

**SIX WEEKS SUMMER TRAINING REPORT**

on

**SHORTEST PATH FINDER**

Submitted by

**KSHITIJ KISHORE BARYA**

**Registration No: 12103024**

**Program Name: B.Tech.(CSE)**

Under the Guidance of

**Mr. Ravi Kant Sahu**

**Assistant Professor**

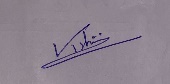
**School of Computer Science and Engineering**

**Lovely Professional University. Phagwara**

**(June- July, 2023)**

**DECLARATION**

I hereby declare that I have completed my six weeks summer training at **Human Resource Development Center, Lovely Professional University, Phagwara (Punjab)** from **6th June 2023** to **17th July 2023** under the guidance of **Mr. Ravi Kant Sahu**. I declare that I have worked with full dedication during these six weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of **B.Tech.(Computer Science & Engineering)** at Lovely Professional University, Phagwara.



Kshitij kishore barya

Registration No: 12103024

Date:- 15/7/2023

Project Code

#include <bits/stdc++.h>

using namespace std;

void ShortestPath(vector<vector<pair<int, int>>>& graph, int source, int destination) {

int numNodes = graph.size();

vector<int> distances(numNodes, INT\_MAX);

vector<int> parents(numNodes, -1);

vector<bool> visited(numNodes, false);

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

distances[source] = 0;

pq.push({0, source});

while (!pq.empty()) {

int current\_node = pq.top().second;

pq.pop();

if (visited[current\_node]) {

continue;

}

visited[current\_node] = true;

for (auto& neighbor : graph[current\_node]) {

int node = neighbor.first;

int edge\_weight = neighbor.second;

// Print the shortest path

cout << "The shortest path from " << source << " to " << destination << " is: ";

for (int i = 0; i < path.size(); i++) {

cout << path[i];

if (i != path.size() - 1) {

cout << " -> ";

}

}

cout << endl;

// Print the total distance

cout << "The total distance is: " << distances[destination] << endl;

}

if (distances[current\_node] + edge\_weight < distances[node]) {

distances[node] = distances[current\_node] + edge\_weight;

parents[node] = current\_node;

pq.push({distances[node], node});

}

}

}

if (distances[destination] == INT\_MAX) {

cout << "There is no path from " << source << " to " << destination << "." << endl;

return;

}

vector<int> path;

int current\_node = destination;

while (current\_node != -1) {

path.insert(path.begin(), current\_node);

current\_node = parents[current\_node];

}

cout << "The shortest path from " << source << " to " << destination << " is: ";

for (int i = 0; i < path.size(); i++) {

cout << path[i];

}

cout << endl; cout << "The total distance is: " << distances[destination] << endl;

}

cout << "The total distance is: " << distances[destination] << endl;

}

int main() {

int numNodes;

cout << "Enter the number of nodes in the graph: ";

cin >> numNodes;

vector<vector<pair<int, int>>> graph(numNodes);

int numEdges;

cout << "Enter the number of edges in the graph: ";

cin >> numEdges;

cout << "Enter the edges and weights in the format 'node1 node2 weight':\n";

for (int i = 0; i < numEdges; i++) {

int node1, node2, weight;

cin >> node1 >> node2 >> weight;

graph[node1].push\_back({node2, weight});

graph[node2].push\_back({node1, weight});

}

int source\_node;

cout << "Enter the Source Node: ";

cin >> source\_node;

int destination\_node;

cout << "Enter the Destination Node: ";

cin >> destination\_node;

ShortestPath(graph, source\_node, destination\_node);

return 0;

}

Input:-

A black screen with white text

Description automatically generated

Graph used:-

A diagram of a number connected to a triangle

Description automatically generated

Output:-

A computer screen with white text

Description automatically generated

Flow chart and DFD DiagramsA diagram of a flowchart

Description automatically generated

DFD for Shortest Path Finder

