Software Testing Project Report

Kanigiri Naveen (IMT2019039)
Srivishnu Sunku (IMT2019086)

CFG –

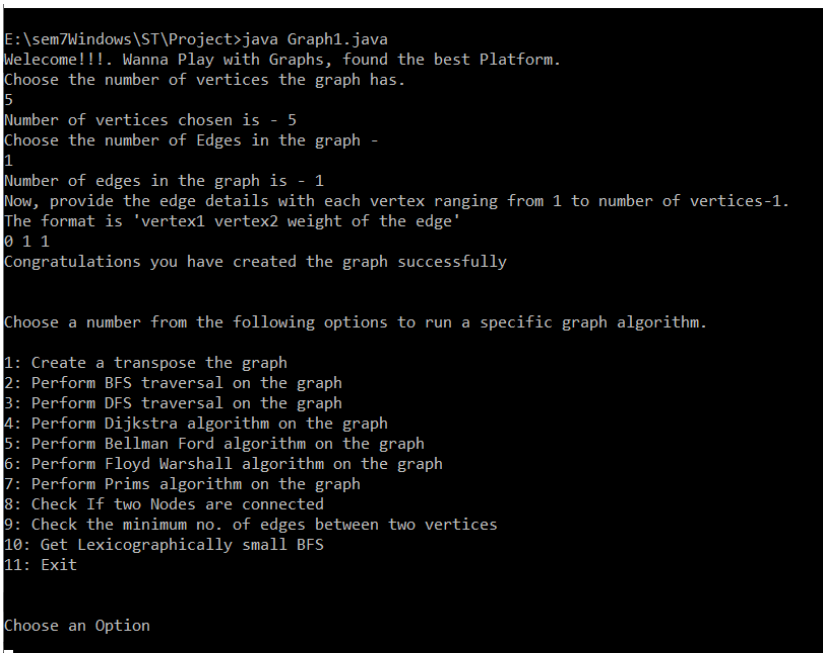
Control Flow Graphs are used to represent the flow of control in the code. Control flow graphs are used for testing to test the control flow of the code. Several test criteria can be applied on CFG's.

The goal of the project is to write and test the Edge coverage and Prime path coverage for source code.

Detailing of Project–

The link to the project directory is found here - Github Repo

The Source code for the project is a terminal based menu interface to execute different graph algorithms. The terminal menu interface looks as follows -



You are asked to provide the details of the graph initially. Firstly, the number of vertices should be mentioned. Secondly, the number of edges in the graph. Finally, it is required to enter all the edges along with the edge weights. After entering the graph information, the menu to execute different graph algorithms pops up.

The source code contains ten methods/functions enumerated as follows –

Method/Function name	Functionality
Transpose	Transposes the given graph.
DFS (Depth First Search)	Simple DFS traversal for the graph
BFS(Breadth First Search)	Simple BFS traversal for the graph.
Dijkstra's algorithm	Finds the shortest paths between nodes in a graph.
FloydWarshall	Finds shortest paths in a directed weighted graph with positive or negative edge weights.
Bellman-Ford's algorithm	Computes shortest paths from a single source vertex to all of the other vertices in a weighted graph.
Prims (Minimum spanning tree)	Finds a minimum spanning tree for a weighted undirected graph
isReachable	Checks whether the destination vertex is reachable from the source vertex.
minEdgeBFS	Computes minimum number of edges between two vertices.
printLexoSmall	Returns BFS in lexicographic order from the start node.

Testing Strategy Used – Control Flow Graphs

Testing Tools Used –

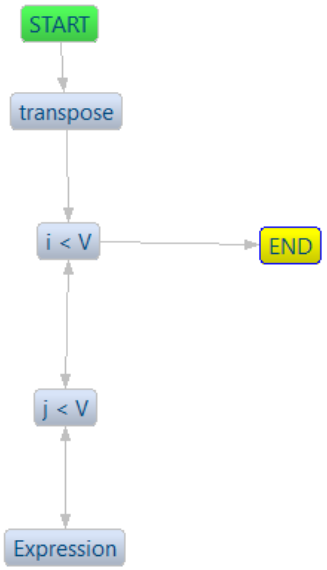
1. JUnit5 for running the test cases
2. CFG generator in eclipse for generating the cfg.
3. A web application to simplify the cfg - Graph Editor
4. A web application that generates Test Requirements and Test Paths for different coverage coverage criteria - Graph Coverage Web Application

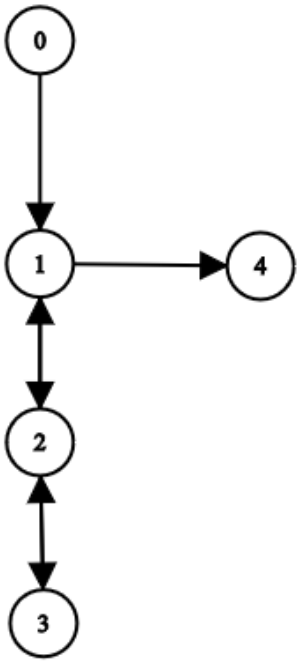
Testing Procedure –

1. A source code that is rich in if conditions and nested loops is written.
2. CFG for each function is generated using eclipse extension.
3. The CFG generated has many redundant nodes which is again simplified using a web application.
4. The simplified CFG is then used to generate the Test Requirements and Test Paths for Edge Coverage and Prime Path Coverage.
5. Of the test paths that are generated, there are infeasible test paths which cannot be executed by any input. Those are mentioned as infeasible. For the test cases that are feasible, Test case is written in JUnit.
6. For all the Test paths, if the path is infeasible, it is mentioned infeasible else the test case number in JUnit is mentioned.
7. The Test requirements are mentioned in the document here - TR Document

The results for Feasibility for different test paths for different functions and if they are feasible the JUnit test case numbers are mentioned as follows –

1. Transpose –





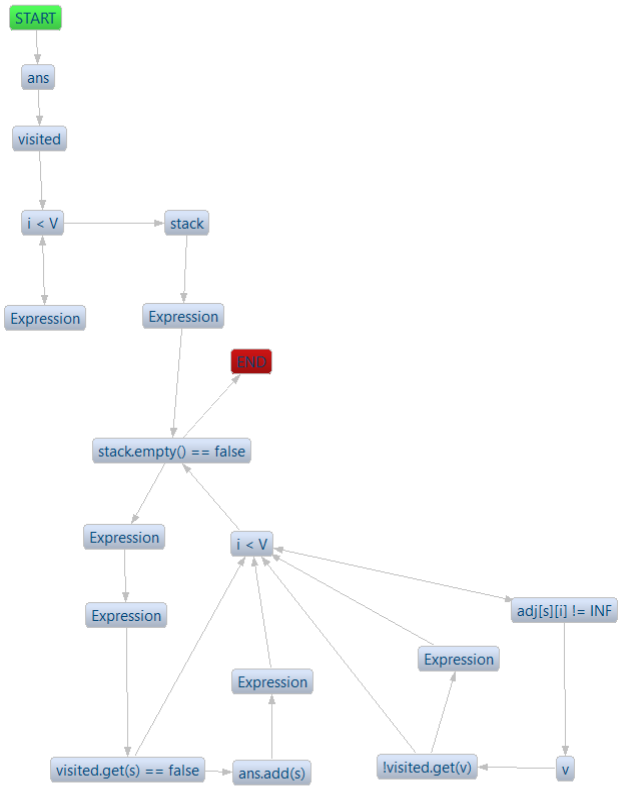
a. Edge coverage -

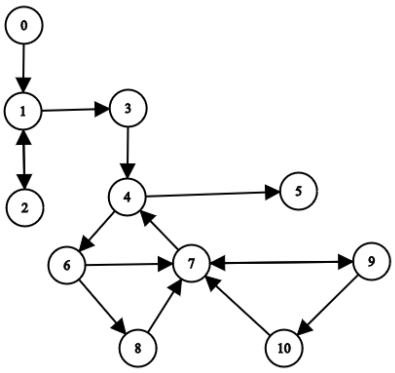
Test path	Feasibility	Test case number in Junit
[0,1,2,1,4]	No	-
[0,1,2,3,2,1,4]	Yes	1

b. Prime Path -

Test path	Feasibility	Test case number in Junit
[0,1,2,1,2,1,4]	No	-
[0,1,2,3,2,3,2,1,4]	Yes	2
[0,1,4]	Yes	3
[0,1,2,3,2,1,4]	Yes	1

2. DFS –





a. Edge coverage -

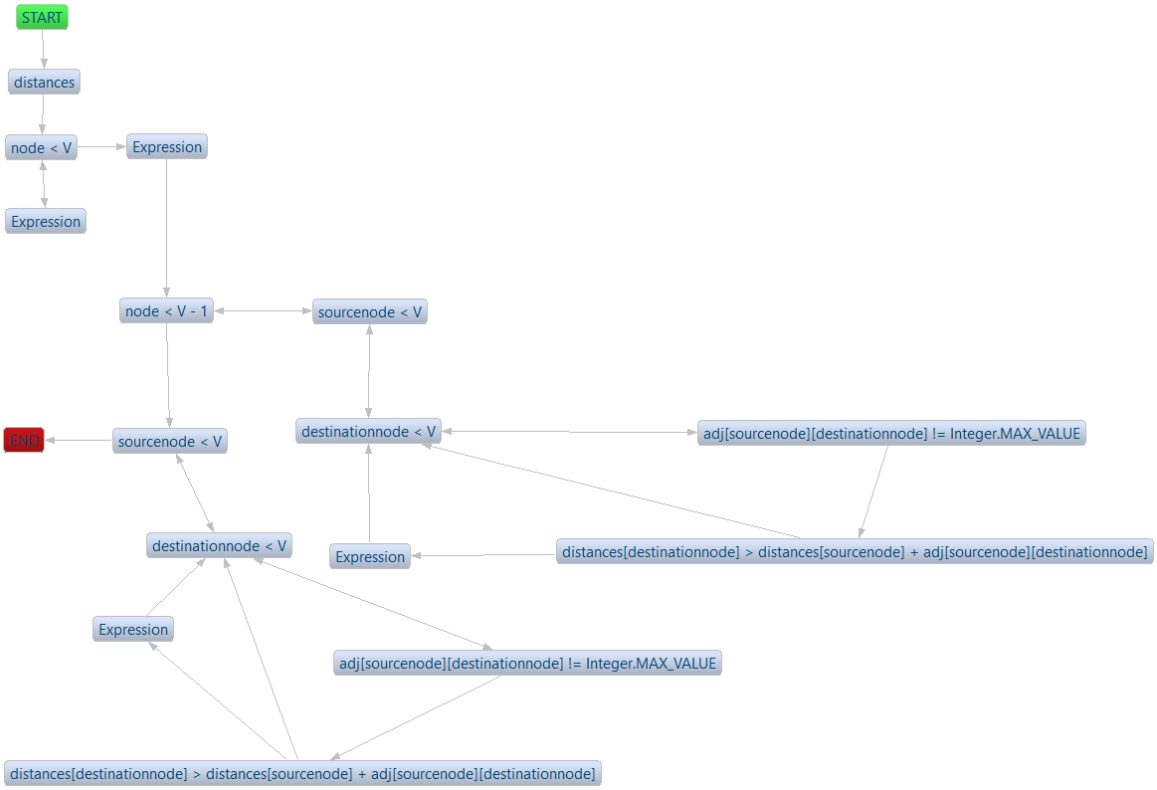
Test paths	Feasibility	Test case number in Junit
[0,1,2,1,3,4,5,6,7,5,13]	No	-
[0,1,3,4,5,6,8,7,9,7,5,13]	No	-
[0,1,3,4,5,6,7,9,10,11,12,7,9,10,11,7,5,13]	No	-

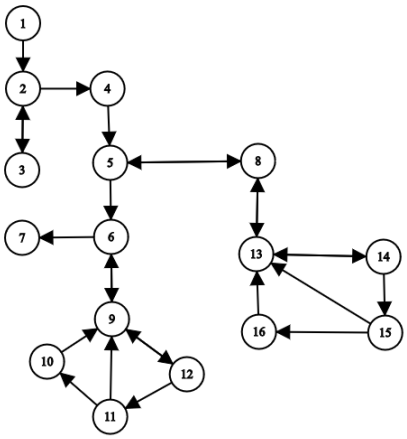
b. Prime Path -

Test case	Feasibility	Test case number in Junit
[0,1,3,4,5,6,7,9,10,11,12,7,9,10,11,7,9,10,11,7,5,6,8,7,5,6,7,5,6,7,5,13]	No	-
[0,1,3,4,5,6,7,9,7,9,7,5,13]	No	-
[0,1,2,1,2,1,3,4,5,13]	No	-
[0,1,2,1,3,4,5,13]	No	-
[0,1,3,4,5,6,8,7,5,6,8,7,5,13]	No	-
[0,1,3,4,5,6,8,7,5,13]	No	-
[0,1,3,4,5,6,7,9,7,5,6,8,7,5,13]	No	-
[0,1,3,4,5,13]	No	-
[0,1,3,4,5,6,7,9,10,11,12,7,9,10,11,12,7,5,13]	No	-
[0,1,3,4,5,6,7,9,10,11,7,5,13]	No	-
[0,1,3,4,5,6,7,9,10,11,12,7,5,13]	No	-
[0,1,3,4,5,6,7,9,10,11,12,7,5,6,8,7,5,13]	No	-
[0,1,2,1,3,4,5,6,8,7,9,10,11,7,5,13]	Yes	8

Note: Test case 9 achieves maximum possible edge coverage.

3. Bellman-Ford –





a. Edge coverage -

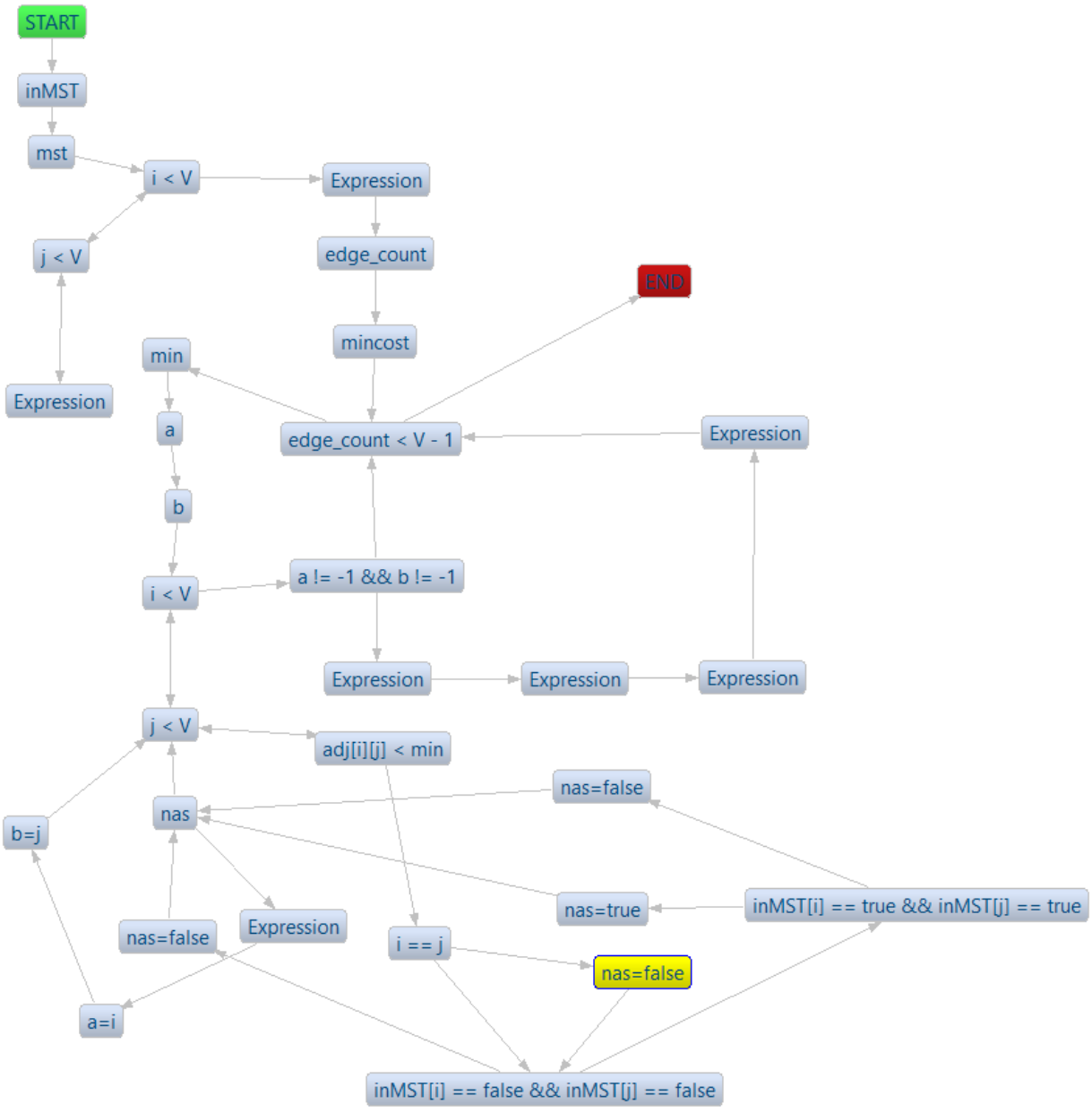
Test paths	Feasibility	Test case number in Junit
[1,2,4,5,8,13,14,15,16,13,14,15,13,14,13,14,13,8,13,8,5,6,7]	No	-
[1,2,4,5,6,9,12,11,10,9,12,9,6,7]	No	-
[1,2,4,5,6,9,6,9,6,7]	No	-
[1,2,4,5,8,5,8,5,6,7]	No	-
[1,2,3,2,4,5,6,7]	No	-
[1,2,4,5,6,9,12,11,9,6,7]	No	-

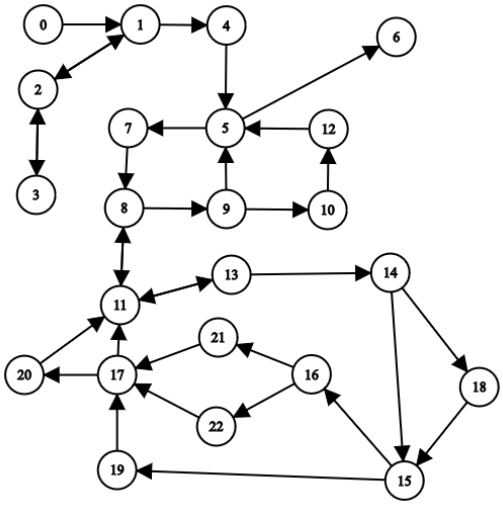
b. Prime Path -

Test path	Feasibility	Test case number in Junit
[1,2,4,5,8,13,8,13,14,13,14,15,13,14,15,16,13,14,15,16,13,14,15,13,8,5,6,9,12,11,10,9,12,11,9,12,11,9,6,7]	No	-
[1,2,4,5,6,9,6,9,6,7]	No	-
[1,2,4,5,8,5,8,5,6,7]	No	-
[1,2,3,2,4,5,6,7]	No	-
[1,2,3,2,3,2,4,5,6,7]	No	-
[1,2,4,5,6,9,12,9,12,9,12,9,12,9,6,7]	No	-
[1,2,4,5,6,9,12,11,10,9,12,11,10,9,6,7]	No	-
[1,2,4,5,6,7]	No	-

[1,2,4,5,6,9,12,11,10,9,6,7]	No	-
[1,2,4,5,8,13,14,13,8,5,6,7]	No	-
[1,2,4,5,8,13,14,15,13,8,5,6,7]	No	-
[1,2,4,5,8,13,14,15,16,13,8,5,6,7]	No	-
[1,2,3,2,4,5,8,13,14,15,16,13,8,5,6,7]	No	-
[1,2,4,5,8,13,14,13,8,5,6,9,12,11,10,9,6,7]	No	-
[1,2,3,2,4,5,6,9,12,11,10,9,6,7]	Yes	17
[1,2,4,5,8,13,14,15,16,13,8,5,6,9,12,11,10,9,6,7]	No	-

4. Prims –


START
inMST
mst
i < V
Expression
edge_count
END
j < V
mincost
min
Expression
a
edge_count < V - 1
Expression
b
i < V
a != -1 && b != -1
Expression
Expression
Expression
j < V
adj[i][j] < min
nas=false
nas
b=j
nas=true
inMST[i] == true && inMST[j] == true
nas=false
Expression
i == j
nas=false
a=i
inMST[i] == false && inMST[j] == false



a. Edge coverage -

Test paths	**Feasibility**	**Test case number in Junit**
[0,1,4,5,7,8,11,13,14,15,16,22,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,19,17,11,13,14,15,16,21,17,20,11,13,11,13,11,8,11,8,9,5,6]	No	-
[0,1,4,5,7,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,9,5,7,8,9,5,6]	No	-
[0,1,2,3,2,1,4,5,6]	Yes	20

b. Prime Path coverage -

Test Paths	**Feasibility**	**Test case number in Junit**
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,13,14,15,19,17,20,11,13,14,15,16,21,17,20,11,13,14,15,19,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,11,13,14,15,16,21,17,11,13,14,18,15,16,22,17,20,11,13,14,15,16,21,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,11,13,14,18,15,19,17,11,13,14,18,15,16,22,17,20,11,13,14,15,16,22,17,20,11,13,14,18,15,16,21,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,11,13,14,15,16,22,17,11,13,14,18,15,16,21,17,11,13,14,15,19,17,11,8,9,5,7,8,9,10,12,5,6]	No	-
[0,1,2,1,2,1,4,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,21,17,20,11,13,14,18,15,16,22,17,20,11,8,11,8,11,8,11,8,9,5,6]	No	-
[0,1,2,3,2,3,2,3,2,1,4,5,6]	No	-
[0,1,4,5,7,8,11,13,11,13,11,13,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,11,13,14,15,19,17,11,13,14,18,15,16,21,17,11,13,14,18,15,19,17,20,11,13,14,18,15,16,22,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,11,13,14,15,16,22,17,11,13,14,15,16,22,17,11,13,14,18,15,16,21,17,20,11,13,14,15,16,22,17,11,8,9,10,12,5,7,8,9,10,12,5,7,8,9,5,6]	No	-
[0,1,2,3,2,1,4,5,6]	Yes	21
[0,1,4,5,7,8,9,5,6]	No	-

[0,1,4,5,6]	No	-
[0,1,4,5,7,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,11,13,14,15,16,21,17,11,13,14,15,19,17,20,11,13,14,18,15,16,22,17,11,13,14,18,15,16,22,17,11,13,14,15,19,17,11,8,9,10,12,5,7,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,11,13,14,15,19,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,11,13,14,15,16,21,17,20,11,13,14,15,16,21,17,11,13,14,18,15,16,21,17,20,11,13,14,18,15,16,22,17,20,11,13,14,18,15,19,17,11,8,9,10,12,5,7,8,9,10,12,5,6]	No	–
[0,1,4,5,7,8,11,13,11,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,11,8,9,10,12,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,11,13,14,15,16,22,17,11,13,14,15,16,21,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,11,13,14,15,16,22,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,20,11,8,9,5,7,8,11,13,14,18,15,19,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,11,13,14,18,15,19,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,13,14,18,15,19,17,20,11,8,9,5,6]	No	-

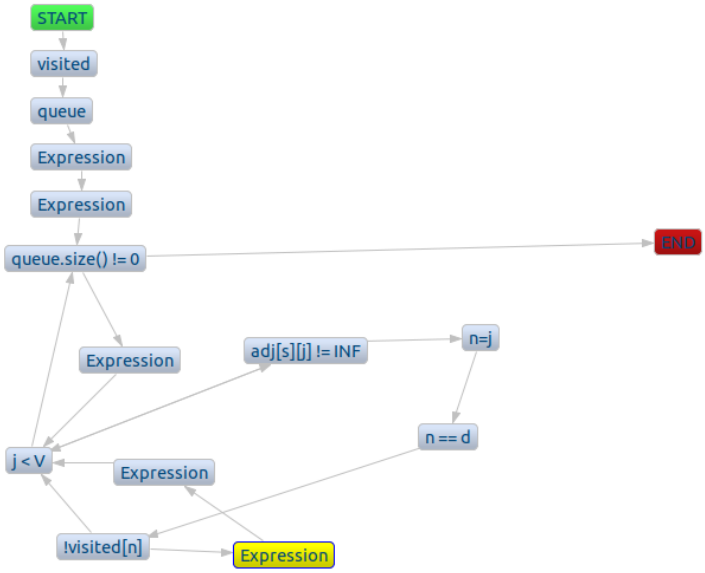
[0,1,4,5,7,8,11,13,14,15,16,22,17,20,11,13,14,15,19,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,13,14,15,16,22,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,13,14,18,15,19,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,19,17,20,11,8,9,10,12,5,7,8,11,13,14,15,16,22,17,20,11,8,9,10,12,5,6]	No	–
[0,1,4,5,7,8,11,13,14,18,15,16,22,17,20,11,8,9,10,12,5,7,8,11,13,14,18,15,16,22,17,20,11,8,9,5,6]	No	-
[0,1,2,3,2,1,4,5,7,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,13,14,18,15,16,21,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,13,14,18,15,16,21,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,20,11,13,14,18,15,16,21,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,20,11,13,14,18,15,16,22,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,20,11,13,14,18,15,19,17,11,8,9,5,6]	No	-
[0,1,2,3,2,1,4,5,7,8,11,13,14,15,19,17,20,11,8,9,10,12,5,6]	No	-

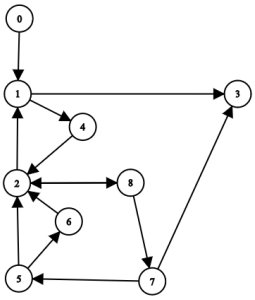
[0,1,2,3,2,1,4,5,7,8,11,13,14,15,16,22,17,20,11,8,9,10,12,5,7,8,11,13,14,15,19,17,20,11,8,9,10,12,5,7,8,9,10,12,5,6]	No	-
[0,1,2,3,2,1,4,5,7,8,11,13,14,18,15,16,21,17,20,11,8,9,10,12,5,7,8,11,13,14,18,15,16,21,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,21,17,20,11,8,9,10,12,5,6]	No	-
[0,1,2,3,2,1,4,5,7,8,11,13,14,15,16,21,17,20,11,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,9,5,7,8,11,13,14,15,19,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,19,17,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,19,17,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,21,17,20,11,8,9,5,7,8,11,13,14,18,15,16,21,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,22,17,20,11,8,9,5,7,8,11,13,14,18,15,16,22,17,20,11,8,9,5,6]	No	-
[0,1,2,3,2,1,4,5,7,8,11,13,14,18,15,16,22,17,20,11,8,9,10,12,5,6]	No	-

[0,1,2,3,2,3,2,1,4,5,7,8,11,13,14,18,15,19,17,20,11,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,9,5,7,8,11,13,14,15,16,22,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,9,5,7,8,11,13,14,15,16,21,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,20,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,21,17,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,19,17,11,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,22,17,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,19,17,20,11,8,9,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,19,17,11,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,22,17,11,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,11,8,9,10,12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,11,8,9,10,12,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,9,10,12,5,7,8,11,13,14,15,16,21,17,20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,22,17,11,8,9,10,12,5,7,8,9,5,6]	No	-

[0,1,4,5,7,8,11,13,14,18,15,16,22,17,11,8,9,10, 12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,21,17,11,8,9,10, 12,5,7,8,9,5,6]	No	-
[0,1,4,5,7,8,9,10,12,5,7,8,11,13,14,18,15,19,17, 20,11,8,9,5,6]	No	-
[0,1,4,5,7,8,11,13,14,18,15,16,21,17,11,8,9,10, 12,5,6]	No	-
[0,1,4,5,7,8,11,13,14,15,16,21,17,20,11,8,9,10, 12,5,7,8,9,5,6]	No	-

5. IsReachable –





a. Edge-coverage:

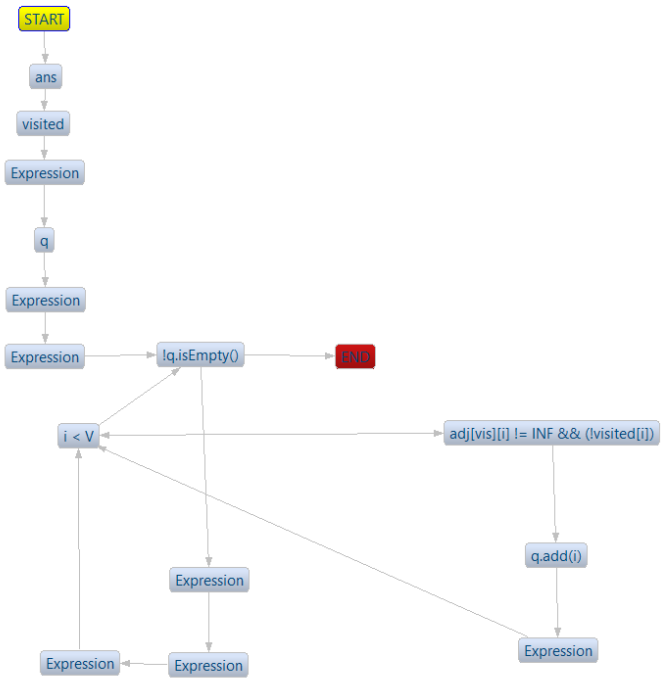
Test path	Feasibility	Test case number in Junit
[0,1,4,2,1,3]	No	-
[0,1,4,2,8,2,8,7,3]	Yes	10
[0,1,4,2,8,7,5,2,8,7,3]	Yes	11
[0,1,4,2,8,7,5,6,2,8,7,3]	No	-

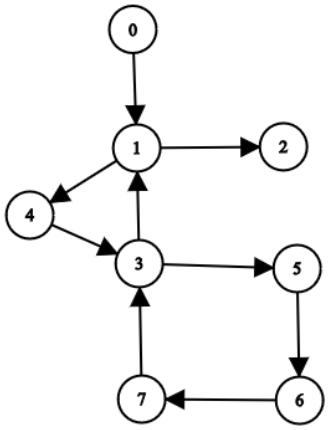
b. Prime path-coverage:

Test paths	Feasibility	Test case number in Junit
[0,1,4,2,8,7,3]	Yes	12
[0,1,4,2,8,7,5,6,2,1,4,2,8,7,3]	No	-
[0,1,4,2,8,7,5,6,2,1,3]	No	-
[0,1,4,2,8,7,5,6,2,8,7,5,6,2,8,7,3]	No	-
[0,1,4,2,8,7,5,6,2,8,7,5,2,8,7,3]	Yes	15
[0,1,4,2,8,7,5,2,1,4,2,8,7,3]	No	-
[0,1,4,2,8,7,5,2,1,3]	Yes	14
[0,1,4,2,8,7,5,2,8,7,5,2,8,7,3]	No	-
[0,1,4,2,1,4,2,8,7,3]	No	-
[0,1,4,2,8,2,1,3]	Yes	13
[0,1,4,2,8,2,1,4,2,8,7,3]	No	-
[0,1,4,2,1,3]	No	-

Test paths	Feasibility	Test case number in Junit
[0,1,4,2,8,7,3]	Yes	12
[0,1,4,2,8,7,5,6,2,1,4,2,8,7,3]	No	-
[0,1,4,2,8,2,8,7,3]	Yes	10
[0,1,3]	No	-

6. BFS –





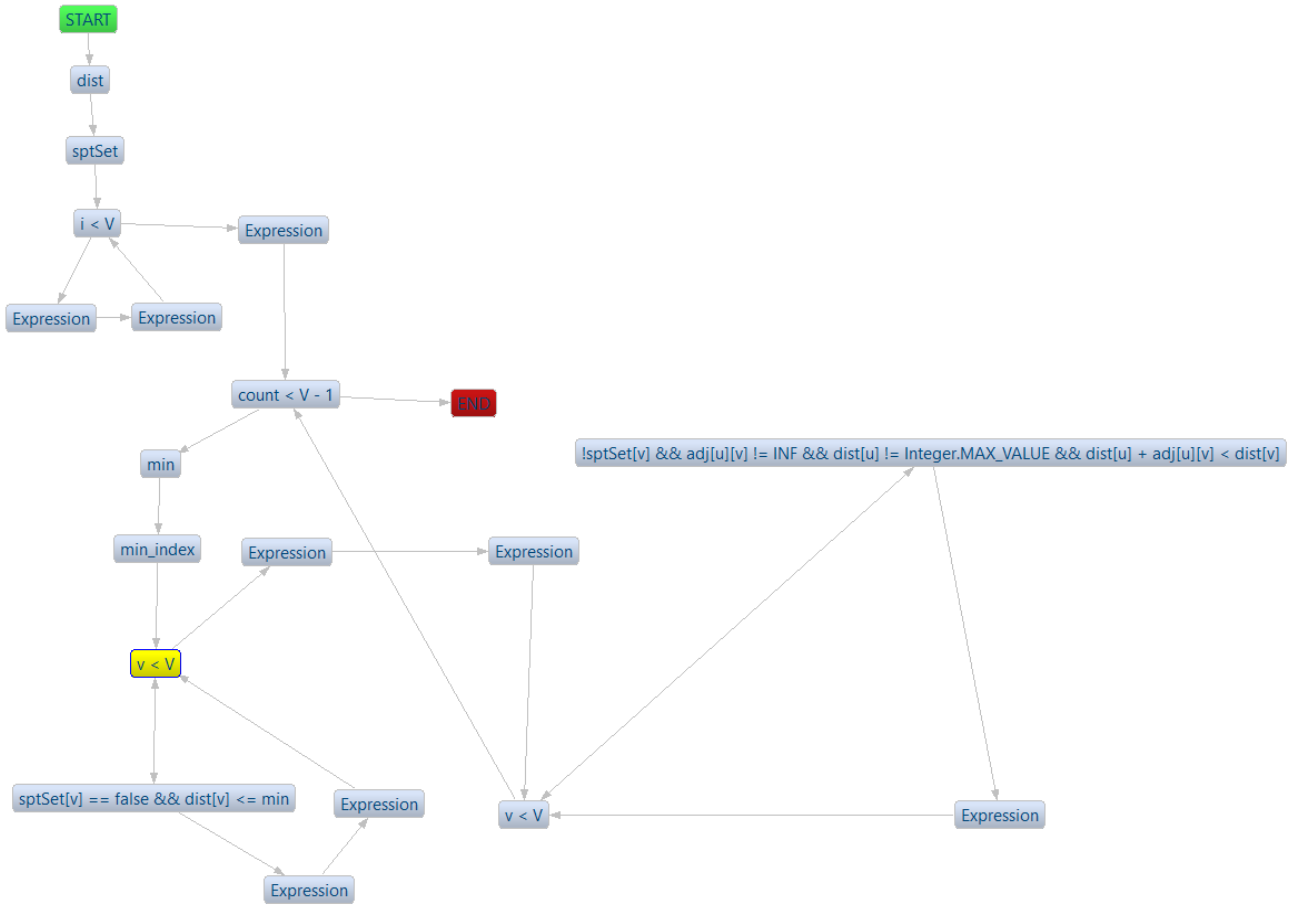
a. Edge-coverage:

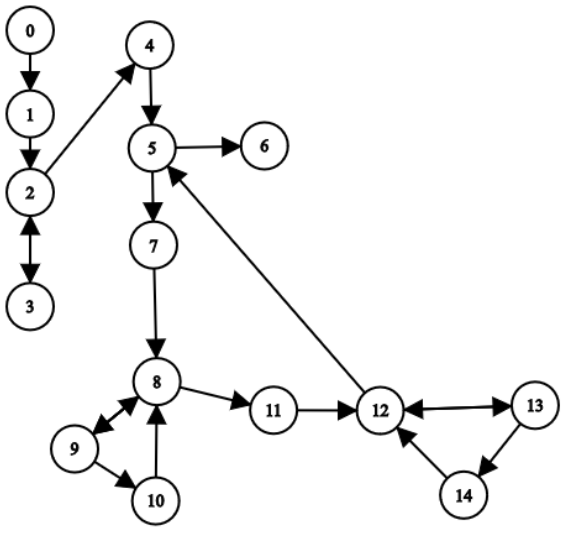
Test path	Feasibility	Test case number in Junit
[0,1,4,3,5,3,1,2]	Yes	16
[0,1,4,3,5,6,7,3,1,2]	Yes	6

b. Prime path-coverage:

Test path	Feasibility	Test case number in Junit
[0,1,4,3,5,6,7,3,1,4,3,1,2]	No	-
[0,1,4,3,5,6,7,3,5,3,1,2]	No	-
[0,1,4,3,5,6,7,3,5,6,7,3,1,2]	No	-
[0,1,4,3,1,4,3,1,2]	No	-
[0,1,4,3,5,3,1,4,3,1,2]	No	-
[0,1,2]	No	-
[0,1,4,3,5,3,5,3,1,2]	Yes	7

7. Dijkstra –


START
dist
sptSet
i < V
Expression
Expression
Expression
count < V - 1
END
min
!sptSet[v] && adj[u][v] != INF && dist[u] != Integer.MAX_VALUE && dist[u] + adj[u][v] < dist[v]
min_index
Expression
Expression
v < V
sptSet[v] == false && dist[v] <= min
Expression
v < V
Expression
Expression


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14

a. Edge coverage:

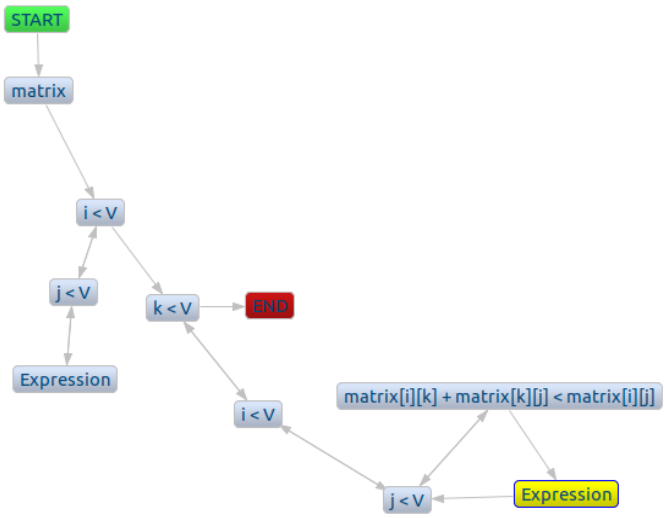
Test paths	**Feasibility**	**Test case number in Junit**
[0,1,2,4,5,7,8,11,12,13,14,12,5,6]	No	-
[0,1,2,4,5,7,8,11,12,13,14,12,5,6]	No	-
[0,1,2,4,5,7,8,9,10,8,9,8,11,12,5,6]	No	-
[0,1,2,4,5,7,8,11,12,5,7,8,11,12,13,12,5,6]	No	-
[0,1,2,3,2,4,5,6]	Yes	18

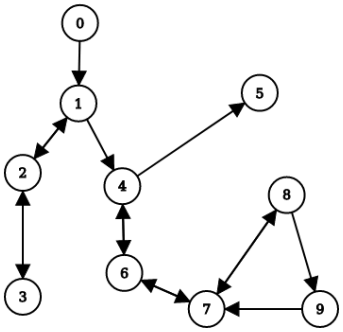
b. Prime path coverage:

Test Paths	**Feabile/ Not Feasible**	**Test Case No.**
[0,1,2,4,5,7,8,11,12,13,12,5,7,8,11,12,5,7,8,11,12,13,14,12,5,6]	No	-
[0,1,2,3,2,3,2,4,5,6]	No	-
[0,1,2,4,5,7,8,11,12,13,12,13,12,5,6]	No	-
[0,1,2,4,5,7,8,9,8,9,8,9,8,11,12,5,6]	No	-
[0,1,2,3,2,4,5,7,8,9,10,8,11,12,5,7,8,9,10,8,9,10,8,9,8,11,12,13,14,12,13,14,12,13,12,5,7,8,9,10,8,11,12,5,6]	No	-
[0,1,2,3,2,4,5,6]	Yes	19
[0,1,2,4,5,7,8,11,12,13,14,12,5,6]	No	-
[0,1,2,4,5,7,8,11,12,5,6]	No	-
[0,1,2,4,5,6]	No	-
[0,1,2,4,5,7,8,9,8,11,12,5,6]	No	-

[0,1,2,4,5,7,8,9,8,11,12,5,7,8,11,12,5,6]	No	-
[0,1,2,4,5,7,8,9,10,8,11,12,13,14,12,5,6]	No	-
[0,1,2,4,5,7,8,9,10,8,11,12,5,6]	No	-
[0,1,2,4,5,7,8,11,12,13,14,12,5,7,8,11,12,5,6]	No	-
[0,1,2,4,5,7,8,11,12,13,14,12,5,7,8,9,10,8,11,12,5,6]	No	-
[0,1,2,3,2,4,5,7,8,11,12,13,14,12,5,6]	No	-

8. Floyd Warshall –





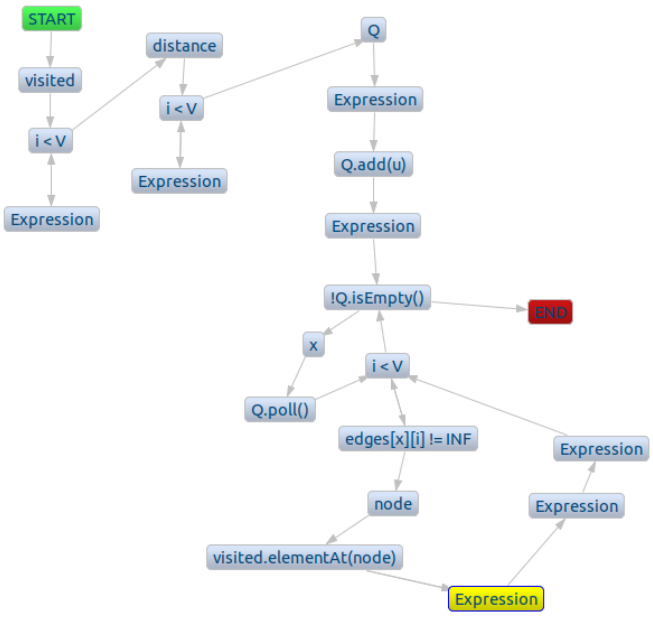
a. Edge coverage

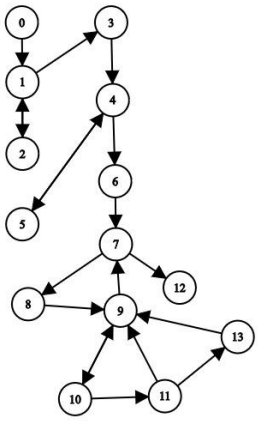
Test path	Feasibility	Test case number in Junit
[0,1,2,3,2,1,4,5]	No	-
[0,1,4,6,7,8,7,6,4,5]	No	-
[0,1,4,6,7,8,9,7,6,4,5]	No	-
[0,1,2,3,2,1,4,6,7,8,9,7,6,4,5]	Yes	5

b. Prime path coverage

Test path	Feasibility	Test case number in Junit
[0,1,2,3,2,1,4,6,7,8,9,7,6,4,5]	Yes	5
[0,1,4,5]	Yes	4
[0,1,4,6,7,8,9,7,8,7,6,4,5]	No	-
[0,1,4,6,7,8,9,7,8,9,7,6,4,5]	No	-
[0,1,2,3,2,3,2,1,4,5]	No	-
[0,1,2,1,2,1,4,5]	No	-
[0,1,4,6,7,8,7,8,7,6,4,5]	No	-
[0,1,4,6,4,6,4,5]	No	-
[0,1,4,6,7,6,7,6,4,5]	No	-

9. minEdgesBFS –





a. Edge coverage:

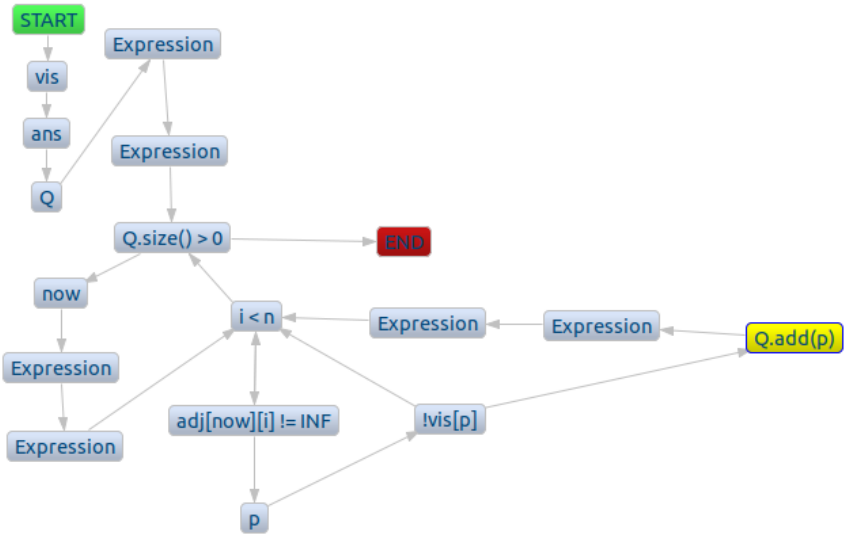
Test paths	Feasibility	Test case number in Junit
[0,1,3,4,6,7,8,9,10,11,13,9,10,9,10,9,7,12]	No	-
[0,1,2,1,3,4,5,4,6,7,8,9,10,9,7,12]	Yes	22
[0,1,3,4,6,7,8,9,10,11,9,7,12]	No	-

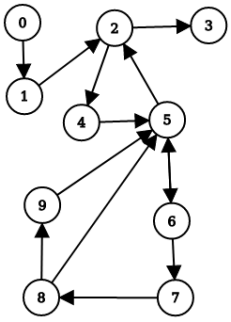
b. Prime path coverage:

Test Paths	Feasibility	Test case number in Junit
[0,1,3,4,6,7,8,9,10,11,9,10,11,9,7,12]	No	-
[0,1,3,4,5,4,5,4,6,7,12]	No	-
[0,1,3,4,6,7,8,9,10,9,10,9,7,8,9,7,12]	No	-

[0,1,2,1,2,1,3,4,5,4,6,7,12]	No	-
[0,1,2,1,3,4,6,7,12]	No	-
[0,1,3,4,5,4,6,7,8,9,10,11,13,9,10,11,13,9,10,11,9,7,12]	No	-
[0,1,3,4,6,7,8,9,7,8,9,7,8,9,7,8,9,7,12]	No	-
[0,1,3,4,6,7,8,9,7,12]	No	-
[0,1,3,4,6,7,8,9,10,9,7,12]	No	-
[0,1,3,4,6,7,8,9,10,11,9,7,8,9,7,12]	No	-
[0,1,3,4,6,7,8,9,10,11,13,9,7,8,9,7,12]	No	-
[0,1,3,4,6,7,8,9,10,11,13,9,7,12]	No	-
[0,1,3,4,6,7,12]	No	-
[0,1,2,1,3,4,5,4,6,7,8,9,10,11,9,7,12]	Yes	23

10. printLexoSmall –





a. Edge Coverage -

Test Paths	**Feasibility**	**Test case number in Junit**
[0,1,2,4,5,6,7,8,9,5,6,7,8,5,2,4,5,6,5,6,5,2,3]	Yes	25
[0,1,2,3]	No	-

b. Prime Path Coverage -

Test Paths	**Feasibility**	**Test case number in Junit**
[0,1,2,4,5,6,7,8,5,6,7,8,9,5,6,7,8,5,2,3]	No	-
[0,1,2,4,5,6,5,6,5,2,3]	Yes	26
[0,1,2,4,5,6,7,8,5,6,7,8,5,6,7,8,9,5,6,7,8,5,2,4,5,2,4,5,2,3]	No	-
[0,1,2,4,5,6,5,2,4,5,2,3]	No	-
[0,1,2,4,5,2,3]	No	-
[0,1,2,3]	No	-
[0,1,2,4,5,6,7,8,9,5,6,7,8,9,5,2,3]	No	-
[0,1,2,4,5,6,7,8,9,5,2,4,5,2,3]	No	-
[0,1,2,4,5,6,7,8,9,5,2,3]	No	-

Junit Result -

Conclusion –

We have generated the Test Requirements and Test paths for the control flow graphs for the source code. Checked for the feasibility and have written the test cases in JUnit. We have observed so many infeasible test cases. So, we have written extra test cases to cover different requirements.

Contributions –

Kanigiri Naveen (IMT2019039)

- DFS
- Bellman-Ford's algorithm
- Transpose
- Prims
- minEdgesBFS

Srivishnu Sunku (IMT2019086)

- BFS
- FloydWarshall's Algorithm
- Dijkstra's algorithm
- IsReachable
- printLexoSmall

While verifying whether a test path is feasible, together we have verified the paths.

Directory Structure –

1. src folder contains the folder for the source code and JUnit.
2. The cfg folder contains the cfg's generated from eclipse.
3. The nodes folder contains the simplified cfg's.
4. A report named (IMT2019039-IMT2019086) for the submission.
5. A pdf named (TR's.pdf) for the Test Requirements