

**CS 5343 Algorithm Analysis and Data Structures**

**Assignment #4**



*Submitted by*

**Naga Mutya Kumar Kumtsam**  
**(nxk210028)**

## Implementation of Dijkstra's algorithm

Program :

```
// Naga Mutya Kumar Kumtsam(nxk210028)
#include<iostream>
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
using namespace std;
typedef pair<int, int> twin;
void Edges(vector <pair<int, int> > sides[], int i, int j, int distance)
{
    sides[i].push_back(make_pair(j, distance));
    sides[j].push_back(make_pair(i, distance));
    cout<<"("<<i<<" "<<j<<" "<<distance<<"<<"<<"<<endl;
}
void display(int parent[],int i)
{
    if (parent[i]==-1)
    {
        return;
    }
    display(parent, parent[i]);
    printf("%d ", i);
}
void mindistance(vector<pair<int,int> > sides[], int k, int source)
{
    priority_queue< twin, vector <twin> , greater<twin> > prior_q;
    vector<int> distance(k, INF);
    int parent[k],i;
    for(int i=0;i<k;i++)
        parent[i]=-1;
    prior_q.push(make_pair(0, source));
    distance[source] = 0;
    while (!prior_q.empty())
    {
        int u = prior_q.top().second;
        prior_q.pop();
        for (auto x : sides[u])
        {
            int v = x.first;
            int dist = x.second;
            if (distance[v] > distance[u] + dist)
            {
                distance[v] = distance[u] + dist;
                parent[v]=u;
            }
        }
    }
}
```

```

        prior_q.push(make_pair(distance[v], v));
    }
}
}
cout<<"Source\t\tDestination\t\tmin_distance\t\tPath";
for ( i = 0; i < k; ++i)
{
    printf("\n%d\t\t%d\t\t\t\t\t",source, i, distance[i]);
    display(parent, i);
}
}
int main()
{
    int vertices = 10;
    vector<twinn> sides[vertices];
    cout <<"Vertices and their Edges:\n" ;
    Edges(sides, 0, 1, 8);
    Edges(sides, 0, 6, 10);
    Edges(sides, 9, 6, 2);
    Edges(sides, 1, 2, 12);
    Edges(sides, 1, 6, 1);
    Edges(sides, 2, 3, 6);
    Edges(sides, 9, 7, 12);
    Edges(sides, 2, 7, 5);
    Edges(sides, 2, 5, 9);
    Edges(sides, 3, 4, 4);
    Edges(sides, 3, 6, 7);
    Edges(sides, 4, 5, 2);
    Edges(sides, 4, 6, 8);
    Edges(sides, 5, 6, 5);
    Edges(sides, 6, 7, 9);
    Edges(sides, 7, 8, 2);
    Edges(sides, 7, 9, 6);
    Edges(sides, 8, 9, 4);
    Edges(sides, 9, 8, 1);
    Edges(sides, 1, 3, 4);
    cout<<"\n\n" ;
    cout<<"After Running Dijkstra's Algorithm : \n";
    mindistance(sides,vertices, 0);
    return 0;
}

```

## Executions

### List of vertices and edges(pair of vertices)

```
"F:\UTD Academics\F21-1st Semester\5343\Assignment4\Assignment4.exe"
Vertices and their Edges:
(0,1,8)(0,6,10)(9,6,2)(1,2,12)(1,6,1)(2,3,6)(9,7,12)(2,7,5)(2,5,9)(3,4,4)(3,6,7)(4,5,2)(4,6,8)(5,6,5)(6,7,9)(7,8,2)(7,9,6)(8,9,4)(9,8,1)(1,3,4)

Process returned 0 (0x0)   execution time : 0.400 s
Press any key to continue.
```

### After Running dijkstra's algorithm

```
"F:\UTD Academics\F21-1st Semester\5343\Assignment4\Assignment4.exe"
Vertices and their Edges:
(0,1,8)(0,6,10)(9,6,2)(1,2,12)(1,6,1)(2,3,6)(9,7,12)(2,7,5)(2,5,9)(3,4,4)(3,6,7)(4,5,2)(4,6,8)(5,6,5)(6,7,9)(7,8,2)(7,9,6)(8,9,4)(9,8,1)(1,3,4)

After Running Dijkstra's Algorithm :
Source      Destination      min_distance      Path
0           0                0                1
0           1                8                1 3 2
0           2               18               1 3
0           3               12               1 3 4
0           4               16               1 6 5
0           5               14               1 6
0           6                9               1 6 9 8 7
0           7               14               1 6 9 8
0           8               12               1 6 9
0           9               11               1 6 9

Process returned 0 (0x0)   execution time : 0.644 s
Press any key to continue.
```