Dijkstra Algorithm

```
#include <iostream>
using namespace std;

#define MAX 100
#define TEMP 0
#define PERM 1
#define infinity 9999
#define NIL -1

void findpath(int s, int v);
void djk(int s);
int mtemp();
void cgraph();
int a, adj[MAX][MAX], predce[MAX], pl[MAX], status[MAX];

int main()
{
	int s, v, n;
	cgraph();
	cout << "Enter source vertex : ";
	cin >> s;
	djk(s);

	while(1)
	{
		cout << "Enter destination vertex : ";
		cin >> v;
		if(v==-1)
		{
			break;
		}
		if( v<0 || v>=n)
		{
			cout << "This vertex does not exists" << endl;
		}
		else if(v==s)
		{
			cout << "Source and Destination vertices are same" << endl;
		}
		else if(pl[v]==infinity)
		{
			cout << "There is no path between Source and Destination" << endl;
		}
		else
		{
			findpath(s, v);
		}
	}
}
```

```
void djk(int s)
{
	int i, current, n;

	for(i=0; i<n; i++)
	{
		predce[i] = NIL;
		pl[i] = infinity;
		status[i] = TEMP;
	}
	pl[s] = 0;

	while(1)
	{
		current = mtemp();
		if(current==NIL)
			return;
		status[current] = PERM;

		for(i=0; i<n; i++)
		{
			if((adj[current][i]) != 0 && status[i] == TEMP)
			{
				if(pl[current] + adj[current][i] < pl[i])
				{
					predce[i] = current;
					pl[i] = pl[current] + adj[current][i];
				}
			}
		}
	}
}

int mtemp()
{
	int i, min = infinity, k = NIL, n;

	for(i=0; i<n; i++)
	{
		if(status[i] == TEMP && pl[i] < min)
		{
			min = pl[i];
			k = i;
		}
	}

	return k;
}

void findpath(int s, int v)
{
	int i, u, path[MAX], shortdist = 0, count = 0;
```

```
        while(v!=s)
        {
                count++;
                path[count] = v;
                u = predce[v];
                shortdist += adj[u][v];
                v = u;
        }

        count++;
        path[count] = s;
        cout << "Shortest path is : ";

        for(i=count; i>=1; i--)
        {
                cout << path[i];
        }

        cout << "Shortest distance is : " << shortdist;
}
void cgraph()
{
        int i, maxe, og, dst, wt, n;

        cout << "Enter no. of vertices : ";
        cin >> n;

        maxe = n*n - n;

        for(int i = 1; i <= maxe; i++)
        {
                cout << "Enter edges : ";
                cin >> og >> dst;

                if((og == -1) && (dst == -1))
                        break;

                cout << "Enter weight for this edge : ";
                cin >> wt;

                if(og >= n || dst >= n || og < 0 || dst < 0)
                {
                        cout << "Invalid edge" << endl;
                        i--;
                }
                else
                {
                        adj[og][dst] = wt;
                }
        }
}
```

```
/*output
Enter no. of vertices : 2
Enter edges : 1
0
Enter weight for this edge : 25
Enter edges : 1
0
Enter weight for this edge : 3
Enter source vertex : 1
distance is : 12.21
*/
```