Sardar Patel Institute of Technology

SEM IV: DESIGN AND ANALYSIS OF ALGORITHMS.

Name	SMRUTI SONEKAR
UID no.	2021700064
BRANCH:	SE CSE (DATA SCIENCE)
Experiment No.	6

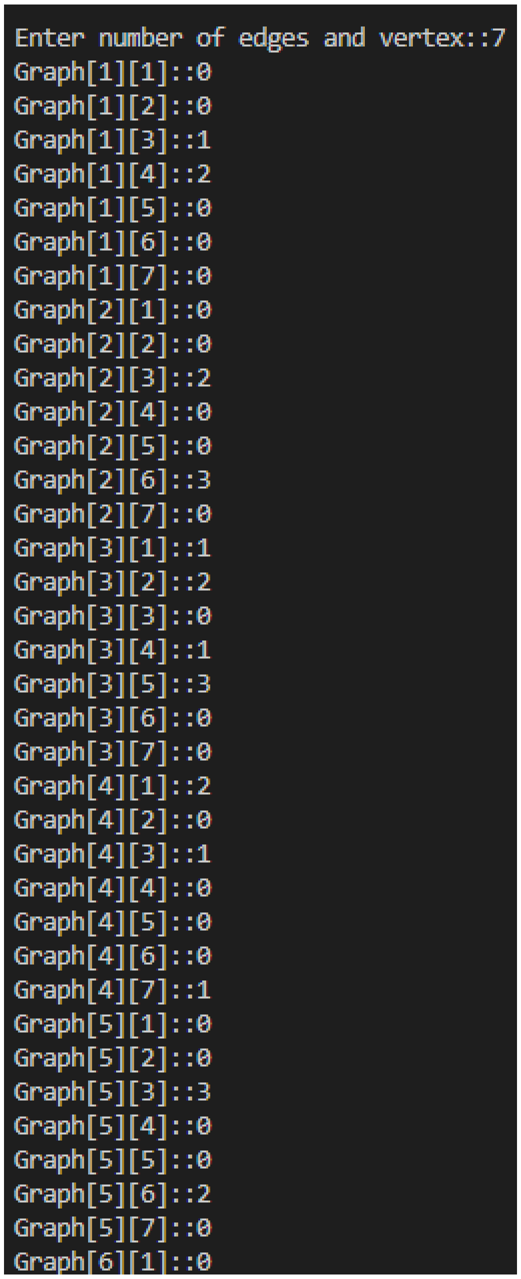
TOPIC:	DIJKSTRA Algorithm
QUERY:	Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph. We need to maintain the path distance of every vertex. We can store that in an array of size v, where v is the number of vertices. We also want to be able to get the shortest path, not only know the length of the shortest path. For this, we map each vertex to the vertex that last updated its path length. Once the algorithm is over, we can backtrack from the destination vertex to the source vertex to find the path. Dijkstra's Algorithm Complexity Time Complexity: O(E *Log V) where, E is the number of edges and V is the number of vertices. Space Complexity: O(V)

PROGRAM:

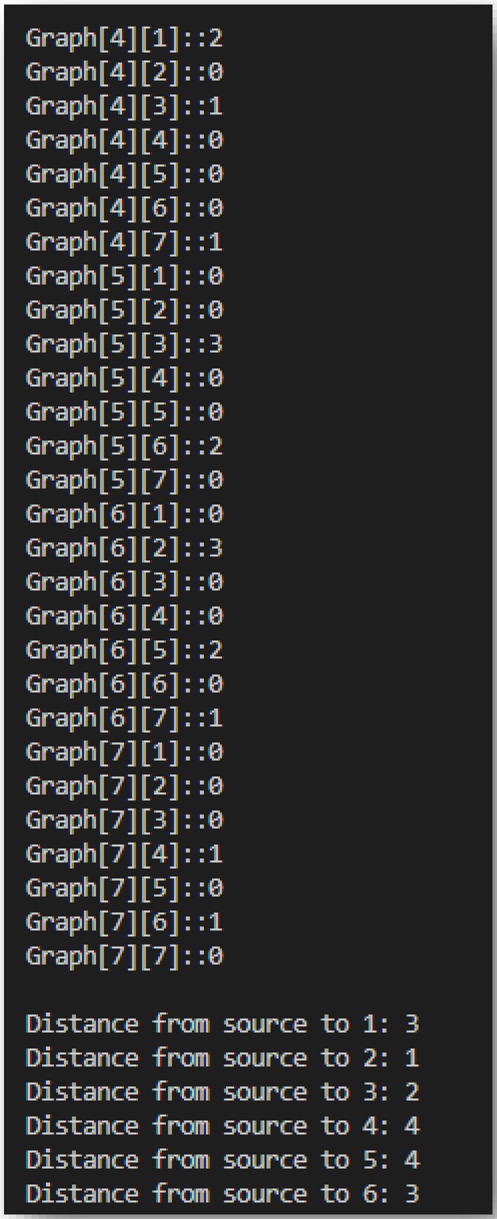
```c
// Dijkstra's Algorithm in C

#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++)
```

```c
      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, distance[i]);
    }
}
int main()
{
  int Graph[MAX][MAX], i, j, n, u;


  printf("\nEnter number of edges and vertex::");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    for(j=0;j<n;j++)
    {
        printf("Graph[%d][%d]::",i+1,j+1);
        scanf("%d",&Graph[i][j]);
    }
  }


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

  return 0;
}
```

RESULT:



```
Enter number of edges and vertex::7
Graph[1][1]::0
Graph[1][2]::0
Graph[1][3]::1
Graph[1][4]::2
Graph[1][5]::0
Graph[1][6]::0
Graph[1][7]::0
Graph[2][1]::0
Graph[2][2]::0
Graph[2][3]::2
Graph[2][4]::0
Graph[2][5]::0
Graph[2][6]::3
Graph[2][7]::0
Graph[3][1]::1
Graph[3][2]::2
Graph[3][3]::0
Graph[3][4]::1
Graph[3][5]::3
Graph[3][6]::0
Graph[3][7]::0
Graph[4][1]::2
Graph[4][2]::0
Graph[4][3]::1
Graph[4][4]::0
Graph[4][5]::0
Graph[4][6]::0
Graph[4][7]::1
Graph[5][1]::0
Graph[5][2]::0
Graph[5][3]::3
Graph[5][4]::0
Graph[5][5]::0
Graph[5][6]::2
Graph[5][7]::0
Graph[6][1]::0
```



```
Graph[4][1]::2
Graph[4][2]::0
Graph[4][3]::1
Graph[4][4]::0
Graph[4][5]::0
Graph[4][6]::0
Graph[4][7]::1
Graph[5][1]::0
Graph[5][2]::0
Graph[5][3]::3
Graph[5][4]::0
Graph[5][5]::0
Graph[5][6]::2
Graph[5][7]::0
Graph[6][1]::0
Graph[6][2]::3
Graph[6][3]::0
Graph[6][4]::0
Graph[6][5]::2
Graph[6][6]::0
Graph[6][7]::1
Graph[7][1]::0
Graph[7][2]::0
Graph[7][3]::0
Graph[7][4]::1
Graph[7][5]::0
Graph[7][6]::1
Graph[7][7]::0

Distance from source to 1: 3
Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 4
Distance from source to 5: 4
Distance from source to 6: 3
```

CONCLUSION:

Dijkstra Algorithm is best method to find shortest path available and its concept is applied through c program.