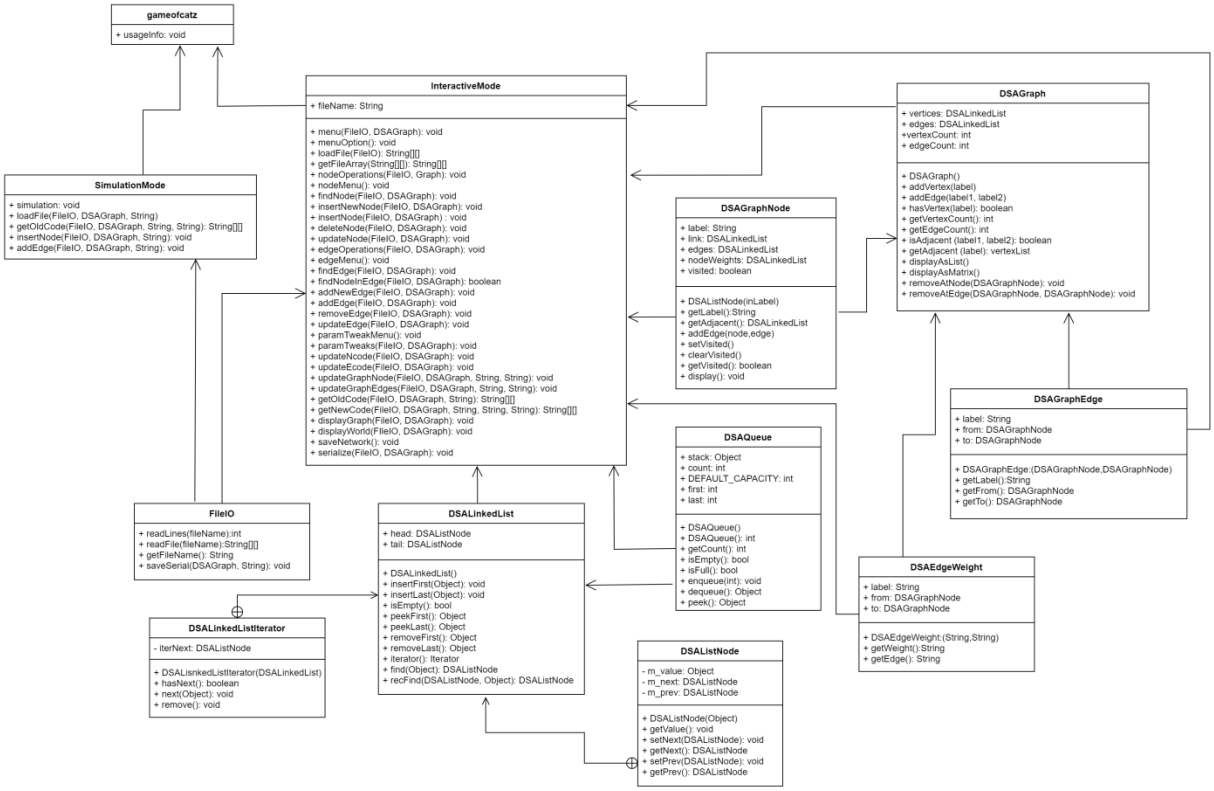
GameofCatz : Report

G.G.T.Shashen 20534534

UML Diagram


gameofcatz
+ usageInfo: void
InteractiveMode
+ fileName: String
+ menu(FileIO, DSAGraph): void
+ menuOption(): void
+ loadFile(FileIO): String[][]
+ getFileArray(String[][]): String[][]
+ nodeOperations(FileIO, Graph): void
+ nodeMenu(): void
+ findNode(FileIO, DSAGraph): void
+ insertNewNode(FileIO, DSAGraph): void
+ insertNode(FileIO, DSAGraph) : void
+ deleteNode(FileIO, DSAGraph): void
+ updateNode(FileIO, DSAGraph): void
+ edgeOperations(FileIO, DSAGraph): void
+ edgeMenu(): void
+ findEdge(FileIO, DSAGraph): void
+ findNodeInEdge(FileIO, DSAGraph): boolean
+ addNewEdge(FileIO, DSAGraph): void
+ addEdge(FileIO, DSAGraph): void
+ removeEdge(FileIO, DSAGraph): void
+ updateEdge(FileIO, DSAGraph): void
+ paramTweakMenu(): void
+ paramTweaks(FileIO, DSAGraph): void
+ updateNcode(FileIO, DSAGraph): void
+ updateEcode(FileIO, DSAGraph): void
+ updateGraphNode(FileIO, DSAGraph, String, String): void
+ updateGraphEdges(FileIO, DSAGraph, String, String): void
+ getOldCode(FileIO, DSAGraph, String): String[][]
+ getNewCode(FileIO, DSAGraph, String, String, String): String[][]
+ displayGraph(FileIO, DSAGraph): void
+ displayWorld(FileIO, DSAGraph): void
+ saveNetwork(): void
+ serialize(FileIO, DSAGraph): void
SimulationMode
+ simulation: void
+ loadFile(FileIO, DSAGraph, String)
+ getOldCode(FileIO, DSAGraph, String, String): String[][]
+ insertNode(FileIO, DSAGraph, String): void
+ addEdge(FileIO, DSAGraph, String): void
DSAGraph
+ vertices: DSALinkedList
+ edges: DSALinkedList
+vertexCount: int
+ edgeCount: int
+ DSAGraph()
+ addVertex(label)
+ addEdge(label1, label2)
+ hasVertex(label): boolean
+ getVertexCount(): int
+ getEdgeCount(): int
+ isAdjacent (label1, label2): boolean
+ getAdjacent (label): vertexList
+ displayAsList()
+ displayAsMatrix()
+ removeAtNode(DSAGraphNode): void
+ removeAtEdge(DSAGraphNode, DSAGraphNode): void
DSAGraphNode
+ label: String
+ link: DSALinkedList
+ edges: DSALinkedList
+ nodeWeights: DSALinkedList
+ visited: boolean
+ DSAListNode(inLabel)
+ getLabel():String
+ getAdjacent(): DSALinkedList
+ addEdge(node,edge)
+ setVisited()
+ clearVisited()
+ getVisited(): boolean
+ display(): void
DSAGraphEdge
+ label: String
+ from: DSAGraphNode
+ to: DSAGraphNode
+ DSAGraphEdge:(DSAGraphNode,DSAGraphNode)
+ getLabel():String
+ getFrom(): DSAGraphNode
+ getTo(): DSAGraphNode
DSAQueue
+ stack: Object
+ count: int
+ DEFAULT_CAPACITY: int
+ first: int
+ last: int
+ DSAQueue()
+ DSAQueue(): int
+ getCount(): int
+ isEmpty(): bool
+ isFull(): bool
+ enqueue(int): void
+ dequeue(): Object
+ peek(): Object
DSAEdgeWeight
+ label: String
+ from: DSAGraphNode
+ to: DSAGraphNode
+ DSAEdgeWeight:(String,String)
+ getWeight():String
+ getEdge(): String
FileIO
+ readLines(fileName):int
+ readFile(fileName):String[][]
+ getFileName(): String
+ saveSerial(DSAGraph, String): void
DSALinkedList
+ head: DSAListNode
+ tail: DSAListNode
+ DSALinkedList()
+ insertFirst(Object): void
+ insertLast(Object): void
+ isEmpty(): bool
+ peekFirst(): Object
+ peekLast(): Object
+ removeFirst(): Object
+ removeLast(): Object
+ iterator(): Iterator
+ find(Object): DSAListNode
+ recFind(DSAListNode, Object): DSAListNode
DSALinkedListIterator
- iterNext: DSAListNode
+ DSALisnkedListIterator(DSALinkedList)
+ hasNext(): boolean
+ next(Object): void
+ remove(): void
DSAListNode
- m_value: Object
- m_next: DSAListNode
- m_prev: DSAListNode
+ DSAListNode(Object)
+ getValue(): void
+ setNext(DSAListNode): void
+ getNext(): DSAListNode
+ setPrev(DSAListNode): void
+ getPrev(): DSAListNode

User Guide:

The program lets you visualize a graph from a given file. User can enter a data file with information about the graph, the program then takes the user given file, process it and output processed information to the user.

User has 2 options to choose from

- Interactive Mode
- Simulation Mode

Interactive Mode

- Lets the user input a data file and it automatically inserts the data to the graph, then the user has the choice to edit the graph. The user can edit, add, remove, update the graph nodes, edges. The user can edit the graph parameter values to display the graph as the user likes. The user has the ability to generate routes from the processed data file to get the ranked path from the graph. The program also lets the user to save the whole network to a file of choice.

Simulation Mode

- In this mode, the user cannot interact or edit the graph from the data file. This mode does not show a menu but rather takes input from the command line as arguments. The user can enter the data file which contains the graph information and the output file name to store the ranked path of the processed graph from the data file.

Features

- Visualize graph
- Edit Nodes
- Edit Edges
- Display Routes
- Generate Routes
- Save Graph Network

Description of Classes

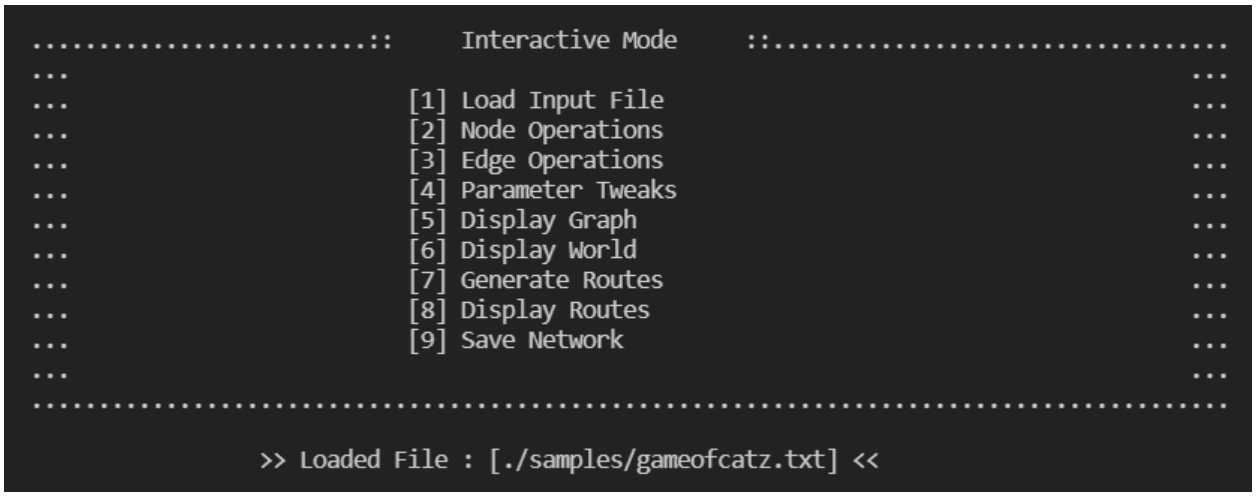
Class Names

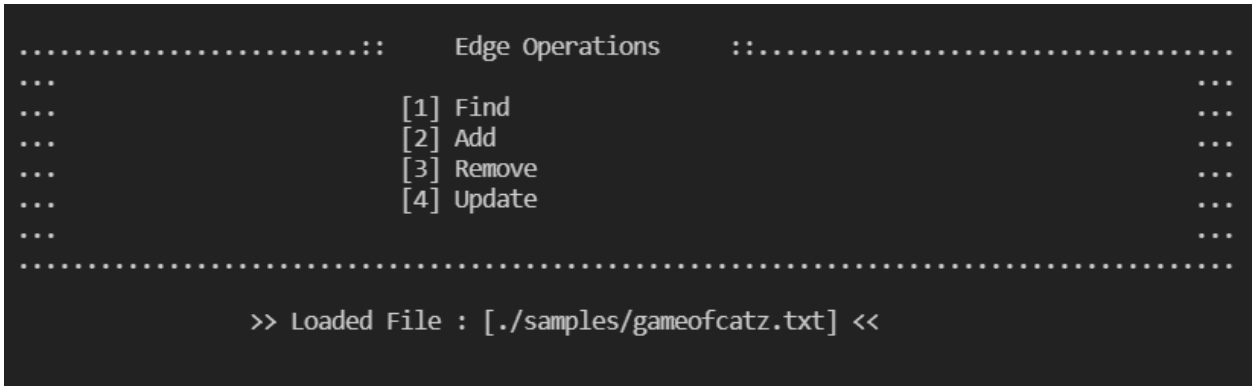
- gameofcatz
    - Contains the main function of the whole program which associates with different classes
- InteractiveMode
    - This class contains the whole logic of the program. This class is quite large in size because it contains all the functionalities of the Interactive Mode which uses many different other classes. This class also contains the user interface of the whole Interactive Mode.
- SimulationMode
    - This class contains the simulation part of the program, which doesn't have many functions since this mode just give the user the ranked path of the graph and also this class does not contain any user input nor any user interface, this only takes command lines arguments
- DSAGraph
    - This class contains the functionality to produce the graph. This class is taken from DSA Practical 6.
- DSAGraphNode
    - This class handles the node functions to represent in the graph. This class is taken from DSA practical 6
- DSAGraphEdge
    - This class handles the edge function in the graph, this associate with the DSA Graph, DSAGraphNode classes. This class is taken from DSA practical 6
- DSAEdgeWeight
    - This class handles the edge weights in the graph, this associate with all the DSA Graph classes. This class is taken from DSA practical 6
- DSALinkedList
    - This class contains the linked list implementation and store the values in the graph class. This class is taken from DSA practical 6
- DSAListNode
    - This class is a private inner class of the DSALinkedList class which contains the nodes where the values are store when created.
- DSALinkedListIterator
    - This class is a private inner class of the DSALinkedList class which contains the iteratable interface and its functionalities
- DSAQueue
    - Used for the depth first search algorithm to store values
- FileIO
    - This class contains the file input and output to get the data from the data file and also output the data to a new data file

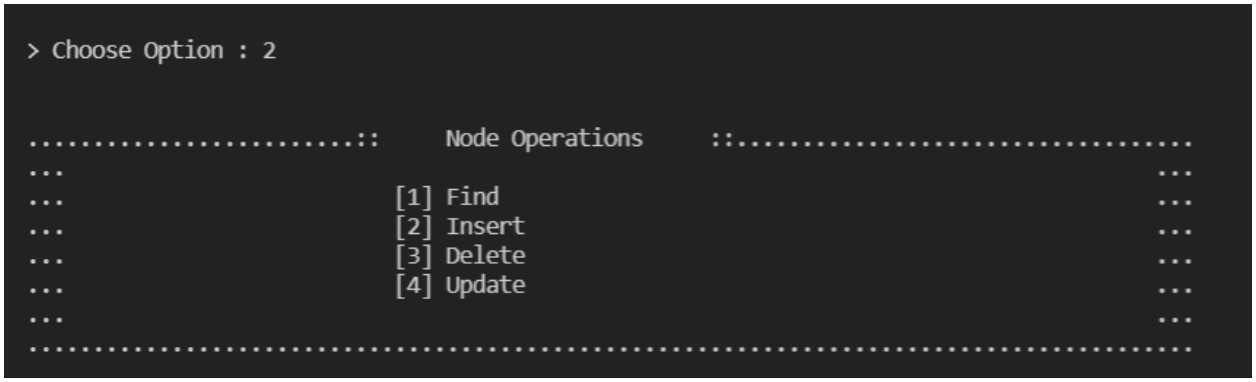
Traceability Matrix

		Requirements	Design/Code	Test
1	Driver/Menu & Modes	1.1 Displays usage if called without arguments	gameofcatz.main()	PASSED
		1.2 System displays the interactive menu when enter –i	gameofcatz.main()	PASSED
		1.3 System displays the simulation menu when enter –s input output	gameofcatz.main()	PASSED
2	Load Data	2.1 System correctly input the data	InteractiveMode.readFile()	PASSED
3	Node Operations	3.1 Correctly operate the node operations	InteractiveMode.nodeOperations()+all related methods inside it	PASSED
4	Edge Operations	4.1 Correctly operates edge operations	InteractiveMode.edgeOperations()+all related methods inside it	PASSED
5	Parameter Tweaks	5.1 Correctly lets the user edit and edit data	InteractiveMode.paramTweaks() + all related methods inside it	PASSED
6	Display Graph	6.1 Correctly displays the matrix graph with any sample file	InteractiveMode.displayGraph()	PASSED
7	Display World	7.1 Correctly displays the world as a list	InteractiveMode.displayWorld()	PASSED
8	Generate Routes	8.1 Generate Routes	InteractiveMode.depthFirstSearch()	-
9	Display Routes	[NOT IMPLEMENTED[	[NOT IMPLEMENTED]	-
10	Save Network	10.1 Correctly output a serialized file with user's choice filename	Interactive.serialize()	PASSED

Showcase


Interactive Mode
[1] Load Input File
[2] Node Operations
[3] Edge Operations
[4] Parameter Tweaks
[5] Display Graph
[6] Display World
[7] Generate Routes
[8] Display Routes
[9] Save Network
>> Loaded File : [./samples/gameofcatz.txt] <<


Edge Operations
[1] Find
[2] Add
[3] Remove
[4] Update
>> Loaded File : [./samples/gameofcatz.txt] <<


> Choose Option : 2
Node Operations
[1] Find
[2] Insert
[3] Delete
[4] Update

```
> Choose Option : 5

0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 1 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 1 1 0 0 0
0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0
```

Matrix for gameofcatz2.txt

```
> Choose Option : 6


Node : A [0] -> Edges : B [1] E [1]

Node : B [1] -> Edges : A [1] E [1] F [1] C [1]

Node : C [0] -> Edges : B [1] F [1] G [1] D [1]

Node : D [0] -> Edges : C [1] G [1]

Node : E [0] -> Edges : A [1] B [1] F [1] H [1]

Node : F [0] -> Edges : B [1] C [1] E [1] G [1] H [1] I [1]

Node : G [-1] -> Edges : C [1] D [1] F [1] I [1]

Node : H [100] -> Edges : E [1] F [1] I [1] J [1]

Node : I [-1] -> Edges : F [1] G [1] H [1] J [1]

Node : J [0] -> Edges : H [1] I [1]
```

Display of world for gameofcatz.txt