Contents

[Maxim Carr 2021 NEA – London Underground Route Planner 2](#_Toc67094117)

[Analysis 2](#_Toc67094118)

[Introduction 2](#_Toc67094119)

[Researching the Problem 2](#_Toc67094120)

[Objectives 7](#_Toc67094121)

[Documented Design 8](#_Toc67094122)

[Data – Station Names 8](#_Toc67094123)

[Data – Variables 8](#_Toc67094124)

[Annotated UI 9](#_Toc67094125)

[Data Structures 13](#_Toc67094126)

[Algorithms 14](#_Toc67094127)

[File IO 16](#_Toc67094128)

[Source Code 20](#_Toc67094129)

[Program.cs 20](#_Toc67094130)

[buildTubeMap.cs 20](#_Toc67094131)

[startScreen.cs 28](#_Toc67094132)

[Autogen Code for startScreen.cs 28](#_Toc67094133)

[mapScreen.cs 30](#_Toc67094134)

[dijkstraAlgorithm.cs 32](#_Toc67094135)

[display.cs 33](#_Toc67094136)

[Autogen code for mapScreen.cs 34](#_Toc67094137)

[PriorityQueue.cs 39](#_Toc67094138)

[loadFileForm.cs 40](#_Toc67094139)

[Autogen Code for loadFileForm.cs 41](#_Toc67094140)

[saveFileForm.cs 43](#_Toc67094141)

[Autogen Code for saveFileForm.cs 43](#_Toc67094142)

[Testing 45](#_Toc67094143)

[Tests 45](#_Toc67094144)

[Proof of Testing 49](#_Toc67094145)

[Evaluation 60](#_Toc67094146)

[End User Feedback 60](#_Toc67094147)

[Status of Analysis Objectives 61](#_Toc67094148)

[Evaluation based on Objectives and User Feedback 62](#_Toc67094149)

[References 63](#_Toc67094150)

# Maxim Carr 2021 NEA – London Underground Route Planner

## Analysis

### Introduction

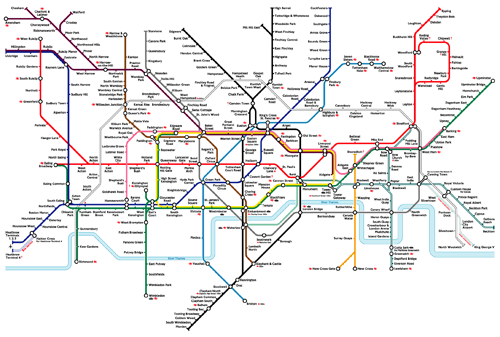
The London Underground is a public transport network which serves 5 million people per day. The network has 270 stations connected by 11 lines, not including the DLR and Overground services. This quantity of stations, along with the complexity of the lines connecting them, can make it difficult for users of the service to find the shortest route between their location and destination. The optimal route between two stations would wave as few intermediate stations as possible while also taking the least amount of time to traverse.

Figure : Map of the London Underground

Ben is an ideal end user of an Underground route-finding programme as he uses the London Underground during his visits to the city. He has asked to stay semi-anonymous for this project and as such will be referred to by his first name only.

### Researching the Problem

#### Abstraction of the Problem

The Underground can be represented as an undirected, weighted graph. Each station would be represented as a vertex and the edges connecting them will have a weight equal to the time taken to travel between the two adjacent vertices. The programme will have to traverse this graph and find the shortest route between two vertices.

#### Interview with end user

Interviewing Ben provided useful insights into what a potential end user would want from a route-finding solution. For the interview, Ben was asked a series of questions on what he expected from an Underground route-finding application.

First, Ben was asked what aspect of a route-finding solution he found most important. He responded, saying that the most important thing for the solution would be that the route generated would be the shortest possible in terms of travel time.

Ben was then asked if he had any preferences on how the user interface looked. His response was that a large portion of the screen should be taken up with an image of the Underground map. He also said that he would prefer it if the generated route was represented graphically on the map.

The user was then asked if he had any platform preference for the solution to run on. He normally plans his routes prior to leaving home when he is visiting London, he said, meaning that he did not have strong feelings towards a certain platform. He did add that he slightly preferred using his laptop over his phone as the larger screen makes it easier to visualise the route.

Finally, Ben was asked if he had any further remarks to make about the project, such as features. One feature he suggested was a function that allows users of the programme to save routes along with the ability to load and view those routes in the future.

#### Existing System Analysis

The need for a navigation system for the Underground has existed for nearly as long as the network has itself, there are many existing route planners and navigation programmes. Ben was asked to use one of these systems – Transport for London’s journey planner (Transport for London, n.d.) – and report how what he thought about the system. TFL’s journey planner was chosen as it is made by the same organisation that operates the Underground itself.

After entering several routes with random endpoints, along with some that he frequently travelled, Ben stated his findings about the existing route finder. These were:

* The route generated seemed to be accurate.
* TFL’s system did not have a map of the Underground and even when he clicked the “Map View” button, it showed the route on a standard map of London.
* The journey planner included methods of transportation other than the Underground. This was not important to Ben though, as he was only interested in finding a route using the Underground.
* By chance he found that the route generated by the journey planner was not consistent.

He was then asked what he would like to see changed as a user of the transport system. Ben wanted to have a consistent route generated by the programme, as it did not matter to his whether a one route was slightly faster than another. He then reiterated his earlier point on having an image of the Underground map, and for the resultant route to be displayed on it.

#### Graph Traversal Research

As stated in the abstraction of the problem, the programme will have to traverse the graph that represents the Underground. The two algorithms that were most mentioned on the Internet for this purpose were Dijkstra’s Algorithm (Brilliant, n.d.) and Bellman Ford’s Algorithm (Brilliant, n.d.).

The main difference between these algorithms is that Bellman Ford’s Algorithm can traverse graphs that contain edges with negative weights. This is not a consideration for traversing the Underground as the edges will be representing tunnels between stations whose weights would be the time taken to travel through said tunnel. Time taken to travel through a tunnel is never going to be a negative as you cannot have negative time, therefore no edges on the graph will have a weight < 0. In addition, Dijkstra’s Algorithm is less time consuming than Bellman Ford’s (Geeks For Geeks, 2020). Due to these two factors, Dijkstra’s Algorithm will be used for this programme.

#### Pseudocode

Below is the pseudocode for Dijkstra’s Algorithm as found on the Brilliant website (Brilliant, n.d.).



Figure : Dijkstra's Algorithm Pseudocode

#### Modelling of Dijkstra’s Algorithm as a Flowchart

Using the pseudocode found in the Brilliant article on Dijkstra’s Algorithm (Brilliant, n.d.), this flowchart for the algorithm was created to ease design and programming.

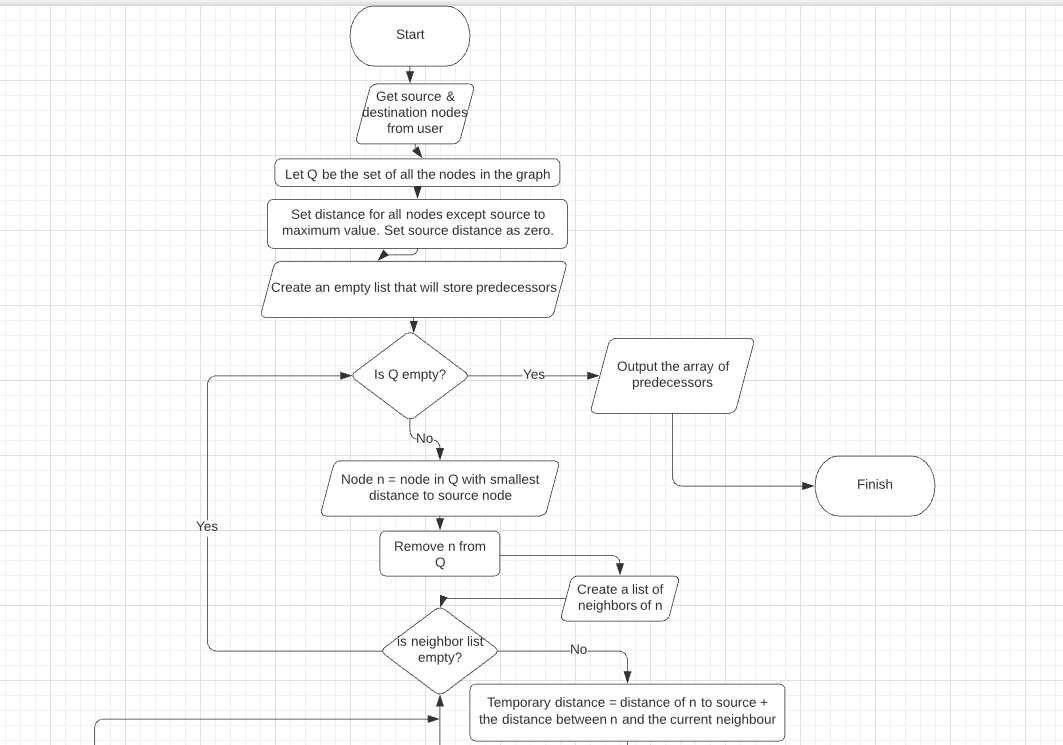
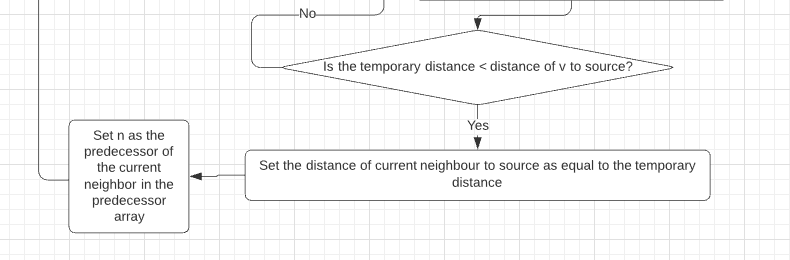


Figure : A flowchart for Dijkstra's Algorithm

#### Preliminary UI Idea

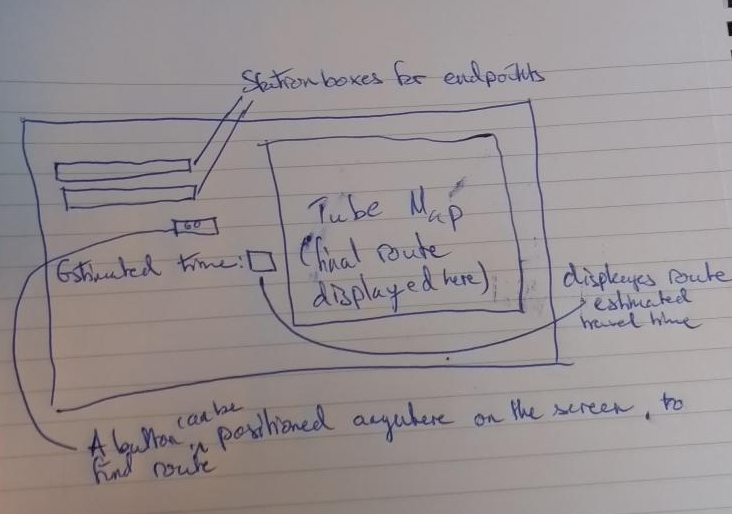


Figure : An initial sketch of what the UI could look like, created with the supervision of the end user.

#### Dijkstra Data Dictionary

Using the Dijkstra pseudocode, this initial data dictionary was created. It contains the variables that are used as part of Dijkstra’s algorithm. The whole of the Underground network has too many nodes to be used as the graph example here, so this Dijkstra data dictionary will use a graph with five nodes defined as 1, 2, 3, 4 and 5.

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Variable | Type | Description | Example |
| Graph | Graph | The adjacency list or matrix that represents the graph. | 1 2 3 4 5  1 0 2 4 0 1  2 2 0 7 1 4  3 4 7 0 2 6  4 0 1 2 0 8  5 1 4 6 8 0 |
| Source | int | The source node’s identifier. | 1 |
| V | int | A node in the graph | 5 |
| dist | int array | An array of distances from the source node | It is originally represented as {0, +infinity, +infinity, +infinity, +infinity}  When taking 1 as the source node. |
| Q | Sorted List | A list of nodes which is ordered by the node’s distance from the source node. | Q can originally be ordered as {1,4,5,3,2} all nodes but 1 have a distance of positive infinity. Node 1 has a distance of 0 taking node 1 as the source node. |
| U | int | A node that is a neighbour of another node V | 3 |
| alt | int | The sum of the distance from node V to source and the distance between node V and node U | For V = 2, U = 5 & source = 1, alt = 2+4 = 6. |

### Objectives

1. Objectives to do with the Route & Stations:
   1. The system must use Dijkstra’s Algorithm to find the shortest path between two user-defined stations.
   2. The user must be able to enter any of the stations shown on the system’s map.
   3. The route generated must connect the two endpoints provided by the user.
      1. This route must be the shortest possible in terms of travel time.
   4. The order that the stations are entered in must not impact the route generated.
   5. The route must be consistent - each pair of endpoints produces the same route each time they are used together.
   6. Invalid stations & routes must be caught and dealt safely.
      1. The system must not attempt to find a route between a station and itself.
      2. The system must not attempt to find a route between anything and a blank input.
      3. The system must not attempt to find a route when any number of the input stations do not exist on its Underground map.
      4. All of the above conditions should provide an error message to the user informing them what went wrong.
   7. The estimated time of the route must be the sum of the edge weights of the route.
   8. Stations must be able to be entered in both capital and lowercase.
2. Objectives to do with File Management:
   1. The user must be able to store a generated route with a user-specified filename.
   2. The user must be able to recover and view stored routes by accessing them with the name they entered earlier.
   3. The load and save route functions must deal with any user error such as an invalid filename robustly proceeding to generate an error message that informs the user what happened.
   4. Cancelling current file operation.
      1. The user should be able to change their mind and not store a route.
      2. The user should also not be forced to load a route once the load menu is opened.

## Documented Design

### Data – Station Names

Creating a list of station names that the programme would use was extremely important to the project as without one, it would not know what to do with any stations the user entered into the system. The list was created by noting each station down from Figure 1. The list was then saved as a text file, sorted into alphabetical order using a simple piece of code and saved into a different text file. The list contained within the last text file was the one used in the system under the name stations.

### Data – Variables

This table contains important variables used in the system.

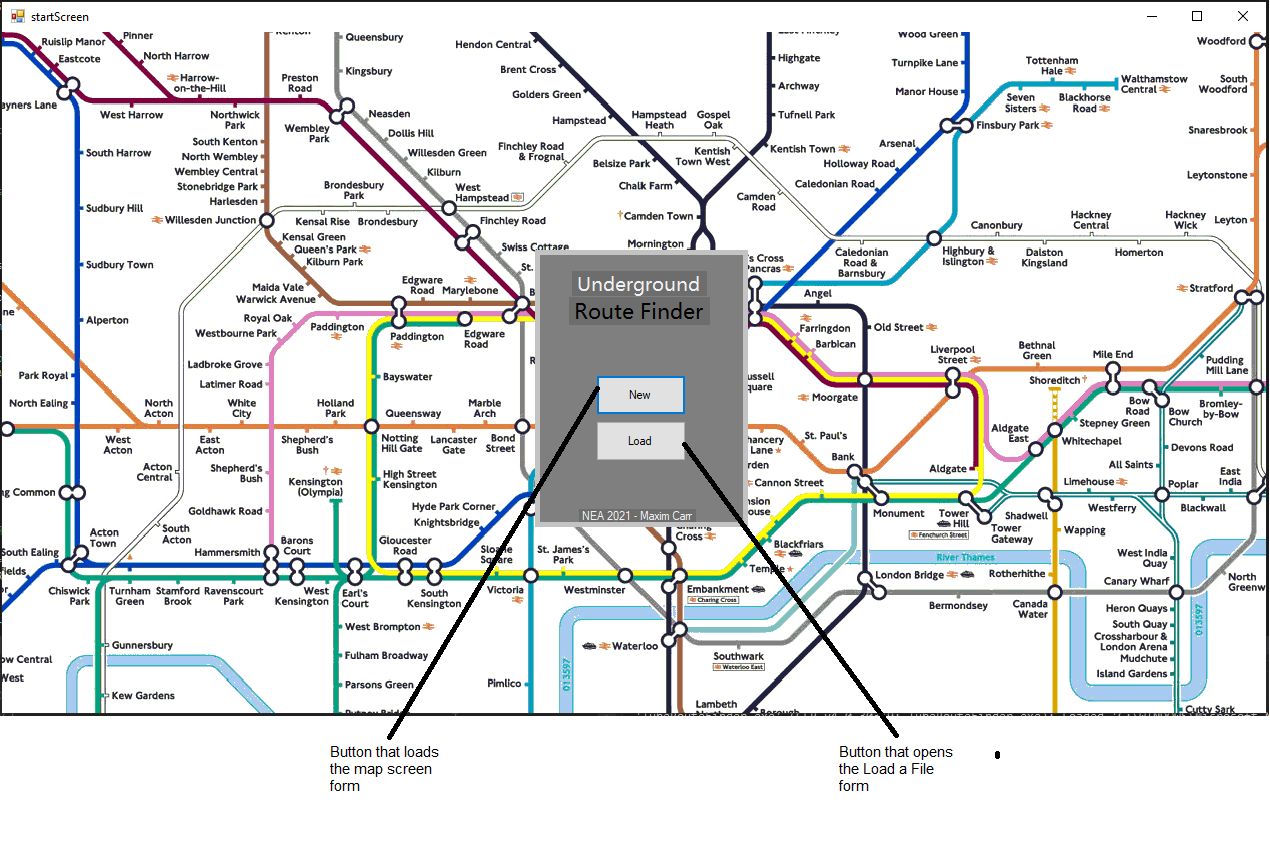
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | | Description | Usage |
| stations | String[] | | A list of station names. | Converting stations to and from their integer IDs. |
| sourceStation | String | | The string that the user inputs into the Source Station text box. | To be converted into a corresponding integer ID for use with the Dijkstra algorithm. |
| destStation | String | | The string that the user enters into the Destination Station text box. | To be converted into a corresponding integer ID for use with the Dijkstra algorithm. |
| sourceStationID | Integer | | The ID of the source station. This is generated by finding the position of sourceStation in the Stations list. | Easier and less error-prone way to refer to stations in the graph is to give them a numerical ID. This ID will be used for graph construction & Dijkstra. |
| destStationID | Integer | | The ID of the destination station. This is generated by finding the position of destStation in the Stations list. | Easier and less error-prone way to refer to stations in the graph is to give them a numerical ID. This ID will be used for graph construction & Dijkstra. |
| currentStation | Integer | | The numerical ID of the current station. | Used in Dijkstra’s Algorithm as a temporary storage while this station’s neighbours are compared for shortest route back to source. |
| neighbour | Integer | | The numerical ID of a neighbour of currentStation. | Used in Dijkstra’s algorithm and is enqueued to routeQueue. |
| Adjacency | List<  List<int>> | | Adjacency list which stores the neighbours of a station. | Dijkstra algorithm needs to know what stations are connected to the current station.  Adjacency[currentStation][neighborID] |
| distanceList | List<  List<int>> | | List similar to the one above, but has distances in an easy to access way once the neighbour is known. | Algorithm needs to know the cost of an edge.  distanceList[currentStation][neighborID+1] = distance between station and neighbour. |
| distanceArray | Int[] | | The members of this array are the distance between the station distanceArray[stationID] and the source station. | Used to find the current shortest route & gives the final estimated time. |
| visited | bool[] | | Storage for visited stations. | Stops the route from doubling back on itself. |
| predecessorArray | | Int[] | Storage of predecessors of current station | Used to build route from destination to source. |
| routeQueue | Priority  Queue\* | | Priority queue of stations that is sorted by ascending distance from source station | Used to store and return stations in the correct order for Dijkstra’s Algorithm |
| node | Subclass/ member for route Queue | | Member for routeQueue & will have two values – a station ID & the distance from source node | As a part of routeQueue. Will be ordered based on distance from source node. |
| path | List<int> | | List for destination-to-source shortest path | To be displayed on the map of the Underground. |
| pathOfStationNames | String[] | | Final path in a form that the user can read it | Display the path in a form that the user can easily read. |

\*Priority Queues are not part of .NET by default, so one will be created using a sorted list.

### Annotated UI

#### Start-up Menu

The start-up menu is a simple feature that allows users to choose to either create a new route or load one that they had previously saved.

****

#### Map Form

The Map Form is the main form of the system. It contains the input boxes, the tube map, the Error and Estimated Time labels and the button that creates the event to run Dijkstra’s algorithm.

This image has been rotated 90 degrees to the left in order to make the annotations readable.

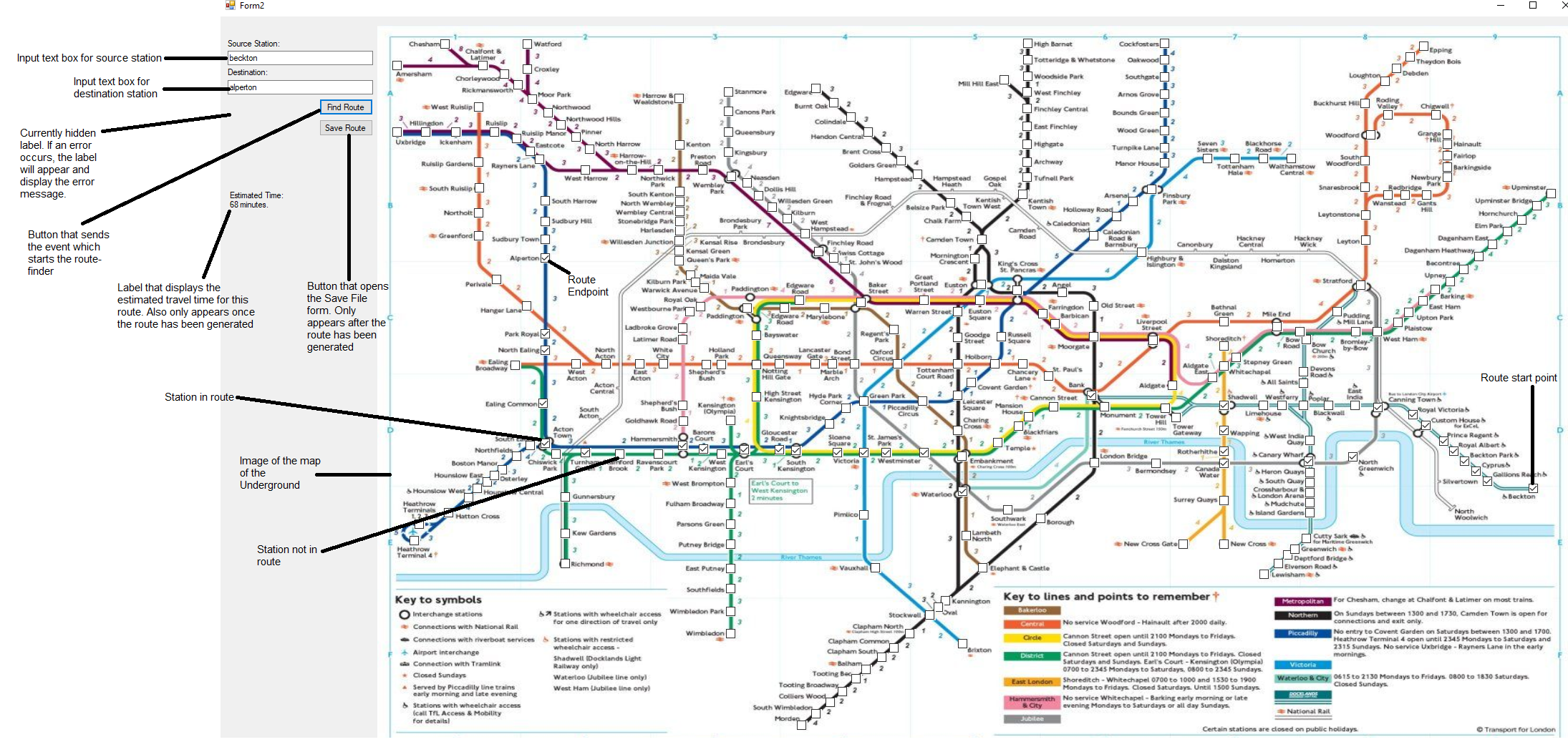


Figure : An annotated image of the Map Screen. An example route has been generated to show certain features

##### Station Marker

The station marker is one of the most important aspects of the UI. It is though these markers that the route is displayed on the map. It is vital that a user of the system can differentiate between stations that are part of the route and those that are not. Therefore, Windows Forms checkboxes are used to represent stations. Not only do they look different in their checked and unchecked states, but they can easily be toggled between these states in code. Also, checkboxes are easily tied to station IDs.



Figure : A station that is not part of a route.



Figure : A station that is part of a route

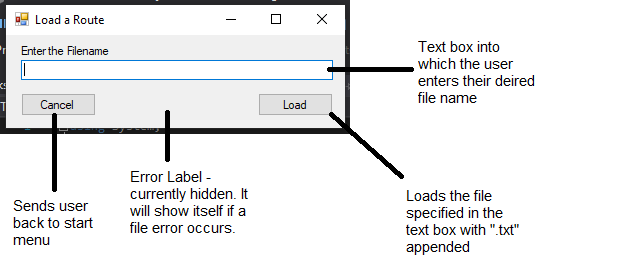


Figure : A checkbox ID. The number at the end is also the ID of the check boxes’ station

When a route is generated, the first thing the system does is clear all previous checkboxes

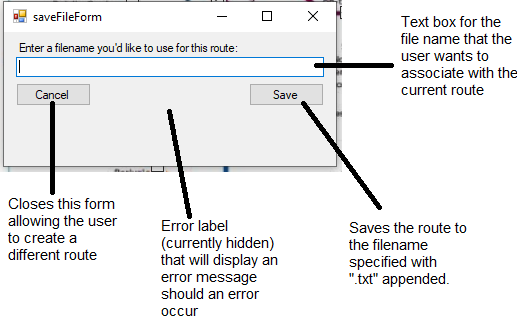
#### Load Route

This form allows the user to enter the name of a file that contains a route that already exists. The contents of this file are then sent to the Map Screen to be displayed.



#### Save Route

This form allows users of the system to enter a filename. This filename is then used to create a text file that stores information about the route.



### Data Structures

#### Priority Queue

For Dijkstra’s Algorithm to be able to find the shortest route between two nodes in a weighted graph, it needs to be able to select the node with the lowest distance from itself to the source node. One way of doing this is to enqueue vertices that are being used into a priority queue.

Each station has two attributes: a station ID and a distance back to the source node, therefore the members of the priority queue are classes with the same attributes.

Every time a station is enqueued or dequeued from the priority queue, it will sort itself in ascending order of distance to source node. The splitting of a member node into a station ID and a priority makes it simple to order the list as the station ID can be ignored during the sorting process except when swapping nodes in the list. The sorting algorithm used to accomplish this is discussed in the Algorithms subsection.

#### Graph Representation

The London Underground is to be represented as a weighted graph in this system as per the Abstraction of the Problem subsection of the Analysis. An adjacency list is being used to accomplish this, along with a separate list for edge weights. This approach was chosen as integer IDs were already being used for the stations and each ID corresponds with a list of neighbours for that node. For example: adjacencyList[0] returns the list of neighbours for the station that has a stationID of 0, which is Acton Town.

The distance list is similar to the adjacency list but is only used to find the cost/distance of a connection. The distance is inserted after the neighbour allowing the distance to be returned. For example, Acton Town (stationID = 0) connects to Chiswick Park (stationID = 51). distancesList[0][(position of 51)+1] returns the time cost of travelling between Acton Town and Chiswick Park which is 2 minutes.

#### Other Data Structures

Lists and arrays are both used many times in the system. Lists are used for collections of data that have a variable size while arrays are used for collections whose sizes are constant and already known.

### Algorithms

#### Dijkstra’s Shortest Path Algorithm



Figure : The same image of the pseudocode for Dijkstra’s algorithm that was used in the Analysis section

Although this algorithm was already discussed in the Analysis section, several things must be altered for it to be used with the rest of the system. To start, ‘v’ and ‘u’ are not helpful variable names, so they have been replaced with ‘currentStation’ and ‘neighbour’ respectively. To reduce memory constraints, Q in the pseudocode above has been replaced with ‘routeQueue’ – a priority queue - and only encountered nodes are enqueued to it. It still returns the same route, but with a smaller queue. These changes, along with the implementation of data structures from the Data Structures section have altered the algorithm to the point that new pseudocode shall be more useful than what can be found in Figure 9.

##### New Pseudocode for Dijkstra’s Algorithm

function dijkstra(adjacencyList, distancesList, sourceStation, destinationStation, stationList)

new bool array visited of length = number of stations

new int array predecessors of length = number of stations

new int array distancesArray of length = number of stations

new PriorityQueue routeQueue

for each stationID in stationList

visited[stationID] = false

predecessors[stationID] = positive Infinity

distancesArray = positive Infinity

visited[sourceStation] = true

distancesArray[sourceStation] = 0

add sourceStation to routeQueue with priority 0

while routeQueue is not empty

new integer currentStation = station in routeQueue with lowest priority

for each neighbour of currentStation

if visited[neighbour] is false

new integer connectionDistance = distance between currentStation and neighbour

visited[neighbour] = true

distance[neighbour] = distance[currentStation] + connectionDistance

predecessor[neighbour] = currentStation

add neighbour to routeQueue with priority of distance[neighbour]

if neighbour is destinationStation

return predecessors

return distance[neighbour]

return Not Connected

#### Algorithm for Priority Queue

The main algorithm that is used as part of a priority queue is a sorting algorithm. This is to keep the list that holds the priority queue in the correct order. For this system, node members of the priority queue must be ordered in ascending order based on their distance from the source node.

Insertion sort was used in this case. The queue does not seem to grow large enough that insertion sort becomes slow. In addition, the queue’s list will not grow quickly and will be sorted after each dequeue and enqueue operation meaning that few swaps will occur each time the sorting algorithm is used. Pseudocode has been created for use in the technical solution.

#### Pseudocode for Insertion Sort

function insertionSort

new int positionInQueue

new member of queue (node) = nodeToInsert

for each point in PriorityQueue

nodeToInsert = PriorityQueue[point]

positionInQueue = point

while((positionInQueue is greater than 0) AND (PriorityQueue[positionInQueue-1] is greater than nodeToInsert.priority))

PriorityQueue[positionInQueue] = PriorityQueue[positionInQueue-1]

Decrement positionInQueue by 1

PriorityQueue[positionInQueue] = nodeToInsert

### File IO

#### Writing a Route to and Reading a Route from a File

One of the objectives of this system, as stated in the Analysis section, is that users of the system can store generated routes in a file.

The main challenge of storing the route as a file is that routes can get quite long. In addition, routes are made up of integer station IDs when they are originally outputted from the Dijkstra algorithm. The conversion of up to as many as fifty IDs to stations is inefficient. It is even more so due to the stations having to be converted back to IDs from names when the route is loaded in the future. This method also resulted in large files that take time to perform operations on.

This problem was solved by realising that there was no need to store the entirety of the route in the text file as the whole route was not used by the system. Instead, only the source and destination station names are now written to the text file. This produced the same route when reading the route from the file as when the whole path was stored as well as faster file operation speed. By cutting the file size drastically while also being able to recover the original route, this size reduction has in effect losslessly compressed the saved routes of the system.

#### Security and Integrity

When files are stored on a computer system, it is important to consider security and integrity of information.

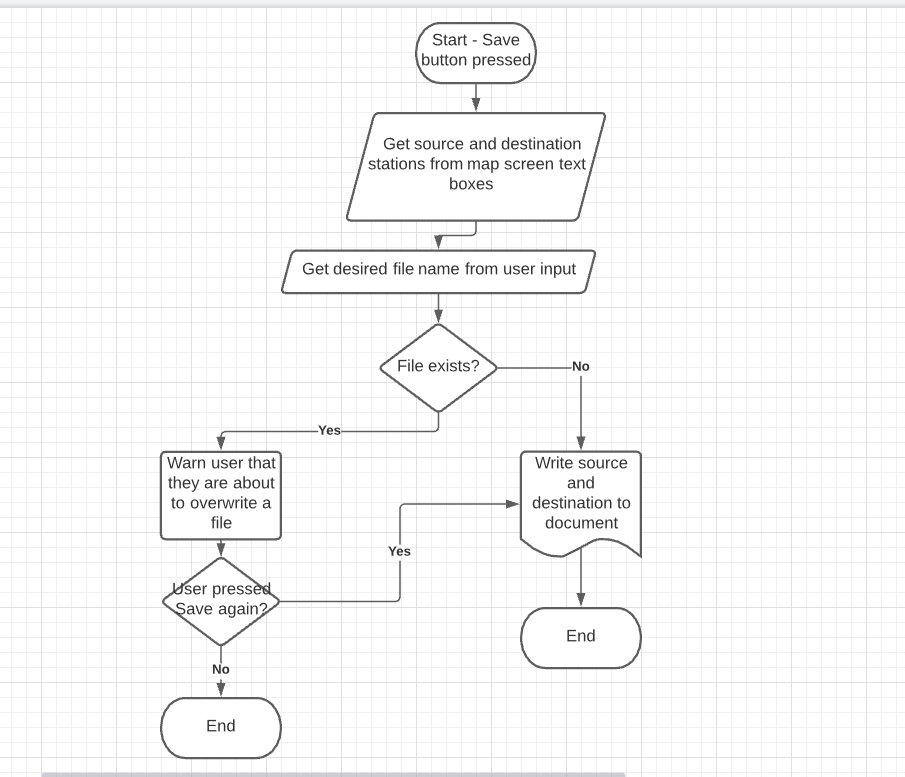
With this system, security is not a major concern. No sensitive data such as names or other personal information is stored in the files produced by this system. All that is stored in the documents generated by the programme are the endpoints of a route that hundreds of thousands, if not millions, of people take each day. Therefore, these files pose a minimal security risk and can safely be stored as text files.

Integrity does impact on how this system will operate. However, if a file is tampered with or is corrupted, the worst thing that can happen is that the user will see an invalid route message instead of a route when attempting to load a file. The user can just amend the incorrect field’s value back to what they wanted it to be in the first place. The route storage feature is more for convenience than for replacing the user’s memory of where they need to go and as such file integrity is not a major issue for this system as well.

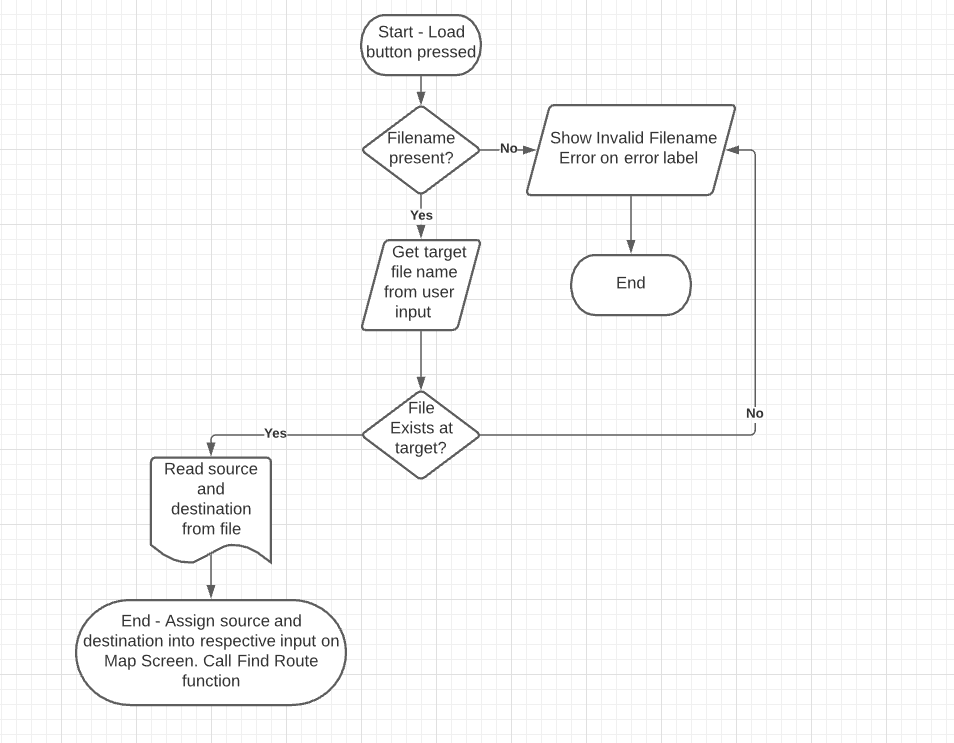
#### Full System Flowchart

##### Dijkstra for the London Underground flowchart

##### Flowchart for File Saving System



##### Flowchart for File Loading System



## Source Code

This section contains the system’s final source code.

### Program.cs

namespace TubeRouteFinder

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new startScreen());

}

}

}

### buildTubeMap.cs

class tubeBuild

{

static void addEdge(List<List<int>> adj, int i, int j, int k)

{

//bidirectional

//Stores edges in adjacency list

adj[i].Add(j);

adj[j].Add(i);

//Stores distances between source & neighbor in separate distances list.

mapScreen.distanceList[i].Add(j); mapScreen.distanceList[i].Add(k);

mapScreen.distanceList[j].Add(i); mapScreen.distanceList[j].Add(k);

// i= station, j = station, k = time between stations in minutes

}

public void buildGraph(List<List<int>> adjStations)//Constructs graph and stores it in adjacency list.

{

//The variables are : adjacency list, source station, neighbor, time between statios

//A

addEdge(adjStations, 0, 72, 2); addEdge(adjStations, 0, 229, 3); addEdge(adjStations, 0, 263, 3); addEdge(adjStations, 0, 51, 2);

addEdge(adjStations, 1, 155, 2); addEdge(adjStations, 1, 261, 4);

addEdge(adjStations, 2, 261, 2); addEdge(adjStations, 2, 294, 2); addEdge(adjStations, 2, 155, 4);

addEdge(adjStations, 3, 69, 2); addEdge(adjStations, 3, 199, 2);

addEdge(adjStations, 4, 192, 2); addEdge(adjStations, 4, 250, 3);

addEdge(adjStations, 5, 45, 4);

addEdge(adjStations, 6, 143, 2); addEdge(adjStations, 6, 187, 3);

addEdge(adjStations, 7, 123, 2); addEdge(adjStations, 7, 262, 1);

addEdge(adjStations, 8, 30, 3); addEdge(adjStations, 8, 238, 3);

addEdge(adjStations, 9, 94, 1); addEdge(adjStations, 9, 127, 2);

//#A

//B

addEdge(adjStations, 10, 162, 1); addEdge(adjStations, 10, 210, 1); addEdge(adjStations, 10, 82, 3); addEdge(adjStations, 10, 241, 4); addEdge(adjStations, 10, 103, 3); addEdge(adjStations, 10, 27, 2); addEdge(adjStations, 10, 93, 6);

addEdge(adjStations, 11, 55, 2); addEdge(adjStations, 11, 255, 1);

addEdge(adjStations, 12, 155, 2); addEdge(adjStations, 12, 242, 2); addEdge(adjStations, 12, 223, 2); addEdge(adjStations, 12, 156, 2); addEdge(adjStations, 12, 167, 3); addEdge(adjStations, 12, 277, 4);

addEdge(adjStations, 13, 91, 1); addEdge(adjStations, 13, 167, 2);

addEdge(adjStations, 14, 76, 2); addEdge(adjStations, 14, 267, 4);

addEdge(adjStations, 15, 90, 2); addEdge(adjStations, 15, 174, 2);

addEdge(adjStations, 16, 109, 2); addEdge(adjStations, 16, 288, 2); addEdge(adjStations, 16, 73, 3);

addEdge(adjStations, 17, 185, 2); addEdge(adjStations, 17, 191, 2);

addEdge(adjStations, 18, 96, 4);

addEdge(adjStations, 19, 64, 2); addEdge(adjStations, 19, 215, 3);

addEdge(adjStations, 20, 66, 3); addEdge(adjStations, 20, 267, 2);

addEdge(adjStations, 21, 46, 2); addEdge(adjStations, 21, 110, 2);

addEdge(adjStations, 22, 40, 2); addEdge(adjStations, 22, 156, 3);

addEdge(adjStations, 23, 155, 3); addEdge(adjStations, 23, 163, 2);

addEdge(adjStations, 24, 160, 1); addEdge(adjStations, 24, 253, 2);

addEdge(adjStations, 25, 258, 2); addEdge(adjStations, 25, 272, 2);

addEdge(adjStations, 26, 77, 3); addEdge(adjStations, 26, 199, 3);

addEdge(adjStations, 27, 161, 1); addEdge(adjStations, 27, 190, 1); addEdge(adjStations, 27, 104, 2);

addEdge(adjStations, 28, 83, 1); addEdge(adjStations, 28, 156, 2);

addEdge(adjStations, 29, 180, 2); addEdge(adjStations, 29, 188, 3);

addEdge(adjStations, 30, 299, 2);

addEdge(adjStations, 31, 69, 1); addEdge(adjStations, 31, 202, 1);

addEdge(adjStations, 32, 35, 2); addEdge(adjStations, 32, 163, 1);

addEdge(adjStations, 33, 99, 3); addEdge(adjStations, 33, 118, 2);

addEdge(adjStations, 34, 246, 2);

addEdge(adjStations, 35, 284, 2);

addEdge(adjStations, 36, 157, 3); addEdge(adjStations, 36, 300, 3);

addEdge(adjStations, 37, 57, 2); addEdge(adjStations, 37, 80, 3);

//#B

//C

addEdge(adjStations, 38, 127, 2); addEdge(adjStations, 38, 143, 5);

addEdge(adjStations, 39, 46, 2); addEdge(adjStations, 39, 88, 3); addEdge(adjStations, 39, 138, 2); addEdge(adjStations, 39, 169, 1);

addEdge(adjStations, 40, 214, 1); addEdge(adjStations, 40, 251, 2); addEdge(adjStations, 40, 41, 3);

addEdge(adjStations, 41, 119, 1); addEdge(adjStations, 41, 287, 2); addEdge(adjStations, 41, 177, 3);

addEdge(adjStations, 42, 77, 2); addEdge(adjStations, 42, 217, 3); addEdge(adjStations, 42, 177, 3); addEdge(adjStations, 42, 284, 3);

addEdge(adjStations, 43, 160, 1); addEdge(adjStations, 43, 165, 2);

addEdge(adjStations, 44, 205, 2); addEdge(adjStations, 44, 244, 2);

addEdge(adjStations, 45, 49, 8); addEdge(adjStations, 45, 52, 4);

//addEdge(adjStations, 46, x); Just here to keep count - all neighbours of 46 already counted

addEdge(adjStations, 47, 125, 1); addEdge(adjStations, 47, 242, 2);

addEdge(adjStations, 48, 86, 1); addEdge(adjStations, 48, 195, 2); addEdge(adjStations, 48, 150, 2);

//addEdge(adjStations, 49, x); All 49 neighbors have been counted

addEdge(adjStations, 50, 102, 2); addEdge(adjStations, 50, 213, 2);

addEdge(adjStations, 51, 263, 2);

addEdge(adjStations, 52, 212, 4);

addEdge(adjStations, 53, 54, 2); addEdge(adjStations, 53, 55, 2);

addEdge(adjStations, 54, 246, 1);

//55

addEdge(adjStations, 56, 186, 4);

addEdge(adjStations, 57, 118, 2);

addEdge(adjStations, 58, 235, 2); addEdge(adjStations, 58, 256, 2);

addEdge(adjStations, 59, 125, 1); addEdge(adjStations, 59, 150, 1);

addEdge(adjStations, 60, 170, 1); addEdge(adjStations, 60, 233, 1);

addEdge(adjStations, 61, 166, 4); addEdge(adjStations, 61, 278, 3);

addEdge(adjStations, 62, 201, 1); addEdge(adjStations, 62, 217, 1);

addEdge(adjStations, 63, 106, 1); addEdge(adjStations, 63, 134, 1);

addEdge(adjStations, 64, 96, 1);

//#C

//D

addEdge(adjStations, 65, 66, 4); addEdge(adjStations, 65, 84, 3);

//66

addEdge(adjStations, 67, 157, 2); addEdge(adjStations, 67, 254, 3);

addEdge(adjStations, 68, 85, 1); addEdge(adjStations, 68, 106, 2);

//69

addEdge(adjStations, 70, 171, 2); addEdge(adjStations, 70, 295, 2);

//#D

//E

addEdge(adjStations, 71, 72, 4); addEdge(adjStations, 71, 281, 3);

addEdge(adjStations, 72, 176, 3);

addEdge(adjStations, 73, 98, 3); addEdge(adjStations, 73, 121, 3); addEdge(adjStations, 73, 137, 3); addEdge(adjStations, 73, 282, 2); addEdge(adjStations, 73, 288, 1);

addEdge(adjStations, 74, 175, 2); addEdge(adjStations, 74, 293, 3);

addEdge(adjStations, 75, 92, 4); addEdge(adjStations, 75, 123, 2);

addEdge(adjStations, 76, 268, 2);

//77

addEdge(adjStations, 78, 203, 2); addEdge(adjStations, 78, 237, 2);

addEdge(adjStations, 79, 208, 2); addEdge(adjStations, 79, 220, 2);

//80

addEdge(adjStations, 81, 162, 2); addEdge(adjStations, 81, 191, 3);

addEdge(adjStations, 82, 191, 4);

addEdge(adjStations, 83, 135, 3); addEdge(adjStations, 83, 147, 2);

addEdge(adjStations, 84, 128, 2);

addEdge(adjStations, 85, 151, 1);

addEdge(adjStations, 86, 253, 2); addEdge(adjStations, 86, 277, 2); addEdge(adjStations, 86, 292, 2);

addEdge(adjStations, 87, 254, 2);

addEdge(adjStations, 88, 143, 2); addEdge(adjStations, 88, 169, 2); addEdge(adjStations, 88, 275, 1);

addEdge(adjStations, 89, 103, 2); addEdge(adjStations, 89, 143, 2);

//#E

//F

addEdge(adjStations, 90, 108, 2);

addEdge(adjStations, 91, 143, 4);

addEdge(adjStations, 92, 164, 3); addEdge(adjStations, 92, 283, 2);

addEdge(adjStations, 93, 252, 2); addEdge(adjStations, 93, 280, 7); addEdge(adjStations, 93, 285, 1);

addEdge(adjStations, 94, 122, 2); addEdge(adjStations, 94, 159, 2); addEdge(adjStations, 94, 222, 4);

addEdge(adjStations, 95, 193, 2); addEdge(adjStations, 95, 282, 1);

//#F

//G

//96

addEdge(adjStations, 97, 174, 2); addEdge(adjStations, 97, 209, 3);

addEdge(adjStations, 98, 121, 4); addEdge(adjStations, 98, 231, 1);

addEdge(adjStations, 99, 110, 4);

addEdge(adjStations, 100, 109, 2); addEdge(adjStations, 100, 225, 1);

addEdge(adjStations, 101, 257, 1); addEdge(adjStations, 101, 275, 2);

addEdge(adjStations, 102, 108, 3);

//103

addEdge(adjStations, 104, 132, 2); addEdge(adjStations, 104, 190, 2); addEdge(adjStations, 104, 195, 1); addEdge(adjStations, 104, 271, 2); addEdge(adjStations, 104, 292, 3);

addEdge(adjStations, 105, 181, 2); addEdge(adjStations, 105, 194, 2);

//106

addEdge(adjStations, 107, 140, 3); addEdge(adjStations, 107, 263, 3);

//#G

//H

//108

addEdge(adjStations, 109, 207, 2); addEdge(adjStations, 109, 263, 2);

//110

addEdge(adjStations, 111, 175, 3); addEdge(adjStations, 111, 194, 2);

addEdge(adjStations, 112, 247, 2); addEdge(adjStations, 112, 296, 2);

addEdge(adjStations, 113, 139, 3);

addEdge(adjStations, 114, 182, 2); addEdge(adjStations, 114, 286, 2); addEdge(adjStations, 114, 178, 3);

addEdge(adjStations, 115, 116, 3); addEdge(adjStations, 115, 117, 3); addEdge(adjStations, 115, 131, 4);

addEdge(adjStations, 116, 117, 5);

//117

//118

addEdge(adjStations, 119, 233, 1);

addEdge(adjStations, 120, 259, 3);

addEdge(adjStations, 121, 185, 3);

addEdge(adjStations, 122, 143, 4);

//123

addEdge(adjStations, 124, 133, 2); addEdge(adjStations, 124, 269, 3);

addEdge(adjStations, 125, 221, 2); addEdge(adjStations, 125, 257, 2);

addEdge(adjStations, 126, 185, 2); addEdge(adjStations, 126, 224, 1);

//127

addEdge(adjStations, 128, 266, 2);

addEdge(adjStations, 129, 130, 2); addEdge(adjStations, 129, 131, 2);

addEdge(adjStations, 130, 188, 2);

//131

addEdge(adjStations, 132, 145, 2);

//#H

//I

addEdge(adjStations, 133, 218, 3);

addEdge(adjStations, 134, 170, 1);

//#I

//No J names

//K

addEdge(adjStations, 135, 189, 2); addEdge(adjStations, 135, 277, 3);

addEdge(adjStations, 136, 204, 3); addEdge(adjStations, 136, 296, 3);

//137

addEdge(adjStations, 138, 262, 2);

addEdge(adjStations, 139, 232, 2);

addEdge(adjStations, 140, 211, 3);

addEdge(adjStations, 141, 285, 2); addEdge(adjStations, 141, 295, 2);

addEdge(adjStations, 142, 158, 2); addEdge(adjStations, 142, 204, 2);

addEdge(adjStations, 143, 221, 2);

addEdge(adjStations, 144, 205, 2); addEdge(adjStations, 144, 280, 3);

addEdge(adjStations, 145, 231, 3);

//#K

//L

addEdge(adjStations, 146, 149, 1); addEdge(adjStations, 146, 290, 2);

addEdge(adjStations, 147, 277, 1);

addEdge(adjStations, 148, 161, 3); addEdge(adjStations, 148, 206, 1);

addEdge(adjStations, 149, 225, 2);

addEdge(adjStations, 150, 195, 2); addEdge(adjStations, 150, 257, 1);

//151

addEdge(adjStations, 152, 153, 3); addEdge(adjStations, 152, 248, 2);

addEdge(adjStations, 153, 228, 2); addEdge(adjStations, 153, 273, 2);

addEdge(adjStations, 154, 223, 3); addEdge(adjStations, 154, 291, 3);

addEdge(adjStations, 155, 167, 2);

addEdge(adjStations, 156, 239, 2);

//157

//#L

//M

addEdge(adjStations, 158, 276, 1);

addEdge(adjStations, 159, 264, 3);

//160

//161

//162

addEdge(adjStations, 163, 245, 2); addEdge(adjStations, 163, 248, 4);

//164

addEdge(adjStations, 165, 261, 2);

addEdge(adjStations, 166, 183, 3); addEdge(adjStations, 166, 212, 4);

addEdge(adjStations, 167, 187, 1);

addEdge(adjStations, 168, 235, 4);

//169

//170

//#M

//N

addEdge(adjStations, 171, 280, 4);

addEdge(adjStations, 172, 251, 4);

addEdge(adjStations, 173, 251, 4);

//174

addEdge(adjStations, 175, 281, 2);

addEdge(adjStations, 176, 192, 2);

//177

addEdge(adjStations, 178, 197, 3);

addEdge(adjStations, 179, 232, 2); addEdge(adjStations, 179, 279, 2);

addEdge(adjStations, 180, 229, 1);

addEdge(adjStations, 181, 234, 3);

addEdge(adjStations, 182, 200, 2);

addEdge(adjStations, 183, 184, 2);

addEdge(adjStations, 184, 197, 2);

addEdge(adjStations, 185, 206, 2);//Notting Hill Gate

//#N

//O

addEdge(adjStations, 186, 238, 2);//oakwood

//186

//187

//188

addEdge(adjStations, 189, 246, 3);//oval

addEdge(adjStations, 190, 195, 2); addEdge(adjStations, 190, 210, 2); addEdge(adjStations, 190, 257, 2); addEdge(adjStations, 190, 275, 2);//oxford

addEdge(adjStations, 191, 216, 1); addEdge(adjStations, 191, 276, 2);

//192

addEdge(adjStations, 193, 203, 3);

//194

//195

addEdge(adjStations, 196, 270, 1); addEdge(adjStations, 196, 271, 3);

//197

addEdge(adjStations, 198, 268, 2); addEdge(adjStations, 198, 284, 2);

addEdge(adjStations, 199, 287, 1); addEdge(adjStations, 199, 291, 1);

addEdge(adjStations, 200, 280, 3);

addEdge(adjStations, 201, 215, 1);

addEdge(adjStations, 202, 248, 2);

//203,204,205,206

addEdge(adjStations, 207, 243, 2);

addEdge(adjStations, 208, 230, 3); addEdge(adjStations, 208, 286, 3);

addEdge(adjStations, 209, 273, 2);

//210,211,212

addEdge(adjStations, 213, 300, 3);

addEdge(adjStations, 214, 274, 1);

//215

addEdge(adjStations, 216, 290, 2);

//217

addEdge(adjStations, 218, 220, 2);

addEdge(adjStations, 219, 234, 1); addEdge(adjStations, 219, 289, 3);

//220,221

addEdge(adjStations, 222, 258, 3);

addEdge(adjStations, 223, 260, 3); addEdge(adjStations, 223, 274, 1); addEdge(adjStations, 223, 294, 2);

addEdge(adjStations, 224, 293, 3);

//225

addEdge(adjStations, 226, 294, 2);

addEdge(adjStations, 227, 231, 2); addEdge(adjStations, 227, 271, 2);

addEdge(adjStations, 228, 236, 2);

//229

addEdge(adjStations, 230, 249, 2);

//231-235

addEdge(adjStations, 236, 300, 2);

addEdge(adjStations, 237, 298, 3);

//238

addEdge(adjStations, 239, 277, 1);

addEdge(adjStations, 240, 271, 2); addEdge(adjStations, 240, 292, 2);

addEdge(adjStations, 241, 252, 1);

//242

addEdge(adjStations, 243, 263, 1);

//244

addEdge(adjStations, 245, 294, 3);

addEdge(adjStations, 246, 270, 3);

addEdge(adjStations, 247, 279, 3);

addEdge(adjStations, 248, 284, 7);

addEdge(adjStations, 249, 250, 3);

//250

addEdge(adjStations, 255, 256, 2);

//256

addEdge(adjStations, 259, 301, 3);

addEdge(adjStations, 264, 299, 2);

addEdge(adjStations, 265, 266, 3);

addEdge(adjStations, 277, 292, 3);

addEdge(adjStations, 283, 301, 1);

addEdge(adjStations, 287, 291, 2);

addEdge(adjStations, 297, 298, 3);

/\* IGNORE CONTENTS OF COMMENT - REMNANT OF NON-GUI PROTOTYPE

Console.WriteLine("Enter your starting station:\n\n"); //replace in interface with data collection from text box / item tapped (DONE)

string userInputStation1 = Console.ReadLine();

Console.WriteLine("Enter your destination:\n\n"); //replace in interface with data collection from text box / item tapped (DONE)

string userInputStation2 = Console.ReadLine();

tubeTracer.printShortestDistance(adjStations, sourceStation, destStation, stationNumber, stationList, distanceList); //All variables needed for algorithm sent to algorithm class

\*/

}

public void removeStation(string stationName) //Removes a station, if the need arises

{

for (int h = 0; h < stations.Length; h++)

{

if (stations[h] == stationName)

{

stations[h] = "";

}

}

}

public int getStationID(string statName) //Returns numerical IDs for routefinder

{

int statID = 0;

int i = 0;

bool statFound = false;

while (!(statFound))

{

if (statName.ToLower() == stations[i].ToLower())

{

statFound = true;

statID = i;

return statID;

}

else

i++;

}

return statID;

}

public string getStationName(int stationID) //Returns actual station names for people to read

{

return stations[stationID];

}

public string[] stations = new string[]//Collection of stations for the map

{"Acton Town","Aldgate","Aldgate East",

"All Saints","Alperton","Amersham",

"Angel","Archway","Arnos Grove",

"Arsenal","Baker Street","Balham",

"Bank","Barbican","Barking",

"Barkingside", "Barons Court", "Bayswater",

"Beckton", "Beckton Park", "Becontree",

"Belsize Park", "Bermondsey", "Bethnal Green",

"Blackfriars", "Blackhorse Road", "Blackwall",

"Bond Street", "Borough", "Boston Manor",

"Bounds Green", "Bow Church", "Bow Road",

"Brent Cross", "Brixton", "Bromley-By-Bow",

"Buckhurst Hill", "Burnt Oak",

"Caledonian Road", "Camden Town", "Canada Water",

"Canary Wharf", "Canning Town", "Cannon Street",

"Canons Park", "Chalfont & Latimer", "Chalk Farm",

"Chancery Lane", "Charing Cross", "Chesham",

"Chigwell", "Chiswick Park", "Chorleywood",

"Clapham Common", "Clapham North", "Clapham South",

"Cockfosters", "Colindale", "Colliers Wood",

"Covent Garden", "Crossharbour & London Arena", "Croxley",

"Custom House", "Cutty Sark", "Cyprus",

"Dagenham East", "Dagenham Heathway", "Debden",

"Deptford Bridge", "Devons Road", "Dollis Hill",

"Ealing Broadway", "Ealing Common", "Earl's Court",

"East Acton", "East Finchley", "East Ham",

"East India", "East Putney", "Eastcote",

"Edgware", "Edgware Road (B)", "Edgware Road (C)",

"Elephant & Castle", "Elm Park", "Elverson Road",

"Embankment", "Epping", "Euston",

"Euston Square", "Fairlop", "Farringdon",

"Finchley Central", "Finchley Road", "Finsbury Park",

"Fulham Broadway", "Gallions Reach", "Gants Hill",

"Gloucester Road", "Golders Green", "Goldhawk Road",

"Goodge Street", "Grange Hill", "Great Portland Street",

"Green Park", "Greenford", "Greenwich",

"Gunnersbury", "Hainault", "Hammersmith",

"Hampstead", "Hanger Lane", "Harlesden",

"Harrow & Wealdston", "Harrow-on-the-Hill", "Hatton Cross",

"Heathrow Terminal 4", "Heathrow Terminals 1, 2 & 3", "Hendon Central",

"Heron Quays", "High Barnet", "High Street Kensington",

"Highbury & Islington", "Highgate", "Hillingdon",

"Holborn", "Holland Park", "Holloway Road",

"Hornchurch", "Hounslow Central", "Hounslow East",

"Hounslow West", "Hyde Park Corner", "Ickenham",

"Island Gardens", "Kennington", "Kensal Green", "Kensington (Olympia)", "Kentish Town",

"Kenton", "Kew Gardens", "Kilburn",

"Kilburn Park", "King's Cross St. Pancras", "Kingsbury",

"Knightsbridge", "Ladbroke Grove", "Lambeth North","Lancaster Gate",

"Latimer Road", "Leicester Square", "Lewisham",

"Leyton", "Leytonstone", "Limehouse",

"Liverpool Street", "London Bridge", "Loughton",

"Maida Vale", "Manor House", "Mansion House",

"Marble Arch", "Marylebone","Mile End",

"Mill Hill East", "Monument", "Moor Park",

"Moorgate", "Morden", "Mornington Crescent",

"Mudchute", "Neasden", "New Cross",

"New Cross Gate", "Newbury Park", "North Acton",

"North Ealing", "North Greenwich","North Harrow",

"North Wembley", "Northfields", "Northolt",

"Northwick Park", "Northwood", "Northwood Hills",

"Notting Hill Gate", "Oakwood", "Old Street",

"Osterley", "Oval", "Oxford Circus",

"Paddington", "Park Royal", "Parsons Green",

"Perivale", "Picadilly Circus", "Pimlico",

"Pinner", "Plaistow", "Poplar",

"Preston Road", "Prince Regent", "Pudding Mill Lane",

"Putney Bridge", "Queen's Park", "Queensbury",

"Queensway", "Ravenscourt Park", "Rayners Lane",

"Redbridge", "Regent's Park", "Richmond",

"Rickmansworth", "Roding Valley", "Rotherhithe",

"Royal Albert", "Royal Oak", "Royal Victoria",

"Ruislip", "Ruislip Gardens", "Ruislip Manor",

"Russell Square", "Seven Sisters", "Shadwell",

"Shepherd's Bush (C)", "Shepherd's Bush (H)", "Shoreditch",

"Sloane Square", "Snaresbrook", "South Ealing",

"South Harrow", "South Kensington", "South Kenton",

"South Quay", "South Ruislip", "South Wimbledon",

"South Woodford",

"Southfields", "Southgate", "Southwark",

"St. James's Park", "St. John's Wood", "St. Paul's",

"Stamford Brook", "Stanmore", "Stepney Green",

"Stockwell", "Stonebridge Park", "Stratford",

"Sudbury Hill", "Sudbury Town", "Surrey Quays",

"Swiss Cottage", "Temple", "Theydon Bois",

"Tooting Bec", "Tooting Broadway", "Tottenham Court Road",

"Tottenham Hale", "Totteridge & Whetstone", "Tower Gateway",

"Tower Hill", "Tufnell Park",

"Turnham Green", "Turnpike Lane", "Upminster",

"Upminster Bridge", "Upney", "Upton Park",

"Uxbridge", "Vauxhall", "Victoria",

"Walthamstow Central", "Wanstead", "Wapping",

"Warren Street", "Warwick Avenue", "Waterloo",

"Watford", "Wembley Central", "Wembley Park",

"West Acton", "West Brompton", "West Finchley",

"West Ham","West Hampstead","West Harrow",

"West India Quay","West Kensington","West Ruislip",

"Westbourne Park","Westferry","Westminster",

"White City","Whitechapel","Willesden Green",

"Willesden Junction","Wimbledon","Wimbledon Park",

"Wood Green","Woodford","Woodside Park"

};

}

### startScreen.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Windows.Forms;

namespace TubeRouteFinder

{

public partial class startScreen : Form

{

public static loadFileForm lF = new loadFileForm();

public static bool hasAFileBeenLoaded = false;

public startScreen()

{

InitializeComponent();

}

private void buttonNew\_Click(object sender, EventArgs e)

{

mapScreen mapscreen = new mapScreen();

mapscreen.Show();

this.Hide();

}

private void buttonLoad\_Click(object sender, EventArgs e)

{

hasAFileBeenLoaded = true;

lF.Show();

this.Hide();

}

}

}

### Autogen Code for startScreen.cs

partial class startScreen

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.buttonNew = new System.Windows.Forms.Button();

this.buttonLoad = new System.Windows.Forms.Button();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// buttonNew

//

this.buttonNew.Location = new System.Drawing.Point(594, 343);

this.buttonNew.Name = "buttonNew";

this.buttonNew.Size = new System.Drawing.Size(90, 40);

this.buttonNew.TabIndex = 0;

this.buttonNew.Text = "New";

this.buttonNew.UseVisualStyleBackColor = true;

this.buttonNew.Click += new System.EventHandler(this.buttonNew\_Click);

//

// buttonLoad

//

this.buttonLoad.Location = new System.Drawing.Point(594, 389);

this.buttonLoad.Name = "buttonLoad";

this.buttonLoad.Size = new System.Drawing.Size(90, 40);

this.buttonLoad.TabIndex = 1;

this.buttonLoad.Text = "Load";

this.buttonLoad.UseVisualStyleBackColor = true;

this.buttonLoad.Click += new System.EventHandler(this.buttonLoad\_Click);

//

// label1

//

this.label1.AutoSize = true;

this.label1.BackColor = System.Drawing.SystemColors.GrayText;

this.label1.ForeColor = System.Drawing.SystemColors.ButtonFace;

this.label1.Location = new System.Drawing.Point(577, 477);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(117, 13);

this.label1.TabIndex = 2;

this.label1.Text = "NEA 2021 - Maxim Carr";

//

// label2

//

this.label2.AutoSize = true;

this.label2.BackColor = System.Drawing.SystemColors.GrayText;

this.label2.Font = new System.Drawing.Font("Microsoft YaHei", 14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label2.ForeColor = System.Drawing.SystemColors.ButtonFace;

this.label2.Location = new System.Drawing.Point(570, 239);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(135, 25);

this.label2.TabIndex = 3;

this.label2.Text = "Underground";

//

// label3

//

this.label3.AutoSize = true;

this.label3.BackColor = System.Drawing.SystemColors.GrayText;

this.label3.Font = new System.Drawing.Font("Microsoft YaHei", 15.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label3.Location = new System.Drawing.Point(567, 265);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(141, 28);

this.label3.TabIndex = 4;

this.label3.Text = "Route Finder";

//

// startScreen

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.SystemColors.WindowFrame;

this.BackgroundImage = global::TubeRouteFinder.Properties.Resources.tubeMapBackground\_new;

this.ClientSize = new System.Drawing.Size(1264, 681);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.buttonLoad);

this.Controls.Add(this.buttonNew);

this.Name = "startScreen";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "startScreen";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button buttonNew;

private System.Windows.Forms.Button buttonLoad;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

}

### mapScreen.cs

public partial class mapScreen : Form

{

//Global variables

static tubeBuild currentTubeBuild = new tubeBuild();

static int stationNumber = currentTubeBuild.stations.Count();

static List<List<int>> adjacency = new List<List<int>>(stationNumber);

public static List<List<int>> distanceList = new List<List<int>>(stationNumber); //Format distanceList[station][neighbor or distance]

List<CheckBox> stationCheckBoxes = new List<CheckBox>();

public static bool confirmation = false;

public mapScreen()

{

InitializeComponent();

saveButton.Hide();//Save Route button is hidden until route is calculated.

createCheckArray(); //Checkbox array for map

for (int i = 0; i < stationNumber; i++)

{

adjacency.Add(new List<int>());

distanceList.Add(new List<int>());

}

currentTubeBuild.buildGraph(adjacency);

if (startScreen.hasAFileBeenLoaded == true)

{

if ((startScreen.lF.source.Length > 0) & (startScreen.lF.dest.Length > 0))

{

int sourceStat = currentTubeBuild.getStationID(startScreen.lF.source);

int destStat = currentTubeBuild.getStationID(startScreen.lF.dest);

sourceField.Text = startScreen.lF.source;

destinationField.Text = startScreen.lF.dest;

launchDijkstra(sourceStat, destStat);

}

}

}

private void goButton\_Click(object sender, EventArgs e) //Event handler for clicking Go button

{

foreach(CheckBox x in stationCheckBoxes) //Resets checkboxes

{

x.Checked = false;

}

exceptionLabel.Text = "";//Resetting exception label (for when program runs > 1 times without shutdown)

timeEstimateLabel.Text = "";//Same as above but for the estimated time label.

if (sourceField.TextLength > 0 && sourceField.TextLength > 0)

{

try

{

string sourceStationInput = sourceField.Text, destinationStationInput = destinationField.Text;

if (sourceStationInput == destinationStationInput)//No same stations

{

sameSourceAndDestinationException();

}

else

{

int sourceStat = currentTubeBuild.getStationID(sourceStationInput);

int destStat = currentTubeBuild.getStationID(destinationStationInput);

launchDijkstra(sourceStat, destStat);//Off to Dijkstra

}

}

catch(IndexOutOfRangeException)//Catches non existent stations

{

nonExistentStationException();

}

}

else

noStationEnteredException();

}

void saveButton\_Click(object sender, EventArgs e)//Save a file begins here

{

saveFileForm sF = new saveFileForm();

sF.source = sourceField.Text;

sF.dest = destinationField.Text;

sF.Show();

}

//Exceptions for different user input errors

void nonExistentStationException()

{

exceptionLabel.Text = "One or both of the stations\nyou have entered do not exist\non this tube map.\nPlease verify.";

}

void noStationEnteredException()

{

exceptionLabel.Text = "Please enter stations";

}

void sameSourceAndDestinationException()

{

exceptionLabel.Text = "You have entered the same\nstation twice.\nPlease enter two different stations";

}

void nodesUnconnectedException()

{

exceptionLabel.Text = "Nodes are unconnected";

}

}

### dijkstraAlgorithm.cs

partial class mapScreen

{

public void launchDijkstra(int sourceStat, int destStat)//Generates fresh arrays for the Dijkstra algorithm, launches Dijkstra & then generates a path from output

{

int[] predecessorArray = new int[stationNumber];

int[] distancesArray = new int[stationNumber];

//Should not run as there are no isolated nodes on the map

//If one is found, due to a connection not being added properly in tubeBuild.buildGraph, this will stop the programme here

if (dijkstraTube(sourceStat, destStat, predecessorArray, distancesArray) == false)

{

nodesUnconnectedException();

return;

}

List<int> path = new List<int>();//Path for storing shortest route

int crawl = destStat;

path.Add(crawl);

while (predecessorArray[crawl] != -1)//Builds a path through each predecessor back to source

{

path.Add(predecessorArray[crawl]);

crawl = predecessorArray[crawl];

}

string[] pathOfStationNames = new string[path.Count];

//Converts integer station IDs to readable string station names

//Also sets path order to source-first whereas before it was destination-first.

for (int i = path.Count - 1; i >= 0; i--)

{

pathOfStationNames[i] = currentTubeBuild.getStationName(path[i]);

}

displayResult(path, distancesArray[destStat]);

}

//Implement Dijkstra

private static bool dijkstraTube(int sourceStation, int destStation, int[] predecessorArray, int[] distancesArray)

{

PriorityQueue routeQueue = new PriorityQueue(); //Creates a priority queue for stations

bool[] visited = new bool[stationNumber];

for (int i = 0; i < stationNumber; i++) //Resets arrays. Sets distances from all stations stat to source as maximum value. Clears predecessor array.

{

visited[i] = false; //Sets all nodes as unvisited

distancesArray[i] = int.MaxValue;

predecessorArray[i] = -1;//No station is in a route right now, so none of them have predecessors.

}

//Sets source as visited, distance to itself = 0 & adds this info to the priority queue.

visited[sourceStation] = true;

distancesArray[sourceStation] = 0;

routeQueue.enQueue(sourceStation, 0);

while (routeQueue.Count() != 0)

{

//returns station (and predecessors) with smallest distance from source still in queue

int currentStation = routeQueue.deQueue();

for (int neighbour = 0; neighbour < adjacency[currentStation].Count; neighbour++)//For each neighbor of current Station

{

if (visited[adjacency[currentStation][neighbour]] == false)

{

int connectionDistance = findDistance(0, currentStation, neighbour);//Gives distance between the current station and the neighbor currently being checked

visited[adjacency[currentStation][neighbour]] = true;//Marks this neigbor as visited

distancesArray[adjacency[currentStation][neighbour]] = distancesArray[currentStation] + connectionDistance; //Total distance from source

predecessorArray[adjacency[currentStation][neighbour]] = currentStation;

routeQueue.enQueue(adjacency[currentStation][neighbour], distancesArray[adjacency[currentStation][neighbour]]);//Enqueue this neigbor & its distance from source

if (adjacency[currentStation][neighbour] == destStation) //Is neigbour = destination? If yes, end. If no, continue.

return true;

}

}

}

//Only ends here if there is no connection. This should not happen.

return false;

}

static int findDistance(int j, int currentStation, int neighbour)//Gives distance between the current station and the neighbor currently being checked

{

int connectionDistance = 0; bool distFound = false;

while (!distFound)

{

if (distanceList[currentStation][j] == adjacency[currentStation][neighbour])

{

connectionDistance = distanceList[currentStation][j + 1];

distFound = true;

}

j++;

}

return connectionDistance;

}

}

### display.cs

partial class mapScreen// Once again split to separate parts of class with different functions

{

void displayResult(List<int> stationIDArray, int estimatedTime)

{

foreach(int i in stationIDArray)

{

stationCheckBoxes[i].Checked = true;

}

timeEstimateLabel.Text = "Estimated Time:\n" + estimatedTime + " minutes.";

saveButton.Show();

}

void createCheckArray()//Creates a list of all checkboxes present on map

{

foreach(CheckBox checkBox in this.Controls.OfType<CheckBox>())

{

stationCheckBoxes.Add(checkBox);

}

stationCheckBoxes.Reverse(); //By default the list is backwards with stationCheckBoxes[0] = checkBox301. This makes it the right way round.

}

}

### Autogen code for mapScreen.cs

namespace TubeRouteFinder

{

partial class mapScreen

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

Private void InitialiseComponent() // This is removed from this document as it is very long.

#endregion

//Many checkboxes

private System.Windows.Forms.PictureBox pictureBox1;

private System.Windows.Forms.CheckBox checkBox0;

private System.Windows.Forms.CheckBox checkBox1;

private System.Windows.Forms.CheckBox checkBox2;

private System.Windows.Forms.CheckBox checkBox3;

private System.Windows.Forms.CheckBox checkBox4;

private System.Windows.Forms.CheckBox checkBox5;

private System.Windows.Forms.CheckBox checkBox6;

private System.Windows.Forms.CheckBox checkBox7;

private System.Windows.Forms.CheckBox checkBox8;

private System.Windows.Forms.CheckBox checkBox9;

private System.Windows.Forms.CheckBox checkBox10;

private System.Windows.Forms.CheckBox checkBox11;

private System.Windows.Forms.CheckBox checkBox12;

private System.Windows.Forms.CheckBox checkBox13;

private System.Windows.Forms.CheckBox checkBox14;

private System.Windows.Forms.CheckBox checkBox15;

private System.Windows.Forms.CheckBox checkBox16;

private System.Windows.Forms.CheckBox checkBox17;

private System.Windows.Forms.CheckBox checkBox18;

private System.Windows.Forms.CheckBox checkBox19;

private System.Windows.Forms.CheckBox checkBox20;

private System.Windows.Forms.CheckBox checkBox21;

private System.Windows.Forms.CheckBox checkBox22;

private System.Windows.Forms.CheckBox checkBox23;

private System.Windows.Forms.CheckBox checkBox24;

private System.Windows.Forms.CheckBox checkBox25;

private System.Windows.Forms.CheckBox checkBox26;

private System.Windows.Forms.CheckBox checkBox27;

private System.Windows.Forms.CheckBox checkBox28;

private System.Windows.Forms.CheckBox checkBox29;

private System.Windows.Forms.CheckBox checkBox30;

private System.Windows.Forms.CheckBox checkBox31;

private System.Windows.Forms.CheckBox checkBox32;

private System.Windows.Forms.CheckBox checkBox33;

private System.Windows.Forms.CheckBox checkBox34;

private System.Windows.Forms.CheckBox checkBox35;

private System.Windows.Forms.CheckBox checkBox36;

private System.Windows.Forms.CheckBox checkBox37;

private System.Windows.Forms.CheckBox checkBox38;

private System.Windows.Forms.CheckBox checkBox39;

private System.Windows.Forms.CheckBox checkBox40;

private System.Windows.Forms.CheckBox checkBox41;

private System.Windows.Forms.CheckBox checkBox42;

private System.Windows.Forms.CheckBox checkBox43;

private System.Windows.Forms.CheckBox checkBox44;

private System.Windows.Forms.CheckBox checkBox45;

private System.Windows.Forms.CheckBox checkBox46;

private System.Windows.Forms.CheckBox checkBox47;

private System.Windows.Forms.CheckBox checkBox48;

private System.Windows.Forms.CheckBox checkBox49;

private System.Windows.Forms.CheckBox checkBox50;

private System.Windows.Forms.CheckBox checkBox51;

private System.Windows.Forms.CheckBox checkBox52;

private System.Windows.Forms.CheckBox checkBox53;

private System.Windows.Forms.CheckBox checkBox54;

private System.Windows.Forms.CheckBox checkBox55;

private System.Windows.Forms.CheckBox checkBox56;

private System.Windows.Forms.CheckBox checkBox57;

private System.Windows.Forms.CheckBox checkBox58;

private System.Windows.Forms.CheckBox checkBox59;

private System.Windows.Forms.CheckBox checkBox60;

private System.Windows.Forms.CheckBox checkBox61;

private System.Windows.Forms.CheckBox checkBox62;

private System.Windows.Forms.CheckBox checkBox63;

private System.Windows.Forms.CheckBox checkBox64;

private System.Windows.Forms.CheckBox checkBox65;

private System.Windows.Forms.CheckBox checkBox66;

private System.Windows.Forms.CheckBox checkBox67;

private System.Windows.Forms.CheckBox checkBox68;

private System.Windows.Forms.CheckBox checkBox69;

private System.Windows.Forms.CheckBox checkBox70;

private System.Windows.Forms.CheckBox checkBox71;

private System.Windows.Forms.CheckBox checkBox72;

private System.Windows.Forms.CheckBox checkBox73;

private System.Windows.Forms.CheckBox checkBox74;

private System.Windows.Forms.CheckBox checkBox75;

private System.Windows.Forms.CheckBox checkBox76;

private System.Windows.Forms.CheckBox checkBox77;

private System.Windows.Forms.CheckBox checkBox78;

private System.Windows.Forms.CheckBox checkBox79;

private System.Windows.Forms.CheckBox checkBox80;

private System.Windows.Forms.CheckBox checkBox81;

private System.Windows.Forms.CheckBox checkBox82;

private System.Windows.Forms.CheckBox checkBox83;

private System.Windows.Forms.CheckBox checkBox84;

private System.Windows.Forms.CheckBox checkBox85;

private System.Windows.Forms.CheckBox checkBox86;

private System.Windows.Forms.CheckBox checkBox87;

private System.Windows.Forms.CheckBox checkBox88;

private System.Windows.Forms.CheckBox checkBox89;

private System.Windows.Forms.CheckBox checkBox90;

private System.Windows.Forms.CheckBox checkBox91;

private System.Windows.Forms.CheckBox checkBox92;

private System.Windows.Forms.CheckBox checkBox93;

private System.Windows.Forms.CheckBox checkBox94;

private System.Windows.Forms.CheckBox checkBox95;

private System.Windows.Forms.CheckBox checkBox96;

private System.Windows.Forms.CheckBox checkBox97;

private System.Windows.Forms.CheckBox checkBox98;

private System.Windows.Forms.CheckBox checkBox99;

private System.Windows.Forms.CheckBox checkBox100;

private System.Windows.Forms.CheckBox checkBox101;

private System.Windows.Forms.CheckBox checkBox102;

private System.Windows.Forms.CheckBox checkBox103;

private System.Windows.Forms.CheckBox checkBox104;

private System.Windows.Forms.CheckBox checkBox105;

private System.Windows.Forms.CheckBox checkBox106;

private System.Windows.Forms.CheckBox checkBox107;

private System.Windows.Forms.CheckBox checkBox108;

private System.Windows.Forms.CheckBox checkBox109;

private System.Windows.Forms.CheckBox checkBox110;

private System.Windows.Forms.CheckBox checkBox111;

private System.Windows.Forms.CheckBox checkBox112;

private System.Windows.Forms.CheckBox checkBox113;

private System.Windows.Forms.CheckBox checkBox114;

private System.Windows.Forms.CheckBox checkBox115;

private System.Windows.Forms.CheckBox checkBox116;

private System.Windows.Forms.CheckBox checkBox117;

private System.Windows.Forms.CheckBox checkBox118;

private System.Windows.Forms.CheckBox checkBox119;

private System.Windows.Forms.CheckBox checkBox120;

private System.Windows.Forms.CheckBox checkBox121;

private System.Windows.Forms.CheckBox checkBox122;

private System.Windows.Forms.CheckBox checkBox123;

private System.Windows.Forms.CheckBox checkBox124;

private System.Windows.Forms.CheckBox checkBox125;

private System.Windows.Forms.CheckBox checkBox126;

private System.Windows.Forms.CheckBox checkBox127;

private System.Windows.Forms.CheckBox checkBox128;

private System.Windows.Forms.CheckBox checkBox129;

private System.Windows.Forms.CheckBox checkBox130;

private System.Windows.Forms.CheckBox checkBox131;

private System.Windows.Forms.CheckBox checkBox132;

private System.Windows.Forms.CheckBox checkBox133;

private System.Windows.Forms.CheckBox checkBox134;

private System.Windows.Forms.CheckBox checkBox135;

private System.Windows.Forms.CheckBox checkBox136;

private System.Windows.Forms.CheckBox checkBox137;

private System.Windows.Forms.CheckBox checkBox138;

private System.Windows.Forms.CheckBox checkBox139;

private System.Windows.Forms.CheckBox checkBox140;

private System.Windows.Forms.CheckBox checkBox141;

private System.Windows.Forms.CheckBox checkBox142;

private System.Windows.Forms.CheckBox checkBox143;

private System.Windows.Forms.CheckBox checkBox144;

private System.Windows.Forms.CheckBox checkBox145;

private System.Windows.Forms.CheckBox checkBox146;

private System.Windows.Forms.CheckBox checkBox147;

private System.Windows.Forms.CheckBox checkBox148;

private System.Windows.Forms.CheckBox checkBox149;

private System.Windows.Forms.CheckBox checkBox150;

private System.Windows.Forms.CheckBox checkBox151;

private System.Windows.Forms.CheckBox checkBox152;

private System.Windows.Forms.CheckBox checkBox153;

private System.Windows.Forms.CheckBox checkBox154;

private System.Windows.Forms.CheckBox checkBox155;

private System.Windows.Forms.CheckBox checkBox156;

private System.Windows.Forms.CheckBox checkBox157;

private System.Windows.Forms.CheckBox checkBox158;

private System.Windows.Forms.CheckBox checkBox159;

private System.Windows.Forms.CheckBox checkBox160;

private System.Windows.Forms.CheckBox checkBox161;

private System.Windows.Forms.CheckBox checkBox162;

private System.Windows.Forms.CheckBox checkBox163;

private System.Windows.Forms.CheckBox checkBox164;

private System.Windows.Forms.CheckBox checkBox165;

private System.Windows.Forms.CheckBox checkBox166;

private System.Windows.Forms.CheckBox checkBox167;

private System.Windows.Forms.CheckBox checkBox168;

private System.Windows.Forms.CheckBox checkBox169;

private System.Windows.Forms.CheckBox checkBox170;

private System.Windows.Forms.CheckBox checkBox171;

private System.Windows.Forms.CheckBox checkBox172;

private System.Windows.Forms.CheckBox checkBox173;

private System.Windows.Forms.CheckBox checkBox174;

private System.Windows.Forms.CheckBox checkBox175;

private System.Windows.Forms.CheckBox checkBox176;

private System.Windows.Forms.CheckBox checkBox177;

private System.Windows.Forms.CheckBox checkBox178;

private System.Windows.Forms.CheckBox checkBox179;

private System.Windows.Forms.CheckBox checkBox180;

private System.Windows.Forms.CheckBox checkBox181;

private System.Windows.Forms.CheckBox checkBox182;

private System.Windows.Forms.CheckBox checkBox183;

private System.Windows.Forms.CheckBox checkBox184;

private System.Windows.Forms.CheckBox checkBox185;

private System.Windows.Forms.CheckBox checkBox186;

private System.Windows.Forms.CheckBox checkBox187;

private System.Windows.Forms.CheckBox checkBox188;

private System.Windows.Forms.CheckBox checkBox189;

private System.Windows.Forms.CheckBox checkBox190;

private System.Windows.Forms.CheckBox checkBox191;

private System.Windows.Forms.CheckBox checkBox192;

private System.Windows.Forms.CheckBox checkBox193;

private System.Windows.Forms.CheckBox checkBox194;

private System.Windows.Forms.CheckBox checkBox195;

private System.Windows.Forms.CheckBox checkBox196;

private System.Windows.Forms.CheckBox checkBox197;

private System.Windows.Forms.CheckBox checkBox198;

private System.Windows.Forms.CheckBox checkBox199;

private System.Windows.Forms.CheckBox checkBox200;

private System.Windows.Forms.CheckBox checkBox201;

private System.Windows.Forms.CheckBox checkBox202;

private System.Windows.Forms.CheckBox checkBox203;

private System.Windows.Forms.CheckBox checkBox204;

private System.Windows.Forms.CheckBox checkBox205;

private System.Windows.Forms.CheckBox checkBox206;

private System.Windows.Forms.CheckBox checkBox207;

private System.Windows.Forms.CheckBox checkBox208;

private System.Windows.Forms.CheckBox checkBox209;

private System.Windows.Forms.CheckBox checkBox210;

private System.Windows.Forms.CheckBox checkBox211;

private System.Windows.Forms.CheckBox checkBox212;

private System.Windows.Forms.CheckBox checkBox213;

private System.Windows.Forms.CheckBox checkBox214;

private System.Windows.Forms.CheckBox checkBox215;

private System.Windows.Forms.CheckBox checkBox216;

private System.Windows.Forms.CheckBox checkBox217;

private System.Windows.Forms.CheckBox checkBox218;

private System.Windows.Forms.CheckBox checkBox219;

private System.Windows.Forms.CheckBox checkBox220;

private System.Windows.Forms.CheckBox checkBox221;

private System.Windows.Forms.CheckBox checkBox222;

private System.Windows.Forms.CheckBox checkBox223;

private System.Windows.Forms.CheckBox checkBox224;

private System.Windows.Forms.CheckBox checkBox225;

private System.Windows.Forms.CheckBox checkBox226;

private System.Windows.Forms.CheckBox checkBox227;

private System.Windows.Forms.CheckBox checkBox228;

private System.Windows.Forms.CheckBox checkBox229;

private System.Windows.Forms.CheckBox checkBox230;

private System.Windows.Forms.CheckBox checkBox231;

private System.Windows.Forms.CheckBox checkBox232;

private System.Windows.Forms.CheckBox checkBox233;

private System.Windows.Forms.CheckBox checkBox234;

private System.Windows.Forms.CheckBox checkBox235;

private System.Windows.Forms.CheckBox checkBox236;

private System.Windows.Forms.CheckBox checkBox237;

private System.Windows.Forms.CheckBox checkBox238;

private System.Windows.Forms.CheckBox checkBox239;

private System.Windows.Forms.CheckBox checkBox240;

private System.Windows.Forms.CheckBox checkBox241;

private System.Windows.Forms.CheckBox checkBox242;

private System.Windows.Forms.CheckBox checkBox243;

private System.Windows.Forms.CheckBox checkBox244;

private System.Windows.Forms.CheckBox checkBox245;

private System.Windows.Forms.CheckBox checkBox246;

private System.Windows.Forms.CheckBox checkBox247;

private System.Windows.Forms.CheckBox checkBox248;

private System.Windows.Forms.CheckBox checkBox249;

private System.Windows.Forms.CheckBox checkBox250;

private System.Windows.Forms.CheckBox checkBox251;

private System.Windows.Forms.CheckBox checkBox252;

private System.Windows.Forms.CheckBox checkBox253;

private System.Windows.Forms.CheckBox checkBox254;

private System.Windows.Forms.CheckBox checkBox255;

private System.Windows.Forms.CheckBox checkBox256;

private System.Windows.Forms.CheckBox checkBox257;

private System.Windows.Forms.CheckBox checkBox258;

private System.Windows.Forms.CheckBox checkBox259;

private System.Windows.Forms.CheckBox checkBox260;

private System.Windows.Forms.CheckBox checkBox261;

private System.Windows.Forms.CheckBox checkBox262;

private System.Windows.Forms.CheckBox checkBox263;

private System.Windows.Forms.CheckBox checkBox264;

private System.Windows.Forms.CheckBox checkBox265;

private System.Windows.Forms.CheckBox checkBox266;

private System.Windows.Forms.CheckBox checkBox267;

private System.Windows.Forms.CheckBox checkBox268;

private System.Windows.Forms.CheckBox checkBox269;

private System.Windows.Forms.CheckBox checkBox270;

private System.Windows.Forms.CheckBox checkBox271;

private System.Windows.Forms.CheckBox checkBox272;

private System.Windows.Forms.CheckBox checkBox273;

private System.Windows.Forms.CheckBox checkBox274;

private System.Windows.Forms.CheckBox checkBox275;

private System.Windows.Forms.CheckBox checkBox276;

private System.Windows.Forms.CheckBox checkBox277;

private System.Windows.Forms.CheckBox checkBox278;

private System.Windows.Forms.CheckBox checkBox279;

private System.Windows.Forms.CheckBox checkBox280;

private System.Windows.Forms.CheckBox checkBox281;

private System.Windows.Forms.CheckBox checkBox282;

private System.Windows.Forms.CheckBox checkBox283;

private System.Windows.Forms.CheckBox checkBox284;

private System.Windows.Forms.CheckBox checkBox285;

private System.Windows.Forms.CheckBox checkBox286;

private System.Windows.Forms.CheckBox checkBox287;

private System.Windows.Forms.CheckBox checkBox288;

private System.Windows.Forms.CheckBox checkBox289;

private System.Windows.Forms.CheckBox checkBox290;

private System.Windows.Forms.CheckBox checkBox291;

private System.Windows.Forms.CheckBox checkBox292;

private System.Windows.Forms.CheckBox checkBox293;

private System.Windows.Forms.CheckBox checkBox294;

private System.Windows.Forms.CheckBox checkBox295;

private System.Windows.Forms.CheckBox checkBox296;

private System.Windows.Forms.CheckBox checkBox297;

private System.Windows.Forms.CheckBox checkBox298;

private System.Windows.Forms.CheckBox checkBox299;

private System.Windows.Forms.CheckBox checkBox300;

private System.Windows.Forms.CheckBox checkBox301;

private System.Windows.Forms.Label sourceStationLabel;

private System.Windows.Forms.TextBox sourceField;

private System.Windows.Forms.Label destinationStaticLabel;

private System.Windows.Forms.TextBox destinationField;

private System.Windows.Forms.Button goButton;

private System.Windows.Forms.Label exceptionLabel;

protected System.Windows.Forms.Label timeEstimateLabel;

private System.Windows.Forms.Button saveButton;

}

### PriorityQueue.cs

class PriorityQueue //Priority queue to store stations.

{

int queueCount = 0;

class node //Member class for queue

{

public int station { get; set; }

public int distanceAsPriority { get; set; }

}

List<node> stationsAsPrioQueue = new List<node>();//Creates emprt list to serve as container for priority queue

public void enQueue(int newStation, int newDistance) //Enqueues a new station as node

{

node newNode = new node { station = newStation, distanceAsPriority = newDistance };

stationsAsPrioQueue.Add(newNode);

queueCount++;

insertionSort();

}

public int Count()

{

return queueCount;

}

public int deQueue()//Dequeues station with smallest distance to source

{

int stationToReturn = stationsAsPrioQueue[0].station;

stationsAsPrioQueue.RemoveAt(0);

queueCount--;

insertionSort();

return stationToReturn;

}

//Insertion sort queue members

//The queue length increments meaning few swaps each time this is run. Queue will not be very long.

void insertionSort()

{

int positionInQueue;

node nodeToInsert;

for(int i = 1; i<this.Count(); i++)

{

nodeToInsert = stationsAsPrioQueue[i];

positionInQueue = i;

while ((positionInQueue > 0) && (stationsAsPrioQueue[positionInQueue-1].distanceAsPriority > nodeToInsert.distanceAsPriority))

{

//swapping until correct position is found

stationsAsPrioQueue[positionInQueue] = stationsAsPrioQueue[positionInQueue - 1];

positionInQueue--;

}

stationsAsPrioQueue[positionInQueue] = nodeToInsert;

}

}

}

### loadFileForm.cs

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TubeRouteFinder

{

public partial class loadFileForm : Form

{

startScreen start = new startScreen();

public string source = string.Empty, dest;

public loadFileForm()

{

InitializeComponent();

}

private void loadFileButton\_Click(object sender, EventArgs e)

{

string fileName = fileNameBox.Text;

try

{

using (StreamReader sr = new StreamReader(fileName + ".txt"))

{

string line;

while ((line = sr.ReadLine()) != null)

{

if (string.IsNullOrEmpty(source))

{

source = line;

}

else

{

dest = line;

}

}

sr.Close();

}

mapScreen mapScreen = new mapScreen();

mapScreen.Show();

this.Hide();

}

catch(Exception)

{

fileNotFoundLabel.Text = "File Not Found\ntry again or cancel.";

}

}

private void cancelButton\_Click(object sender, EventArgs e)

{

startScreen.hasAFileBeenLoaded = false;

start.Show();

this.Hide();

}

}

}

### Autogen Code for loadFileForm.cs

partial class loadFileForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.EnteraNameLabel = new System.Windows.Forms.Label();

this.fileNameBox = new System.Windows.Forms.TextBox();

this.loadFileButton = new System.Windows.Forms.Button();

this.cancelButton = new System.Windows.Forms.Button();

this.fileNotFoundLabel = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// EnteraNameLabel

//

this.EnteraNameLabel.AutoSize = true;

this.EnteraNameLabel.Location = new System.Drawing.Point(12, 9);

this.EnteraNameLabel.Name = "EnteraNameLabel";

this.EnteraNameLabel.Size = new System.Drawing.Size(95, 13);

this.EnteraNameLabel.TabIndex = 0;

this.EnteraNameLabel.Text = "Enter the Filename";

//

// fileNameBox

//

this.fileNameBox.Location = new System.Drawing.Point(15, 25);

this.fileNameBox.Name = "fileNameBox";

this.fileNameBox.Size = new System.Drawing.Size(312, 20);

this.fileNameBox.TabIndex = 1;

//

// loadFileButton

//

this.loadFileButton.Location = new System.Drawing.Point(252, 58);

this.loadFileButton.Name = "loadFileButton";

this.loadFileButton.Size = new System.Drawing.Size(75, 23);

this.loadFileButton.TabIndex = 2;

this.loadFileButton.Text = "Load";

this.loadFileButton.UseVisualStyleBackColor = true;

this.loadFileButton.Click += new System.EventHandler(this.loadFileButton\_Click);

//

// cancelButton

//

this.cancelButton.Location = new System.Drawing.Point(15, 58);

this.cancelButton.Name = "cancelButton";

this.cancelButton.Size = new System.Drawing.Size(75, 23);

this.cancelButton.TabIndex = 3;

this.cancelButton.Text = "Cancel";

this.cancelButton.UseVisualStyleBackColor = true;

this.cancelButton.Click += new System.EventHandler(this.cancelButton\_Click);

//

// fileNotFoundLabel

//

this.fileNotFoundLabel.AutoSize = true;

this.fileNotFoundLabel.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(64)))), ((int)(((byte)(0)))));

this.fileNotFoundLabel.Location = new System.Drawing.Point(118, 63);

this.fileNotFoundLabel.Name = "fileNotFoundLabel";

this.fileNotFoundLabel.Size = new System.Drawing.Size(0, 13);

this.fileNotFoundLabel.TabIndex = 4;

//

// loadFileForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(339, 93);

this.Controls.Add(this.fileNotFoundLabel);

this.Controls.Add(this.cancelButton);

this.Controls.Add(this.loadFileButton);

this.Controls.Add(this.fileNameBox);

this.Controls.Add(this.EnteraNameLabel);

this.Name = "loadFileForm";

this.Text = "Load a Route";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label EnteraNameLabel;

private System.Windows.Forms.TextBox fileNameBox;

private System.Windows.Forms.Button loadFileButton;

private System.Windows.Forms.Button cancelButton;

private System.Windows.Forms.Label fileNotFoundLabel;

}

### saveFileForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TubeRouteFinder

{

public partial class saveFileForm : Form

{

public string source, dest;

private void saveButton\_Click(object sender, EventArgs e)

{

string path = (fileNameBox.Text)+".txt";

if(!File.Exists(path))//Checks for existing file

{

using(StreamWriter save = File.CreateText(path))//Creates and writes to file

{

save.WriteLine(source);

save.WriteLine(dest);

}

}

else if(mapScreen.confirmation==true)//Overwrites file

{

File.Delete(path);

using (StreamWriter save = File.CreateText(path))

{

save.WriteLine(source);

save.WriteLine(dest);

}

}

else

{

errorLabel.Text = "File already exists\nPress again for overwrite";//Prepeares for overwrite

mapScreen.confirmation = true;

}

this.Hide();

}

private void cancelButton\_Click(object sender, EventArgs e)//Closes form if user has changed their mind

{

this.Hide();

}

public saveFileForm()

{

InitializeComponent();

}

}

}

### Autogen Code for saveFileForm.cs

partial class saveFileForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.saveFileLabel = new System.Windows.Forms.Label();

this.fileNameBox = new System.Windows.Forms.TextBox();

this.saveButton = new System.Windows.Forms.Button();

this.cancelButton = new System.Windows.Forms.Button();

this.errorLabel = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// saveFileLabel

//

this.saveFileLabel.AutoSize = true;

this.saveFileLabel.Location = new System.Drawing.Point(12, 9);

this.saveFileLabel.Name = "saveFileLabel";

this.saveFileLabel.Size = new System.Drawing.Size(226, 13);

this.saveFileLabel.TabIndex = 0;

this.saveFileLabel.Text = "Enter a filename you\'d like to use for this route:";

//

// fileNameBox

//

this.fileNameBox.Location = new System.Drawing.Point(12, 25);

this.fileNameBox.Name = "fileNameBox";

this.fileNameBox.Size = new System.Drawing.Size(308, 20);

this.fileNameBox.TabIndex = 1;

//

// saveButton

//

this.saveButton.Location = new System.Drawing.Point(245, 51);

this.saveButton.Name = "saveButton";

this.saveButton.Size = new System.Drawing.Size(75, 23);

this.saveButton.TabIndex = 2;

this.saveButton.Text = "Save";

this.saveButton.UseVisualStyleBackColor = true;

this.saveButton.Click += new System.EventHandler(this.saveButton\_Click);

//

// cancelButton

//

this.cancelButton.Location = new System.Drawing.Point(12, 51);

this.cancelButton.Name = "cancelButton";

this.cancelButton.Size = new System.Drawing.Size(75, 23);

this.cancelButton.TabIndex = 3;

this.cancelButton.Text = "Cancel";

this.cancelButton.UseVisualStyleBackColor = true;

this.cancelButton.Click += new System.EventHandler(this.cancelButton\_Click);

//

// errorLabel

//

this.errorLabel.AutoSize = true;

this.errorLabel.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(64)))), ((int)(((byte)(0)))));

this.errorLabel.Location = new System.Drawing.Point(90, 95);

this.errorLabel.Name = "errorLabel";

this.errorLabel.Size = new System.Drawing.Size(0, 13);

this.errorLabel.TabIndex = 4;

//

// saveFileForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(332, 134);

this.Controls.Add(this.errorLabel);

this.Controls.Add(this.cancelButton);

this.Controls.Add(this.saveButton);

this.Controls.Add(this.fileNameBox);

this.Controls.Add(this.saveFileLabel);

this.Name = "saveFileForm";

this.Text = "saveFileForm";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label saveFileLabel;

private System.Windows.Forms.TextBox fileNameBox;

private System.Windows.Forms.Button saveButton;

private System.Windows.Forms.Button cancelButton;

private System.Windows.Forms.Label errorLabel;

}

## Testing

### Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | What is Tested | Reason for Test | Input | Expected Result | Actual Result |
| 1 | Route finding –  Objective 1.2 | Normal inputs | “Maida Vale” -> Source Station.  “Aldgate” -> Destination. | Route between Maida Vale & Aldgate. | As expected |
| 2 | Station letter case conversion – Objective 1.7 | Ensuring that the system ignores input letter case. | “uXbRiDgE” -> Source Station.  “ACTON town” -> Destination. | Route between Uxbridge & Acton Town. | As expected |
| 3 | Shortest path by station count vs. shortest path by estimated travel time – Objective 1.2.1 | Earlier versions of system miscalculated this route preferring the fewer stations option – even when it takes longer | “West Ham” -> Source Station.  “Mile End” -> Destination. | Route between Source & Destination stations, NOT via Stratford. | As expected |
| 4 | Reversal of Station Names – Objective 1.3 | Making certain that order of input has no impact on route | “Mile End” -> Source Station.  “West Ham” -> Destination. | Same route as Test 3 | As expected |
| 5 | Consistency of generated route – Objectives 1.4, 1.7 | Confirming no change in route. | “maida vale” -> Source Station.  “aldgate” -> Destination. | Same route as Test 1 | As expected |
| 6 | Combination of Tests 4 & 5 – Objectives 1.3, 1.4, 1.7 | Confirming that the same endpoints always will provide the same route. | “aldgate” -> Source Station.  “maida vale” -> Destination. | Same route as Test 1, Test 5 | As expected |
| 7 |  | Invalid Input – blank stations | “” -> Source  “” -> Destination | No route,  ‘No Stations’ error message displayed | As expected |
| 8 |  | Invalid Input – One station blank | “” -> Source  “King’s Cross St. Pancras” -> Destination | No route, error message displayed | Borderline – No route was displayed, but the error message displayed was the same as in Test 7 |
| 9 | Invalid Input cases – Objectives 1.5.x | Invalid Input – Non-Existent Stations (Either One or Both) | “AStation” -> Source  “AnotherStation” -> Destination | No route, error message displayed. | As expected |
| 10 |  |  | “34” -> Source  “Greenford” -> Destination | No route, error message displayed. | As expected |
| 11 |  |  | “45” -> Source  “55” -> Destination | No route, error message displayed. | As expected |
| 12 |  | Edge case with station names – Marylbon will be entered instead of Marylebone | “Marylbon”-> Source  “Angel” -> Destination | No route, error message displayed. | As expected |
| 13 |  | Source != Destination | “Angel” -> Source  “Angel” -> Destination | No route, error message displayed. | As expected |
| 14 | Time estimate – Objective 1.6 | Estimated time number must be accurate | “Barons Court” -> Source  “Oxford Circus” -> Destination | Route. Hand-drawing route on map gives 3+2+1+3+2+2+2 = 15, so would expect estimated time to be equal | As expected |
| 15 | Combination of Time Estimate, Letter case entered and input order tests – Objectives 1.2, 1.3, 1.4, 1.6, 1.7 | Full route finding system check | “OXFORd cIrcus” -> Source “bARONS cOURT” -> Destination | Same route as in previous test. Same time as previous test. | As expected |
| 16 | Route saving function – Objective 2.1 | Must be able to save a route | “King’s Cross St. Pancras” -> Source  “South Kensington” -> Destination  “test16” -> File Name | File called test16.txt created in directory containing everything that is needed for this route. | As expected |
| 17 | Route loading & viewing from a file – Objective 2.2 | Must be able to view a stored route | “test16” -> File Name | Route between Kings Cross & South Kensington | As expected |
| 18 | Error Catching in the File System –  Objective 2.3 | Should not break if invalid filename entered into load function | “test17” -> File Name  This file does not exist in the project’s directory | Error message displayed | As expected |
| 19 |  | If an already used name is entered into the save function it should warn the user before overwriting | Different route from Test 16  “test16” -> File Name | Warning to User | Fail – Although it does not overwrite the file, it does not warn the user |
| 20 | Cancel file operation (write) – Objective 2.4 | Objective 2.4 fulfilment | Pressing Cancel button on the Save File Form | Form closes itself allowing user to change routes | As expected |
| 21 | Cancel file operation (read) – Objective 2.4 | Objective 2.4 fulfilment | Pressing Cancel button on the Load File Form | Form closes itself and returns to the main menu | As expected |
| 22a | Whole system test –  Objectives 1.2, 1.3, 1.4, 1.6, 1.7, 2.1, 2.2 | Test the whole system in one go | “OakWOod” -> Source  “BanK” -> Destination | Route from Oakwood to Bank. Estimated time of 32 minutes. | As expected |
| 22b |  |  | Open file save form  “test22” -> FileName | Text file called test22 is created, containing info for route | As expected |
| 22c |  |  | Open file load form  “test22” -> FileName | Route is loaded from test22.txt and is displayed | As expected |
| 23 | Random station testing | Objective 1.1 | Random stations selected and entered into input boxes  Generator used can be found in notes. | Route between the two input stations. | As Expected |

#### Testing Notes

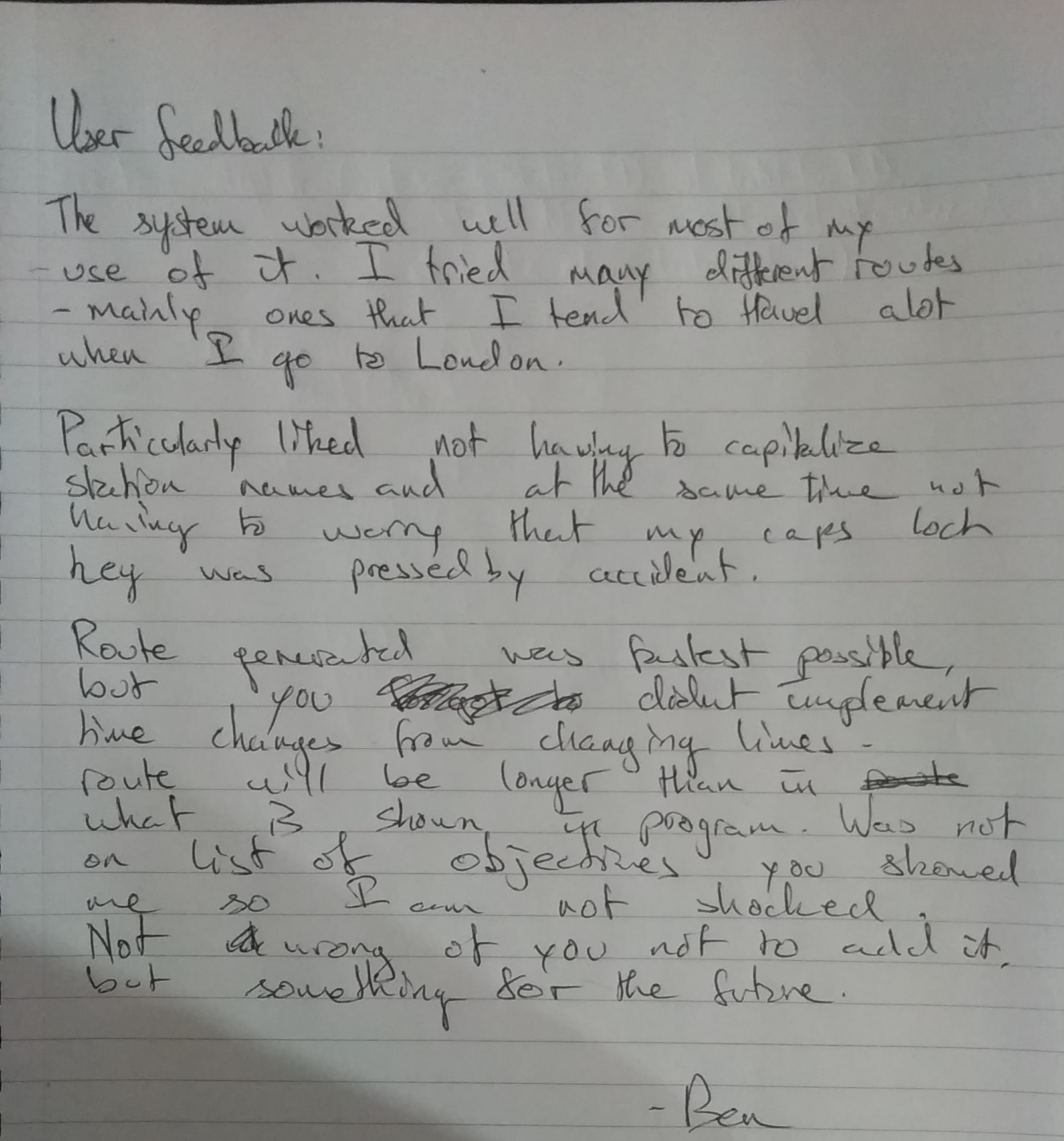
* Random station generator used for Test 23 can be found here: <http://www.generatorland.com/usergenerator.aspx?id=5176>
* Objective 1.1 was tested not only as part of Test 23, but across the whole regimen of tests due to the range of endpoints used.
* Some of the pictures in the next section have been rotated 90 degrees in order to fit into the input picture boxes.

### Proof of Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | | Inputs | Actual Result | |
| 1 | |  |  | |
| 2 | |  |  | |
| 3 | |  |  | |
| 4 | |  |  | |
| 5 | |  |  | |
| 6 | |  |  | |
| 7 | |  |  | |
| 8 | |  |  | |
| 9 | |  |  | |
| 10 | |  |  | |
| 11 | |  |  | |
| 12 | |  |  | |
| 13 | |  |  | |
| 14 | |  |  | |
| 15 | |  |  | |
| 16 | |  | Only endpoints need to be stored | |
| 17 | |  |  | |
| 18 | |  |  | |
| 19 | |  |  | |
| 20 |  | |  | |
| 21 | |  |  | |
| 22a | |  |  | |
| 22b | |  |  | |
| 22c | |  |  | |
| 23 | | One of many runs is shown here. | |  |

## Evaluation

### End User Feedback



As can be seen above Ben, who was interviewed as part of the Analysis at the start of this project, provided his feedback after trialling the system for just over half an hour. He tried every feature of the system from route generation to saving a route to loading said route back into the system. He also tried reversing the order of the input stations to see if he could get a different route out of the system, similar to a number of the tests that were already performed on the system. Just as in the tests, the result was the same no matter the order of inputs.

His main point of features that could be improved was the addition of a check to see if a line change had occurred – something that takes considerable time when using the Underground. However, as the user stated, this was not part of the objectives set out in the Analysis.

### Status of Analysis Objectives

|  |  |  |
| --- | --- | --- |
| Objective Name & Description | Test #s involving this Objective | Objective Status & Any notes if required.  Green = Successfully Completed  Yellow = Borderline  Red = Objective Failed |
| 1.0 – The system must use Dijkstra’s algorithm to find the shortest path between two user-defined stations. | All tests excluding 7,8,9,10,11,12,13, 16, 18 & 19. |  |
| 1.1 - The user must be able to enter any of the stations shown on the system’s map. | All tests excluding 7,8,9,10,11,12,13, 16, 18 & 19. |  |
| 1.2 - The route generated must connect the two endpoints provided by the user. | All tests excluding 7,8,9,10,11,12,13, 16, 18 & 19. |  |
| 1.2.1 - This route must be the shortest possible in terms of travel time. | All tests excluding 7,8,9,10,11,12,13, 16, 18 & 19. |  |
| 1.3 - The order that the stations are entered in must not impact the route generated. | 4,6,15 |  |
| * 1. - The route must be consistent - each pair of endpoints produces the same route each time they are used together. | 4,5,6,15 |  |
| 1.5.1 - The system must not attempt to find a route between a station and itself. | 13 |  |
| 1.5.2 - The system must not attempt to find a route between anything and a blank input. | 7,8 | Semi borderline with King’s Cross St. Pancras, but an error message was displayed so the user was not left in the dark that they had entered an invalid input. |
| 1.5.3 - The system must not attempt to find a route when any number of the input stations do not exist on its Underground map. | 9,10,11,12 |  |
| 1.5.4 - All of the above conditions should provide an error message to the user informing them what went wrong. | 7,8,9,10,11,12 |  |
| 1.6 - The estimated time of the route must be the sum of the edge weights of the route. | 14, 22(a, b & c), 23 |  |
| 1.7 – Stations must be able to be entered in both capital and lowercase. | 2,5,6,22,23 |  |
| 2.0 - The user must be able to store a generated route with a user-specified filename. | 16,22,23 |  |
| 2.1 - The user must be able to recover and view stored routes by accessing them with the name they entered earlier. | 17,22,23 |  |
| 2.2 - The load and save route functions must deal with any user error such as an invalid filename robustly proceeding to generate an error message that informs the user what happened. | 18, 19 | While the load file function is robust, the save file one is not. It failed to overwrite a file after notifying the user (Test 19). |
| 2.3.1 - The user should be able to change their mind and not store a route. | 20 |  |
| 2.3.2 – The user should also not be forced to load a route once the load menu is opened. | 21 |  |
|  |  |  |

### Evaluation based on Objectives and User Feedback

The initial goal of this project was to create a route planning programme for the London Underground. In that aspect, I can be confident that the final system has been successful in attaining that goal. Based on user feedback and the objective testing, which resulted in only one failure and one borderline result, the system seems to be working mostly as intended. The project has met almost all the objectives laid out during the analysis phase and the end user is pleased with the system with only one suggestion for the future.

The objectives laid out at the beginning of this project have been accomplished through testing of the system both through the tests in the Testing section and by Ben, the end user after he was provided with the list of objectives from the Analysis section.

However, this far from means that there is not room for improvement. The implementation of Dijkstra’s algorithm works – returning the shortest path between two stations.

If I attempted a similar project in the future, I would change or add the following:

* Given more time, I would implement the user’s suggestion from their feedback – a way to check and even possibly limit the number of line changes the generated route has.
* Implement further sorting algorithms for the priority queue just in case it becomes long. I only used Insertion Sort for this system as the queue did not seem to get very long and as modern computers are fast anyway, the now-miniscule time savings that would come with quicksort seems inconsequential.
* Implement the Overground, National Rail and as a further extension bus lines into the algorithm.
* As an extreme extension, a debug menu could be added to allow more curious users to specify which out of a range of shortest-path algorithms to use. Examples could include Dijkstra (already implemented), A\* and Bellman Ford (discussed in Analysis).
* Parts of the code are difficult to follow, even with commenting. If I were to do a future project, I would clean up my code more thoroughly.

# References

Brilliant, n.d. *Bellman-Ford Algorithm.* [Online]   
Available at: https://brilliant.org/wiki/bellman-ford-algorithm/  
[Accessed 4 11 2020].

Brilliant, n.d. *Dijkstra's Shortest Path Algorithm.* [Online]   
Available at: https://brilliant.org/wiki/dijkstras-short-path-finder/  
[Accessed 4 11 2020].

Geeks For Geeks, 2020. *What are the differences between Bellman Ford’s and Dijkstra’s algorithms?.* [Online]   
Available at: https://www.geeksforgeeks.org/what-are-the-differences-between-bellman-fords-and-dijkstras-algorithms/  
[Accessed 4 11 2020].

Transport for London, n.d. *Plan a Journey.* [Online]   
Available at: https://tfl.gov.uk/plan-a-journey/  
[Accessed 6 11 2020].