EXPERIMENT 9

Implementing Travelling Salesman Problem

AIM:

To implement Travelling Salesman Problem

ALGORITHM

1. Import necessary libraries:

   - Import `maxsize` from `sys` to use as an initial value for the minimum path.

   - Import `permutations` from `itertools` to generate all possible permutations of the vertices.

2. Initialize graph and variables:

   - Define the number of vertices (`V`) and the graph representing the distances between them.

   - Set the starting vertex (`s`) to 0.

3. Define the traveling salesman function:

   - Create a list of vertices excluding the starting vertex (`s`).

   - Initialize variables for the minimum path (`min_path`) and the best tour (`best_tour`).

   - Generate all permutations of the remaining vertices.

4. Iterate through permutations:

   - For each permutation, calculate the total weight of the path.

   - Update the minimum path and the corresponding tour if the current path is shorter.

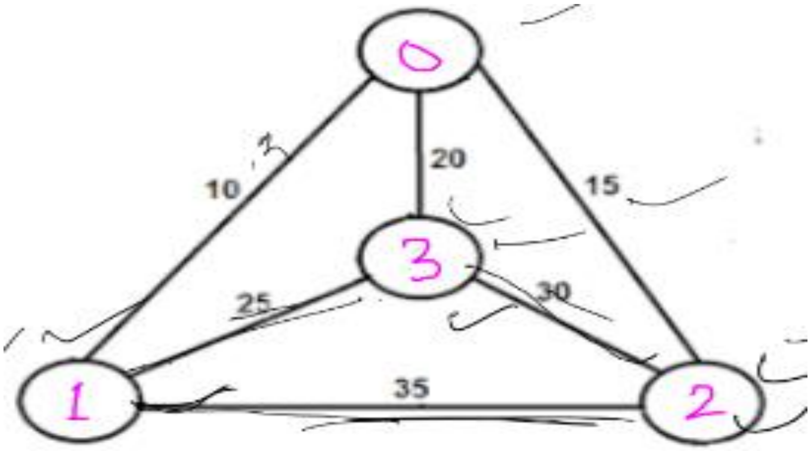
5. Return the result:

   - Return the minimum path and the best tour.

6. Invoke the function and print the result:

   - Call the `travellingSalesmanProblem` function with the given graph and starting vertex.

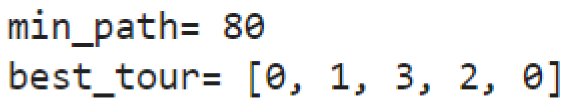
   - Print the minimum path and the best tour.



PROGRAM

```
from sys import maxsize

from itertools import permutations

V = 4

graph = [[0, 10, 15, 20], [10, 0, 35, 25],

            [15, 35, 0, 30], [20, 25, 30, 0]]

s = 0

def travellingSalesmanProblem(graph, s):

    vertex = []

    for i in range(V):


        if i != s:

            #print (i)

            vertex.append(i)

            #print(vertex)


    min_path = maxsize

    best_tour= None
    next_permutation=permutations(vertex)

    for i in next_permutation:
        #print(i)

        current_pathweight = 0
        k = s

        for j in i:

            current_pathweight += graph[k][j]
            #print(current_pathweight)
            k = j

        current_pathweight += graph[k][s]
        #print("weight=",current_pathweight)
        if current_pathweight < min_path:
            min_path = min(min_path, current_pathweight)

            best_tour = [s] + list(i) + [s]

    return min_path,best_tour
min_path,best_tour=(travellingSalesmanProblem(graph, s))
print('min_path=',min_path)
print('best_tour=',best_tour)
```

OUTPUT



```
min_path= 80
best_tour= [0, 1, 3, 2, 0]
```

RESULT:

Travelling Salesman problem is implemented and the output is verified.