

Aman Kalla

RA1911003010640

Artificial Intelligence Lab

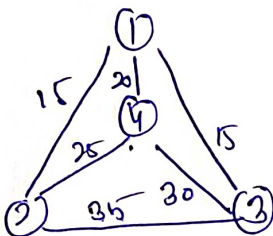
Lab-2

Aim:- Developing Agent programs for Real World Problems -
Travelling Salesman Problem (TSP)

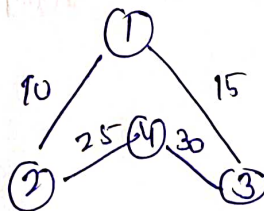
Problem Formulation

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 maximum total weight.

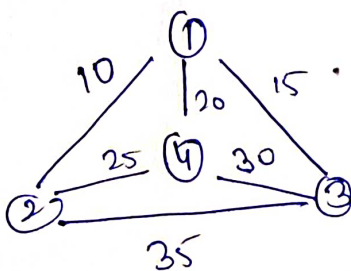
Initial State:-



Final state:-

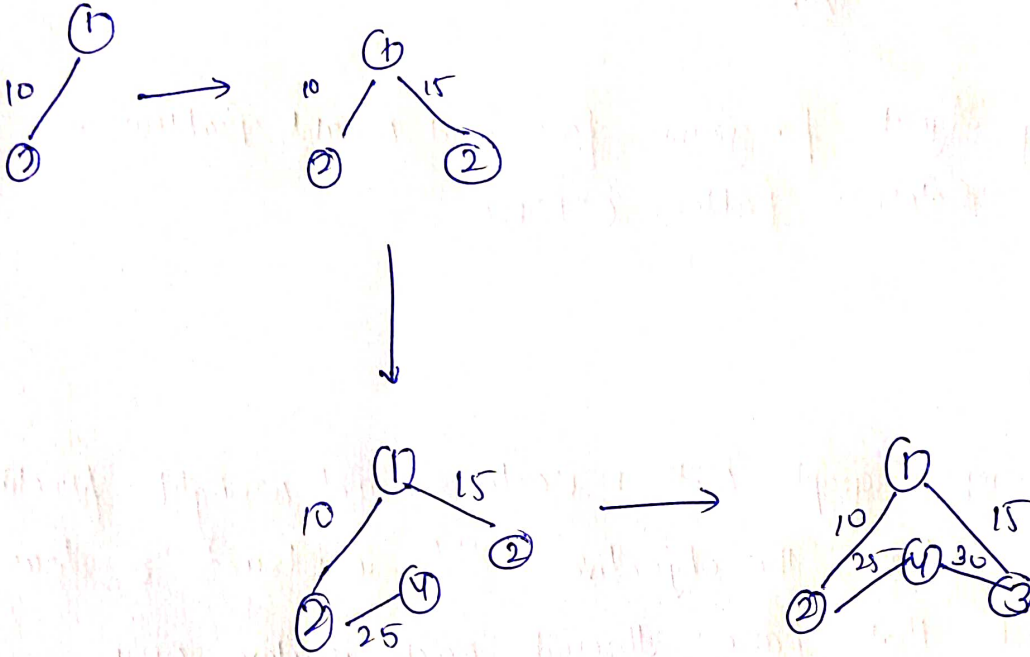


Problem Solving.



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

For path $1 \rightarrow 2$, 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.

ARTIFICIAL INTELLIGENCE

LAB 2

Travelling Salesman Problem

Aman Kalla

RA1911003010640

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
```

```
def travellingSalesmanProblem(graph, s):
```

```
    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

    k = s
    for j in i:
        current_pathweight += graph[k][j]
        k = j
    current_pathweight += graph[k][s]

    min_path = min(min_path, current_pathweight)

return min_path

```

```

if __name__ == "__main__":

```

```

    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
              [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print(travellingSalesmanProblem(graph, s))

```

Output:-

The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'RA1911003010638' with several subfolders and files. The code editor has two tabs: 'AI1.py' and 'TSP-AI2.py'. The 'TSP-AI2.py' tab is active, showing a Python script for solving the Travelling Salesman Problem using brute force. The script imports 'maxsize' from 'sys' and 'permutations' from 'itertools'. It defines a function 'travellingSalesmanProblem(graph, s)' that initializes a 'vertex' list, generates all permutations of vertices, and iterates through them to find the minimum path weight. The terminal at the bottom shows the command 'python RA1911003010640/TSP-AI2.py' being executed, with the output '80' and the message 'Process exited with code: 0'.

```
1 from sys import maxsize
2 from itertools import permutations
3 V = 4
4
5 def travellingSalesmanProblem(graph, s):
6
7     vertex = []
8
9     for i in range(V):
10         if i != s:
11             vertex.append(i)
12
13     min_path = maxsize
14     next_permutation=permutations(vertex)
15     for i in next_permutation:
16
17         current_pathweight = 0
18
19         k = s
20         for j in i:
21             current_pathweight += graph[k][j]
22             k = j
23         current_pathweight += graph[k][s]
24
25     min_path = min(min_path, current_pathweight)
26
27     return min_path
28
29
30
31
32
33
34
35
```

42/47 Python Spaces: 4

bash - "ip-172-31-11-128" x Immediate x RA1911003010640/AI1.py x RA1911003010640/TSP-AI2.py x

Run Command: RA1911003010640/TSP-AI2.py Runner: Python 3 CWD ENV

80

Process exited with code: 0

Result:-

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