IT314: Software Engineering

Lab 9

Mutation Testing

Student ID: **202201207**
Name: **Swayam Hingu**

202201207

Q-1

The code below is part of a method in the ConvexHull class in the VMAP system. The following is a small fragment of a method in the ConvexHull class. For the purposes of this exercise, you do not need to know the intended function of the method. The parameter p is a Vector of Point objects, p.size() is the size of the vector p, (p.get(i)).x is the x component of the ith point appearing in p, similarly for (p.get(i)).y. This exercise is concerned with structural testing of code, so the focus is on creating test sets that satisfy some particular coverage criteria.

```
Vector doGraham(Vector p) {
        int i,j,min,M;

        Point t;
        min = 0;

        // search for minimum:
        for(i=1; i < p.size(); ++i) {
            if( ((Point) p.get(i)).y <
                         ((Point) p.get(min)).y )
            {
                min = i;
            }
        }

        // continue along the values with same y component
        for(i=0; i < p.size(); ++i) {
            if(( ((Point) p.get(i)).y ==
                          ((Point) p.get(min)).y ) &&
                 (((Point) p.get(i)).x >
                          ((Point) p.get(min)).x ))
            {
                 min = i;
            }
        }
```

20201207

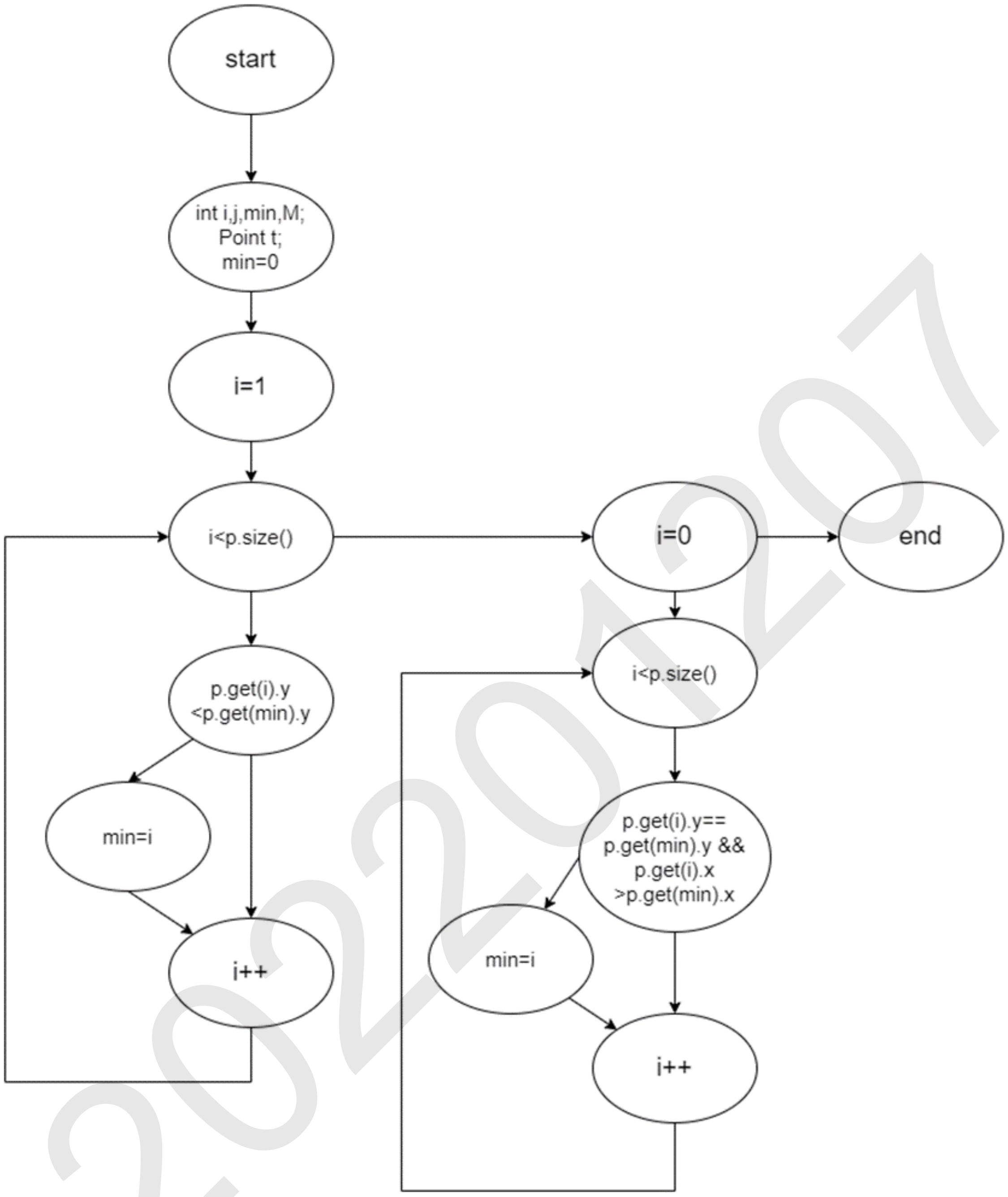
1. **Convert the code comprising the beginning of the doGraham method into a control flow graph (CFG).**

   Code:

```java
public class Point {
    double x;
    double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}

public class ConvexHull {
    public void doGraham(Vector<Point> p) {
        if (p.size() == 0) {
            return;
        }
        int minY = 0;
        for (int i = 1; i < p.size(); i++) {
            if (p.get(i).y < p.get(minY).y || (p.get(i).y == p.get(minY).y
&& p.get(i).x < p.get(minY).x)) {
                minY = i;
            }
        }
        Point temp = p.get(0);
        p.set(0, p.get(minY));
        p.set(minY, temp);
    }
}
```



20201207

Control flow graph of doGraham method

2. **Construct test sets for your flow graph that are adequate for the following criteria:**
   a. Statement Coverage.
   b. Branch Coverage.
   c. Basic Condition Coverage,

a) Statement coverage test sets:

=> Test cases

1. p is an empty vector
2. p is a vector with one point
3. p is a vector with two points having same y component
4. p is a vector with two points having different y components
5. p is a vector with three or more different points with different points with same y components
6. p is a vector with three or more different points with different points with different y components

b) Branch coverage test sets:

=> Test cases

1. p is an empty vector
2. p is a vector with one point
3. p is a vector with two points having same y component
4. p is a vector with two points having different y components
5. p is a vector with three or more different points with different points with same y components and with same x components.
6. p is a vector with three or more different points with different points with different y components and with same x components.
7. p is a vector with three or more points with the same x and y components

c) Basic condition coverage test sets:

=> Test cases

1. p is an empty vector
2. p is a vector with one point
3. p is a vector with two points having same y component

202201207

4. p is a vector with two points having different y components
5. p is a vector with three or more different points with different points with same y components and with same x components.
6. p is a vector with three or more different points with different points with different y components and with same x components.
7. p is a vector with three or more points with the same x and y components
8. p is a vector with some of them having same x component and all of them having same y component

=> Test cases examples:

1) p=[(x=2,y=3),(x=2,y=2),(x=1,y=5),(x=1,y=4)]
2) p=[(x=5,y=6),(x=3,y=2),(x=3,y=4),(x=1,y=2)]
3) p=[(x=5,y=6),(x=3,y=5),(x=1,y=5),(x=4,y=5),(x=2,y=7)]
4) p=[(x=7,y=8)]
5) p=[]

202201207

3. For the test set you have just checked can you find a mutation of the code (i.e. the deletion, change or insertion of some code) that will result in failure but is not detected by your test set. You have to use the mutation testing tool.

1. Deleting a Line of Code

```
/ Original:
Point temp = p.get(0);
p.set(0, p.get(minY));
p.set(minY, temp);

// Mutant:
// Point temp = p.get(0);
// p.set(0, p.get(minY));
// p.set(minY, temp);/
```

Reasoning: If we delete the swap operation, the method won't change the point at index `0` with the one having the minimum Y. This will lead to incorrect results, but the current test cases might not detect it if the input is such that the first point in the list is already the one with the smallest Y value.

2. Inserting a Fault (Code Insertion)

```
// Original:
Point temp = p.get(0);
p.set(0, p.get(minY));
p.set(minY, temp);

// Mutant:
p.add(new Point(0, 0));
```

Reasoning: Adding a new point to the list will disrupt the order of points and cause errors in the convex hull logic. However, if the test set doesn't cover this scenario, it might not catch this fault.

2020127

3.Modifying a Line of Code

```
// Original:
if (p.get(i).y < p.get(minY).y || (p.get(i).y == p.get(minY).y && p.get(i).x <
p.get(minY).x)) {


// Mutant:
if (p.get(i).y < p.get(minY).y || (p.get(i).y == p.get(minY).y && p.get(i).x >
p.get(minY).x)) {
```

Reasoning: By changing the comparison operator from < to >, the algorithm will mistakenly identify the wrong point as the minimum, which will lead to an incorrect convex hull. This change could pass the current test set if it doesn't thoroughly test different Y and X coordinates.

202201207

4. Create a test set that satisfies the path coverage criterion where every loop is explored at least zero, one or two times.

1. **Test Case 1: Empty List**
   Input: []
   Path: The if (p.size() == 0) condition will be true, and the method will return early without any further processing.
   Expected Result: No errors; method exits gracefully.

2. **Test Case 2: Single Point List**
   Input: [new Point(1, 2)]
   Path: The loop for (int i = 1; i < p.size(); i++) will not execute because the list has only one element. The point remains at the 0th position.
   Expected Result: No errors; the method exits without any changes to the list.

3. **Test Case 3: List with Two Points, One with Smaller Y**
   Input: [new Point(2, 3), new Point(1, 2)]
   Path: The loop will iterate once, and the point at index 1 will be identified as the one with the smaller Y value. The points will be swapped.
   Expected Result: The list will be reordered as [new Point(1, 2), new Point(2, 3)].

4. **Test Case 4: List with Multiple Points, Different Y and X Coordinates**
   Input: [new Point(2, 3), new Point(1, 2), new Point(4, 5)]
   Path: The loop will compare points based on Y, and in case of a tie, it will compare the X values. The point (1, 2) should be identified as the point with the smallest Y.
   Expected Result: The list will be reordered as [new Point(1, 2), new Point(2, 3), new Point(4, 5)].

5. **Test Case 5: List with Points Having Same Y but Different X**
   Input: [new Point(3, 4), new Point(2, 4), new Point(1, 4)]
   Path: The method will compare points based on Y first. Since all have the same Y, it will compare their X values, and reorder them accordingly.
   Expected Result: The list will be reordered as [new Point(1, 4), new Point(2, 4), new Point(3, 4)].

20220120