Name : Resham Landge
Roll No : 2339

******************************Assignment No : 8*****************************

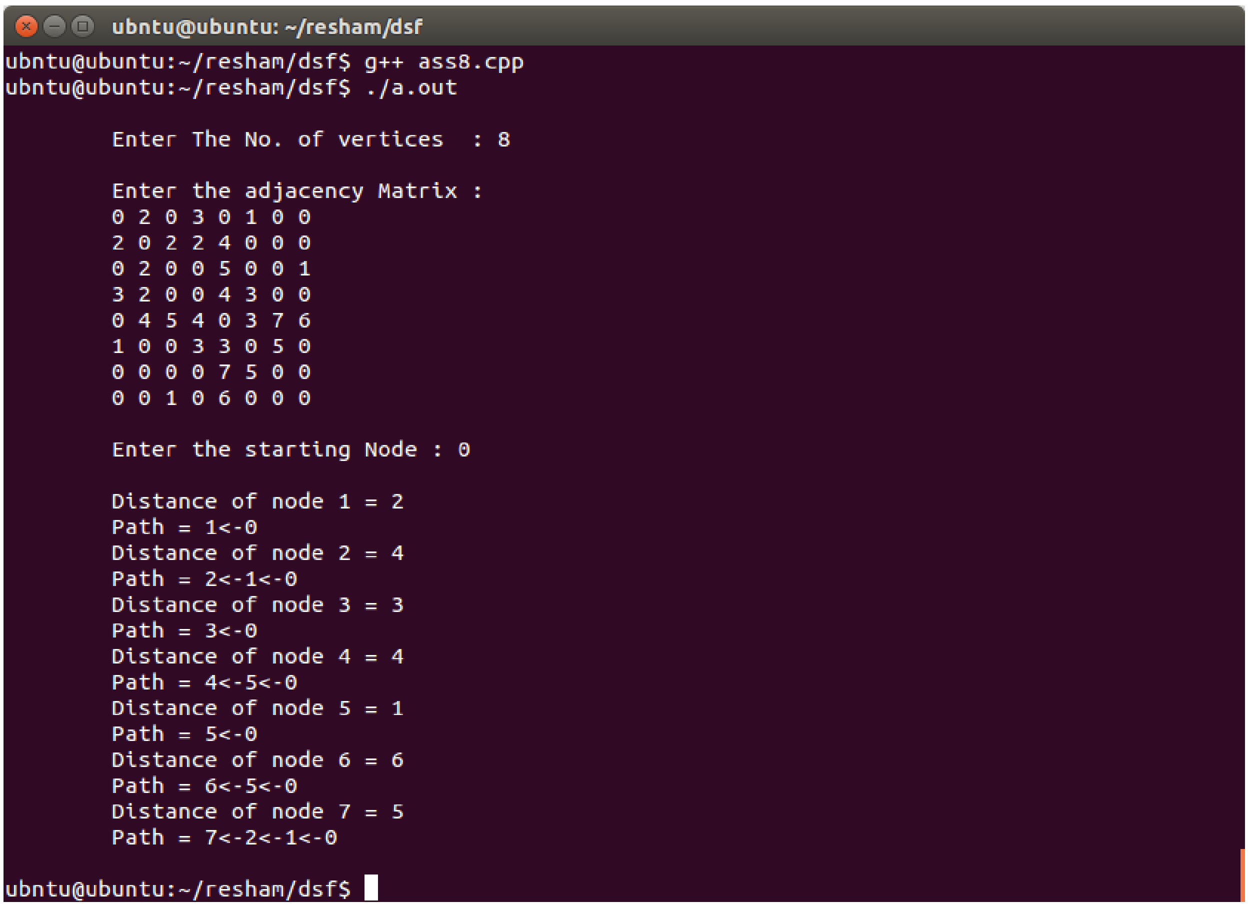
Title : Represent a given graph using adjacency matrix /adjacency list and find the shortest path using Dijkstra's algorithm

****

```
#include<iostream>
#define Infinity 9999
#define MAX 30
using namespace std;

class graph
{
	public:
		int G[MAX][MAX],n;

		graph()
		{
			n=0;
		}

		void create();
		void display();
		void dijikstra(int startnode);
};

void graph::create()				//to create the graph
{
	int i,k;

	cout<<"\n\tEnter The No. of vertices  : ";
	cin>>n;
	cout<<"\n\tEnter the adjacency Matrix : ";
	for(i=0; i<n; i++)
		for(k=0; k<n; k++)
			cin>>G[i][k];
}

void graph::display()				//to display the node of graph
{
	int i,k;
```

```
		for(i=0; i<n; i++)
		{
			cout<<"\n";
			for(k=0; k<n; k++)
				cout<<" "<<G[i][k];
		}
	}

	int main()
	{
		graph g;
		int s;

		g.create();
		cout<<"\n\tEnter the starting Node : ";
		cin>>s;
		g.dijikstra(s);
	}

	void graph::dijikstra(int startnode)
	{
		int   cost[MAX][MAX],distance[MAX],pred[MAX],visited[MAX],count,mindistance,
		nextnode,I k;
		//pred[] stores the predecessor of each node
		//count gives the number of nodes seen so far

		for(i=0; i<n; i++)				//create the cost matrix
			for(k=0; k<n; k++)
				if(G[i][k]==0)
					cost[i][k]=Infinity;
				else
					cost[i][k]=G[i][k];

		for(i=0; i<n; i++)				//initilize pred[],distance[] & visited[]
		{
			distance[i]=cost[startnode][i];
			pred[i]=startnode;
			visited[i]=0;
		}

		distance[startnode]=0;
		visited[startnode]=1;
		count=1;
		while(count < n-1)
		{
			mindistance=Infinity;

			for(i=0; i<n; i++)		//nextnode gives the node at minimum distance
				if(distance[i] < mindistance && !visited[i])
				{
					mindistance=distance[i];
					nextnode=i;
				}

			visited[nextnode]=1;		//check if a better path exists through nexxtnode

			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++;
		}

		for(i=0; i<n; i++)			//Print the path & distance of each node
			if(i!=startnode)
			{
				cout<<"\n\tDistance of node "<<i<<" = "<<distance[i];
				cout<<"\n\tPath = "<<i;

				k=i;

				do
				{
					k=pred[k];
					cout<<"<-"<<k;
				}while(k!=startnode);
			}
		cout<<"\n\n";
	}
```

Output :



```
ubntu@ubuntu:~/resham/dsf$ g++ ass8.cpp
ubntu@ubuntu:~/resham/dsf$ ./a.out

        Enter The No. of vertices  : 8

        Enter the adjacency Matrix :
        0 2 0 3 0 1 0 0
        2 0 2 2 4 0 0 0
        0 2 0 0 5 0 0 1
        3 2 0 0 4 3 0 0
        0 4 5 4 0 3 7 6
        1 0 0 3 3 0 5 0
        0 0 0 0 7 5 0 0
        0 0 1 0 6 0 0 0

        Enter the starting Node : 0

        Distance of node 1 = 2
        Path = 1<-0
        Distance of node 2 = 4
        Path = 2<-1<-0
        Distance of node 3 = 3
        Path = 3<-0
        Distance of node 4 = 4
        Path = 4<-5<-0
        Distance of node 5 = 1
        Path = 5<-0
        Distance of node 6 = 6
        Path = 6<-5<-0
        Distance of node 7 = 5
        Path = 7<-2<-1<-0

ubntu@ubuntu:~/resham/dsf$
```