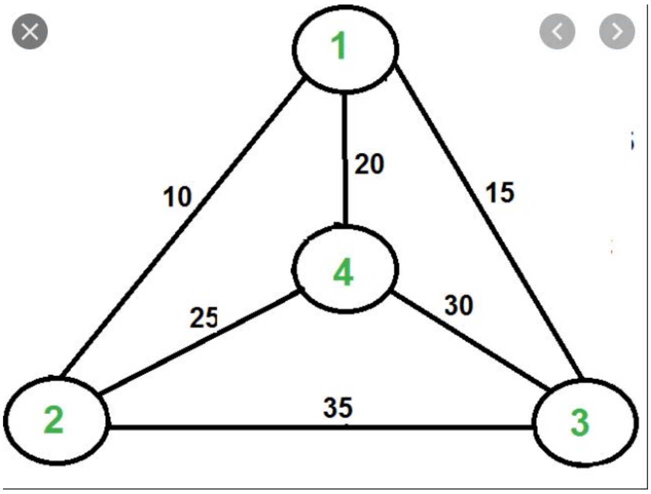
Experiment no: 2
Date: 28/01/2020

IMPLEMENTATION OF REAL WORLD PROBLEM

AIM:

Implementation of Real-World Problem using Travelling Salesman Problem.

METHODOLOGY:

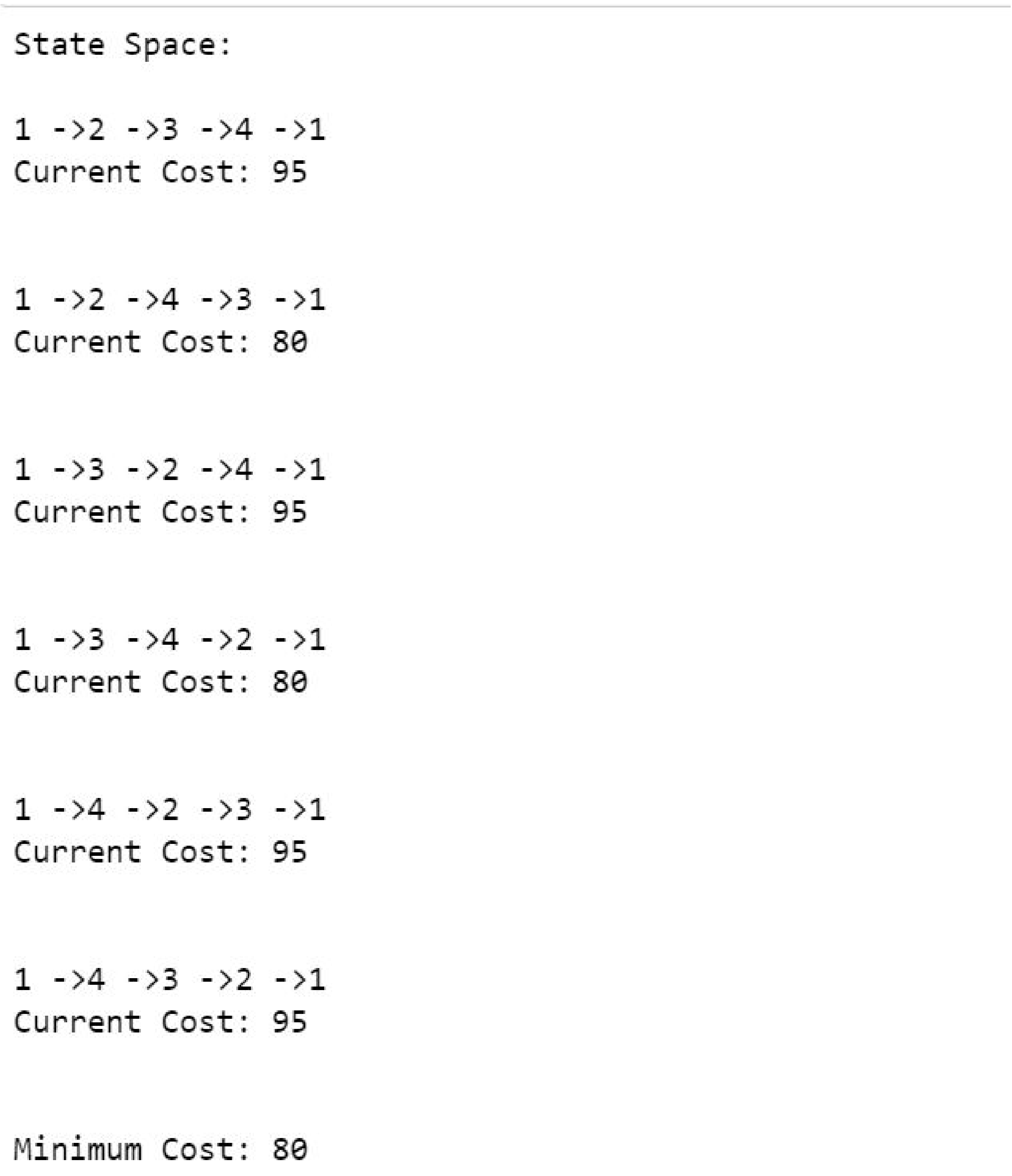


- ➢ The Weight Matrix of the Graph is stored in a list.
- ➢ We will consider City 1 as Source City.
- ➢ Permutation of all other cities will be generated.
- ➢ Cost of each route permutation will be calculated.
- ➢ The Minimum cost among all the routes will be considered for output.

CODE:

```
from sys import maxsize
from itertools import permutations

V = 4


# implementation of traveling Salesman Problem
def travellingSalesmanProblem(graph, s):
    print("State Space:\n")
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)

    min_path = maxsize
    next_permutation = permutations(vertex)
    for i in next_permutation:

        current_pathweight = 0

        # For Display
        print(s + 1, "->", end="")
        # compute current path weight
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            print(j + 1, "->", end="")
            k = j
        print(s + 1)
        current_pathweight += graph[k][s]
        print("Current Cost:", current_pathweight)
        print("\n")

        min_path = min(min_path, current_pathweight)

    return min_path

if __name__ == "__main__":
#if _name_ == "_main_":
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
             [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print("Minimum Cost:", travellingSalesmanProblem(graph, s))
```

OUTPUT:



```
State Space:

1 ->2 ->3 ->4 ->1
Current Cost: 95


1 ->2 ->4 ->3 ->1
Current Cost: 80


1 ->3 ->2 ->4 ->1
Current Cost: 95


1 ->3 ->4 ->2 ->1
Current Cost: 80


1 ->4 ->2 ->3 ->1
Current Cost: 95


1 ->4 ->3 ->2 ->1
Current Cost: 95


Minimum Cost: 80
```

RESULT:
Travelling Salesman Problem is executed successfully.