

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:		Student ID:	
Other name(s):			
Unit name:	Data Structures and Algorithms	Unit ID:	
Lecturer / unit coordinator:	Valerie Maxville		
Assessment:	Assignment		

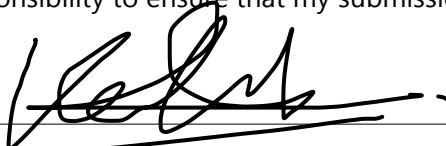
I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: _____



Date of
signature: _____

(By submitting this form, you indicate that you agree with all the above text.)

1) User Guide

When run on 'Silent Mode': The program asks the user to enter the map filename to read and its corresponding journey file. Then, possible routes to travel from start to end depending on the obstacles mentioned in the journey file are printed in ascending order, ranked by distance to "RankedRoutes.txt".

When run on 'Interactive Mode': An interactive menu as displayed in the assignment pdf is displayed to the user.

1. The user has the option to either read a .txt map file or to load a graph from a serialized file.
2. Node operations: 'find' allows user to find a node and get all its adjacent nodes displayed, 'insert' allows user to input the label name of a new node, 'delete', allows the user to delete a given node, all its links with other nodes and its edges. 'Update' can be used to change the label name of the node. NOTE: Only label names are changed, because the node doesn't store any objects or values.
3. Edge operations: find, insert, delete and update allows the user to change the distance between two nodes, security level, barriers and start/end locations of an edge respectively.
4. Parameter tweaks: allows users to multiply or divide all distances between locations in the map by the given factor.
5. Display graph: Displays the graph in the form of an adjacency matrix, with the distance between 2 locations. Users are also given the option to view it as an adjacency list.
6. Display World: All edges and connections between each location in the map are displayed to the user, with the distance, security level and barriers. The user has the option to save it to "World.txt".
7. Journey details: User is asked to enter the filename of the needed journey and journey details are stored in a string.
8. Generate routes: Possible routes to travel adhering to conditions in the journey file are calculated and stored in a string with possible paths in ranked order according to distance.
9. Display routes: String from the previous choice is displayed to the user, and also given the option to save to "RankedRoutes.txt".
10. Save network: The graph object is serialized and save to "SerializedGraph.txt"

2) Description of Classes

1. DSAGraphVertex : Used to create a graph vertex/node in the graph class.
2. DSAGraphEdge : Used to store an edge between 2 graph vertices. It stores an object field to store the weight of the edge, which is stored as a Connection object.
3. DSAGraph : Used to create a graph object for the map file. Each location is stored as a DSAGraphVertex and any details regarding a path between 2 locations is stored in a DSAGraphEdge².
4. Connection : This class contains fields to store the to/from locations, distance, security clearance required and barrier. This class used to create objects to store the weight information between 2 vertices and store it in a DSAGraphEdge.
5. Route : Used to create a Route object which contains a DSALinkedList for a route and a double field for the distance of that route. This is used when ranking the paths according to their total distance from start to end location.
6. Journey : Used to create an object to store the information of the journey obtained by reading the Journey.txt file.
7. MapFileIO : A class with static methods used to read a map file, line by line and create necessary by obtaining relevant data from each line, and add it to a graph object accordingly. Also contains methods to serialize and deserialize graph objects to/from files
8. JourneyFileIO : Again, this is a class with static methods used to read journey files and write the ranked routes, adjacency matrix and graph-world to respective files.
9. DSALinkedList : The class for linked-list. Used to store the list of vertices and edges in DSAGraph and also used when generating routes and ranking them according to distance.

3) Justification of Decision

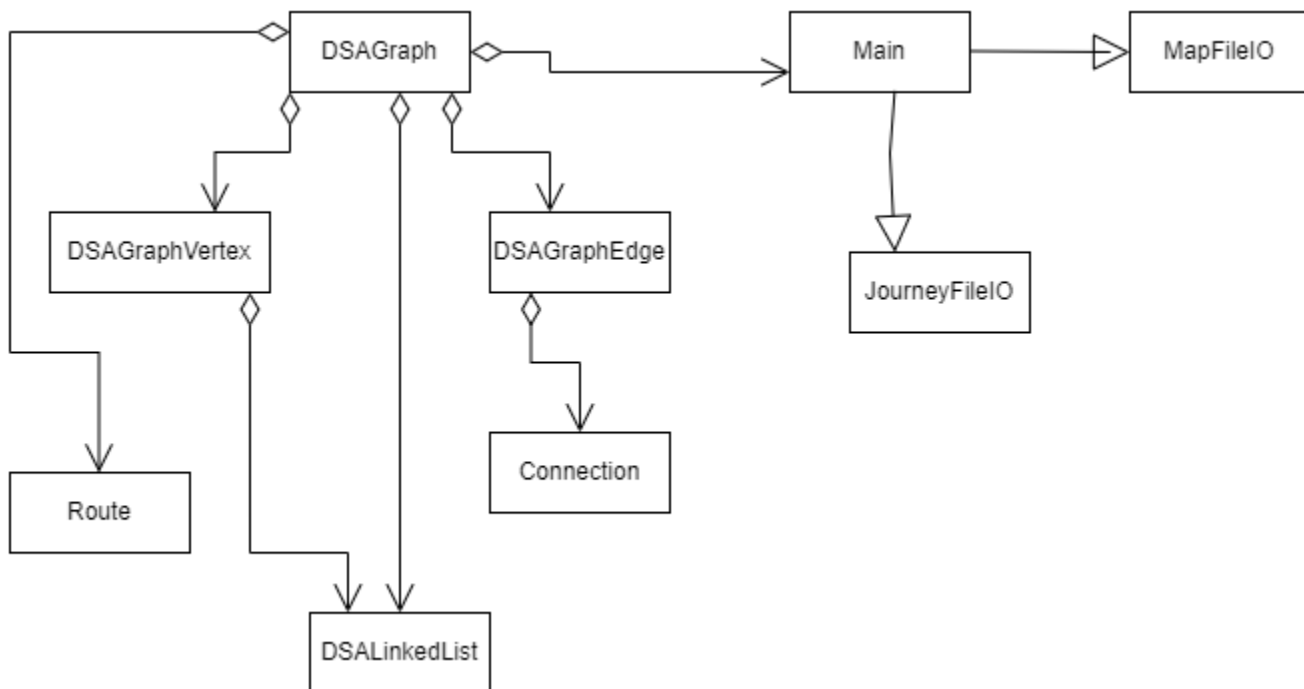
Reason for the choice of LinkedList to store vertices and edges: When using a linked list, you can always add or remove elements anytime, and there's no need to define a fixed length unlike in an array.

Reason for the use of Graph Edge: The edge class is used to store the weight between 2 locations in the map, as a Connection object, i.e the distance, security level, and barrier info.

Reason to use arrays in some cases: In some situations, for example in the method to get the distance a single path, because using an array we can get 2 consecutive nodes and calculate distance between the two, and repeat until the destination is met, in order to calculate total path distance.

4) UML Class diagram

Below shown is a simple demo of the UML class diagram. I cannot include my actual UML diagram into a single page in this document, so it is submitted as a separate pdf named "AssessmentUML.pdf".



5) Traceability Matrix

		Requirements	Design/Code	Test Status
1 Driver/Menu and Modes		1.1 System displays usage when called without arguments	whereNow.java	Passed
		1.2 System enters interactive mode with -i argument	whereNow.java	Passed
		1.3 In interactive mode, user enters command and program responds	whereNow.java	Passed
		1.4 Program requests user only for map and journey file names with -s argument is silent mode	whereNow.java	Passed
2 FileIO		2.1 To read a file, system asks for filename	whereNow.java	Passed
		2.2 User is prompted if file doesn't exist or if an error occurred	MapFileIO.java, JourneyFileIO.java	Passed
		2.3 User is given the choice to read a new Map file or to read from a pre-saved serialized file	whereNow.java	Passed
		2.4 Exception is handled if data is parsed into variables of incompatible data types when a line is split and put into variables	MapFileIO.java, JourneyFileIO.java	Passed
3 Node/Vertex Operations		3.1 Vertex is returned and its label & links are displayed when user selects find option	DSAGraph.java	Passed
		3.2 A new vertex is created and inserted when 'insert' option is selected	DSAGraph.java	Passed
		3.3 Existing vertex gets deleted when user selects delete	DSAGraph.java	Passed
		3.4 Vertex label gets updated with newly entered when update is selected	DSAGraph.java	Passed
		3.5 For above scenarios, if user enters an invalid vertex label, an error message is prompted	DSAGraph.java	Passed
4 Edge Operations		3.1 Edge is displayed when user selects find option	DSAGraph.java	Passed
		3.2 A new edge is created and inserted when 'insert' option is selected	DSAGraph.java	Passed
		3.3 Existing edge gets deleted when user selects delete	DSAGraph.java	Passed
		3.4 Edge information gets updated with newly entered info when update is selected	DSAGraph.java	Passed
5 Display the graph		3.5 For above scenarios, if user enters an invalid vertex label, an error message is prompted	whereNow.java	Passed
		5.2 Adjacency matrix/Adjacency list/World is displayed when respective choice is opted	whereNow.java, DSAGraph.java	Failed for Map_int_314.txt file
		5.3 User is given the option to save Adjacency matrix/World	whereNow.java	Passed
		5.4 If the given file doesn't exist, a new file is created accordingly and saved	JourneyFileIO.java	Passed
6 Generating and ranking routes		6.1 Routes are generated and ranked according to input from journey file	DSAGraph.java	Passed
		6.2 If there are no possible routes, a message is displayed to user	DSAGraph.java	Passed
		6.3 Route Details are saved to file if the user selects the option	DSAGraph.java	Passed
7 Save Network		Graph is saved in a serialized format to a txt file	JourneyFileIO.java	Passed

5) ShowCase

2. Scenario 1 : This scenario is where Map2.txt is read as the map file and Journey2e.txt is read as the journey file when run in INTERACTIVE mode. The opted output is the weighted adjacency matrix saved to 'AdjacencyMatrix.txt' file. Since the location names take up quite the space, locations are displayed as numbers in the adjacency matrix and a legend is afterwards displayed for the user to track specific locations. If there's a connection between 2 locations, the distance in between is displayed, else 0 is displayed, meaning there's no connection. Below is a screenshot of 'AdjacencyMatrix.txt'.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0	3.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	3.0	0	3.0	3.0	3.0	10.0	0	0	0	20.0	50.0	52.0	0	0	0	0	0	0	10.0	0	0	0	0
3	0	3.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	3.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	3.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	10.0	0	0	0	0	10.0	10.0	3.0	10.0	45.0	47.0	0	0	0	0	0	0	10.0	0	0	0	0
7	0	0	0	0	0	10.0	0	2.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	10.0	2.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	3.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	20.0	0	0	0	10.0	0	0	0	0	45.0	47.0	2.0	2.0	4.0	0	0	0	10.0	0	0	0	0
11	0	50.0	0	0	0	45.0	0	0	0	45.0	0	12.0	0	0	0	0	0	0	0	0	4.0	4.0	0
12	0	52.0	0	0	0	47.0	0	0	0	47.0	12.0	0	0	0	0	0	0	0	0	0	0	0	6.0
13	0	0	0	0	0	0	0	0	0	2.0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	2.0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	4.0	0	0	0	0	0	2.0	2.0	2.0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.0	0	0	0	0	0	0	0	0
19	0	10.0	0	0	0	10.0	0	0	0	10.0	0	0	0	0	0	0	0	0	0	2.0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	4.0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	4.0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	6.0	0	0	0	0	0	0	0	0	0	0	0

1	314.221.lab
2	314.1.ext1
3	314.220.lab
4	314.219.lab
5	314.218.lab
6	204.1.ext1
7	204.239.lab
8	204.238.lab
9	204.1.basement
10	210.1.ext1
11	402.1.ext1
12	400.1.ext1
13	210.101.LT
14	210.102.LT
15	213.1.ext1
16	213.101.LT
17	213.102.LT
18	213.103.LT
19	105.1.ext1
20	105.107.LT
21	402.218.FF
22	402.219.FF
23	400.219.FF

3. **Scenario 2** : Scenario 2 is same as above, but the adjacency matrix for Map.txt is given below:

	1	2	3	4	5	6	7	8	9
1	0	3.0	0	0	0	0	0	0	0
2	3.0	0	3.0	3.0	3.0	10.0	0	0	0
3	0	3.0	0	0	0	0	0	0	0
4	0	3.0	0	0	0	0	0	0	0
5	0	3.0	0	0	0	0	0	0	0
6	0	10.0	0	0	0	0	10.0	10.0	3.0
7	0	0	0	0	0	10.0	0	0	0
8	0	0	0	0	0	10.0	0	0	0
9	0	0	0	0	0	3.0	0	0	0

- 1 314.221.lab
- 2 314.1.ext1
- 3 314.220.lab
- 4 314.219.lab
- 5 314.218.lab
- 6 204.1.ext1
- 7 204.238.lab
- 8 204.239.lab
- 9 204.1.basement

4. **Scenario 3** : In this scenario, the ranked routes are printed for Journey2e.txt for Map2.txt, when the program is executed in SILENT mode.

204.1.basement	204.1.ext1	400.1.ext1	400.219.FF	Distance: 56.0			
204.1.basement	204.1.ext1	210.1.ext1	400.1.ext1	400.219.FF	Distance: 66.0		
204.1.basement	204.1.ext1	402.1.ext1	400.1.ext1	400.219.FF	Distance: 66.0		
204.1.basement	204.1.ext1	314.1.ext1	400.1.ext1	400.219.FF	Distance: 71.0		
204.1.basement	204.1.ext1	210.1.ext1	402.1.ext1	400.1.ext1	400.219.FF	Distance: 76.0	
204.1.basement	204.1.ext1	314.1.ext1	402.1.ext1	400.1.ext1	400.219.FF	Distance: 81.0	
204.1.basement	204.1.ext1	314.1.ext1	210.1.ext1	400.1.ext1	400.219.FF	Distance: 86.0	
204.1.basement	204.1.ext1	210.1.ext1	314.1.ext1	400.1.ext1	400.219.FF	Distance: 91.0	
204.1.basement	204.1.ext1	314.1.ext1	210.1.ext1	402.1.ext1	400.1.ext1	400.219.FF	Distance: 96.0
204.1.basement	204.1.ext1	210.1.ext1	314.1.ext1	402.1.ext1	400.1.ext1	400.219.FF	Distance: 101.0
204.1.basement	204.1.ext1	402.1.ext1	210.1.ext1	400.1.ext1	400.219.FF	Distance: 146.0	
204.1.basement	204.1.ext1	402.1.ext1	314.1.ext1	400.1.ext1	400.219.FF	Distance: 156.0	
204.1.basement	204.1.ext1	314.1.ext1	402.1.ext1	210.1.ext1	400.1.ext1	400.219.FF	Distance: 161.0
204.1.basement	204.1.ext1	210.1.ext1	402.1.ext1	314.1.ext1	400.1.ext1	400.219.FF	Distance: 166.0
204.1.basement	204.1.ext1	402.1.ext1	314.1.ext1	210.1.ext1	400.1.ext1	400.219.FF	Distance: 171.0
204.1.basement	204.1.ext1	402.1.ext1	210.1.ext1	314.1.ext1	400.1.ext1	400.219.FF	Distance: 171.0

Start location: 204.1.basement End location: 400.219.FF
Security clearance: 2 Barriers to avoid: stairs, construction

5. **Scenario 4 :** In this, Map_int_314.txt file is read and the display world option is selected, which is then written to World.txt file. Below is a snapshot of part of the text file:

From: 1	To: 2	Distance: 80.0	Security Level: 0	Barrier:
From: 2	To: 3	Distance: 40.0	Security Level: 0	Barrier:
From: 10	To: 2	Distance: 80.0	Security Level: 0	Barrier:
From: 2	To: 20	Distance: 100.0	Security Level: 0	Barrier:
From: 3	To: 4	Distance: 50.0	Security Level: 0	Barrier:
From: 11	To: 3	Distance: 80.0	Security Level: 0	Barrier:
From: 3	To: 21	Distance: 100.0	Security Level: 0	Barrier:
From: 4	To: 5	Distance: 60.0	Security Level: 0	Barrier:
From: 12	To: 4	Distance: 80.0	Security Level: 0	Barrier:
From: 4	To: 22	Distance: 100.0	Security Level: 0	Barrier:
From: 5	To: 6	Distance: 60.0	Security Level: 0	Barrier:
From: 13	To: 5	Distance: 80.0	Security Level: 0	Barrier:
From: 5	To: 23	Distance: 100.0	Security Level: 0	Barrier:
From: 6	To: 7	Distance: 40.0	Security Level: 0	Barrier:
From: 6	To: 24	Distance: 100.0	Security Level: 0	Barrier:
From: 1099	To: 6	Distance: 50.0	Security Level: 0	Barrier:
From: 7	To: 8	Distance: 70.0	Security Level: 0	Barrier:
From: 7	To: 25	Distance: 100.0	Security Level: 0	Barrier:
From: 1100	To: 7	Distance: 50.0	Security Level: 0	Barrier:
From: 9	To: 10	Distance: 20.0	Security Level: 0	Barrier:
From: 10	To: 11	Distance: 40.0	Security Level: 0	Barrier:
From: 400	To: 10	Distance: 30.0	Security Level: 0	Barrier:
From: 11	To: 12	Distance: 50.0	Security Level: 0	Barrier:
From: 356	To: 11	Distance: 50.0	Security Level: 0	Barrier:
From: 12	To: 13	Distance: 60.0	Security Level: 0	Barrier:
From: 17	To: 12	Distance: 50.0	Security Level: 0	Barrier:
From: 13	To: 14	Distance: 30.0	Security Level: 0	Barrier:
From: 1067	To: 13	Distance: 30.0	Security Level: 0	Barrier:
From: 15	To: 16	Distance: 30.0	Security Level: 0	Barrier:
From: 16	To: 17	Distance: 50.0	Security Level: 0	Barrier:
From: 1097	To: 16	Distance: 20.0	Security Level: 0	Barrier:
From: 17	To: 18	Distance: 20.0	Security Level: 0	Barrier:
From: 1098	To: 17	Distance: 20.0	Security Level: 0	Barrier:
From: 19	To: 20	Distance: 20.0	Security Level: 0	Barrier:
From: 20	To: 21	Distance: 40.0	Security Level: 0	Barrier:
From: 20	To: 401	Distance: 20.0	Security Level: 0	Barrier:
From: 21	To: 22	Distance: 50.0	Security Level: 0	Barrier:
From: 21	To: 28	Distance: 90.0	Security Level: 0	Barrier:
From: 22	To: 23	Distance: 60.0	Security Level: 0	Barrier:
From: 22	To: 29	Distance: 90.0	Security Level: 0	Barrier:
From: 23	To: 24	Distance: 60.0	Security Level: 0	Barrier:
From: 23	To: 30	Distance: 90.0	Security Level: 0	Barrier:
From: 24	To: 25	Distance: 40.0	Security Level: 0	Barrier:

6. **Scenario 6 :** In this case, the program is run on SILENT Mode. The map file is Map2.txt and the journey file is Journey2c.txt. Since there are no valid paths available avoiding the given barriers and adhering to given security levels, the following is written to the RankedRoutes.txt file.

No routes available

Start location: 204.1.basement End location: 314.220.lab
 Security clearance: 2 Barriers to avoid: stairs, construction

7. Conclusion and Future Work

Assumptions made: I assumed that 3 teams of construction workers work 3 8-hour shifts per day, so that construction takes place at any time during the day, therefore it is a barrier which is always present.

Reflection : The adjacency matrix for Map_int_314.txt file takes a very long time to load(over one hour), because it has 1400+ vertices, i.e it is considered not working. The other options do work for this particular file. However, this issue needs to be investigated and a solution should be implemented.

updateEdge method in the DSAGraph class only works for input arguments of object type Connection, which is specific to this situation/assessment only, i.e this should be improved to a way it accepts any Object and still updates the given edge in a working manner.