# Experiment Number 8

| | | | |
|---|---|---|---|
| Name :: | Rishabh Anand | UID :: | 19BCS4525 |
| Branch :: | CSE - IoT | Sec/Grp :: | 1/A |
| Semester :: | 5th | Date :: | 14th Nov, 2021 |
| Subject :: | Adv Programming Lab | CODE :: | CSP-347 |

## 1. Aim :

Implement Travelling Salesperson problem using Dynamic programming.

## 2. Task :

1. Implement Travelling Salesperson problem using Dynamic programming.

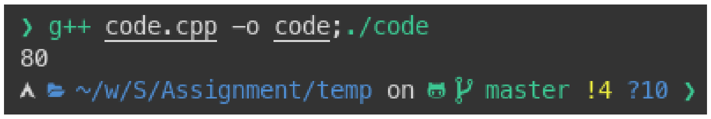
## 3. Algorithm :

- Consider city 1 as the starting and ending point. Since the route is cyclic, we can consider any point as a starting point.
- Generate all (n-1)! permutations of cities.
- Calculate the cost of every permutation and keep track of the minimum cost permutation.
- Return the permutation with minimum cost.

## 4. Source Code :

```
#include <bits/stdc++.h>

using namespace std;

#define V 4

int travllingSalesmanProblem(int graph[][V], int s)
{
    vector<int> vertex;
    for (int i = 0; i < V; i++)
        if (i != s)
            vertex.push_back(i);
    int min_path = INT_MAX;
    do
    {
        int current_pathweight = 0;
        int k = s;
        for (int i = 0; i < vertex.size(); i++)
        {
            current_pathweight += graph[k][vertex[i]];
            k = vertex[i];
        }
        current_pathweight += graph[k][s];
        min_path = min(min_path, current_pathweight);
    } while (
        next_permutation(vertex.begin(), vertex.end()));
    return min_path;
}

int main()
{
    int graph[][V] = {{0, 10, 15, 20},
                      {10, 0, 35, 25},
                      {15, 35, 0, 30},
                      {20, 25, 30, 0}};
    int s = 0;
    cout << travllingSalesmanProblem(graph, s) << endl;
    return 0;
}
```

## 5. Observations :



```
❯ g++ code.cpp -o code;./code
80
~/w/S/Assignment/temp on  master !4 ?10 ❯
```

## Learning Outcomes :

- Travelling salesperson problem and its real world application.
- Learnt concept of Dynamic Programming and its wide application in the field of software.
- Landing to a optimized solution over brute force approach to get most out of the algorithm, its robustness and scalability.

| S. No. | Parameters | Marks Obtained | Maximum Marks |
|---|---|---|---|
| 1. | | | |
| 2. | | | |
| 3. | | | |
| | | | |