



## Department of Computer Science and Engineering (Data Science)

NAME: ALISTAIR SALDANHA  
SAPID: 60009200024  
BATCH: K1

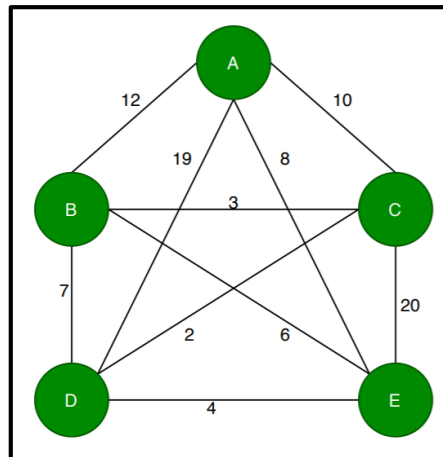
### Experiment 7 (Dynamic Programming)

**Aim:** Implementation of Travelling salesperson problem using dynamic programming.

#### Theory:

In the TSP, given a set of cities and the distance between each pair of cities, a salesman needs to choose the shortest path to visit every city exactly once and return to the city from where he started.

Let's take an example to understand the TSP in more detail:



Here, the nodes represent cities, and the values in each edge denote the distance between one city to another. Let's assume a salesman starts the journey from the city A. According to TSP, the salesman needs to travel all the cities exactly once and get back to the city A by choosing the shortest path. Here the shortest path means the sum of the distance between each city travelled by the salesman, and it should be less than any other path.

With only 5 cities, the problem already looks quite complex. As the graph in the example is a complete graph, from each city, the salesman can reach any other cities in the graph. From each city, the salesman needs to choose a route so that he doesn't have to revisit any city, and the total distance travelled should be minimum.

#### Algorithm:

Step 1: Let  $d[i, j]$  indicates the distance between cities  $i$  and  $j$ . Function  $C[x, V - \{x\}]$  is the cost of the path starting from city  $x$ .  $V$  is the set of cities/vertices in given graph. The aim of TSP is to minimize the cost function.

Step 2: Assume that graph contains  $n$  vertices  $V_1, V_2, \dots, V_n$ . TSP finds a path covering all vertices exactly once, and the same time it tries to minimize the overall traveling distance.



## Department of Computer Science and Engineering (Data Science)

Step 3: Mathematical formula to find minimum distance is stated below:

$$C(i, V) = \min \{ d[i, j] + C(j, V - \{ j \}) \}, j \in V \text{ and } i \notin V.$$

TSP problem possesses the principle of optimality, i.e. for  $d[V1, Vn]$  to be minimum, any intermediate path  $(Vi, Vj)$  must be minimum.

### Pseudocode:

#### Algorithm 1: Dynamic Approach for TSP

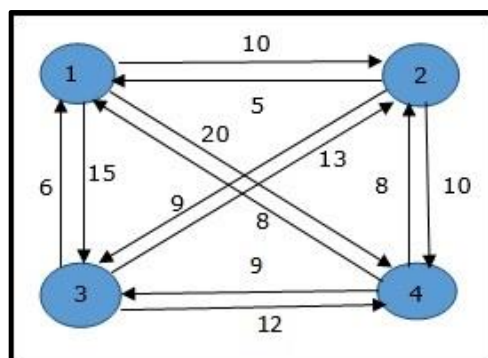
```
Data:  $s$ : starting point;  $N$ : a subset of input cities;  $dist()$ :  
distance among the cities  
Result:  $Cost$ : TSP result  
 $Visited[N] = 0$ ;  
 $Cost = 0$ ;  
Procedure  $TSP(N, s)$   
   $Visited[s] = 1$ ;  
  if  $|N| = 2$  and  $k \neq s$  then  
     $Cost(N, k) = dist(s, k)$ ;  
    Return  $Cost$ ;  
  else  
    for  $j \in N$  do  
      for  $i \in N$  and  $visited[i] = 0$  do  
        if  $j \neq i$  and  $j \neq s$  then  
           $Cost(N, j) = \min ( TSP(N - \{i\}, j) + dist(j, i) )$   
           $Visited[j] = 1$ ;  
        end  
      end  
    end  
  end  
  Return  $Cost$ ;  
end
```

### Complexity:

Time Complexity:  $O(n^2 * 2^n)$

### Lab Assignment to Complete:

A traveler needs to visit all the cities from a list, where distances between all the cities are known and each city should be visited just once. What is the shortest possible route that he visits each city exactly once and returns to the origin city?





## Department of Computer Science and Engineering (Data Science)

Code:

```
#include<stdio.h>
#define I 32767
#define MAX 10

int dist[MAX][MAX],visited[MAX],n,cost=0;
int least(int c)
{
    int i,nc=I;
    int min=I,kmin;
    for(i=0;i<n;i++)
    {
        if((dist[c][i] != 0) && (visited[i] == 0))
            if(dist[c][i] < min)
            {
                min=dist[i][0]+dist[c][i];
                kmin=dist[c][i];
                nc=i;
            }
    }
    if(min!=I)
        cost+=kmin;
    return nc;
}

void mincost(int city)
{
    int i,ncity;
    visited[city]=1;
    printf("%d -->",city+1);
    ncity=least(city);
    if(ncity==I)
    {
        ncity=0;
        printf("%d",ncity+1);
        cost+=dist[city][ncity];
        return;
    }
    mincost(ncity);
}

void main()
{
    int i,j;
```



## Department of Computer Science and Engineering (Data Science)

```
printf("Enter No. of Cities: ");
scanf("%d",&n);
printf("\nEnter Cost Matrix\n");
for(i=0;i < n;i++)
{
    printf("\nEnter Elements of Row %d:\n",i+1);
    for( j=0;j < n;j++)
        scanf("%d",&dist[i][j]);
    visited[i]=0;
}
printf("\nThe cost matrix is:\n");
for( i=0;i < n;i++)
{
    printf("\n\n");
    for(j=0;j < n;j++)
        printf("%d\t",dist[i][j]);
}
printf("\n\nThe Path is:\n\n");
mincost(0);
printf("\n\nMinimum cost:");
printf("%d",cost);
}
```

Output:

```
Enter No. of Cities: 4

Enter Cost Matrix

Enter Elements of Row 1:
0 10 15 20

Enter Elements of Row 2:
5 0 9 10

Enter Elements of Row 3:
6 13 0 12

0      10      15      20
5      0       9       10
6      13      0       12
8      8       9       0

The Path is:

1 -->2 -->4 -->3 -->1

Minimum cost:35
```