ASSIGNMENT - 10

ANIKEIT SETHI (190001003)

* **Code Logic: -** We are given a starting city and we have to find the shortest distance to every city from this city. So, this problem is similar to the single source shortest path problem. We can just use Dijkstra’s Algorithm to achieve the result. Below I have implemented the Dijkstra’s algorithm using priority queue.

Code:-

#include <bits/stdc++.h>

typedef long long ll;

typedef long double ld;

#define fr(i, a, b) for (ll i = a; i < b; i++)

#define rf(i, a, b) for (ll i = a; i >= b; i--)

typedef std::vector<long long> vi;

#define F first

#define S second

#define fast \

ios\_base::sync\_with\_stdio(0); \

cin.tie(0); \

cout.tie(0);

#define mod 1000000007

#define PB push\_back

#define MP make\_pair

#define PI 3.14159265358979323846

#define all(a) a.begin(), a.end()

#define mx(a) \*max\_element(all(a))

#define mn(a) \*min\_element(all(a))

const ll INF = LLONG\_MAX / 2;

const ll N = 2e5 + 1;

using namespace std;

int main()

{

int n, m;

cout << "Input:->\n";

cin >> n >> m;

vector<pair<int, int>> adj[n + 1];

for (int i = 1; i <= m; i++)

{

ll a, b, c;

cin >> a >> b >> c;

adj[a].push\_back({b, c});

adj[b].push\_back({a, c});

}

int distance[n + 1];

for (int i = 1; i <= n; i++)

distance[i] = INT\_MAX;

distance[1] = 0;

priority\_queue<pair<int, int>> pq;

int mark[n + 1] = {0};

pq.push({0, 1});

while (!pq.empty())

{

pair<int, int> s = pq.top();

pq.pop();

int source = s.second, dist = s.first;

mark[source] = 1;

for (auto it : adj[source])

{

int x = it.first, edge\_weight = it.second;

if (mark[x] == 1)

continue;

if (distance[source] + edge\_weight < distance[x])

{

distance[x] = distance[source] + edge\_weight;

pq.push({-distance[x], x});

}

}

}

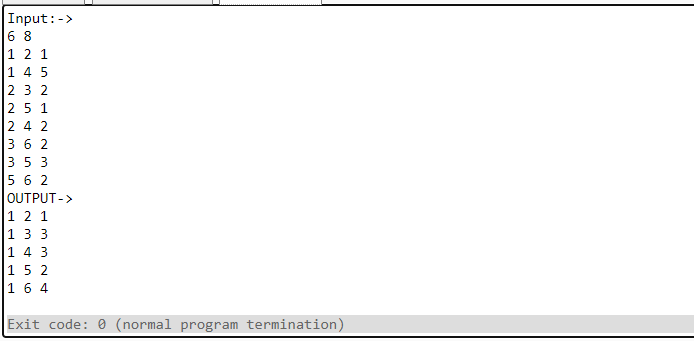
cout << "OUTPUT->\n";

for (int i = 2; i <= n; i++)

cout << "1 " << i << " " << distance[i] << "\n";

}

Output: -



**Time Complexity: - *O (N + M log(N))*** where n is the number of vertices and m is the number of edges.

Graph: -

