**Final Project: Implementing Shortest Algorithm**

CSEN 2328: Data Structures & Algorithms

Texas A&M University- Kingsville

Spring 2019

Dr. Khan

By:

Diego Reyes

**Project description:**

The main objective of our final project was to implement Dijkstra’s shortest path algorithm on the given Navy ship passageway network. Given a source node the algorithm finds the shortest path distance to each exit and which exit is the closest. The algorithm was implemented using the C++ programming language. The given passageway network was comprised of 23 nodes, and 28 edges which were weighted.

**Given project tasks:**

1. Store the adjacency list of the passageway network using a two-dimensional array

2. Implement shortest path algorithm using C++.

3. Run the shortest path algorithm on the passageway network used in Homework 2 and 3.

4. Find the shortest path distances from a given node to all exit nodes and find the nearest exit.

**Given passageway network:**

A close up of a map

Description automatically generated

**Computed distances:**

The distance of the source vertex 15 from the exits are:

Exit A: 36

Exit F: 54

Exit T: 38

**Explanation of results:**

The closest exit from vertex 15 is exit A. The distance is computed by the algorithm which searches the graph for the shortest distance from one vertex to the next.

**Source code:**

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

**using namespace std;**

**#define INF INT\_MAX**

**int G[23][3] = { // Graph array**

**{1,8},**

**{0,22},**

**{3,7,22},**

**{2,15},**

**{6},**

**{6,9},**

**{4,5,7},**

**{2,6,8},**

**{0,7,9},**

**{5,8,10},**

**{9,11,20},**

**{10,12,18},**

**{11,13,19},**

**{12,14,17},**

**{13,15,16},**

**{3,14},**

**{14,17},**

**{13,16,18},**

**{11,17},**

**{12,20},**

**{10,19},**

**{22},**

**{1,2,21}**

**};**

**int W[23][3] = { // Weight array**

**{4,4},**

**{4,4},**

**{8,10,8},**

**{8,12},**

**{8},**

**{16,12},**

**{8,16,8},**

**{10,8,12},**

**{4,12,30},**

**{12,30,12},**

**{12,20,20},**

**{20,8,28},**

**{8,14,12},**

**{14,4,14},**

**{4,8,14},**

**{12,8},**

**{14,4},**

**{14,4,8},**

**{28,8},**

**{12,20},**

**{20,20},**

**{14},**

**{4,8,14}**

**};**

**int Deg[23] = {2,2,3,2,1,2,3,3,3,3,3,3,3,3,3,2,2,3,2,2,2,1,3}; // Degree array**

**int d[23]; // Distance array**

**typedef pair<int, int> node;**

**void ShortestPath(int s) {**

**priority\_queue<node, vector<node>, greater<node> > Q; // the queue gives priority to the node with the smallest distance from the source node**

**for (int v = 0; v < 23; v++) { // Initialize the distance of the nodes to infinity**

**d[v] = INF;**

**}**

**d[s] = 0; // Initialize the distance of the source node to 0**

**Q.push(make\_pair(s,d[s])); // Enqueue s and d[s]**

**while (!Q.empty()) {**

**int u = Q.top().first; // set u to the node with the distance closest to the source**

**Q.pop(); // Dequeue that node**

**for (int a = 0; a < Deg[u]; a++) { // for each node adjacent to u do**

**int v = G[u][a]; // adjacent node to u**

**int weight = W[u][a]; // weight of the edge between node u and node v**

**if (d[v] > d[u] + weight) { // if the distance of node v is greater than the distance of node u + the weight of the edge between the nodes do**

**d[v] = d[u] + weight; // set the distance of node v to the distance of node u + the weight of the edge between the nodes**

**Q.push(make\_pair(v,d[v])); // Enqueue the node**

**}**

**}**

**}**

**char min;**

**if (d[0] > d[5] and d[5] > d[19]) { // compares the distance of the source node to the exits and sets the node with the least distance to min**

**min = 'T';**

**}**

**else if (d[19] > d[0] and d[0] > d[5]) {**

**min = 'F';**

**}**

**else {**

**min = 'A';**

**}**

**cout << "The distance of the source vertex " << s << " from the exits are: " << endl;**

**cout << "Exit A: " << d[0] << endl << "Exit F: " << d[5] << endl << "Exit T: " << d[19] << endl;**

**cout << "The closest exit from the source vertex is: " << min << endl;**

**}**

**int main() {**

**ShortestPath(15);**

**return 0;**

**}**

**Output:**

**The distance of the source vertex 15 from the exits are:**

**Exit A: 36**

**Exit F: 54**

**Exit T: 38**

**The closest exit from the source vertex is: A**