

**DATA STRUCTURE AND  
ALGORITHM  
HW08**

**MOHAMMAD ASHRAF  
YAWAR**

**161044123**

## SCREEN SHOTS:

The screenshot shows the IntelliJ IDEA IDE with a Java file named `part02 - TestClass.java`. The code implements a graph structure with methods for adding and deleting edges, and searching for paths. A tooltip for `Key Promoter X` is visible in the bottom right corner, indicating that the `Rerun "TestClass"` command was pressed 541 times.

```
part02 - TestClass.java

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Run: /home/asraaf/.jdk/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar:38637:/snap/intellij-idea-community/232/bin -Dfile.encoding=UTF-8 -classpath /home/asraaf/Desktop/6_SEMESTER/1)Data
***** UNDIRECTED GRAPH *****
Creating Linked List graph...
Inserting Edges...

*****
using Row node:
[0,0]-->[0,4]-->[0,1]-->[0,2]
[1,0]-->[1,4]-->[1,3]
[2,0]-->[2,2]-->[2,4]
[3,1]-->[3,3]-->[3,4]
[4,1]-->[4,2]-->[4,3]
*****
delete an edge (4,2)...
*****
using Row node:
[0,0]-->[0,4]-->[0,1]-->[0,2]
[1,0]-->[1,4]-->[1,3]
[2,0]-->[2,2]-->[2,4]
[3,1]-->[3,3]-->[3,4]
[4,1]-->[4,3]
*****
breadth first search...
stating numeric
[4, 1, 3, 0, 2]

depth first search...
starting number:2
parent: [3, 1, 4, 0, 2]
inserting Vertex...
*****
using Row node:
[0,0]-->[0,4]-->[0,1]-->[0,2]
[1,0]-->[1,4]-->[1,3]
[2,0]-->[2,2]-->[2,4]
[3,1]-->[3,3]-->[3,4]
[4,1]-->[4,3]
*****
iterating edge 2...
true
0
2
```

Key Promoter X  
Command `Rerun "TestClass"` pressed 541...  
'Ctrl+F5' (Don't show again)

part02 - TestClass.java

FileEditViewNavigateCodeAnalyzeRefactorBuildRunToolsVCSWindowHelp

Run: TestClass

4,1<-->[4,3]

iterating edge 2...

true

0

2

4

iterating edge 4...

has next true

1

3

\*\*\*\*\* DIRECTED GRAPH \*\*\*\*\*

Creating linked list graph...

inserting Edges...

\*\*\*\*\*

using Row node:

0,0-->0,4

1,0-->1,4

2,0-->2,2-->2,4

3,1-->3,3

4,1-->4,2-->4,3

\*\*\*\*\*

delete an edge 4,2--

\*\*\*\*\*

using Row node:

0,0-->0,4

1,0-->1,4

2,0-->2,2-->2,4

3,1-->3,3

4,1-->4,3

\*\*\*\*\*

breath first search...

stating number:4

4, 1, 3, 0

depth first search...

starting number:2

parents [1, 3, 4, 0, 2]

inserting Vertex...

\*\*\*\*\*

using Row node:

```
part02 - TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
7  using Row node:
10  [0,0]-->[0,4]
11  [1,0]-->[1,4]
12  [2,0]-->[2,2]-->[2,4]
13  [3,1]-->[3,3]
14  [4,1]-->[4,3]
15  *****
16  breath first search...
17  stating number:4
18  [4, 1, 3, 0]
19
20  depth first search...
21  starting number:2
22  parent: [1, 3, 4, 0, 2]
23  inserting Vertex...
24  *****
25  using Row node:
26  [0,0]-->[0,4]
27  [1,0]-->[1,4]
28  [2,0]-->[2,2]-->[2,4]
29  [3,1]-->[3,3]
30  [4,1]-->[4,3]
31
32  *****
33  iterating edge 2...
34  true
35  0
36  2
37  4
38  iterating edge 4...
39  has next :true
40  1
41  3
42  is edge testing(2,4)...
43  true
44  is edge testing(0,3)...
45  false
46  is edge testing(2,4)...
47  true
48
49  Process finished with exit code 0
Event Log
LF UTF-8 AWS: No credentials selected 4 spaces master
```

# SOLUTION APPROACH:

- TestClass is the main test class which creates graph and tests all the methods.
- LinkedListGraph Class extends AbstractGraph and adds more necessary methods in LinkedListGraph.
- while creating graph I think of it as 2D linkedlist which has row and columns same as numV which the number of edge that initial graph will have. later on we can increase our numV and number of edge in our graph.
- when inserting or deleting any edge or vertex I first check its graph type whether directed graph or undirected graph and act accordingly.
- I also added ListGraph and MatrixGraph Class because our one function named **createGraph** make use of both these classes and I also added an option addition to the **createGraph** function which is creating graph for my LinkedListGraph as well.

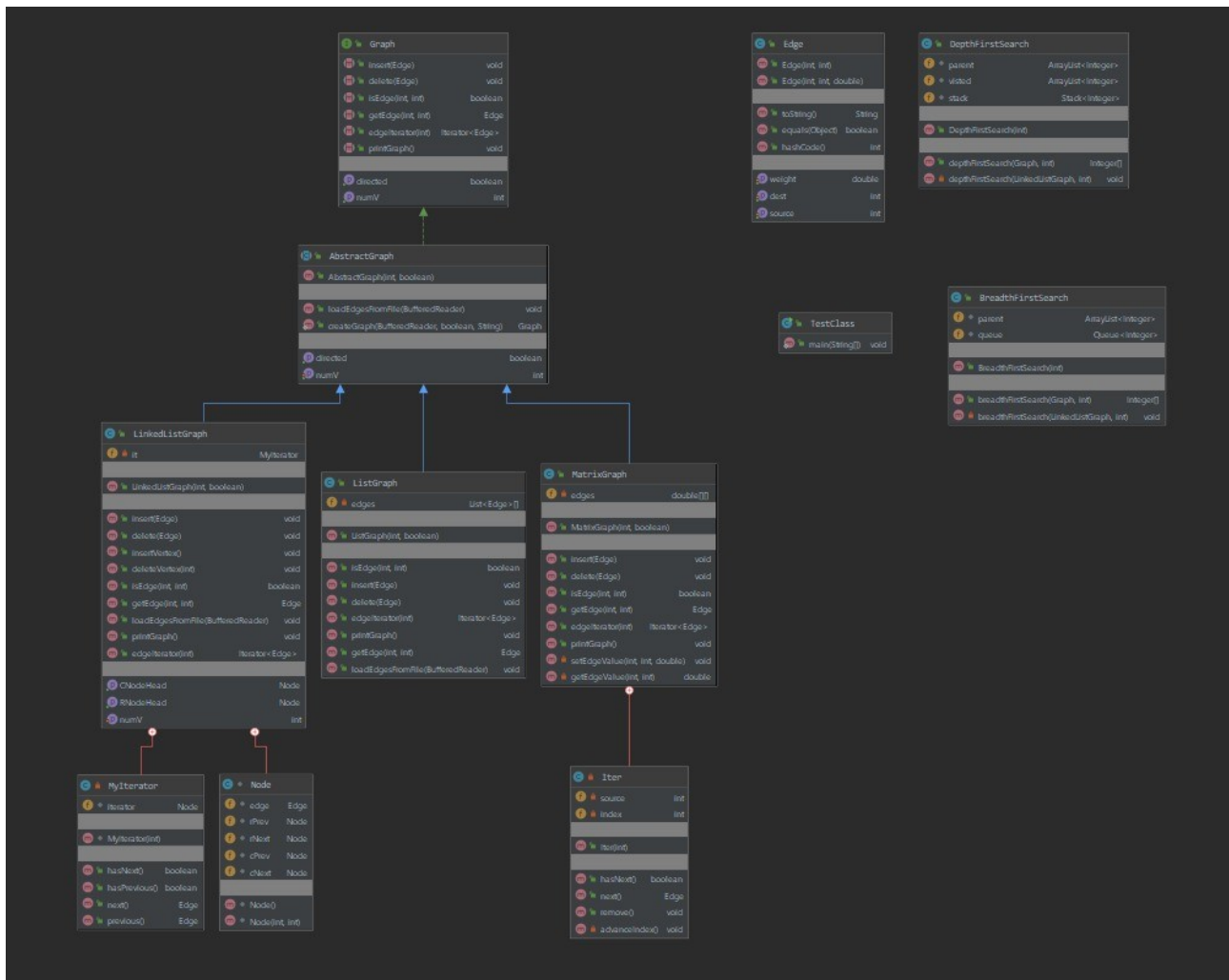
# TEST CASES:

TEST CASE ID	TEST SCENARIO	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
0	Create directed graph	5	Directed graph with 5 edges	same	pass
1	Create undirected graph	5	Undirected graph with 5 edges	Same	pass
2	Insert edge with values between 0 and 4 as source or destination	0,1,2,3,4	Insert edge	Insert edge into graph	Pass
3	Insert edge with values bigger than 4 as source or destination	$N > 4$ where N is any values bigger than 4	Not insert edge and give error	Yes	pass
4	Delete an edge which is exist in the graph	delete(source = 4, destination = 2)	Delete and unbound the edge from the graph	As expected	Pass
5	Delete an edge which does not exist in the graph	Delete(5,3);	Error while deleting	As expected	Pass
6	Insert vertex in to graph	insertVertex( )	Add a new vertex and increment the numV of the graph	As expected	Pass
7	Breath first search	Do breath first search		As expected	Pass

		and return the order of visit as an int array			
8	Depth first search	Do depth first search		As expected	Pass
9	Test iterator for graph	Iterate through an edge		As expected	Pass
10	Find if a particular source and destination values are edge in the graph	IsEdge(2,4)	True	True	Pass
11	Find if a particular source and destination values are edge in the graph	IsEdge(0,3)	False	False	Pass
12	Find if a particular source and destination values are edge in the graph	IsEdge(4,4)	False	False	pass

# PART02

## CLASS DIAGRAM:



AuxiliaryClass	
f	visited
f	listGraph
m	AuxiliaryClass()
m	readFromFile()
m	constructGraph(ArrayList<ArrayList<String>>, int, int)
m	cont(int, int)
m	printMaze(ArrayList<ArrayList<String>>)
P	numV

Edge	
m	Edge(int, int)
m	Edge(int, int, double)
m	toString()
m	equals(Object)
m	hashCode()
P	weight
P	dest
P	source

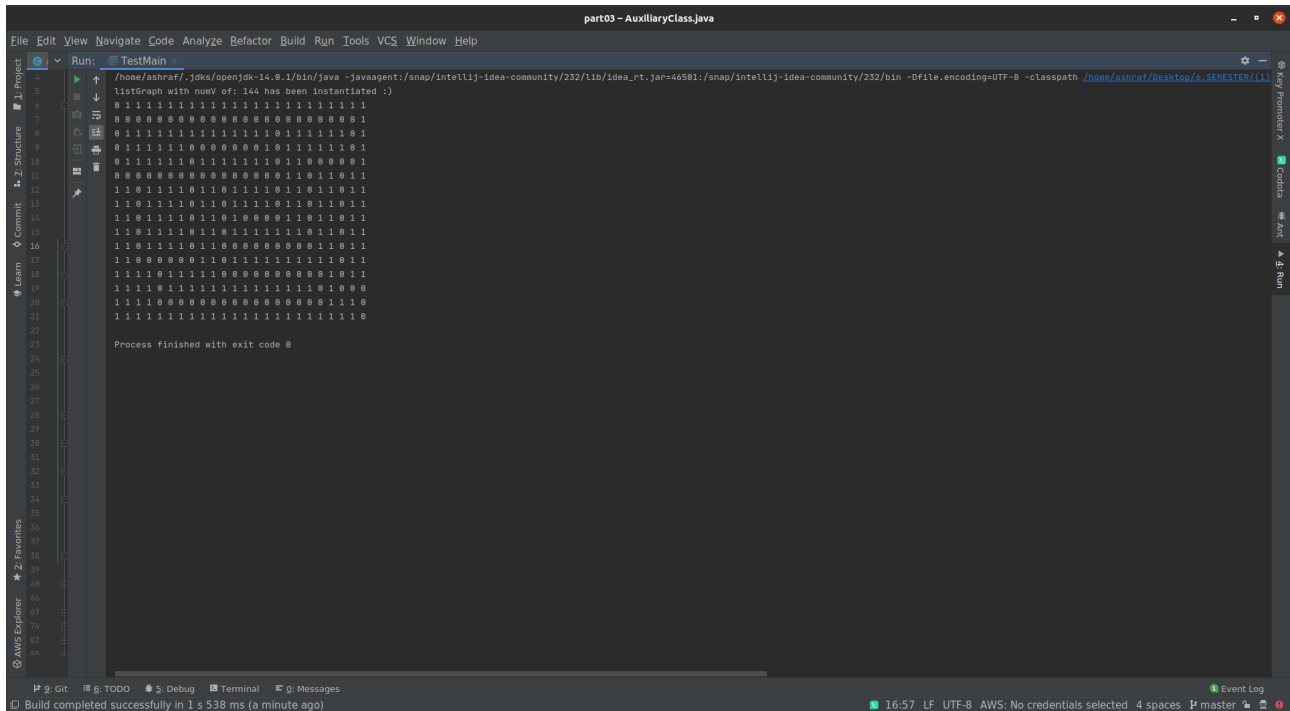
ListGraph	
f	edges
m	ListGraph(int, boolean)
m	isEdge(int, int)
m	insert(Edge)
m	edgeIterator(int)
m	getEdge(int, int)
m	loadEdgesFromFile(BufferedReader)

Graph	
m	insert(Edge)
m	isEdge(int, int)
m	getEdge(int, int)
m	edgeIterator(int)
P	directed
P	numV

AbstractGraph	
m	AbstractGraph(int, boolean)
m	loadEdgesFromFile(BufferedReader)
P	directed
P	numV

TestMain	
m	main(String[])

# SCREEN SHOTS:



The screenshot shows an IDE window titled "part03 - AuxiliaryClass.java". The "Run" window displays the output of a Java program. The output consists of a grid of binary digits (0s and 1s) arranged in a pattern that resembles a stylized 'X' or a similar geometric shape. The grid is 14 columns wide and 14 rows high. The first row is all 1s. The second row is 1s followed by 13 0s. The third row is 1s followed by 12 0s. The fourth row is 1s followed by 11 0s. The fifth row is 1s followed by 10 0s. The sixth row is 1s followed by 9 0s. The seventh row is 1s followed by 8 0s. The eighth row is 1s followed by 7 0s. The ninth row is 1s followed by 6 0s. The tenth row is 1s followed by 5 0s. The eleventh row is 1s followed by 4 0s. The twelfth row is 1s followed by 3 0s. The thirteenth row is 1s followed by 2 0s. The fourteenth row is 1s followed by 1 0. The pattern is symmetric about the center. Below the grid, the text "Process finished with exit code 0" is displayed. The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help), a toolbar, and a sidebar with "Project", "Structure", "Commit", "Learn", "Favorites", and "AWS Explorer". The status bar at the bottom shows "Build completed successfully in 1 s 538 ms (a minute ago)" and "16:57 LF UTF-8 AWS: No credentials selected 4 spaces master".

```
part03 - AuxiliaryClass.java
Run: TestMain
/home/ashraf/.jdk/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=46501:/snap/intellij-idea-community/232/bin -Dfile.encoding=UTF-8 -classpath /home/ashraf/Desktop/6_SENSETE6(1)
listGraph with numV of: 144 has been instantiated :)
01111111111111111111111111111111
00000000000000000000000000000001
01111111111111111111111111111101
0111111000000001011111111101
0111111011111111111111000001
000000000000000000011011011
110111101101111111011011011
1101111011011111011011011011
1101111011010000011011011
11011110110111111111011011
110111101100000000011011
1100000011011111111111011
1110111111100000000001011
11110111111111111111101000
1111000000000000000001110
11111111111111111111111110

Process finished with exit code 0
Build completed successfully in 1 s 538 ms (a minute ago)
16:57 LF UTF-8 AWS: No credentials selected 4 spaces master
```



# **TEST CASES:**