Testing Report



for

Maze DU~~C~~KS

**Version <1.0>**

**Prepared by**

**Group Name: Sitting Ducks**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Devdatta Mahesh Joshi** | | **170953070** | | [**devdattaj007@gmail.com**](mailto:devdattaj007@gmail.com) | |
| **Kunal Khanwalkar** | | **170953162** | | [**khanwalkar@gmail.com**](mailto:khanwalkar@gmail.com) | |
| **Swapnil Samanta** | | **170953120** | | [**swapnil319206@gmail.com**](mailto:swapnil319206@gmail.com) | |
| **Utkarsh Chapekar** | | **170953106** | | [**Utkarshchhapekar2@gmail.com**](mailto:Utkarshchhapekar2@gmail.com) | |
|  | |  | |  | |
| **Instructor:** | | **Rajesh Rao** | |
| **Course:** | | **Rational Unified Process Lab** | |
| **Lab Section:** | | ***B1*** | |
| **Date:** | | **September 16th, 2019** | |
|  | |  | |

White Box Testing

**Maze Visualization**

Located in all the maze generation files of each game mode. This file generates the UI of the maze and the different colours of the tiles.

Code snippet:

for(int y = 0; y < columns; y++){

for(int x = 0; x < rows; x++){

Tile tile = new Tile(x, y);

tile.setSize(panelSize, panelSize);

tile.setLocation((x\*panelSize)+23, (y\*panelSize)+25);

if(map[x][y] == 0)

{

tile.setBackground(Color.GRAY);

}

else if(map[x][y] == 2)

{

tile.setBackground(Color.WHITE);

tile.setWall(false);

p.setLocation((x\*panelSize)+23, (y\*panelSize)+25);

p.y = y;

}

else if(map[x][y] == 3)

{

tile.setBackground(Color.RED);

tile.setWall(false);

endLevelLocx = y;

endLevelLoc = x;

}

else

{

tile.setBackground(Color.WHITE);

tile.setWall(false);

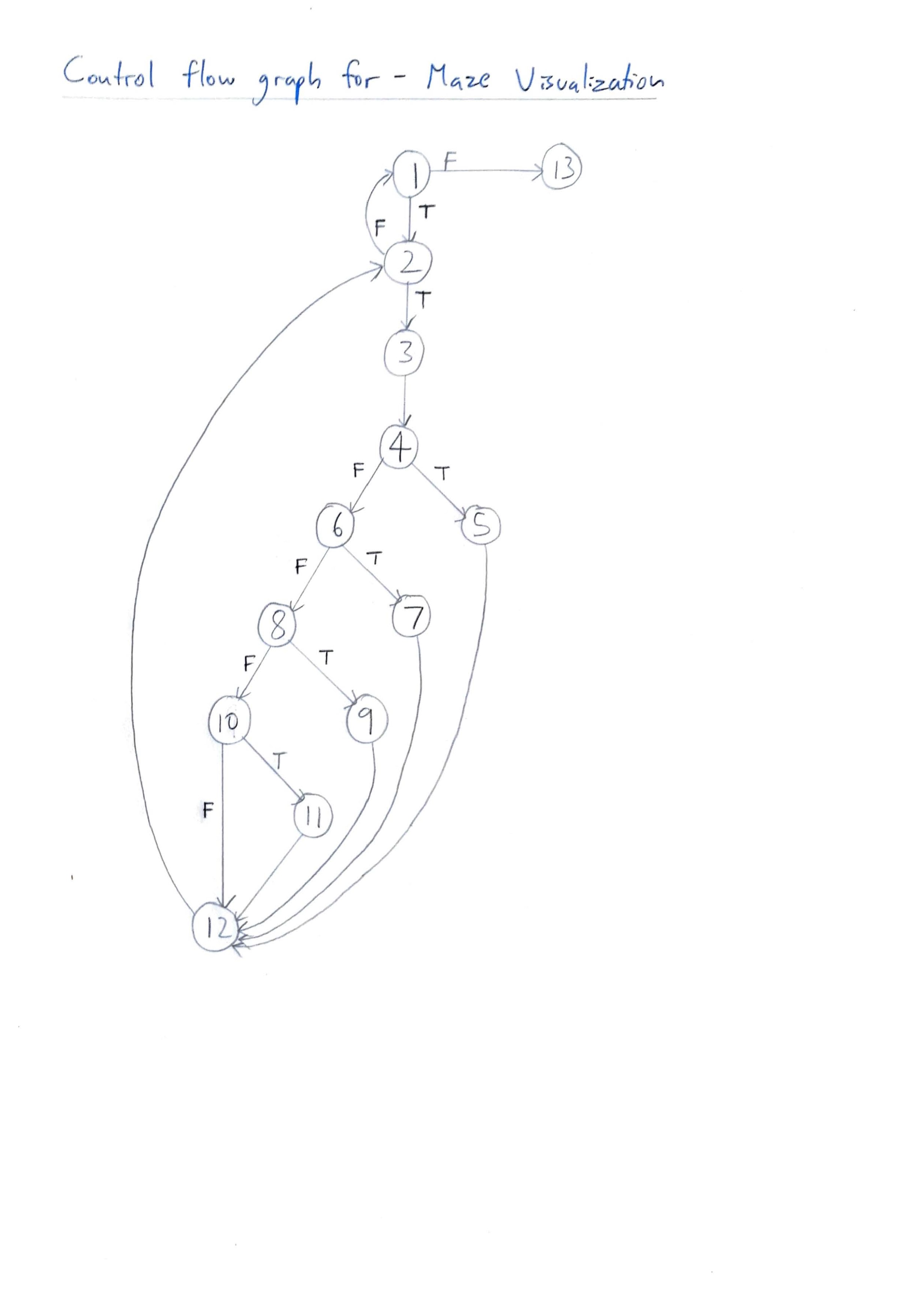
}

tile.setVisible(true);

this.add(tile);

}

}



Cyclomatic Complexity

Cyclomatic Complexity = no of predicates + 1  
 = 6 + 1  
 = 7

Independent Paths:

Path: [1 -> 2 -> 3 -> 4 -> 5 -> 12 -> 2]:

Test cases: map[x][y] = 0  
Expected Output: Tile color set to Grey

Path: [1 -> 2 -> 3 -> 4 -> 6 -> 7 -> 12 -> 2]:

Test cases: map[x][y] = 2  
Expected Output: Tile color set to green

Path: [1 -> 2 -> 3 -> 4 -> 6 -> 8 -> 9 ->12 -> 2]:

Test cases: map[x][y] = 3  
Expected Output: Tile color set to red

Path: [1 -> 2 -> 3 -> 4 -> 6 -> 8 ->10 -> 11 -> 12 -> 2]:

Test cases: map[x][y] = 1  
Expected Output: Tile color set to white

Path: [1 -> 2 -> 3 -> 4 -> 6 -> 8 ->10 -> 11 -> 12 -> 2 -> 1]:

Test cases: x >=rows  
Expected Output: No change in output

Path: [1 -> 2 -> 3 -> 4 -> 6 -> 8 ->10 -> 11 -> 12 -> 2 -> 1 -> 13]:

Test cases: x>=rows & y>=columns  
Expected Output: No change in output

**Maze timer**

Located in all the Timer files of the different game modes of the maze. This helps calculate the time elapsed/remaining for a specific level.  
  
Code snippet:

public void start() {

t.start();

}

public void stop() {

t.stop();

}

public void actionPerformed(ActionEvent e)

{

sec++;

if(sec == 60)

{

min++;

sec = 0;

if(min == 60)

{

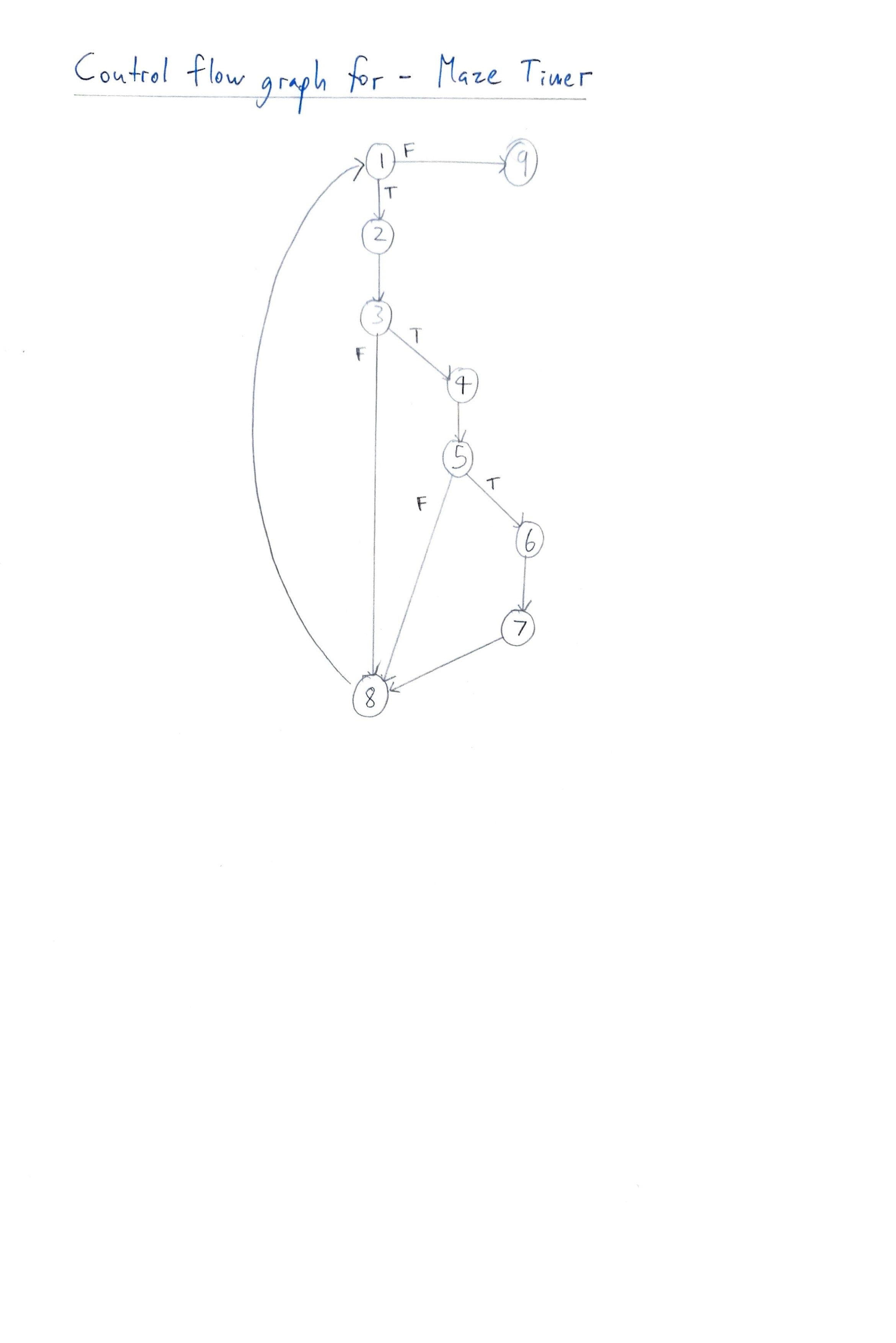
hour++;

min = 0;

}

}

}



Cyclomatic Complexity

Cyclomatic Complexity = no of predicates + 1  
 = 3 + 1  
 = 4

Independent Paths:

Path: [1 -> 9]:

Test cases: sec = any value; min = any value; hour = any value; t.stop is called  
Expected Output: Timer is stopped

Path: [1 -> 2 -> 3 -> 8]:

Test cases: sec = [0,59]; min = any value; hour = any value.  
Expected Output: sec increments by 1

Path: [1 -> 2 -> 3 -> 4 -> 5 -> 8]:

Test cases: sec = 60; min = [0,59]; hour = any value.  
Expected Output: sec becomes 0 and min increments by 1.

Path: [1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8]:

Test cases: sec = 60; min = 60; hour = any value.  
Expected Output: sec becomes 0, min becomes 0 and hour increments by 1.

**Updating Database**

Located throughout the project where the Game has to access or update the Leaderboards.

Code Snippet

try

{

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mazegame?zeroDateTimeBehavior=convertToNull", "root", "1234");

String query1 = "insert into leaderboard values ( '" + username + "' , " + dif +"," + hour + "," + min + "," + sec + ");";

System.out.print(query1);

//Statement st=con.createStatement();

//ResultSet rs=st.executeQuery(query1);

PreparedStatement pstmt = con.prepareStatement(query1);

pstmt.execute();

}

catch (Exception f)

{

System.out.print(f);

}

Cyclomatic Complexity

**Show Leaderboard**

Called when the user is accessing the Leaderboard database using in the In-game User Interface and Menus.

Code Snippet

try

{

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mazegame?zeroDateTimeBehavior=convertToNull", "root", "1234");

String query1 = "select \* from arcadeboard order by score/((hrs\*3600)+(mins\*60)+secs);";

Statement st=con.createStatement();

ResultSet rs=st.executeQuery(query1);

DefaultTableModel model=(DefaultTableModel)leaderboard\_table.getModel();

model.setRowCount(0);

Object [] row = new Object [3];

while(rs.next())

{

row[0] = rs.getString("name");

row[1] = rs.getString("Score");

row[2] = rs.getString("hrs") + ":" + rs.getString("mins") + ":" + rs.getString("secs");

model.addRow(row);

}

} catch (Exception e) {

System.out.print(e);

}

Cyclomatic Complexity

**Adaptive Difficulty**

Adaptive Difficulty uses Machine Learning to predict the appropriate size of the next maze based on the size of the current maze and time taken to solve it.

Code Snippet

Instances dataset = null;

DataSource source;

try

{

source = new DataSource(".\\src\\Maze\_Ducks\\dataset.arff");

dataset = source.getDataSet();

} catch (Exception e) {

System.out.println("Invalid File");

System.exit(0);

}

//set class index to the last attribute

dataset.setClassIndex(dataset.numAttributes()-1);

//Build model

LinearRegression model = new LinearRegression();

model.buildClassifier(dataset);

// Now Predicting the cost

Instance fsize = dataset.lastInstance();

Instance ins = new Instance(3);

ins.setValue(0, 25);

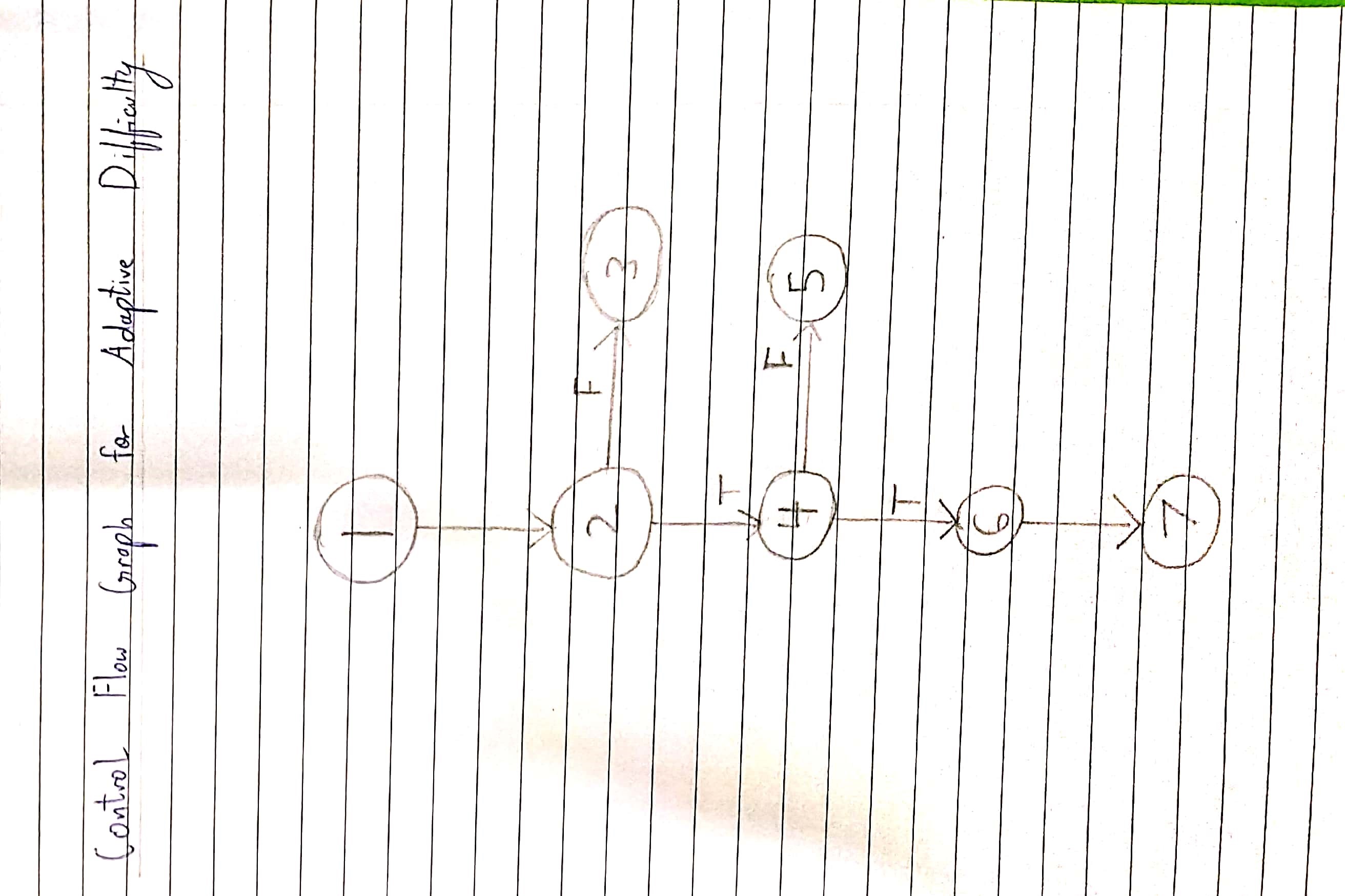
ins.setValue(1, 90);

ins.setValue(2, 28);

ins.setDataset(dataset);

double size = model.classifyInstance(ins);

System.out.println("new maze : "+size);



Cyclomatic Complexity

Cyclomatic Complexity = no of predicates + 1  
 = 2 + 1  
 = 3

Independent Paths:

Path: [1 -> 2 -> 3]:

Test case: Incorrect Dataset

Expected Output: System Error and Safe exit

Path: [1 -> 2 -> 4 -> 5]:

Test case: Regression Model not loaded

Expected Output: Import Error and Safe Exit

Path: [1 -> 2 ->4 -> 6 -> 7]:

Test case: Prediction Complete

Expected Output: Return an appropriate Maze size according to model input

System Testing

**LOGIN**

Precondition:- The user remembers his username.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Action | Expected System Response | Pass/Fail |
| 1. | Click on the username textfield | The cursor on the textfield starts blinking. | Pass |
| 2. | The user enters the username | The textbox displays the text entered by the user | Pass |
| 3. | The user clicks on the login button | The system checks if the username exists or not by checking in the database | Pass |
| 4. | On entering correct username | The user gets redirected to the homepage | Pass |
| 5. | On entering non-existent username in database | New username is inserted into the database and user is then redirected. | Pass |

**Checking Leaderboard rank**

**Precondition: No precondition required**

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Action** | **Expected System Response** | **Pass/Fail** |
| 1. | Click on the leaderboard button from the homepage | User gets redirected to the leaderboard page | Pass |
| 2. | User clicks show result button “by difficulty”. | Data from the database gets fetched to the table and user’s ranking and data is displayed along with the data of all other users on the basis of **difficulty** . | Pass |
| 2. | User clicks show result button “by time”. | Data from the database gets fetched to the table and user’s ranking and data is displayed along with the data of all other users on the basis of **the time they took to complete the maze**. | Pass |
| 4. | User clicks on exit | User is redirected to the respective game mode page he was playing. | Pass |

**Tunnel Vision Mode**

**Precondition:** User is playing in classic mode.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Action | Expected System Response | Pass/Fail |
| 1. | User clicks the enable tunnel vision button | The tunnel vision feature is enabled. | Pass |
| 2. | User clicks start | New maze is generated. | Pass |
| 3. | User moves the player with the arrow keys | The tunnel moves along the direction in which the user presses the keys | Pass |
| 4. | User presses exit | The user is redirected to the game classic game mode. | Pass |

Post Condition: System destroys the maze on exit.

**Choose Difficulty**

**Precondition:** User is playing either “Classic” mode.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Action | Expected System Response | Pass/Fail |
| 1. | User scrolls up or down the Combobox to select the difficulty. | On clicking the up button the difficulty level is increased by one unit and by pressing down, the difficulty level is lowered by one unit. | Pass |
| 2. | User chooses the difficulty from the Combobox. | The value that the user clicks on gets selected. | Pass |
| 3. | User clicks start | The generated maze’s complexity is based on the difficulty level set by the user. | Pass |

Postcondtion: On exiting the game mode, the default difficulty level is set to 10 units.

**Choose time limit**

Precondition: The user is playing in “Timed Mode”.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Action | Expected System Response | Pass/Fail |
| 1. | User scrolls up or down the Combobox to select the time limit for completing as many mazes as possible. | On clicking the up button the time limit is increased by one minute and by pressing down, the time limit is lowered by one minute. | Pass |
| 2. | User chooses the time limit from the Combobox. | The value that the user clicks on gets selected. | Pass |
| 3. | User clicks start | The first maze is generated and the timer starts counting down from the time limit. | Pass |
| 4. | User is redirected to Timed mode page. | The maze is destroyed as soon as the timer count reaches zero | Pass |

Post Condition: The default timer count value is set back to zero as soon as the user is redirected back to the Timed mode page.