11/5/2024

Lab 9

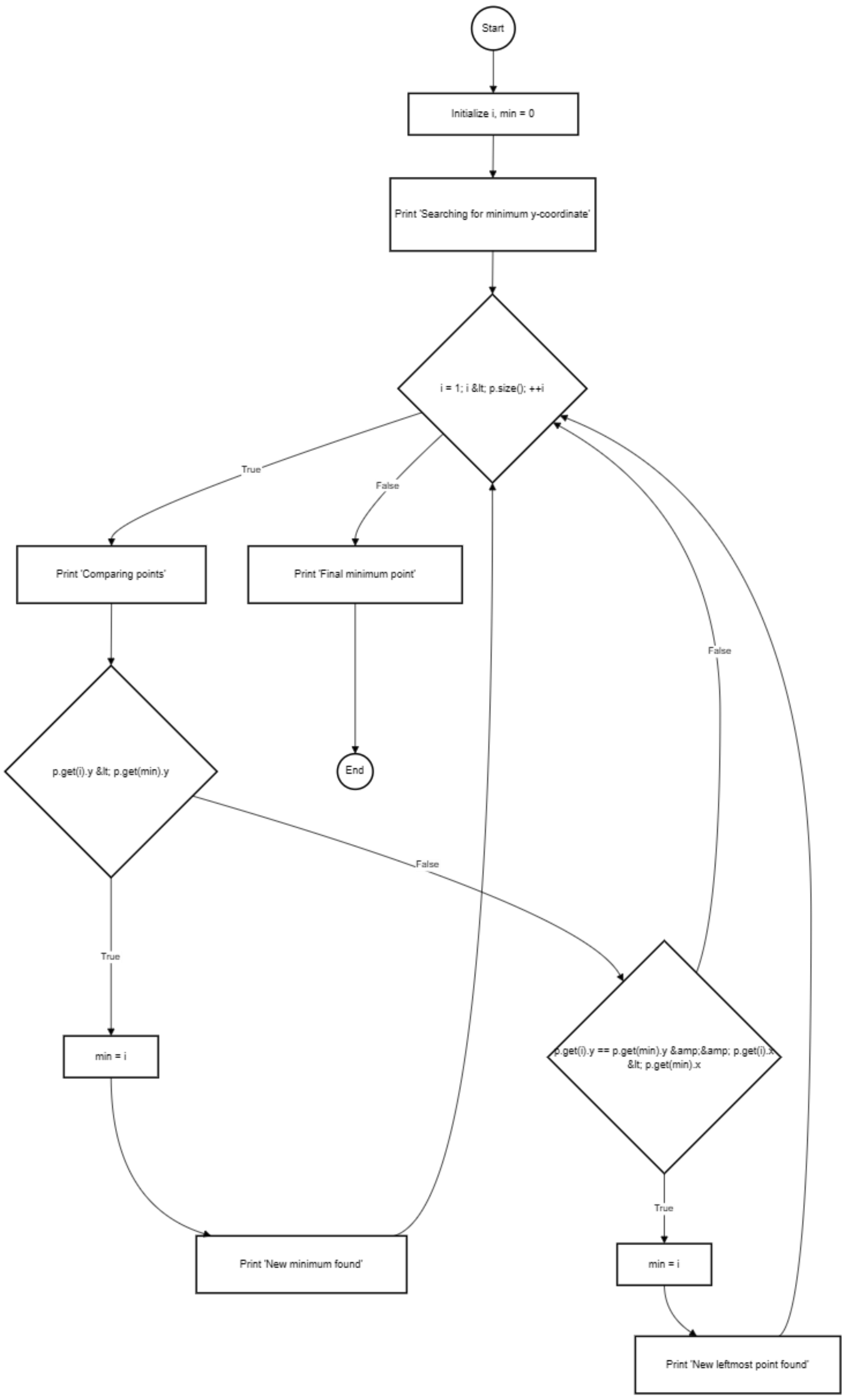
Rabadiya Utsav - 202201081

utsav rabadiya

Code Implementation

```
import java.util.Vector;

class Point {
    int x, y;

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

    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}

public class ConvexHull {
    public static void doGraham(Vector<Point> p) {
        int i, min;
        min = 0;

        System.out.println("Searching for the minimum y-coordinate...");

        for (i = 1; i < p.size(); ++i) {
            System.out.println("Comparing " + p.get(i) + " with " + p.get(min));
            if (p.get(i).y < p.get(min).y) {
                min = i;
                System.out.println("New minimum found: " + p.get(min));
            }
        }

        System.out.println("Searching for the leftmost point with the same minimum y-
coordinate...");

        for (i = 0; i < p.size(); ++i) {
            System.out.println("Checking if " + p.get(i) + " has the same y as " +
p.get(min) + " and a smaller x...");
            if (p.get(i).y == p.get(min).y && p.get(i).x < p.get(min).x) {
                min = i;
                System.out.println("New leftmost minimum point found: " + p.get(min));
            }
        }

        System.out.println("Final minimum point: " + p.get(min));
    }

    public static void main(String[] args) {
        Vector<Point> points = new Vector<>();
        points.add(new Point(1, 2));
        points.add(new Point(3, 1));
        points.add(new Point(0, 1));
        points.add(new Point(-1, 1));
        doGraham(points);
    }
}
```

Control Flow Graph


Start
Initialize i, min = 0
Print 'Searching for minimum y-coordinate'
i = 1; i &lt; p.size(); ++i
True
False
Print 'Comparing points'
Print 'Final minimum point'
End
p.get(i).y &lt; p.get(min).y
False
True
min = i
Print 'New minimum found'
p.get(i).y == p.get(min).y &amp;&amp; p.get(i).x &lt; p.get(min).x
False
True
min = i
Print 'New leftmost point found'

Test Coverage Analysis

1. Statement Coverage

Test cases needed to cover all statements:

```
// Test Case SC1
Vector<Point> points1 = new Vector<>();
points1.add(new Point(2, 2));
points1.add(new Point(1, 1));
points1.add(new Point(3, 3));
// Executes all statements by finding minimum y and checking leftmost point

// Test Case SC2
Vector<Point> points2 = new Vector<>();
points2.add(new Point(0, 1));
points2.add(new Point(1, 1));
// Tests equal y-coordinates scenario
```

2. Branch Coverage

Test cases needed to cover all branches:

```
 // Test Case BC1 - Testing y < min.y branch (True)
Vector<Point> points1 = new Vector<>();
points1.add(new Point(2, 2));
points1.add(new Point(1, 1));

// Test Case BC2 - Testing y == min.y && x < min.x branch (True)
Vector<Point> points2 = new Vector<>();
points2.add(new Point(2, 1));
points2.add(new Point(1, 1));

// Test Case BC3 - Testing both conditions (False)
Vector<Point> points3 = new Vector<>();
points3.add(new Point(1, 1));
points3.add(new Point(2, 2));
```

3. Basic Condition Coverage

Test cases needed to cover all basic conditions:

```
// Test Case BCC1 - y < min.y (True)
Vector<Point> points1 = new Vector<>();
points1.add(new Point(2, 2));
points1.add(new Point(1, 1));

// Test Case BCC2 - y == min.y (True) && x < min.x (True)
Vector<Point> points2 = new Vector<>();
points2.add(new Point(2, 1));
points2.add(new Point(1, 1));
```

```
// Test Case BCC3 - y == min.y (True) && x < min.x (False)
Vector<Point> points3 = new Vector<>();
points3.add(new Point(1, 1));
points3.add(new Point(2, 1));

// Test Case BCC4 - y == min.y (False)
Vector<Point> points4 = new Vector<>();
points4.add(new Point(1, 1));
points4.add(new Point(2, 2));
```

Mutation Testing Analysis

1. Deletion Mutation

```
// Original code

if (p.get(i).y < p.get(min).y) {

    min = i;

}

// Mutated code (deleted condition)

min = i;
```

Analysis:

- This mutation removes the y-coordinate comparison
- Impact: Will incorrectly update the minimum point without checking y-coordinates
- Not detected by test cases that only verify final point selection

2. Change Mutation

```
// Original code

if (p.get(i).y == p.get(min).y && p.get(i).x < p.get(min).x)



// Mutated code

if (p.get(i).y == p.get(min).y && p.get(i).x <= p.get(min).x)
```

Analysis:

- Changed < to <= in x-coordinate comparison
- Impact: Could select wrong point when x-coordinates are equal
- Requires specific test cases with equal x-coordinates to detect

3. Insertion Mutation

```
// Original code

min = i;


// Mutated code

min = i + 1;
```

Analysis:

- Added unnecessary increment to index
- Impact: Could cause array index out of bounds or select wrong point
- Current test cases might not detect this if they don't verify exact index values

Path Coverage Test Cases

```
// Test Case 1: Empty Vector (0 iterations)

Vector<Point> emptyPoints = new Vector<>();


// Test Case 2: Single Point (1 iteration)

Vector<Point> singlePoint = new Vector<>();

singlePoint.add(new Point(1, 1));


// Test Case 3: Two Points (2 iterations)

Vector<Point> twoPoints = new Vector<>();

twoPoints.add(new Point(2, 2));

twoPoints.add(new Point(1, 1));


// Test Case 4: Multiple Points (>2 iterations)
```

```
Vector<Point> multiPoints = new Vector<>();
multiPoints.add(new Point(3, 3));
multiPoints.add(new Point(1, 1));
multiPoints.add(new Point(2, 2));
multiPoints.add(new Point(0, 2));
```