

OSCN LAB 1

MOHAMMED NIZAMUDDIN
M.TECH CSE
2503B05144

Write a C++ program to implement Dijkstra's Single Source Shortest Path Algorithm for a graph represented using an adjacency matrix.

Number of vertices: 5

Edges:

0 1 4

0 2 8

1 4 6

2 3 2

3 4 10

Source vertex: 0

CODE:

```
#include <iostream>
```

```
#include <vector>
```

```
#include <queue>
```

```
#include <climits>
```

```
using namespace std;
```

```
vector<vector<vector<int>>> constructAdj(vector<vector<int>> &edges, int V) {
```

```

vector<vector<vector<int>>> adj(V);

for (const auto &edge : edges) {
    int u = edge[0];
    int v = edge[1];
    int wt = edge[2];

    adj[u].push_back({v, wt});
    adj[v].push_back({u, wt});
}

return adj;
}

vector<int> dijkstra(int V, vector<vector<int>> &edges, int src) {
    vector<vector<vector<int>>> adj = constructAdj(edges, V);

    priority_queue<vector<int>, vector<vector<int>>, greater<vector<int>>> pq;
    vector<int> dist(V, INT_MAX);

    dist[src] = 0;
    pq.push({0, src});

    while (!pq.empty()) {
        int u = pq.top()[1];
        pq.pop();
    }
}

```

```

    for (auto x : adj[u]) {
        int v = x[0];
        int weight = x[1];

        if (dist[v] > dist[u] + weight) {
            dist[v] = dist[u] + weight;
            pq.push({dist[v], v});
        }
    }
}

return dist;
}

```

```

int main() {
    int V = 5;
    int src = 0;

    vector<vector<int>> edges = {
        {0, 1, 4},
        {0, 2, 8},
        {1, 4, 6},
        {2, 3, 2},
        {3, 4, 10}
    };
}

```

```
vector<int> result = dijkstra(V, edges, src);
```

```
cout << "Shortest distances from source vertex " << src << ":\n";
```

```
for (int dist : result)
```

```
    cout << dist << " ";
```

```
cout << endl;
```

```
return 0;
```

```
}
```



The screenshot shows a C++ IDE with a code editor on the left and an output console on the right. The code implements Dijkstra's algorithm using a priority queue. The output console shows the execution result: "Finished in 0 ms" and "Shortest distances from source vertex 0: 0 4 8 10 10".

```
1 #include <iostream>
2 #include <vector>
3 #include <queue>
4 #include <limits>
5
6 using namespace std;
7
8 vector<vector<vector<int>>> constructAdj(vector<vector<int>> &edges, int V) {
9     vector<vector<vector<int>>> adj(V);
10
11     for (const auto &edge : edges) {
12         int u = edge[0];
13         int v = edge[1];
14         int wt = edge[2];
15
16         adj[u].push_back({v, wt});
17         adj[v].push_back({u, wt});
18     }
19
20     return adj;
21 }
22
23 vector<int> dijkstra(int V, vector<vector<int>> &edges, int src) {
24     vector<vector<vector<int>>> adj = constructAdj(edges, V);
25 }
```

Output: Finished in 0 ms
Shortest distances from source vertex 0:
0 4 8 10 10