IT314

SOFTWARE ENGINEERING

LAB-09

Lab Session- Mutation Testing

Name: SMIT FEFAR

ID: 202201253

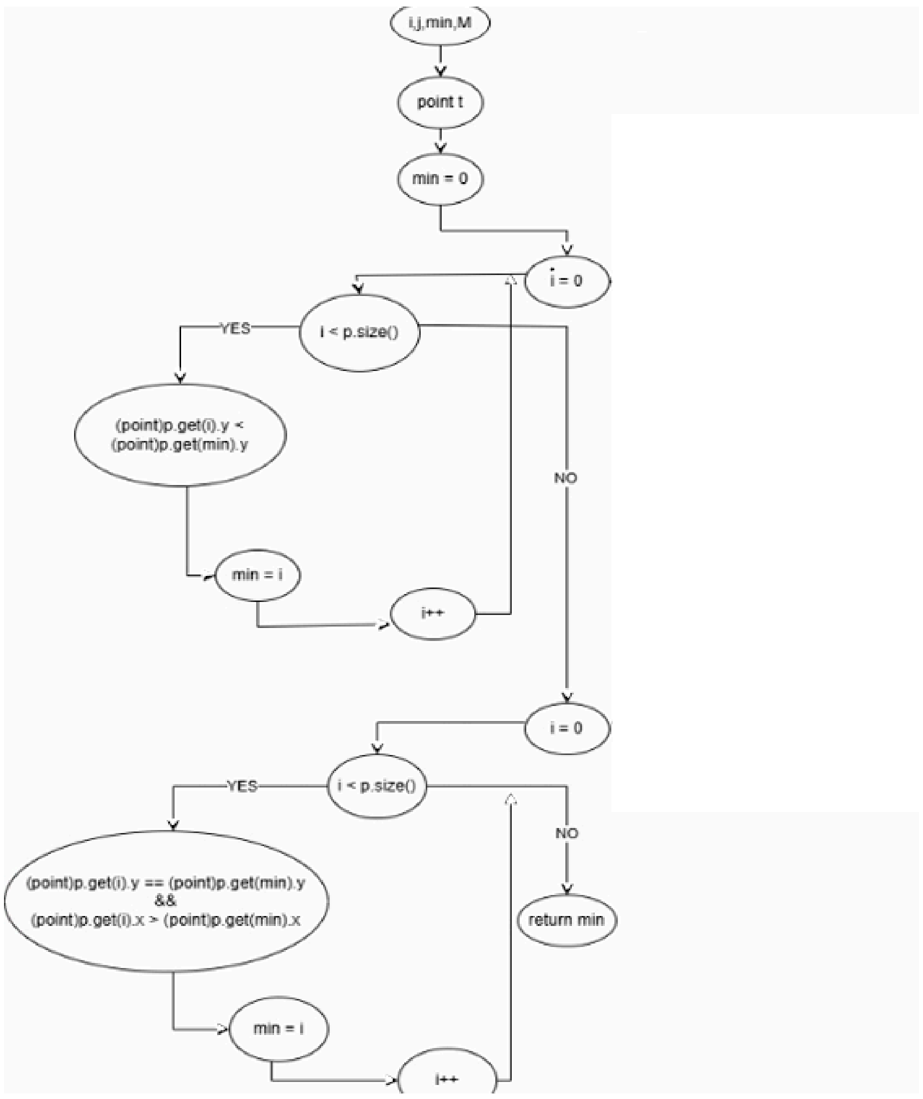
Q-1: Convert the code comprising the beginning of the doGraham method into a control flow graph (CFG). You are free to write the code in any programming language.

Ans:

Code in python Language:

```python
class Point:
 def __init__(self, x, y):
   self.x = x
   self.y = y

 @staticmethod
 def do_graham(point_list):
   lowest_index = 0

   # Search for the point with the minimum y value
   for i in range(1, len(point_list)):
     if point_list[i].y < point_list[lowest_index].y:
       lowest_index = i

   # Continue along points with the same y component to find the minimum x
   for i in range(len(point_list)):
     if point_list[i].y == point_list[lowest_index].y and point_list[i].x > point_list[lowest_index].x:
       lowest_index = i

   return point_list  # Placeholder return
```

CFG:



i,j,min,M
point t
min = 0
i = 0
i < p.size()
YES
NO
(point)p.get(i).y < (point)p.get(min).y
min = i
i++
i = 0
i < p.size()
YES
NO
(point)p.get(i).y == (point)p.get(min).y && (point)p.get(i).x > (point)p.get(min).x
min = i
i++
return min

.

Q-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.

Ans:

1. Statement Coverage:
   - Test Case 1:
     - Input: [(2, 1), (1, 2), (3, 3)]
     - Expected Output: lowest_index = (2, 1).

2. Branch Coverage:
   - Test Case 1 (Min y Updated):
     - Input: [(1, 2), (3, 3), (2, 1)]
     - Expected Output: lowest_index = (2, 1).
   - Test Case 2 (No Update to Min):
     - Input: [(1, 2), (2, 2), (3, 3)]
     - Expected Output: remains the same.
   - Test Case 3 (Tie-Breaking on x Value):
     - Input: [(1, 2), (2, 2), (3, 2)]
     - Expected Output: The lowest_index points to (3, 2).

3. Basic Condition Coverage:

- Test Case 1 (Condition 1 True, Condition 2 Not Checked):
  - Input: [(1, 2), (2, 1), (3, 3)]
- Test Case 2 (Condition 2 True, Condition 3 True):
  - Input: [(1, 2), (2, 2), (3, 2)]
- Test Case 3 (Condition 1 False):
  - Input: [(1, 2), (2, 2), (3, 3)]
- Test Case 4 (Condition 2 True, Condition 3 False):
  - Input:[(3,2),(1,2),(2,2)]

Q-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.

Ans:

Code:

```python
import unittest

class TestHullAlgorithm(unittest.TestCase):

  def test_single_coordinate(self):
    point_list = [Point(0, 0)]
    result = Point.do_graham(point_list)
    expected = [Point(0, 0)]
    self.assertEqual(result, expected)

  def test_multiple_coordinates(self):
    point_list = [Point(1, 1), Point(2, 2)]
    result = Point.do_graham(point_list)
    expected = [Point(1, 1)]
    self.assertEqual(result[0].x, expected[0].x)
    self.assertEqual(result[0].y, expected[0].y)

  def test_same_y_coordinates(self):
    point_list = [Point(2, 1), Point(3, 1)]
    result = Point.do_graham(point_list)
    expected = [Point(2, 1)]
```

```
        self.assertEqual(result[0].x, expected[0].x)
        self.assertEqual(result[0].y, expected[0].y)

if __name__ == "__main__":
    unittest.main()
```

Outcome:

Gives error for collinear points and also gives assertion error.

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

Ans: Test Cases for Path Coverage:

1. Zero Iterations for Both Loops:

   a. Description: No points are provided, leading to zero iterations in both loops.
   b. Expected Output: For loop is not executed and the function should handle this case gracefully, potentially returning an empty list or raising an appropriate error.

2. Zero Iterations for the First Loop, One Iteration for the Second Loop:

   a. Description: Only one point is provided, so the first loop does not iterate, while the second loop runs once.
   b. Expected Output: The function should return the same single point, (0, 0).

3. One Iteration for the First Loop, Two Iterations for the Second Loop:

    a. Description: Two points are provided with different y-coordinates. The first loop iterates once to identify the minimum, and the second loop runs twice.
    b. Expected Output: (1, 1).

4. Two Iterations for the 1st Loop, Three Iterations for the 2nd Loop:

    a. Description: Three points are supplied, allowing the first loop to iterate twice and the second loop to iterate three times.
    b. Expected Output: The minimum y-coordinate point should be (1, 1).

→ Mutation testing:

1. Deletion Mutation:

- Mutation: Remove the line min=0; at the beginning of the method.
- Expected Effect:
    - Skipping the initialization step could lead to min being some random value. This would throw off the min-selection logic in both loops.
- Mutation Outcome: This change might cause the method to start with an incorrect min index, leading to errors in picking the lowest point.

2. Change Mutation:

- Mutation: Change the condition in the first if statement from < to <=:
- Expected Effect:
    - Using <= instead of < means that points with the same y value could be chosen too, which might affect the accuracy in finding the lowest y point.
    - Mutation Outcome: The method might end up picking a point with a lower x value than intended if multiple points share the same y value.

3. Insertion Mutation:

- Mutation: Add an extra min=i; statement at the end of the second for loop.
- Expected Effect:
    - This addition would reset min to the last index of p, which isn't right because min should point to the index of the lowest point.
- Mutation Outcome: The method could incorrectly choose the last point as the minimum, especially if test cases don't check the final min value closely.

→ After generating the control flow graph, check whether your CFG matches with the CFG generated by Control Flow Graph Factory Tool and Eclipse flow graph generator. (In your submission document, mention only "Yes" or "No" for each tool).

Ans: Yes

→ Devise the minimum number of test cases required to cover the code using the aforementioned criteria.

Ans: 4

→ This part of the exercise is very tricky and interesting. The test cases that you have derived in Step 2 are then used to identify the fault when you make some modifications in the code. Here, you need to insert/delete/modify a piece of code that will result in failure but it is not detected by your test set – derived in Step 2. Write/identify a mutation code for each of the three operation separately, i.e., by deleting the code, by inserting the code, by modifying the code.

Ans:

Insertion : if the size of p is 1 then early return is happened.

Deletion : Delete the update of min Y-coordinate.

Modify : Changing < to <= to find min y.