University of Edinburgh, School of Informatics

Informatics 3: Software Testing: Tutorial 4

Structural Testing

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 i^{th} point appearing in `p`, similarly for (`p.get(i)).y`. This exercise is concerned with structural testing of code and so the focus is on creating test sets that satisfy some particular coverage criterion.

```
    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;
            }
        }
```

- **Prerequisites**: before the tutorial you should review the reading on structural testing. In particular, you should read Pezzè and Young's chapter 12 which will provide adequate background for this tutorial.
- **Preparation**: Review the code fragment drawn from the `doGraham` method above. If need be, check the documentation on the Vector class and any other Java documentation you might require.
- **Activities:** Having considered this code fragment you should carry out the following activities that will be facilitated by your tutor.

- **First Activity (around 10 minutes):** Individually, convert the Java code comprising the beginning of the `doGraham` method into a flow graph.
- **Second Activity (around 5 minutes):** Team up with one other member of your tutorial group, swap flow graphs and check them to see they agree, resolve any differences and decide on an agreed flowgraph for the rest of the activities.
- **Third Activity (around 15 minutes):** Split the tutorial group up into three subgroups and construct test sets for your flowgraph that are adequate for the following criteria:
    - Statement coverage.
    - Branch coverage.
    - Basic Condition Coverage.
- **Fourth Activity (around 10 minutes):** Rotate your solutions around the groups (statement coverage tests go to basic condition coverage group; basic condition coverage goes to branch coverage; branch coverage goes to statement coverage group). Check the other group's solution is correct. If you find any errors check with them and agree a resolution to the problem.
- **Fifth Activity (around 10 minutes):** 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.
- **Sixth Activity(if there is any time left):** As a group can you create a test set that satisfies the path coverage criterion where every loop is explored at least zero, one or two times.