**PROJECT ABSTRACT**

**Advanced Data structures Laboratory**

**SHORTEST PATH FINDER IN A MAP**

Swaminathan Navinashok

2019115126

The aim of this project is to develop an application that could find a shortest path in a user input text file (typically a hardcoded and formatted map-like structure) through show casing

**CONCEPTS USED**

1. OOPS concepts
2. Graph algorithms
   * 1. Breadth first search
     2. Depth First Search
     3. Dijkstra’s algorithm
3. STL
   * 1. Vectors
     2. Priority queues
     3. Stacks
     4. Queues

The project finds the shortest path between two nodes of user choice through an algorithm of user choice and prints the path to follow (and also the iterations of the algorithm for verification purposes)

**Classes used**

**Location :**

An Object of type Location for every entry in the input matrix is preserved

Every such object has:

* + 1. name
    2. coordinate in the matrix
    3. parent’s coordinates in the matrix
    4. whether or not it has a neighbor at left, right , above or below
    5. whether or not the node has been visited during the search algorithm
    6. set and get functions for the above
    7. overloaded << operator

**Map :**

An object of type map has/stores :

1. a 2d vector of location objects called locations
2. a user input start and end vertex location objects and their cordinates
3. whether or not a map has been loaded
4. whether or not a path has been found
5. a constructor that takes a filename as input and initializes the locations 2d vector
6. Set and get functions for the above
7. Overloaded () operator that returns a location object at the coordinates set by the arguments

**Functions Used**

**printMap :**

Prints the map

**PrintCoordinates :**

Prints the path to go from start to end vertex by mentioning the coordinates of each step of the path

**Success:**

Sets a path trailer from start to end if path was found

**SearchPath:**

Templated function to search for path in BFS as well as DFS using a queue or stack of locations as a template parameter and calls **Success** if a path is found

**searchDjikstra:**

Searches for a path using dijiksta’s algorithm and calls

successDjikstra if successful

**successDjikstra:**

Sets a path trailer from start to end if path was found and computes total cost for traversal

**optionsMenu:**

To print menu and input choice