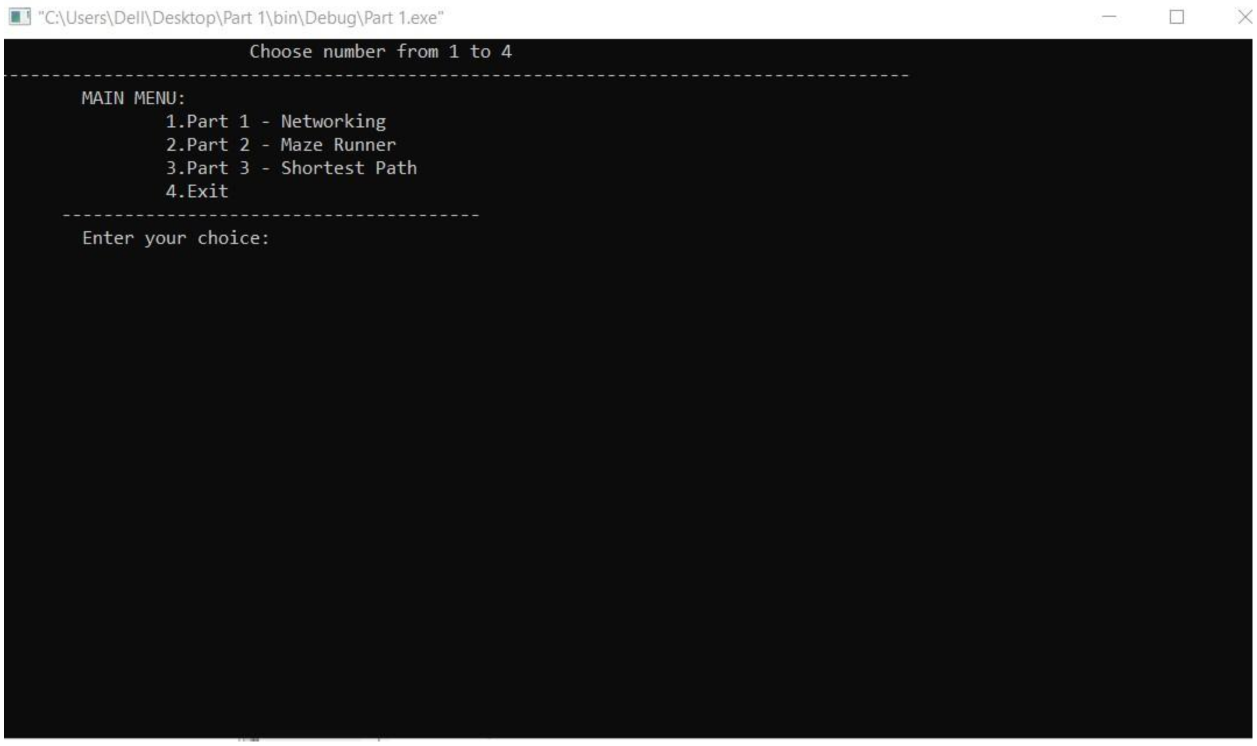
Data Structure 2

Final Project

Main Menu



"C:\Users\Dell\Desktop\Part 1\bin\Debug\Part 1.exe"

```
                       Choose number from 1 to 4
--------------------------------------------------------------------------------------
       MAIN MENU:
               1.Part 1 - Networking
               2.Part 2 - Maze Runner
               3.Part 3 - Shortest Path
               4.Exit
     ---------------------------------------
       Enter your choice:
```

Part 1: Linkedln Networking Site

Main idea is to count how many people are connected with each other from 1st, 2nd and 3rd connection. The user starts adding total number of people(vertices) and the connection between them(edges), then he puts the starting person(vertex) that he wants the connection start from and the length of the connection.

The program starts to fill the array 1’s or 0’s according to the edges and number of vertices, then by using BFS algorithm we start to check each vertex if visited or not and using pair to connect each vertex by the one before it to know how many people are connected.

Algorithm Used: BFS (breadth first search)

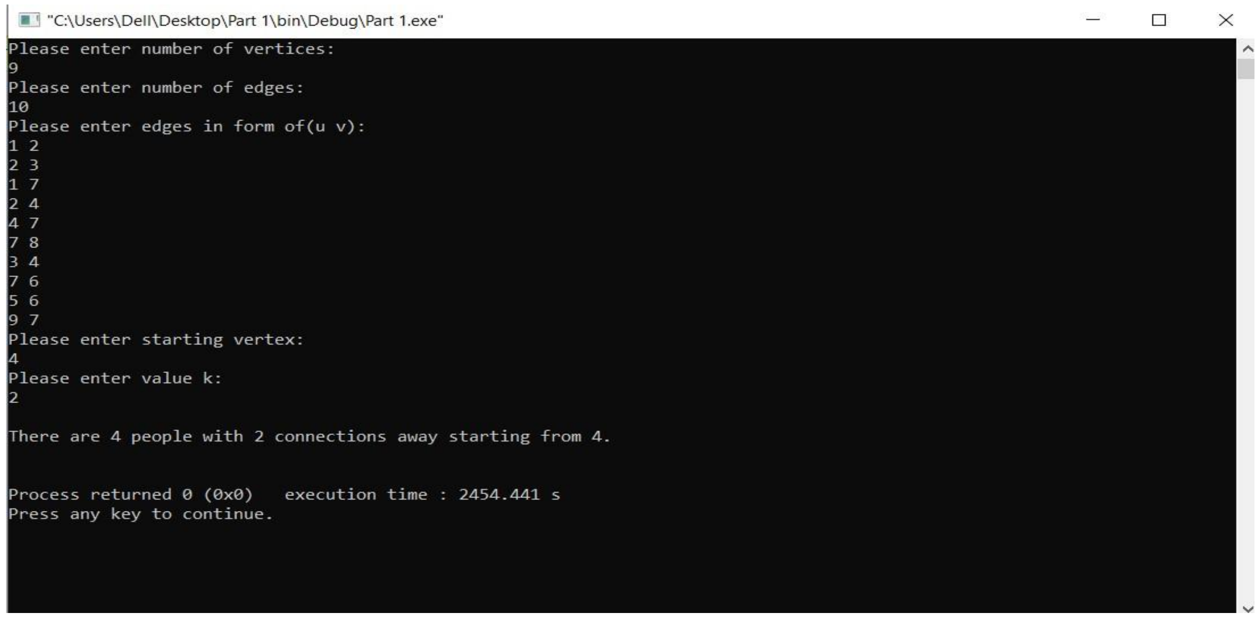
Data Structure Used: adjacency matrix

Running Time: *O*(*V*+*E*)

Input: - number of vertices
- number of edges
- edges
- starting vertex
- length of connection

Output: number of people away from the starting connected person

Program:



"C:\Users\Dell\Desktop\Part 1\bin\Debug\Part 1.exe"

```
Please enter number of vertices:
9
Please enter number of edges:
10
Please enter edges in form of(u v):
1 2
2 3
1 7
2 4
4 7
7 8
3 4
7 6
5 6
9 7
Please enter starting vertex:
4
Please enter value k:
2

There are 4 people with 2 connections away starting from 4.


Process returned 0 (0x0)   execution time : 2454.441 s
Press any key to continue.
```

Part 2: Maze Runner

Algorithm used for scanning the maze is BFS(breadth first search) ,Then the runner starts to move and check if down is valid he moves down until he finds a block or visited node and marks all previous nodes as visited then he will check if right is valid ,then he will continue until he finds a block or visited node and marks all previous nodes as visited so he will continue checking up and left by the same way till he reaches the destination.

Algorithm used: BFS (breadth first search)

Data structure used: adjacency matrix

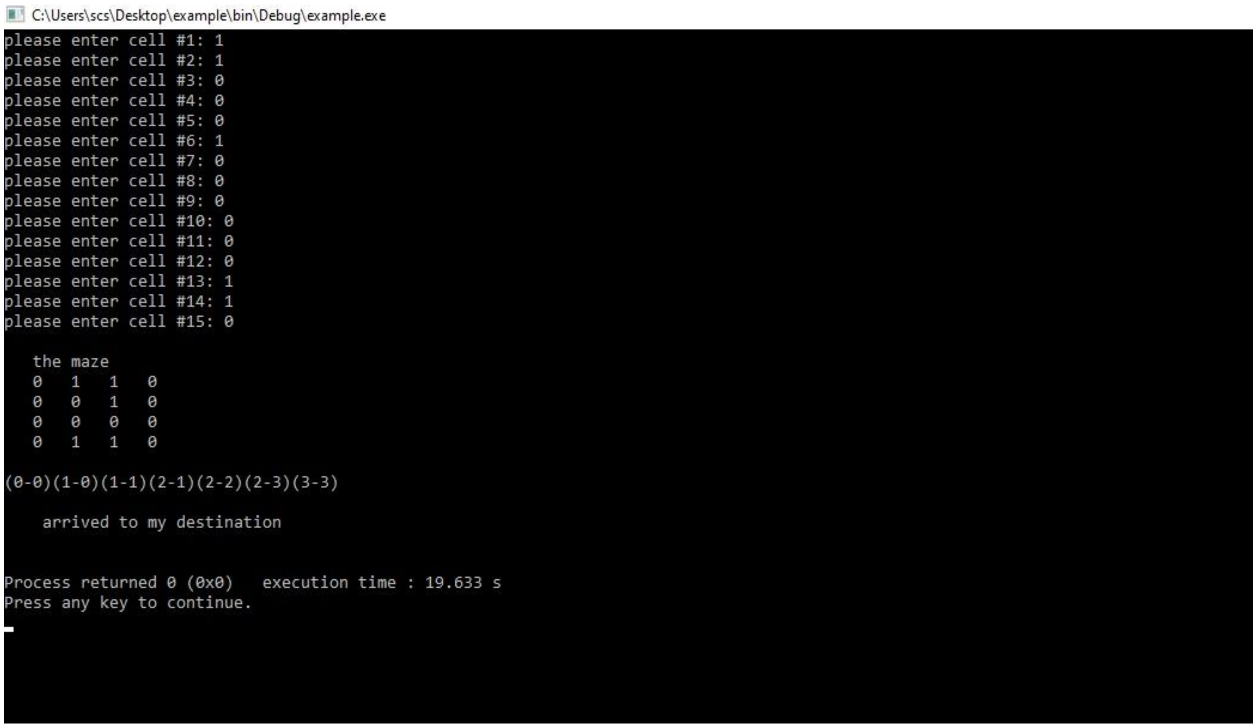
Running time: O(|V|^2|)

Input: -size of maze

-the maze in 1’s and 0’s

Output: the valid paths to escape the maze.

Program:



```
C:\Users\scs\Desktop\example\bin\Debug\example.exe
please enter cell #1: 1
please enter cell #2: 1
please enter cell #3: 0
please enter cell #4: 0
please enter cell #5: 0
please enter cell #6: 1
please enter cell #7: 0
please enter cell #8: 0
please enter cell #9: 0
please enter cell #10: 0
please enter cell #11: 0
please enter cell #12: 0
please enter cell #13: 1
please enter cell #14: 1
please enter cell #15: 0

   the maze
   0   1   1   0
   0   0   1   0
   0   0   0   0
   0   1   1   0

(0-0)(1-0)(1-1)(2-1)(2-2)(2-3)(3-3)

    arrived to my destination


Process returned 0 (0x0)   execution time : 19.633 s
Press any key to continue.
```

Part 3: Shortest Path

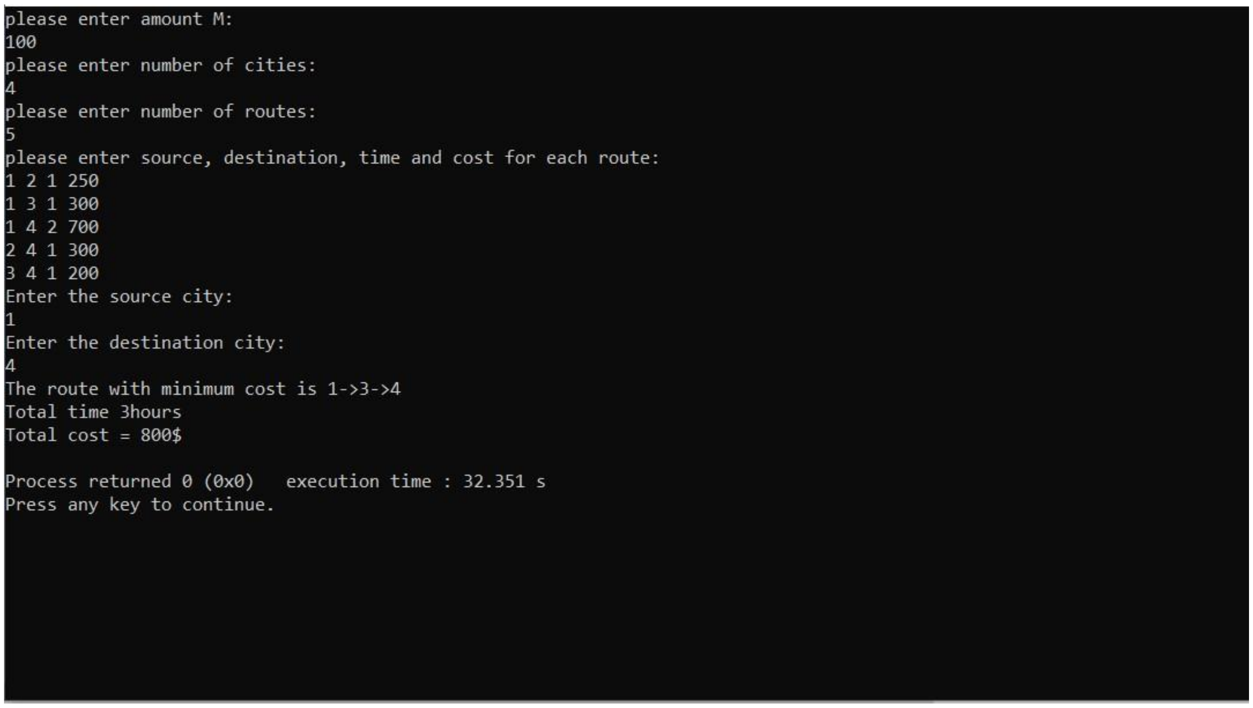
C++ program to implement Dijkstra's algorithm (Shortest path).
It's required to get the path with the minimum cost for the employee during his journey, as the user enters the number of cities and the existing routes to calculate the least cost, and for each hour the employee spends between each city the amount M is added to the final cost.

Algorithm used: Dijkstra's algorithm

Data structure used: adjacency matrix

Running time: O(V (Tins + Tex) + E*Tdec).

Input: - Amount M that he will lose per hour. (m)
- Number of cities. (vertices)
- Number of existing routes. (edges)
- Cost and time for each flight between two cities. (cost, time) - Source and destination Cities. (source, destination) Output:
The route with minimum cost and time.
Program:



```
please enter amount M:
100
please enter number of cities:
4
please enter number of routes:
5
please enter source, destination, time and cost for each route:
1 2 1 250
1 3 1 300
1 4 2 700
2 4 1 300
3 4 1 200
Enter the source city:
1
Enter the destination city:
4
The route with minimum cost is 1->3->4
Total time 3hours
Total cost = 800$

Process returned 0 (0x0)   execution time : 32.351 s
Press any key to continue.
```