Experiment Number 5

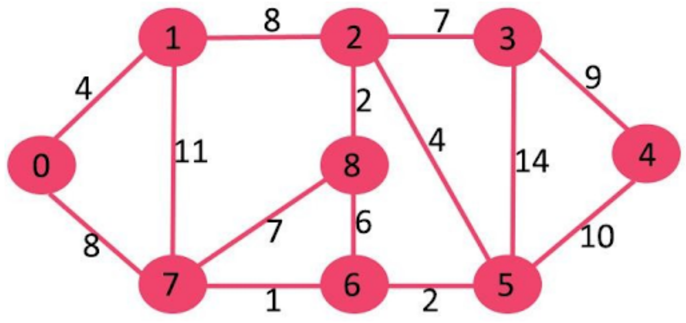
Name ::	Rishabh Anand	UID ::	19BCS4525
Branch ::	CSE - IoT	Sec/Grp ::	1/A
Semester ::	5th	Date ::	14th Nov, 2021
Subject ::	Adv Programming Lab	CODE ::	CSP-347

1. Aim :

a. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.
b. Compute the transitive closure of a given directed graph using Warshall's algorithm.

2. Task :

1. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.
2. Compute the transitive closure of a given directed graph using Warshall's algorithm.



3A. Algorithm :

- Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest-path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
- Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
- While sptSet doesn't include all vertices:
  - Pick a vertex u which is not there in sptSet and has a minimum distance value.
  - Include u to sptSet.
  - Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

3B. Algorithm :

- Instead of an integer resultant matrix (dist[V][V] in floyd warshall), we can create a boolean reach-ability matrix reach[V][V] (we save space). The value reach[i][j] will be 1 if j is reachable from i, otherwise 0.
- Instead of using arithmetic operations, we can use logical operations. For arithmetic operation , logical and is used, and for a min, logical or is used. (We save time by a constant factor. Time complexity is the same though)

4A. Source Code :

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

using namespace std;

#define V 9

int minDist(int dist[], bool sptSet[])
{
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
    return min_index;
}

void printSol(int dist[])
{
    cout << "Vertex \t Distance from Source" << endl;
    for (int i = 0; i < V; i++)
        cout << i << " \t\t" << dist[i] << endl;
}

void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;
    dist[src] = 0;
    for (int count = 0; count < V - 1; count++)
    {
        int u = minDist(dist, sptSet);
        sptSet[u] = true;
        for (int v = 0; v < V; v++)
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u
                ] + graph[u][v] < dist[v])
                 dist[v] = dist[u] + graph[u][v];
    }
    printSol(dist);
}
```

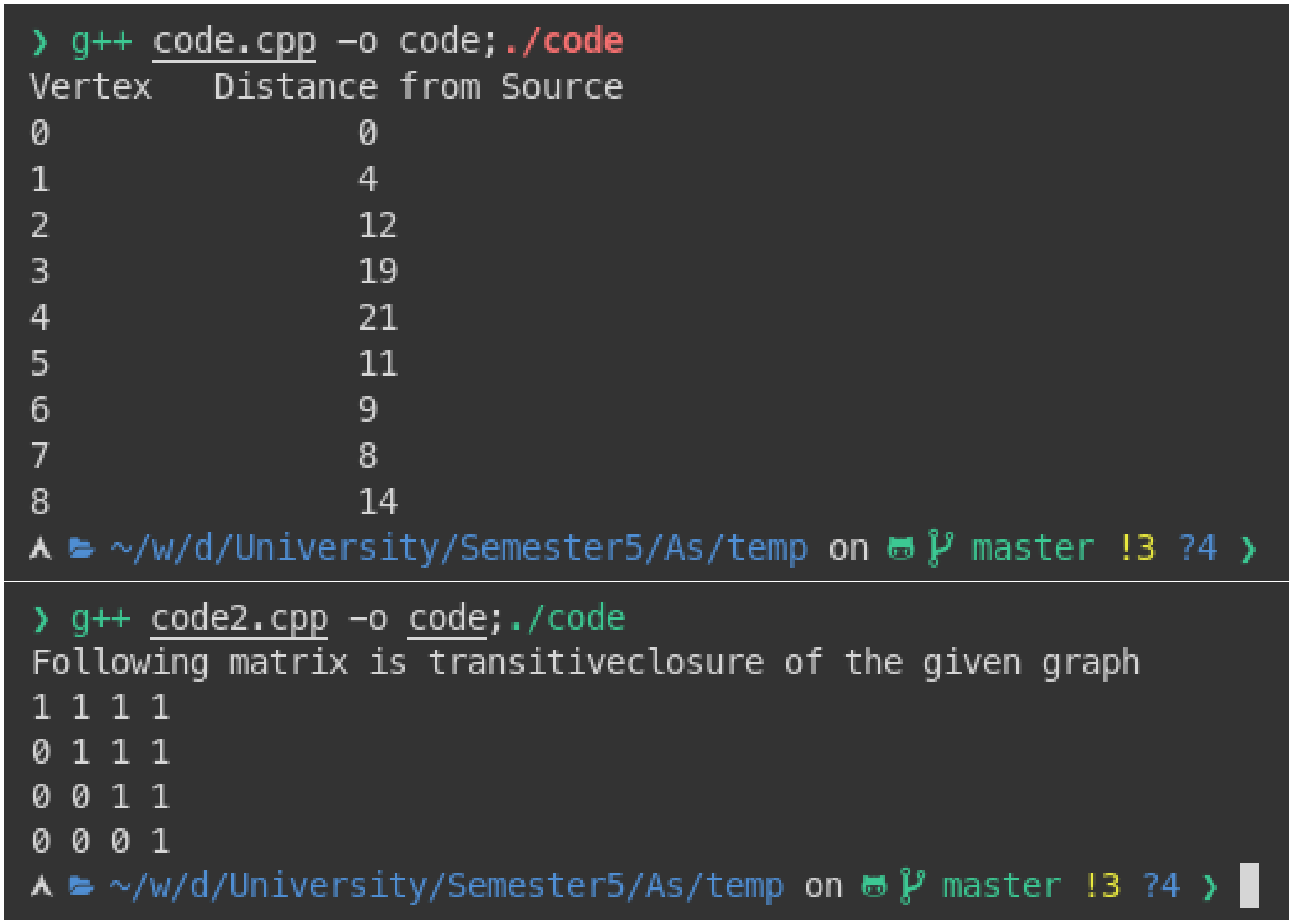
```
int main()
{
    int graph[V][V] = {{0, 4, 0, 0, 0, 0, 0, 8, 0},
                        {4, 0, 8, 0, 0, 0, 0, 11, 0},
                        {0, 8, 0, 7, 0, 4, 0, 0, 2},
                        {0, 0, 7, 0, 9, 14, 0, 0, 0},
                        {0, 0, 0, 9, 0, 10, 0, 0, 0},
                        {0, 0, 4, 14, 10, 0, 2, 0, 0},
                        {0, 0, 0, 0, 0, 2, 0, 1, 6},
                        {8, 11, 0, 0, 0, 0, 1, 0, 7},
                        {0, 0, 2, 0, 0, 0, 6, 7, 0}};
    dijkstra(graph, 0);
    return 0;
}
```

4B. Source Code :

```
#include <stdio.h>

#define V 4

void printSolution(int reach[][V]);

void transitiveClosure(int graph[][V])
{
    int reach[V][V], i, j, k;

    for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
            reach[i][j] = graph[i][j];

    for (k = 0; k < V; k++)
    {
        for (i = 0; i < V; i++)
        {
            for (j = 0; j < V; j++)
                reach[i][j] = reach[i][j] ||
                              (reach[i][k] && reach[k][j]);
        }
    }
    printSolution(reach);
}

void printSolution(int reach[][V])
{
    printf("Following matrix is transitive");
    printf("closure of the given graph\n");
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            if (i == j)
                printf("1 ");
            else
                printf("%d ", reach[i][j]);
        }
        printf("\n");
    }
}
```

```
int main()
{
    int graph[V][V] = {{1, 1, 0, 1},
                       {0, 1, 1, 0},
                       {0, 0, 1, 1},
                       {0, 0, 0, 1}};
    transitiveClosure(graph);
    return 0;
}
```

5. Observations :



```
❯ g++ code.cpp -o code;./code
Vertex   Distance from Source
0               0
1               4
2               12
3               19
4               21
5               11
6               9
7               8
8               14
⋏ 🗁 ~/w/d/University/Semester5/As/temp on  master !3 ?4 ❯
```

```
❯ g++ code2.cpp -o code;./code
Following matrix is transitiveclosure of the given graph
1 1 1 1
0 1 1 1
0 0 1 1
0 0 0 1
⋏ 🗁 ~/w/d/University/Semester5/As/temp on  master !3 ?4 ❯
```

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