider city 1 as the starting and ending point.
Step 2: Generate all (n-1)! Permutations of cities.
Step 3: Calculate cost of every permutation and keep track of minimum cost permutation.
Step 4: Return the permutation with minimum cost.

Source code:

```
from sys import maxsize
from itertools import permutations
V = 4

# implementation of traveling Salesman Problem
def travellingSalesmanProblem(graph, s):

    # store all vertex apart from source vertex
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)
    # store minimum weight Hamiltonian Cycle
    min_path = maxsize
    next_permutation=permutations(vertex)
    # for i in next_permutation:
    #     print(i," ")
    for i in next_permutation:

        # store current Path weight(cost)
        current_pathweight = 0

        # compute current path weight
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]

        # update minimum
        min_path = min(min_path, current_pathweight)

    return min_path


# Driver Code
if __name__ == "__main__":

    # matrix representation of graph
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
            [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print("Minimum weight for visiting all the cities",
travellingSalesmanProblem(graph, s))
```

Output:



```
RA1911003010633/TravellingSalesmanProblem_Lab2.py
```

```
Minimum weight for visiting all the cities 80


Process exited with code: 0
```

Result:

Hence, the implementation of Travelling Salesman Problem is done successfully.