LAB REPORT

CSE 114 : Data Structure and Algorithms Sessional

PREPARED BY

Mehrin Farzana
ID: 2101013
Session: 2021-2022
Date: 31/10/2023

SUPERVISED BY

Suman Saha
Lecturer
Department of IRE, BDU

BANGABANDHU SHEIKH MUJIBUR

RAHMAN DIGITAL UNIVERSITY

(BDU)

List of Problems

1. Floyd warshall algorithm.

2. Dijsktra algorithm.

Problem No.: 01

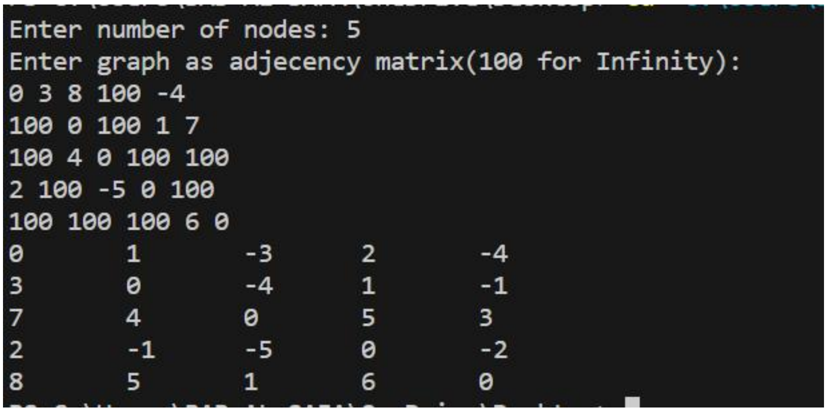
Problem Statement:

Floyd warshall algorithm.

Code:

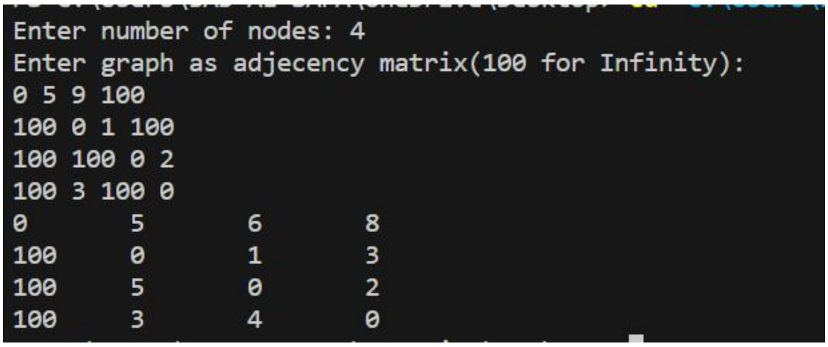
```
#include<stdio.h>
int main(){
  int n;
  printf("Enter number of vatices: ");
  scanf("%d", &n);
  int a[n][n];
  printf("Enter graph as adjecency matrix(100 for Infinity): \n");
  for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
      scanf("%d", &a[i][j]);
    }
  }
  for(int k=0; k<n; k++){
    for(int i=0; i<n; i++){
      for(int j=0; j<n; j++){
        if(a[i][j]>(a[i][k]+a[k][j])){
          a[i][j]=a[i][k]+a[k][j];
        }
      }
    }
  }
  for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
      printf("%d\t", a[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

Output:



```
Enter number of nodes: 5
Enter graph as adjecency matrix(100 for Infinity):
0 3 8 100 -4
100 0 100 1 7
100 4 0 100 100
2 100 -5 0 100
100 100 100 6 0
0       1       -3      2       -4
3       0       -4      1       -1
7       4       0       5       3
2       -1      -5      0       -2
8       5       1       6       0
```

Fig 1.1: Output on console for case 1.



```
Enter number of nodes: 4
Enter graph as adjecency matrix(100 for Infinity):
0 5 9 100
100 0 1 100
100 100 0 2
100 3 100 0
0       5       6       8
100     0       1       3
100     5       0       2
100     3       4       0
```

Fig 1.1: Output on console for case 1.

Problem No.: 02

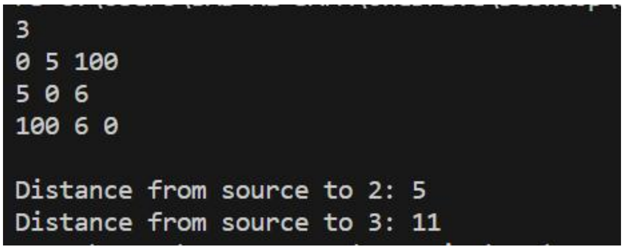
Problem Statement:

Dijsktra algorithm.

Code:

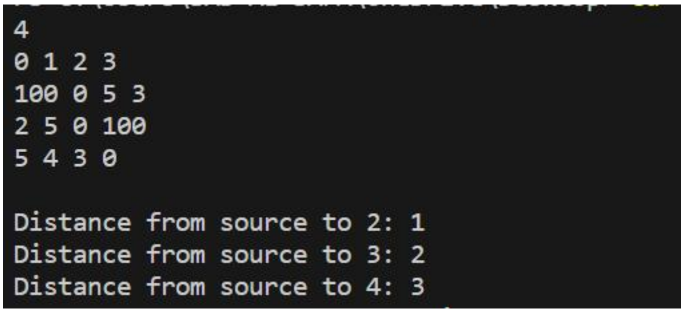
```
#include <stdio.h>
#define INFINITY 9999
#define MAX 10

void Dijkstra(int Graph[MAX][MAX], int n, int start);

void Dijkstra(int Graph[MAX][MAX], int n, int start) {
  int cost[MAX][MAX], distance[MAX], pred[MAX];
  int visited[MAX], count, mindistance, nextnode, i, j;

  // Creating cost matrix
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (Graph[i][j] == 0)
        cost[i][j] = INFINITY;
      else
        cost[i][j] = Graph[i][j];

  for (i = 0; i < n; i++) {
    distance[i] = cost[start][i];
    pred[i] = start;
    visited[i] = 0;
  }

  distance[start] = 0;
  visited[start] = 1;
  count = 1;

  while (count < n - 1) {
    mindistance = INFINITY;

    for (i = 0; i < n; i++)
      if (distance[i] < mindistance && !visited[i]) {
        mindistance = distance[i];
        nextnode = i;
      }
    visited[nextnode] = 1;
    for (i = 0; i < n; i++)
      if (!visited[i])
        if (mindistance + cost[nextnode][i] < distance[i]) {
          distance[i] = mindistance + cost[nextnode][i];
          pred[i] = nextnode;
        }
    count++;
  }

  // Printing the distance
  for (i = 0; i < n; i++)
    if (i != start) {
      printf("\nDistance from source to %d: %d", i+1, distance[i]);
    }
}
int main() {
  int Graph[MAX][MAX], i, j, n, u;
  scanf("%d", &n);

  for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
      scanf("%d", &Graph[i][j]);
    }
  }

  u = 0;
  Dijkstra(Graph, n, u);

  return 0;
}
```

Output:



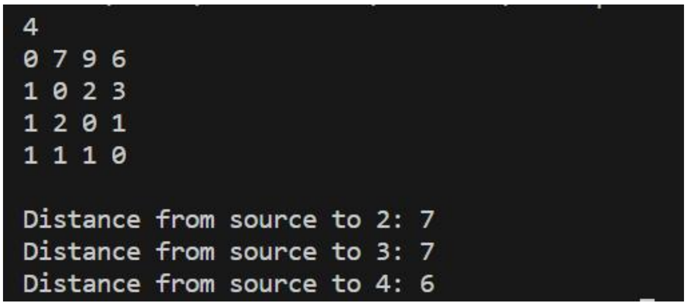
```
3
0 5 100
5 0 6
100 6 0

Distance from source to 2: 5
Distance from source to 3: 11
```

Fig 1.1: Output on console for case 1.



```
4
0 1 2 3
100 0 5 3
2 5 0 100
5 4 3 0

Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 3
```

Fig 1.2: Output on console for case 2.



```
4
0 7 9 6
1 0 2 3
1 2 0 1
1 1 1 0

Distance from source to 2: 7
Distance from source to 3: 7
Distance from source to 4: 6
```

Fig 1.3: Output on console for case 3.