Practical 6

DAA

Name: Madhura Mahatme

Roll no.: 36

Section: A1-B3

GIT LINK:

https://github.com/mahatmemadhura4-bot/TSP-and-Bellman-Ford-Problem-DAA-Pract6

Aim-1:

A traveling salesman is getting ready for a big sales tour. Starting at his hometown, suitcase in hand, he will conduct a journey in which each of his target cities is visited exactly once before he returns home. Given the pairwise distances between cities, what is the best order in which to visit them, so as to minimize the overall distance traveled?

Input:

Ver 1 2 3 4

1 0 10 15 20

2 5 0 9 10

3 6 13 0 12

4 8 8 9 0

Consider different starting vertex: 1, 2, 3, 4

Sample Output:

Path: 1→2→4→3→1, Cost of travelling is: 35

Code:

```
#include <stdio.h>
#include <stdlib.h>
#define INF 1000000

int minCost;
int *bestPath;
void tsp(int **dist, int *path, int *visited, int N, int pos, int count, int cost) {
   if (count==N &&dist[pos][path[0]] >0) { // complete tour
      cost += dist[pos][path[0]];
      if (cost<minCost) {
         minCost = cost;
         for (int i= 0; i<= N; i++)
            bestPath[i] = path[i];
      }
      return;
   }

   for (int i= 0; i< N; i++) {
      if (!visited[i] && dist[pos][i]> 0) {
         visited[i] = 1;
         path[count] = i;
         tsp(dist, path, visited, N, i, count+ 1, cost+ dist[pos][i]);
         visited[i]= 0;
      }
```

```
    }
}
int main() {
    int N;
    printf("Enter number of cities: ");
    scanf("%d", &N);

    int **dist = (int **)malloc(N * sizeof(int *));
    for (int i = 0; i < N; i++)
        dist[i] = (int *)malloc(N * sizeof(int));

    printf("Enter distance matrix:\n");
    for (int i= 0; i< N;i++)
        for (int j= 0; j< N; j++)
            scanf("%d", &dist[i][j]);

    bestPath=(int *)malloc((N + 1) * sizeof(int));

    for (int start =0; start <N; start++) {
        minCost =INF;
        int *path =(int *)malloc((N + 1) * sizeof(int));
        int *visited =(int *)calloc(N, sizeof(int));

        path[0] =start;
        visited[start] =1;

        tsp(dist, path, visited, N, start, 1, 0);

        printf("Starting at city %d:\n", start + 1);
```

```
        printf("Path: ");
        for (int i =0; i <=N; i++)
            printf("%d%s", bestPath[i] + 1, (i< N) ? " -> " : "\n");
        printf("Cost of travelling is: %d\n\n", minCost);

        free(path);
        free(visited);
    }
    for (int i= 0; i< N; i++)
        free(dist[i]);
    free(dist);
    free(bestPath);
    return 0;
}
```

Output:

```
PS C:\Users\Madhura\OneDrive\Desktop\C> cd "c:\U
le } ; if ($?) { .\tempCodeRunnerFile }
Enter number of cities: 4
Enter distance matrix:
0 12 13 15
23 0 45 16
34 23 0 42
23 12 10 0
Starting at city 1:
Path: 1 -> 4 -> 3 -> 2 -> 724183337
Cost of travelling is: 71

Starting at city 2:
Path: 2 -> 1 -> 4 -> 3 -> 724183337
Cost of travelling is: 71

Starting at city 3:
Path: 3 -> 2 -> 1 -> 4 -> 724183337
Cost of travelling is: 71

Starting at city 4:
Path: 4 -> 3 -> 2 -> 1 -> 724183337
Cost of travelling is: 71

PS C:\Users\Madhura\OneDrive\Desktop\C>
```

Aim-2:

Construction of Single Source Shortest Path

Problem Statement:

Develop a system to optimize the delivery routes for a fleet of vehicles in a metropolitan area. The system should efficiently calculate the shortest paths between multiple pickup and delivery points, taking into account traffic congestion and road conditions.

Implement the Bellman-Ford algorithm to find the shortest path from a central depot to each delivery location while considering varying transportation costs and time constraints.

Consider the following criteria for determining connections within the same state in India:

i. Determine the latitude and longitude of addresses within the same city. Select 6 to 8 addresses, with one designated as zero mile, and construct a fully connected graph.

ii. Designate the zero-mile location as the pickup point.

iii. Calculate the shortest paths between the pickup point and delivery points.

Code:

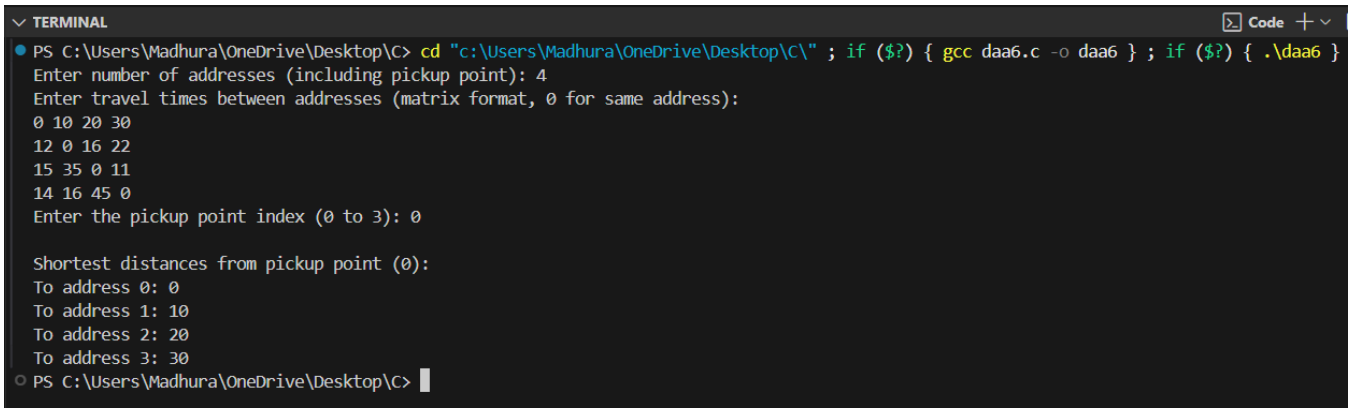
```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

typedef struct {
   int src, dest, weight;
}Edge;

void bellmanFord(Edge edges[], int V, int E, int src) {
```

```
int dist[V];

for (int i= 0;i <V; i++)
   dist[i] = INT_MAX;
dist[src] = 0;

for (int i =1; i <=V-1; i++) {
   for (int j= 0; j< E; j++) {
      int u= edges[j].src;
      int v= edges[j].dest;
      int w= edges[j].weight;
      if (dist[u] != INT_MAX && dist[u]+w<dist[v])
         dist[v]=dist[u] + w;
   }
}

for (int j=0; j<E; j++) {
   int u= edges[j].src;
   int v= edges[j].dest;
   int w= edges[j].weight;
   if (dist[u] != INT_MAX && dist[u] + w < dist[v]) {
      printf("Graph contains negative weight cycle\n");
      return;
   }
}

printf("\nShortest distances from pickup point (0):\n");
for (int i= 0; i< V; i++) {
   printf("To address %d: ", i);
```

```
        if (dist[i]==INT_MAX)
            printf("Unreachable\n");
        else
            printf("%d\n", dist[i]);
    }
}

int main() {
    int V;
    printf("Enter number of addresses (including pickup point): ");
    scanf("%d", &V);

    int E=V*(V-1);
    Edge edges[E];
    int k=0;

    printf("Enter travel times between addresses (matrix format, 0 for same address):\n");
    int travelTime[V][V];
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            scanf("%d", &travelTime[i][j]);
        }
    }
    for (int i= 0; i< V; i++) {
        for (int j= 0; j< V; j++) {
            if (i!= j) {
                edges[k].src = i;
                edges[k].dest = j;
                edges[k].weight = travelTime[i][j];
```

```
            k++;
        }
    }
}
int pickupPoint;
printf("Enter the pickup point index (0 to %d): ", V-1);
scanf("%d", &pickupPoint);
bellmanFord(edges, V, E, pickupPoint);

return 0;
}
```

Output:



Leetcode Submission: