Name: Shaan Agarwal
UID: 2021300001
Branch: Computer Engineering (Comps A)
Batch: A
Experiment No: 6

Aim: To implement Dijkstra's Algorithm

Algorithm:

Create a priority queue to store the vertices in the increasing order of their distance from the source vertex. Initialize the distance of all the vertices to infinity and the distance of the source vertex to 0.

Insert the source vertex into the priority queue.

While the priority queue is not empty, do the following:

a. Extract the vertex with the minimum distance from the priority queue. Let's call this vertex u.

b. For each neighbour v of u, do the following:

i. Calculate the distance from the source vertex to v through u. This is equal to the distance to u plus the weight of the edge from u to v.

ii. If this distance is less than the current distance to v, update the distance to v to this new distance and insert v into the priority queue.

After the while loop, the distances of all the vertices from the source vertex have been calculated.

Program:

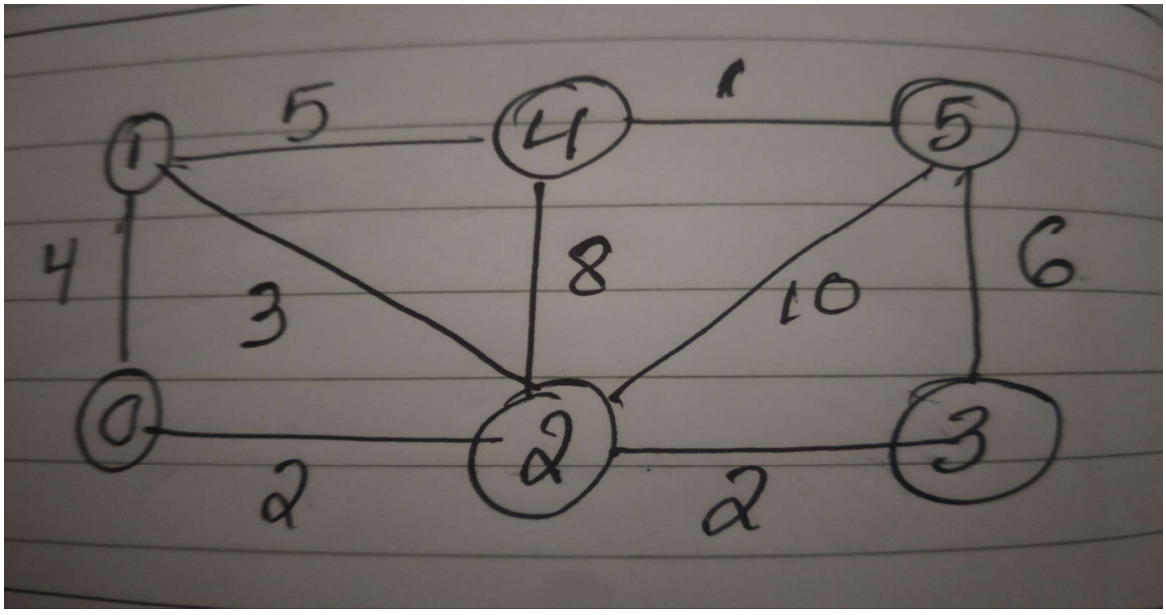
```
#include<bits/stdc++.h>

using namespace std;

void dijkstra(vector<vector<pair<int, int>>> graph, int source) {
    int V = graph.size();
    vector<int> dist(V, INT_MAX);
    dist[source] = 0;
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,
int>>> pq;
    pq.push(make_pair(0, source));
    while (!pq.empty()) {
        int u = pq.top().second;
        pq.pop();
        for (auto i = graph[u].begin(); i != graph[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 << "Vertex \t Distance from Source\n";
    for (int i = 0; i < V; i++)
        cout << i << " \t\t " << dist[i] << endl;
}

int main() {
    int V, E;
    cout << "Enter the number of vertices and edges: ";
    cin >> V >> E;
    vector<vector<pair<int, int>>> graph(V);
    cout << "Enter the edges and their weights (source, destination,
weight):\n";
    for (int i = 0; i < E; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        graph[u].push_back(make_pair(v, w));
        graph[v].push_back(make_pair(u, w)); // for undirected graph
    }
    int source;
    cout << "Enter the source vertex: ";
    cin >> source;
    dijkstra(graph, source);
    return 0;
```

```
}
```

Output

```
Enter the number of vertices and edges: 6 9
Enter the edges and their weights (source, destination, weight):
0 1 4
0 2 2
1 2 3
1 4 5
2 3 2
2 4 8
2 5 10
3 5 6
4 5 1
Enter the source vertex: 0
Vertex   Distance from Source
0                0
1                4
2                2
3                4
4                9
5                10
```

Representation of the edges and vertices in the form of a graph



Conclusion: After performing the above experiment, I have understood the concept of Dijkstra's Algorithm, and have implemented the algorithm using a C++ Program.