NAME : JAGADEESH R
REGNO : 2021506314
ADS LAB 11a : Dijkstra's Algorithm Graph

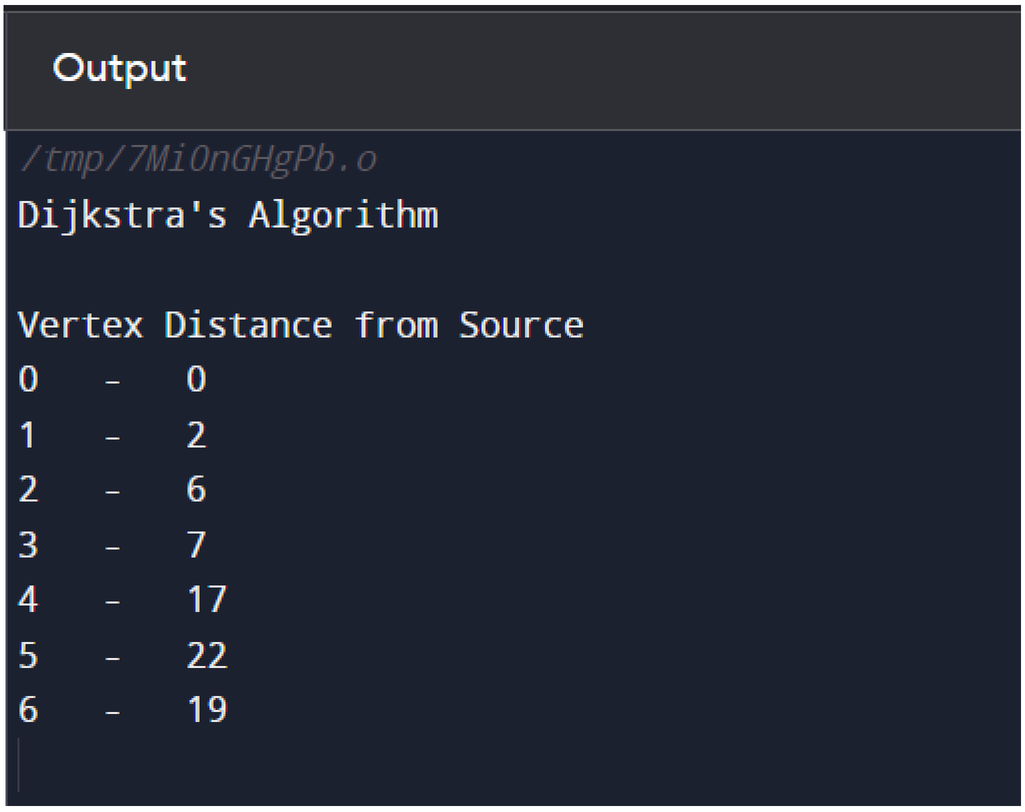
SOURCE CODE :

```
#include <bits/stdc++.h>
#include<iostream>
using namespace std;
#define INF 9999999
typedef pair<int, int> iPair;
class Graph {
	int V;
	list<pair<int, int> >* adj;
public:
	Graph(int V);
	void addEdge(int u, int v, int w);
	void shortestPath(int s);
};
Graph::Graph(int V)
{
	this->V = V;
	adj = new list<iPair>[V];
}
void Graph::addEdge(int u, int v, int w)
{
	adj[u].push_back(make_pair(v, w));
	adj[v].push_back(make_pair(u, w));
}
void Graph::shortestPath(int src)
{
	priority_queue<iPair, vector<iPair>, greater<iPair> >pq;
	vector<int> dist(V, INF);
	pq.push(make_pair(0, src));
	dist[src] = 0;
	while (!pq.empty()) {
		int u = pq.top().second;
		pq.pop();
		list<pair<int, int> >::iterator i;
		for (i = adj[u].begin(); i != adj[u].end(); ++i) {
			int v = (*i).first;
			int weight = (*i).second;
			if (dist[v] > dist[u] + weight) {
				dist[v] = dist[u] + weight;
				pq.push(make_pair(dist[v], v));
			}
		}
```

```
        }
        cout << "Dijkstra's Algorithm \n\n" << "Vertex Distance from Source" << endl;
        for (int i = 0; i < V; ++i)
                cout << i << "\t-\t" << dist[i] << endl;
}

int main()
{
        int V = 7;
        Graph g(V);
        g.addEdge(0, 1, 2);
        g.addEdge(0, 2, 6);
        g.addEdge(1, 3, 5);
        g.addEdge(2, 3, 8);
        g.addEdge(3, 4, 10);
        g.addEdge(3, 5, 15);
        g.addEdge(4, 6, 2);
        g.addEdge(5, 6, 6);
        g.shortestPath(0);
        return 0;
}
```

OUTPUT :



```
/tmp/7Mi0nGHgPb.o
Dijkstra's Algorithm

Vertex Distance from Source
0   -   0
1   -   2
2   -   6
3   -   7
4   -   17
5   -   22
6   -   19
```