

*ASSIGNMENT ON*

**Assignment Title:** **Find My Way**

**Course name: Algorithm**

**Summer 2018**

***Submitted To:***

Mohammad Azizur Rahman

**Lecturer, DIU**

**Submitted By:**

**Name ID**

**Rakibul Alam 181-16-285**

Dept. of CIS

Daffodil International University

Date of Submission: 06/08/2018

Table of Contents

[**Task 1** 3](#_Toc521347227)

[**1.** **The way I implement map in my program:** 3](#_Toc521347228)

[**2.** **The data structure, I will use to implement the map in the program:** 4](#_Toc521347229)

[**3.** **The Algorithm, I will use to solve the problem:** 5](#_Toc521347230)

[**The reason behind choose it:** 5](#_Toc521347231)

[**How it works:** 5](#_Toc521347232)

[**Time Complexity of the Algorithm:** 5](#_Toc521347233)

[**4.** **Algorithm:** 6](#_Toc521347234)

[**Task 2** 7](#_Toc521347235)

# **Task 1**

## **The way I implement map in my program:**

At first, I will initialize a node array as structure and which will hold parents, time and color values for the node.

1. A time value is assigned for every node and the value is set to 0 for the initial node and to infinity for other nodes. If Gulistan will the source vertex, it time value will be 0 and infinity for all other vertex.
2. All unvisited nodes marked as white color and initial node marked as Black color. If Gulistan will the source vertex it color will be Black.
3. For the initial or the nodes currently working, consider all its adjacent vertices which are unvisited and calculate their temporary times and after comparing minimum time will be assigned. For example, if Shahbag nearby neighbor will Hatirpool and if Shagbag to Hatirpool edge value will 5 and also if Hatirpool to Danmondi 32 edge value will 3. So Shahbag to Danmodi 32 edge value will be 5 + 3 = 8. If Danmodi 32 previously assigned value more than 8, then the value upadated. Othewise, keep the value as it is.
4. A visited node never be checked again. So, after finishing above steps with all the neighbors of the current node, make that node as visited and remove it from the unvisited set.
5. The destination node will be marked and for unvisited vertices will apply iterative process from step III, thus, shortest path will be found from source to destination.

Here, I will use adjacency matrix to represent the graph. In program, 2D-Array is used to represent adjacency matrix. Matrix row and column number are same as vertices number.

For scenario Graph, adjancecy matrix will be:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8X8 | Gulistan | High Court | PressClub | Shahbag | Nilkhet | Science Lab | Hatirpool | Danmondi 32 |
| Gulistan | 0 | 5 | 7 | 0 | 0 | 0 | 0 | 0 |
| High Court | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| Press Club | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 |
| Shahbag | 0 | 0 | 0 | 0 | 0 | 6 | 3 | 0 |
| Nilkhet | 0 | 0 | 0 | 0 | 0 | 3 | 5 | 0 |
| Science Lab | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 |
| Hatirpool | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 |
| Danmondi 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## **The data structure, I will use to implement the map in the program:**

In the program, I am going to use two-dimensional array for taking input vertices edges weight (**delay time)** and for representation of graph as adjacency matrices form. I will use liner array several times in the program. To store regions/area, I will use pointer array of character in the program. To store node and path information, I will use single dimensional array. To define node, I will used custom data structure called structure. In structure, I will store nodes information. As, Structure is user defined data structure, it can be designed as needed before using it. So, it will help me to work with node information.

## **The Algorithm, I will use to solve the problem:**

I am going to use Dijkstra’s algorithm to solve the problem.

### **The reason behind choose it:**

Dijkstra’s algorithm solves the single-source shortest-paths problem on a weighted directed graph G = (V, E) for the case in which all edge weights are nonnegative. In the scenario graph, have no any negative weighted edge. So, Dijistra’s algorithm will be efficient for the scenario. The problem can be solved using Bellman-Frod algortithm. But Dijkstra’s run time is less than Bellman-Frod Algorithm.

### **How it works:**

Dijkstra’s works with a set of vertices (S) which final shortest-path weights from the source s have already determined. The algorithm iteratively selects source vertex from the set of vertices to demine shortest path and add source to set of vertices. Dijkstra algorithm relaxes edges after initializing source parents and edge value. Using array to contain all the vertices. The work will be occurred repeatedly. Thus, it will update parents of vertices and estimate values (here, time). Dijistra’s algorithm always chooses the lightest and closest vertex for it source from set of vertices(S).

### **Time Complexity of the Algorithm:**

Here, I used an array for vertices set then the complexity is O (E + V2) = O ( V2) as each increase and decrease operation takes O(1) time, and each EXTRACT\_MIN operation takes O (V) time as here have to search entire array.

E = number of edges

V = number of vertices

In Dijkstra algorithm, the efficiency depends on the number of vertices and updating for priority queues that are used. If a standard binary heap is used then the complexity is O(ElogV).

Implementing, min-priority queue with a Fibonacci heap, its run time will be

O (VlogV + E).

## **Algorithm:**

INITIALIZE-SINGLE-SOURCE (G, s)

1. For each vertex v G.V
2. v.d =
3. v.= NIL
4. s.d = 0

RELAX (u, v, w)

1. if v.d > u.d + w (u, v)
2. v.d = u.d + w (u, v)
3. v. = u

DIJKSTRA (G, w, s)

1. INITIALIZE-SINGLE-SOURCE (G, s)
2. S =
3. Q = G.V
4. While Q
5. u = EXTRACT-MIN(Q)
6. S = S {u}
7. for each vertex v G.Adj[u]
8. RELAX (u, v, w)

# **Task 2**

Sample Input:

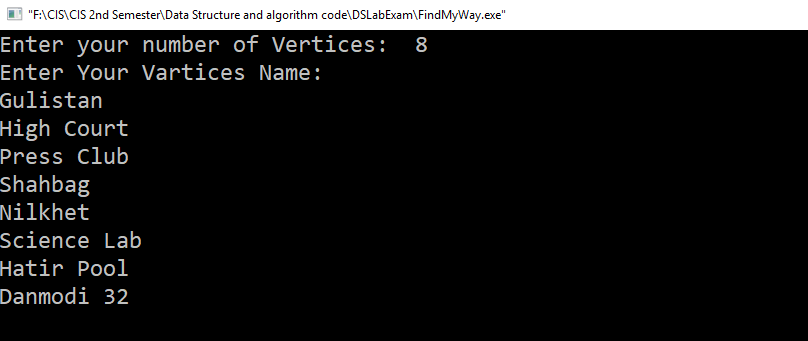


Figure: Sample Input 1

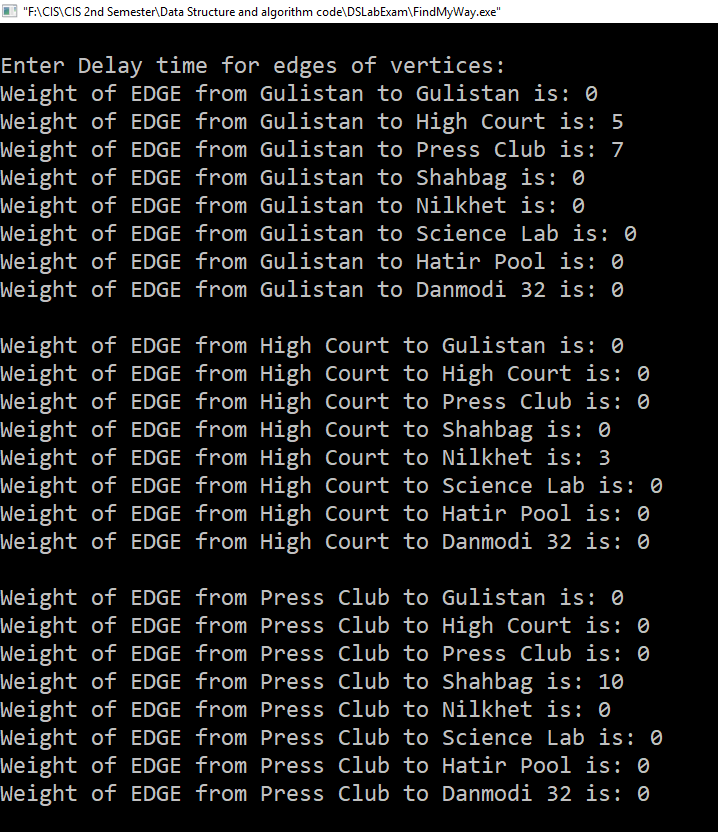


Figure: Sample Input 2

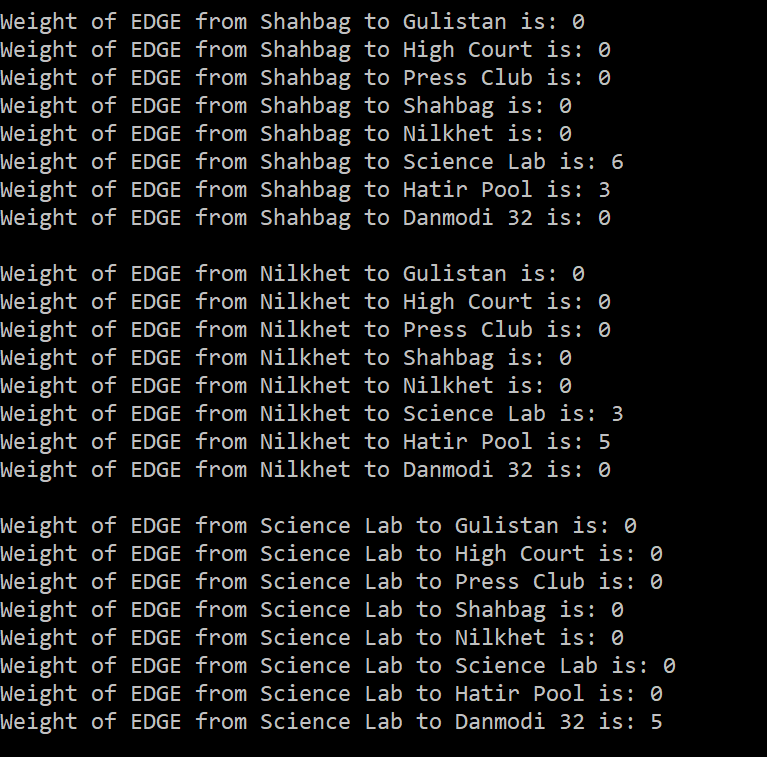


Figure: Sample Input 3

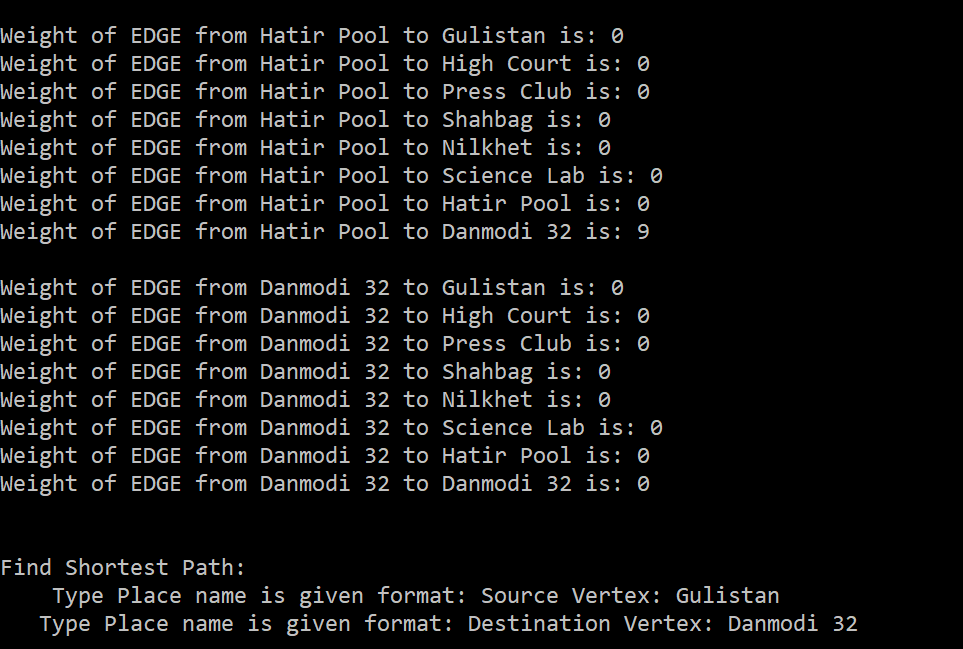


Figure: Sample Input 4

Sample Output:

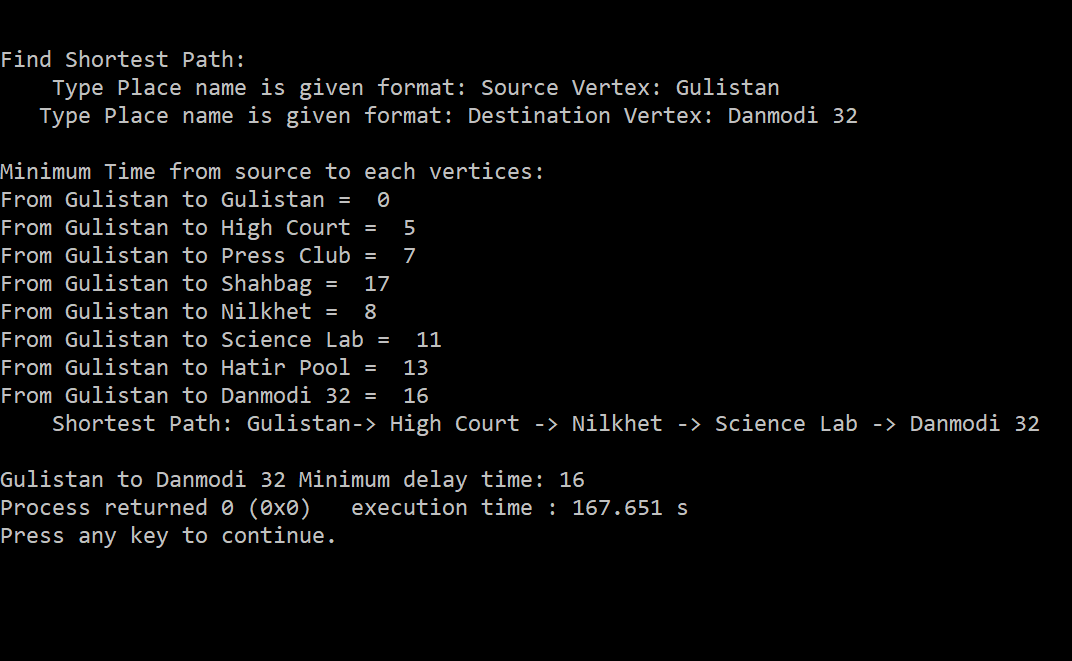


Figure: Sample Output