LAB 2 – TRAVELLING SALESMAN PROBLEM

AI LAB

SHUBHAM SHARMA

RA1911003010649

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.

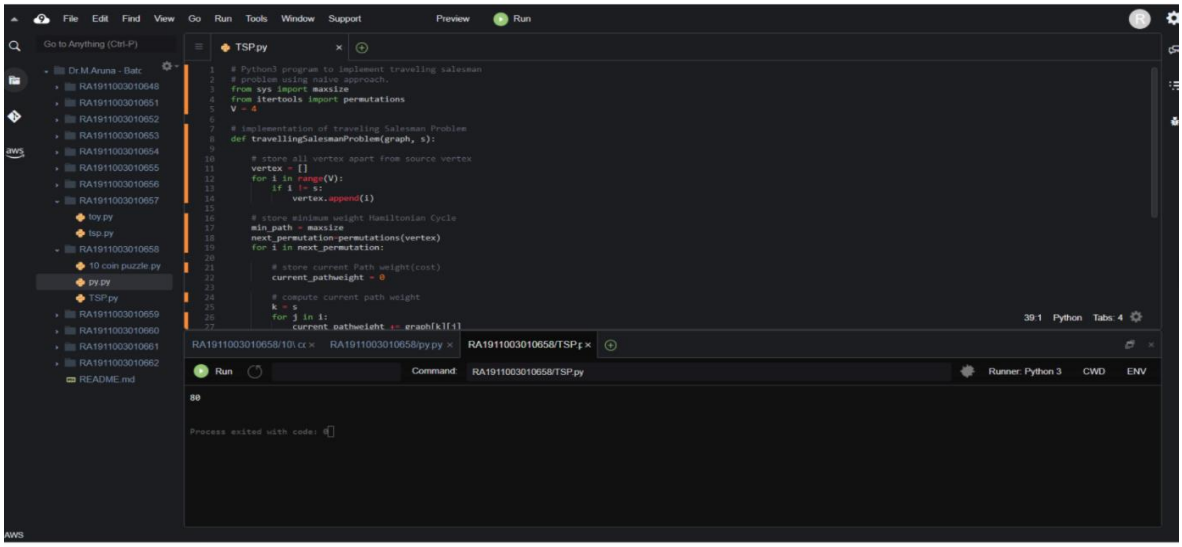
Step 4: Return the permutation with minimum cost.

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:

# 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: ",travellingSalesmanProblem(graph, s))
```

OUTPUT:



RESULT: Hence, the implementation of Travelling Salesman Person was successfully done.

| Shubham Sharma
RA1911003010649 |

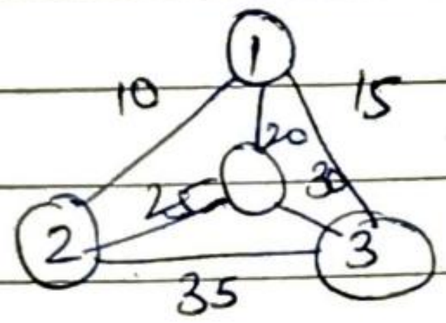
DATE: / /20
PAGE No

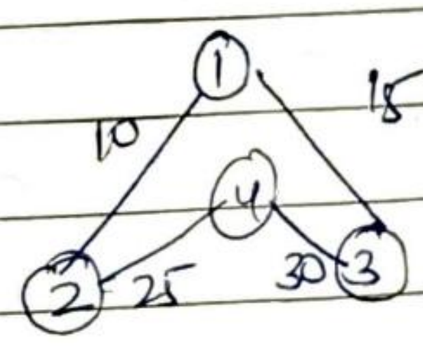
AI Lab-2

Aim – To develop agent programs for real world problem ie Travelling Salesman Problem.

Problem formulation →

for a given complete graph with n vertices & weight fn defined on the edges, the objective is to construct a tour, ie a circuit that passes through each vertex only once and return back to the starting with minimum total weight.





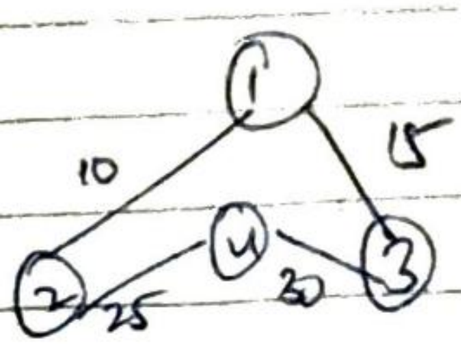
Initial state final state

Problem Solving :→

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

Now first we find a path for the same condition and try various permutations to find the min path out of various path. For this example, it is

1-2-4-3)



Min. weight = 80