1 Experimental requirements

1. Install Major Mutation Framework. The instruction of how to install and use Major can be seen in http://mutation-testing.org/doc/major.pdf
2. Coding a program named 'UpgradedTriangle'. Given the length value (integer) of 3 sides of a triangle. Finish 2 functions respectively, (1) classifying the triangle and (2) calculating the area of valid triangle.
    1. In function (1), given 3 length of sides(integers), output the shape of triangle made up by given sides. (Output a String, the shape could be "SCALENE","EQUILATERAL", "ISOSCELES","INVALID".)
    2. In function (2), given 3 length of sides(integers), if these 3 sides can make up a valid triangle, output the area of the triangle (double or float), otherwise, return 0. (reference : Heron's formula)
3. Write testing cases for 2 functions with Junit according to your previous study (MC/DC, boundary value, equivalence partitioning, etc.), guarantee the sufficiency and diversity of your test set. Each function should have at least 10 test cases. Then run mutants on the test sets with Major Mutation Framework.
4. Analyzing the report provided by Major. Calculate these values:
    - The number of mutants generated
    - The number of mutants covered by the test suite
    - The number of mutants killed by the test suite
    - The number of live mutants
    - The overall mutation score / adequacy of the test suite

    Discuss and explain your results: (Here are some Viewpoints you could discuss)
    - What do the results tell you about your test suite?
    - Does the test suite exhibit weaknesses? How can it be improved?
    - Does the test suite exhibit strengths? How do you recognize them?
    - Do you have any other interesting insights or opinions on the experience?
    - Among the generated mutants, If both killed and unkilled mutants were generated, what was the type of operator used? How was it applied to the code (how did the code change)?
    - According to your mutation analysis result, which part of the source code need to be strengthened in further coding? Which test case in your suite are more important compared with others.

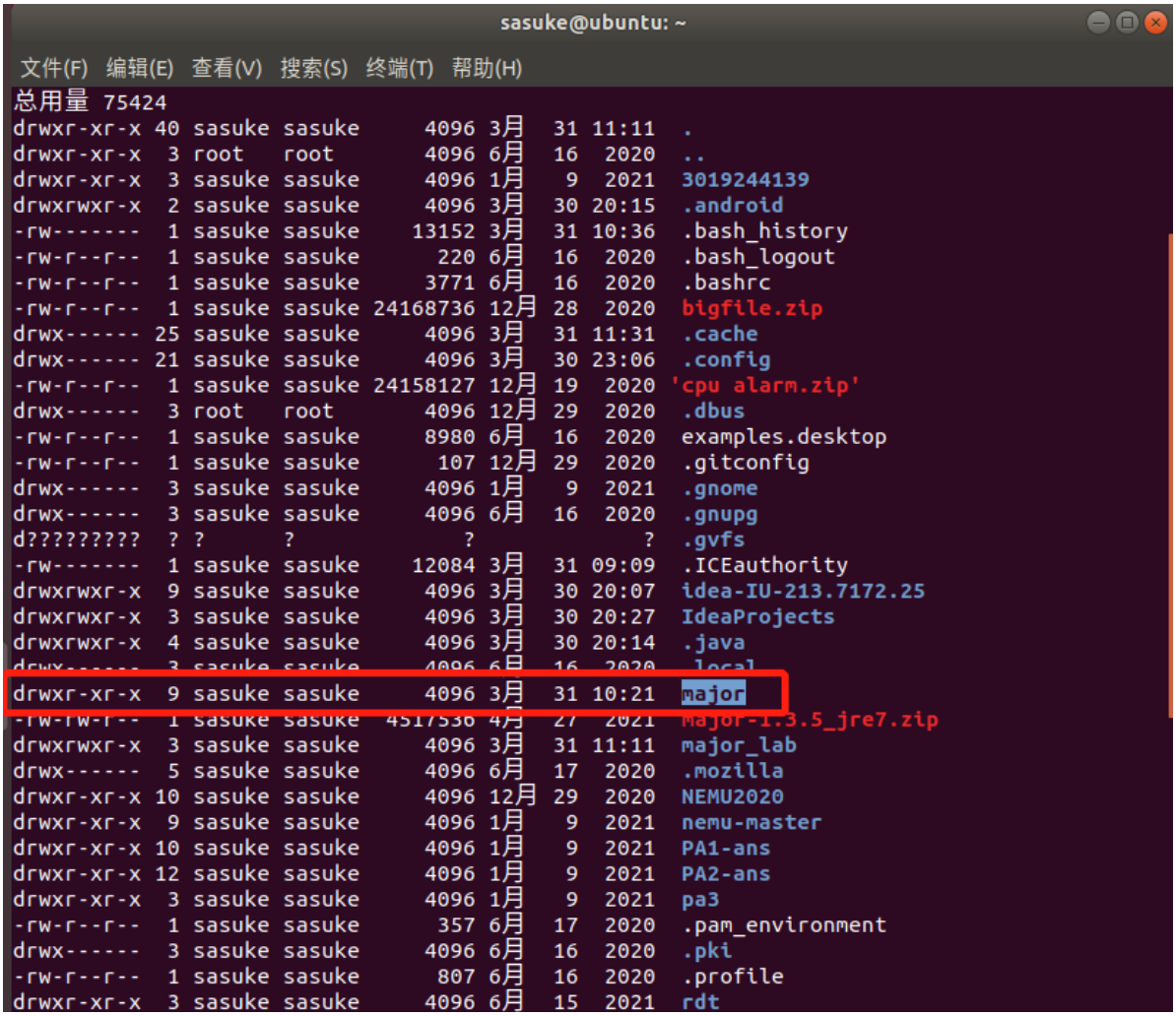
2 Configuration

2.1 配置major

1、linux下安装jdk，根据提示命令行安装openjdk-8

2、配置major

（1）在linux环境下，将major压缩包解压，放在主目录下，如图



（2）配置环境变量

命令行输入以下命令

```
sudo gedit /etc/profile
```

在文件末尾加上以下命令，注意JAVA_HOME的位置

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_321
export MAJOR_HOME=/home/sasuke/major
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$PATH:$JAVA_HOME/bin
export PATH=${MAJOR_HOME}/bin:$PATH
```

命令行输入以下命令，使环境生效

```
source /etc/profile
```

3、验证major是否配置成功

命令行输入以下命令

```
javac -version
```

出现如图所示的输出，即为配置成功

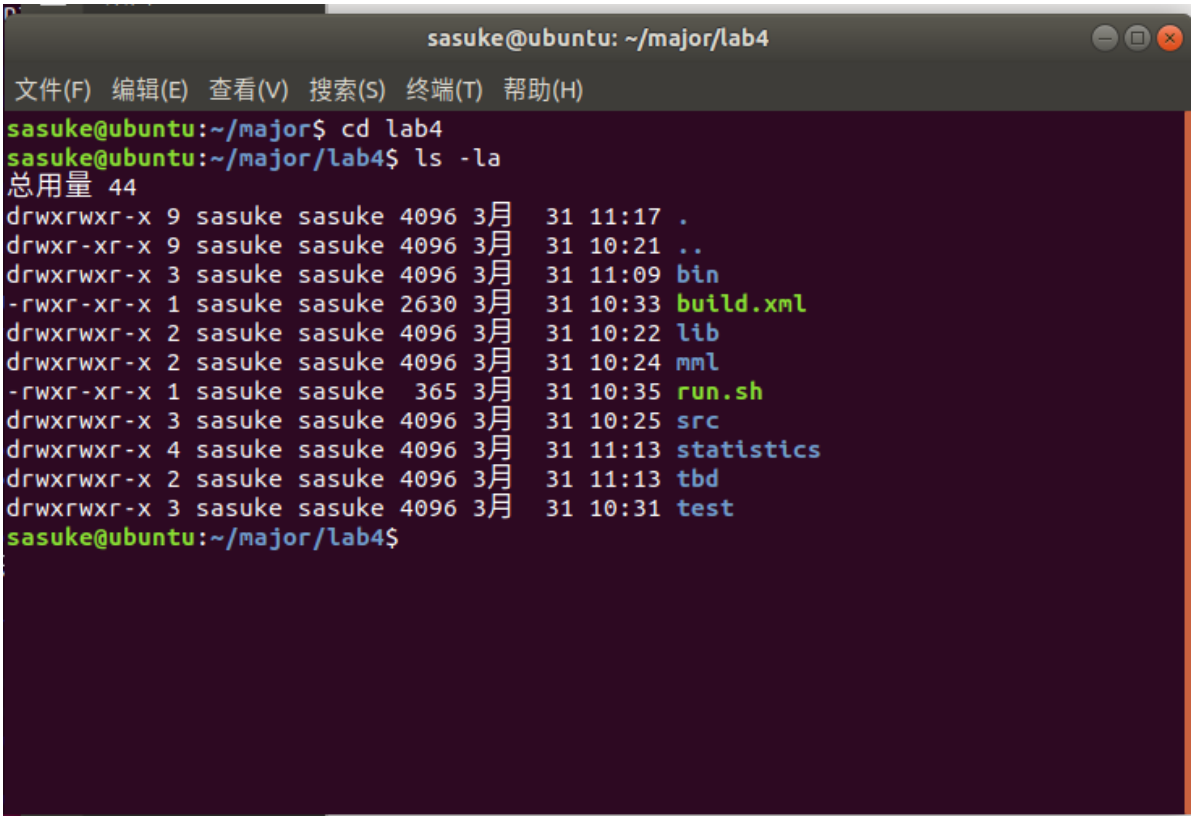
```
sasuke@ubuntu:~/major$ source /etc/profile
sasuke@ubuntu:~/major$ javac -version
javac 1.7.0-Major-v1.3.5
sasuke@ubuntu:~/major$
```

2.2 编写代码并运行major

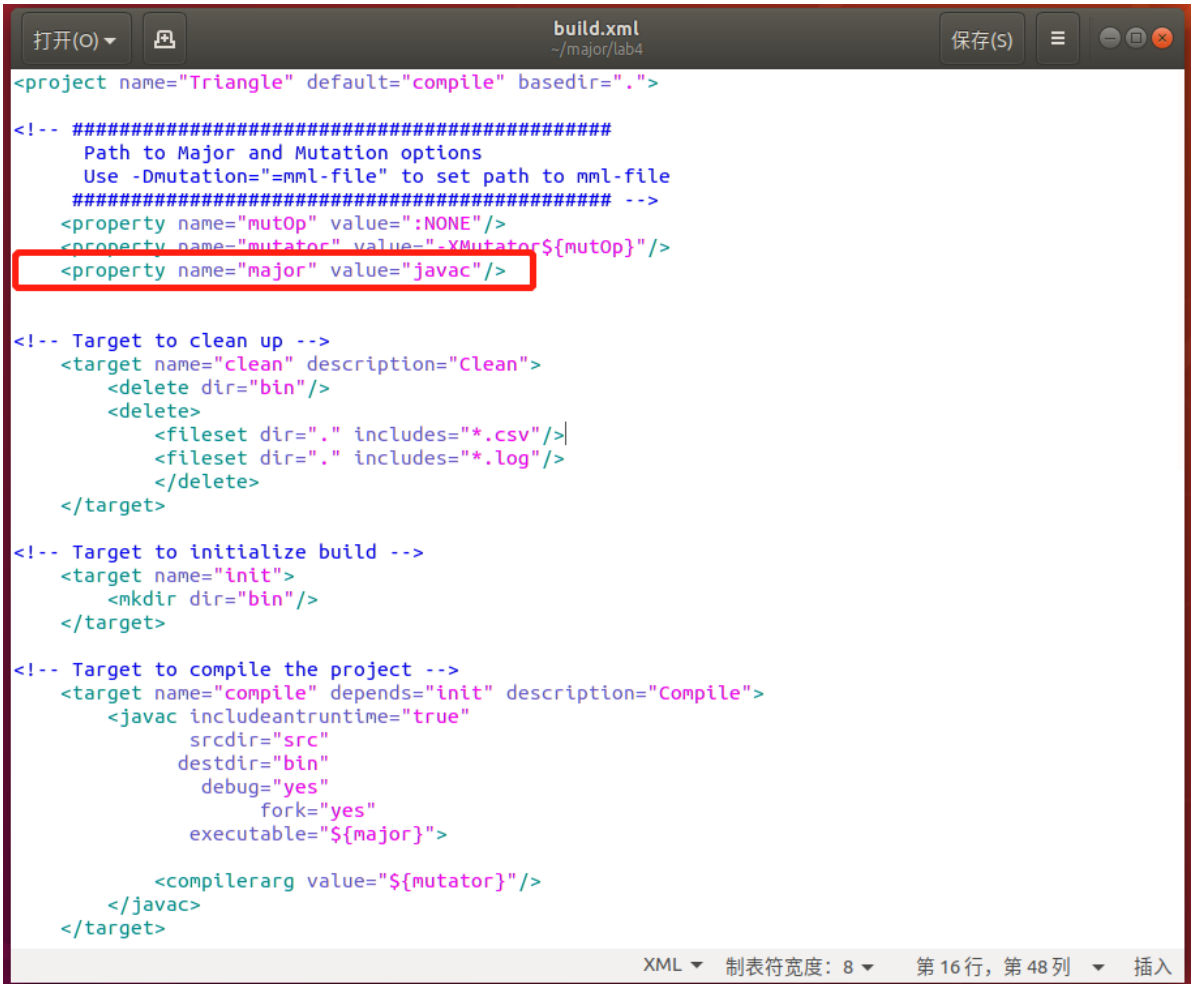
1、源码见4 source code

2、运行major

（1）在major文件夹下创建lab4文件夹，然后创建如下目录



（2）复制build.xml到新建目录，将如图位置修改为途中代码



（3）生成变异并运行

编写 mml 脚本

triangle1.mml（triangle.UpgradedTriangle::classify）

```
targetOp{
    // Define the replacements for ROR
    BIN(>)->{>=,!=,FALSE};
    BIN(<)->{<=,!=,FALSE};
    BIN(>=)->{>,==,TRUE};
    BIN(<=)->{<,==,TRUE};
    BIN(==)->{<=,>=,FALSE,LHS,RHS};
    BIN(!=)->{<,>,TRUE,LHS,RHS};
    // Define the replacements for COR
    BIN(&&)->{==,LHS,RHS,FALSE};
    BIN(||)->{!=,LHS,RHS,TRUE};
    // Define the type of statement that STD should delete
    DEL(RETURN);

    // Enable the STD, COR, and ROR mutation operators
    STD;
    COR;
    ROR;
}
// Call the defined operator group for the target method
targetOp<"triangle.UpgradedTriangle::classify(int,int,int)">;
```

triangle2.mml（triangle.UpgradedTriangle::area）

```
targetOp{

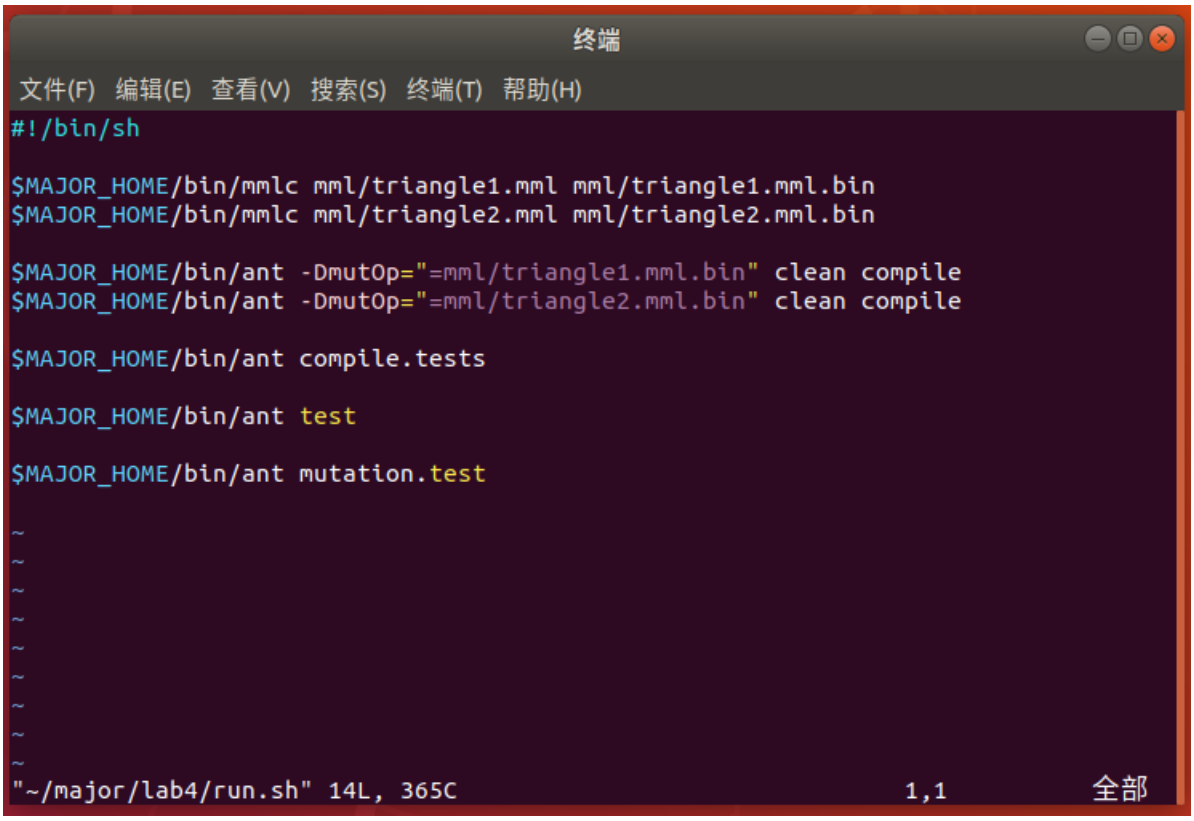
    BIN(>)->{>=,!=,FALSE};
    BIN(<)->{<=,!=,FALSE};
    BIN(>=)->{>,==,TRUE};
    BIN(<=)->{<,==,TRUE};
    BIN(==)->{<=,>=,FALSE,LHS,RHS};
    BIN(!=)->{<,>,TRUE,LHS,RHS};
    BIN(||)->{!=,LHS,RHS,TRUE};
    DEL(RETURN);
    BIN (*) -> {/ ,%};
    BIN (/) -> {* ,%};
    BIN (+) -> {-};
    BIN (-) -> {+};
    AOR;
    STD;
    COR;
    ROR;
}
targetOp<"triangle.UpgradedTriangle::area(int,int,int)">;
```

将triangle1.mml和triangle2.mml放在mml文件夹中，命令行输入

```
mmlc triangle1.mml triangle1.mml.bin
mmlc triangle2.mml triangle2.mml.bin
```

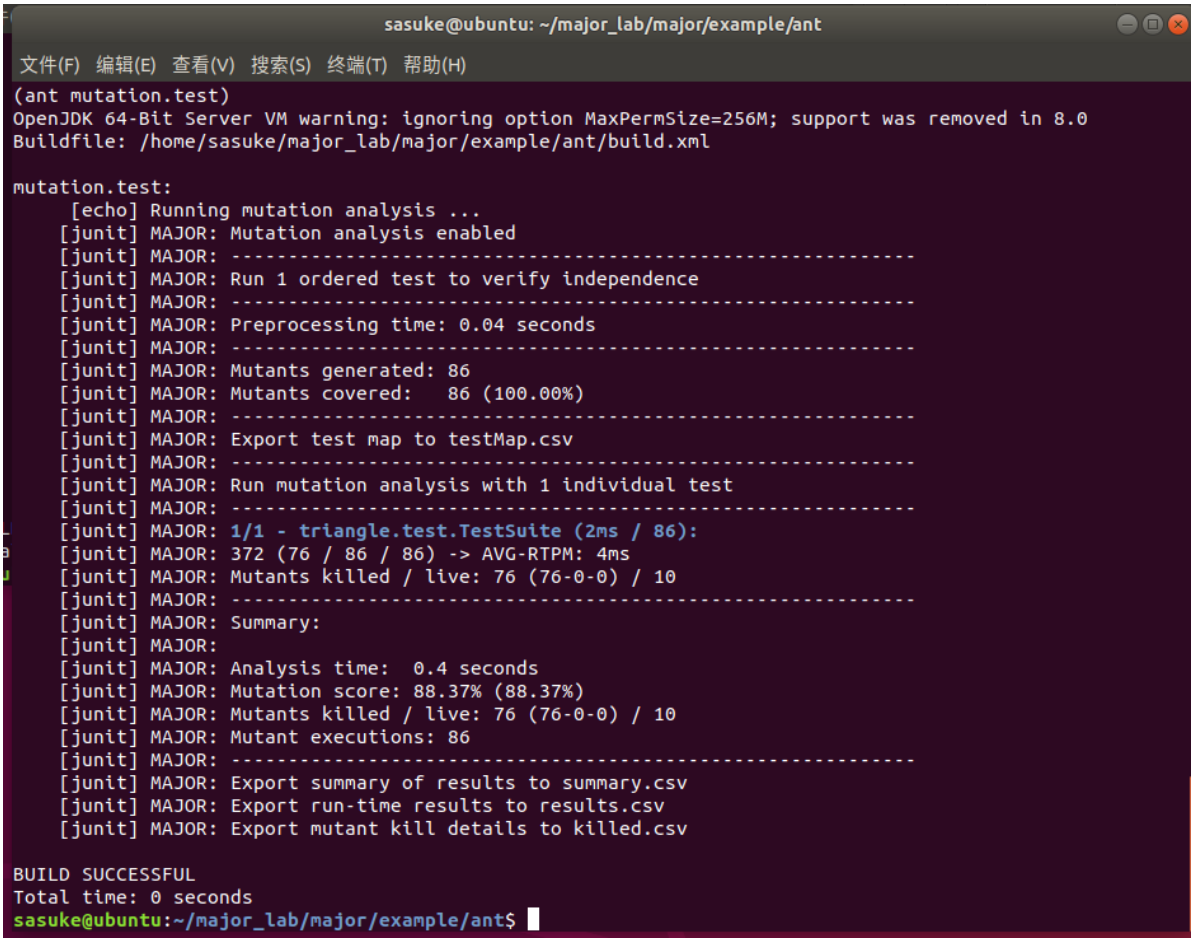
分别编译两个.mml文件，生成两个.bin文件

run.sh文件写入如下命令

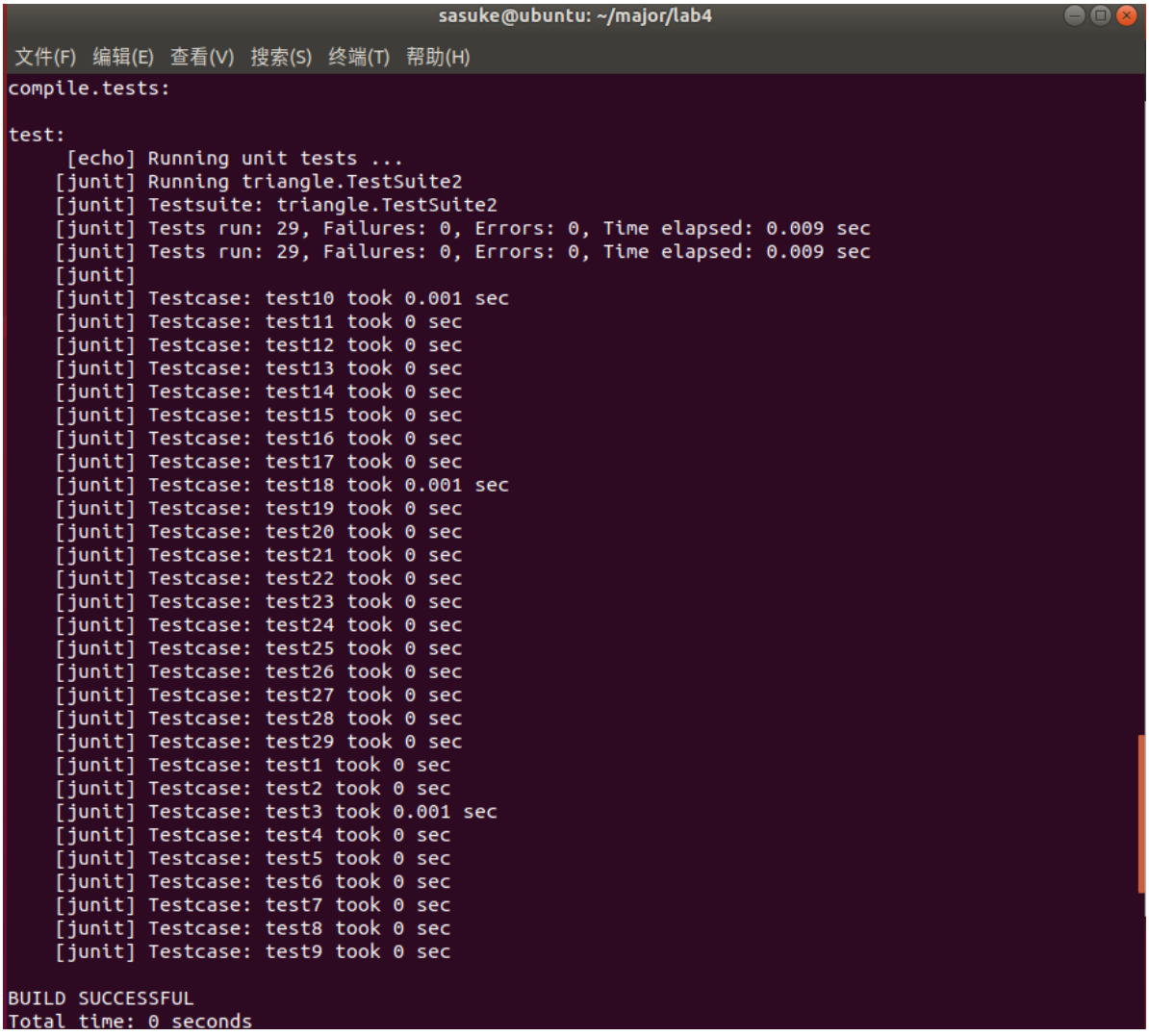


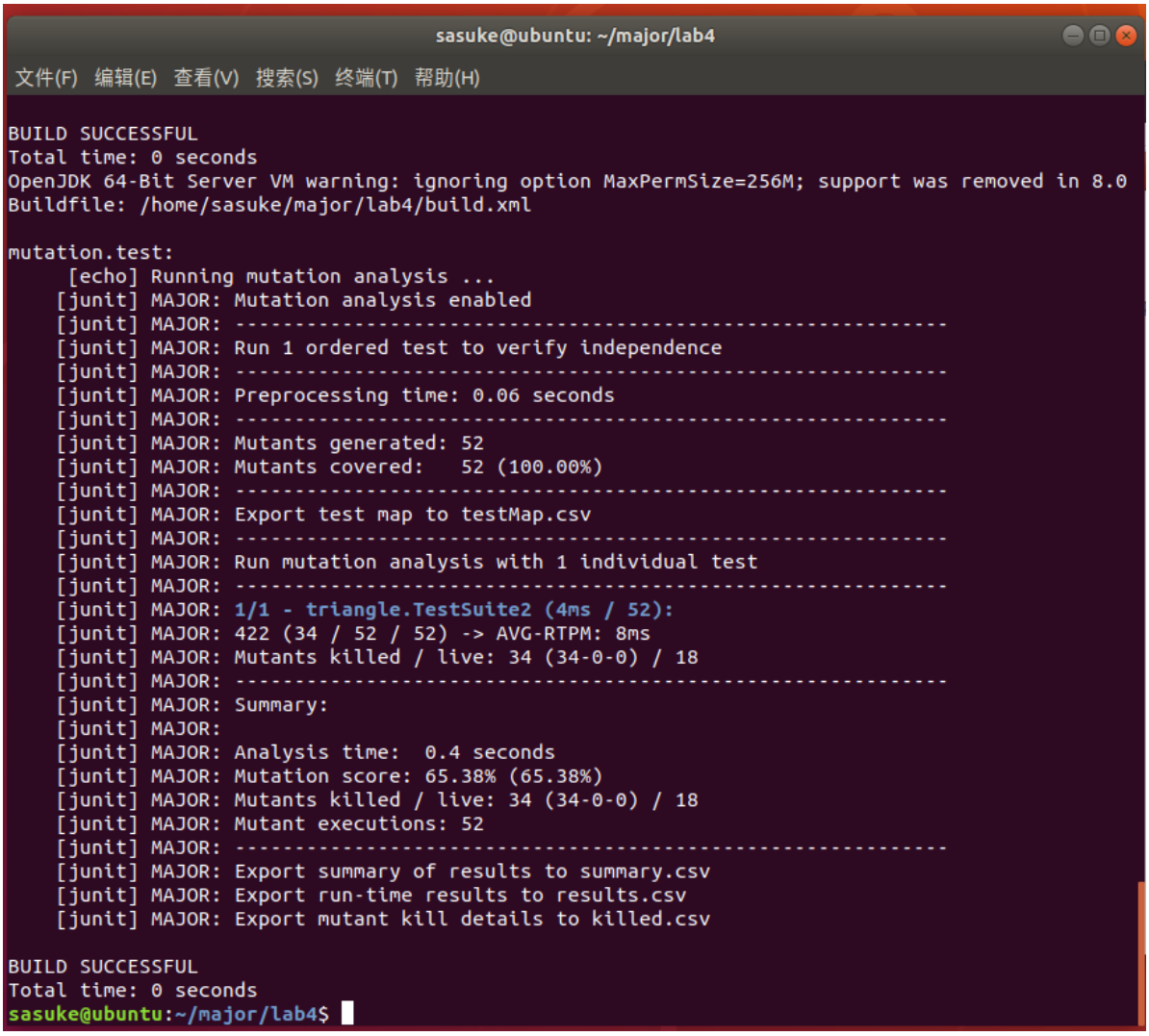
命令行执行run.sh，可得如下结果

方法一：

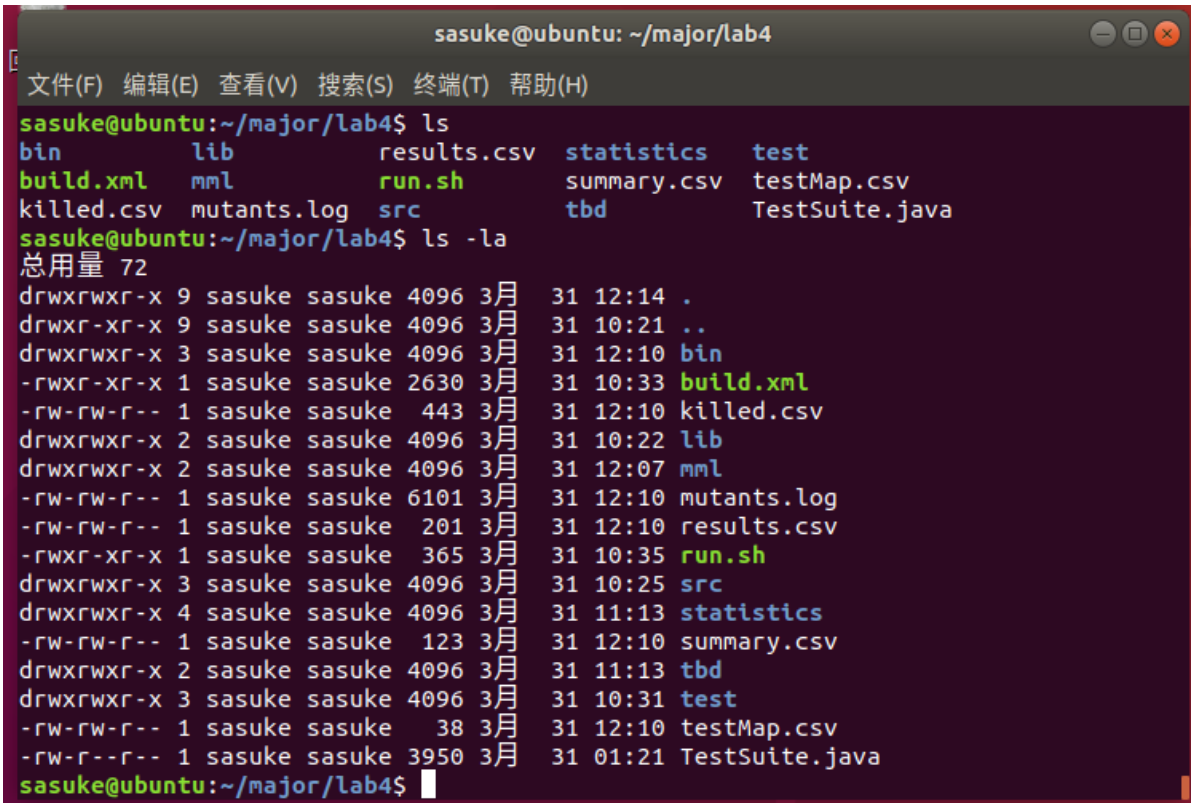

sasuke@ubuntu: ~/major_lab/major/example/ant
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
(ant mutation.test)
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=256M; support was removed in 8.0
Buildfile: /home/sasuke/major_lab/major/example/ant/build.xml
mutation.test:
[echo] Running mutation analysis ...
[junit] MAJOR: Mutation analysis enabled
[junit] MAJOR: ----------
[junit] MAJOR: Run 1 ordered test to verify independence
[junit] MAJOR: ----------
[junit] MAJOR: Preprocessing time: 0.04 seconds
[junit] MAJOR: ----------
[junit] MAJOR: Mutants generated: 86
[junit] MAJOR: Mutants covered: 86 (100.00%)
[junit] MAJOR: ----------
[junit] MAJOR: Export test map to testMap.csv
[junit] MAJOR: ----------
[junit] MAJOR: Run mutation analysis with 1 individual test
[junit] MAJOR: ----------
[junit] MAJOR: 1/1 - triangle.test.TestSuite (2ms / 86):
[junit] MAJOR: 372 (76 / 86 / 86) -> AVG-RTPM: 4ms
[junit] MAJOR: Mutants killed / live: 76 (76-0-0) / 10
[junit] MAJOR: ----------
[junit] MAJOR: Summary:
[junit] MAJOR:
[junit] MAJOR: Analysis time: 0.4 seconds
[junit] MAJOR: Mutation score: 88.37% (88.37%)
[junit] MAJOR: Mutants killed / live: 76 (76-0-0) / 10
[junit] MAJOR: Mutant executions: 86
[junit] MAJOR: ----------
[junit] MAJOR: Export summary of results to summary.csv
[junit] MAJOR: Export run-time results to results.csv
[junit] MAJOR: Export mutant kill details to killed.csv
BUILD SUCCESSFUL
Total time: 0 seconds
sasuke@ubuntu:~/major_lab/major/example/ant$

方法二：


sasuke@ubuntu: ~/major/lab4
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
compile.tests:
test:
[echo] Running unit tests ...
[junit] Running triangle.TestSuite2
[junit] Testsuite: triangle.TestSuite2
[junit] Tests run: 29, Failures: 0, Errors: 0, Time elapsed: 0.009 sec
[junit] Tests run: 29, Failures: 0, Errors: 0, Time elapsed: 0.009 sec
[junit]
[junit] Testcase: test10 took 0.001 sec
[junit] Testcase: test11 took 0 sec
[junit] Testcase: test12 took 0 sec
[junit] Testcase: test13 took 0 sec
[junit] Testcase: test14 took 0 sec
[junit] Testcase: test15 took 0 sec
[junit] Testcase: test16 took 0 sec
[junit] Testcase: test17 took 0 sec
[junit] Testcase: test18 took 0.001 sec
[junit] Testcase: test19 took 0 sec
[junit] Testcase: test20 took 0 sec
[junit] Testcase: test21 took 0 sec
[junit] Testcase: test22 took 0 sec
[junit] Testcase: test23 took 0 sec
[junit] Testcase: test24 took 0 sec
[junit] Testcase: test25 took 0 sec
[junit] Testcase: test26 took 0 sec
[junit] Testcase: test27 took 0 sec
[junit] Testcase: test28 took 0 sec
[junit] Testcase: test29 took 0 sec
[junit] Testcase: test1 took 0 sec
[junit] Testcase: test2 took 0 sec
[junit] Testcase: test3 took 0.001 sec
[junit] Testcase: test4 took 0 sec
[junit] Testcase: test5 took 0 sec
[junit] Testcase: test6 took 0 sec
[junit] Testcase: test7 took 0 sec
[junit] Testcase: test8 took 0 sec
[junit] Testcase: test9 took 0 sec
BUILD SUCCESSFUL
Total time: 0 seconds



也生成了.csv文件，如图



3 Result analysis

1、结果分析

由于方法一，在major源文件中已经给出，所以直接分析方法二

由上图可知

- 所产生的突变体的数量：52
- 测试套件所覆盖的突变体的数量：52
- 被测试套件杀死的突变体的数量：34
- 活的突变体的数量：18
- 测试套件的总体突变评分/充分性：65.38%

2、结果讨论

- What do the results tell you about your test suite?
  - 测试并不能达到100%杀死突变体
- Does the test suite exhibit weaknesses? How can it be improved?
  - test suite本身有弱点，可以通过完善测试用例来提高mutation score
  - 查看mutants.log和killed.csv可以查看存活的突变体，可以有针对性的完善测试用例
- Does the test suite exhibit strengths? How do you recognize them?
  - test suite提高了测试的自动化程度，能够自动生成突变体，并对测试用例进行测试，还能够显示出存活的突变体，便于开发人员完善测试代码，提高了开发的效率，降低了测试的成本
- Do you have any other interesting insights or opinions on the experience?
  - 对于此次的实验，我遇到的最大的困难是配置环境，关于路径和名称等的问题，以官方文档和网上博客作为参考，还是遇到了许多信息不全面的问题。
- Among the generated mutants, If both killed and unkilled mutants were generated, what was the type of operator used? How was it applied to the code (how did the code change)?
  - 根据config.java中的代码，分情况处理
  - ```
    /*
         * The mutant identifier:
         *
         * __M_NO <  0 -> Run original version
         *
         * __M_NO == 0 -> Run original version and gather coverage information
         *
         * __M_NO >  0 -> Execute mutant with the corresponding id
         */
    ```
  - 对比kill.csv和mutants.log，以前几个突变体为例
MutantNo	[FAIL | TIME | EXC | LIVE]
1	LIVE
2	LIVE
3	FAIL
4	LIVE
  -
    ```

```
1:ROR:<=(int,int):<(int,int):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 |==> a < 0
2:ROR:<=(int,int):==(int,int):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 |==> a == 0
3:ROR:<=(int,int):TRUE(int,int):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 |==> true
4:ROR:<=(int,int):<(int,int):triangle.UpgradedTriangle@area(int,int,int):41:b <= 0 |==> b < 0
```

- 可知对于第一个突变体，将代码的第41行的a<=0变成了a<0，并且存活了下来。
-
```
public static double area(int a, int b, int c) {
    if (a <= 0 || b <= 0 || c <= 0) {
        return 0;
    }
```

- According to your mutation analysis result, which part of the source code need to be strengthened in further coding? Which test case in your suite are more important compared with others.
  - 通过分析kill.csv和mutants.log

MutantNo	[FAIL | TIME | EXC | LIVE]
1	LIVE
2	LIVE
3	FAIL
4	LIVE
5	LIVE
6	FAIL
7	LIVE
8	LIVE
9	LIVE
10	FAIL
11	LIVE
12	LIVE
13	FAIL
14	LIVE
15	LIVE
16	LIVE

```
1:ROR:<=(int,int):<(int,int):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 |==> a < 0
2:ROR:<=(int,int):==(int,int):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 |==> a == 0
3:ROR:<=(int,int):TRUE(int,int):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 |==> true
4:ROR:<=(int,int):<(int,int):triangle.UpgradedTriangle@area(int,int,int):41:b <= 0 |==> b < 0
5:ROR:<=(int,int):==(int,int):triangle.UpgradedTriangle@area(int,int,int):41:b <= 0 |==> b == 0
6:ROR:<=(int,int):TRUE(int,int):triangle.UpgradedTriangle@area(int,int,int):41:b <= 0 |==> true
7:COR:||(boolean,boolean):!=(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 || b <= 0 |==> a <= 0 != b <= 0
8:COR:||(boolean,boolean):LHS(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 || b <= 0 |==> a <= 0
9:COR:||(boolean,boolean):RHS(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 || b <= 0 |==> b <= 0
10:COR:||(boolean,boolean):TRUE(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 || b <= 0 |==> true
11:ROR:<=(int,int):<(int,int):triangle.UpgradedTriangle@area(int,int,int):41:c <= 0 |==> c < 0
12:ROR:<=(int,int):==(int,int):triangle.UpgradedTriangle@area(int,int,int):41:c <= 0 |==> c == 0
13:ROR:<=(int,int):TRUE(int,int):triangle.UpgradedTriangle@area(int,int,int):41:c <= 0 |==> true
14:COR:||(boolean,boolean):!=(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 || b <= 0 || c <= 0 |==> (a <= 0 || b <= 0) !
15:COR:||(boolean,boolean):LHS(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 || b <= 0 || c <= 0 |==> a <= 0 || b <= 0
16:COR:||(boolean,boolean):RHS(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):41:a <= 0 || b <= 0 || c <= 0 |==> c <= 0
```

  - 可知，第41行，存活的突变体最多，说明此处的代码需要提高

30	FAIL
31	FAIL
32	LIVE
33	FAIL
34	FAIL
35	LIVE
36	FAIL
37	FAIL
38	FAIL
39	FAIL
40	FAIL
41	FAIL
42	FAIL
43	FAIL
44	FAIL
45	FAIL
46	FAIL
47	FAIL
48	FAIL
49	FAIL
50	FAIL
51	FAIL
52	FAIL

30:COR:||(boolean,boolean):TRUE(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):44:a + b <= c || a + c <= b |==> true
31:AOR:+(int,int):-(int,int):triangle.UpgradedTriangle@area(int,int,int):44:b + c |==> b - c
32:ROR:<=(int,int):<(int,int):triangle.UpgradedTriangle@area(int,int,int):44:b + c <= a |==> b + c < a
33:ROR:<=(int,int):==(int,int):triangle.UpgradedTriangle@area(int,int,int):44:b + c <= a |==> b + c == a
34:ROR:<=(int,int):TRUE(int,int):triangle.UpgradedTriangle@area(int,int,int):44:b + c <= a |==> true
35:COR:||(boolean,boolean):!=(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):44:a + b <= c || a + c <= b || b + c <= a |==>
36:COR:||(boolean,boolean):LHS(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):44:a + b <= c || a + c <= b || b + c <= a |==
37:COR:||(boolean,boolean):RHS(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):44:a + b <= c || a + c <= b || b + c <= a |=
38:COR:||(boolean,boolean):TRUE(boolean,boolean):triangle.UpgradedTriangle@area(int,int,int):44:a + b <= c || a + c <= b || b + c <= a |=
39:STD:<RETURN>:<NO-OP>:triangle.UpgradedTriangle@area(int,int,int):45:return 0; |==> <NO-OP>
40:AOR:+(int,int):-(int,int):triangle.UpgradedTriangle@area(int,int,int):48:a + b |==> a - b
41:AOR:+(int,int):-(int,int):triangle.UpgradedTriangle@area(int,int,int):48:a + b + c |==> a + b - c
42:AOR:/(double,double):%(double,double):triangle.UpgradedTriangle@area(int,int,int):48:(a + b + c) / 2.0 |==> (a + b + c) % 2.0
43:AOR:/(double,double):*(double,double):triangle.UpgradedTriangle@area(int,int,int):48:(a + b + c) / 2.0 |==> (a + b + c) * 2.0
44:AOR:-(double,double):+(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p - a |==> p + a
45:AOR:*(double,double):%(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p * (p - a) |==> p % (p - a)
46:AOR:*(double,double):/(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p * (p - a) |==> p / (p - a)
47:AOR:-(double,double):+(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p - b |==> p + b
48:AOR:*(double,double):%(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p * (p - a) * (p - b) |==> p * (p - a) % (p - b)
49:AOR:*(double,double):/(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p * (p - a) * (p - b) |==> p * (p - a) / (p - b)
50:AOR:-(double,double):+(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p - c |==> p + c
51:AOR:*(double,double):%(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p * (p - a) * (p - b) * (p - c) |==> p * (p - a) *
52:AOR:*(double,double):/(double,double):triangle.UpgradedTriangle@area(int,int,int):49:p * (p - a) * (p - b) * (p - c) |==> p * (p - a) /

- 分析可知，对于能够识别出突变体并杀死的测试用例更加重要，对于48,49行的突变体，测试用例基本上都能够杀死它们，所以，对于能够执行到48，49行的测试用例，也就是满足形成三角形可得面积的测试用例，更加重要。

4 Source Code

全部文件请看附件

UpgradedTriangle.java

```
1 package triangle;
2
```

```java
/**
 * @author Han Ruiqian
 * @date 2022/3/30
 */
public class UpgradedTriangle {

    public static enum Type {
        INVALID, SCALENE, EQUILATERAL, ISOSCELES
    };

    public static Type classify(int a, int b, int c) {
        int trian;
        if (a <= 0 || b <= 0 || c <= 0)
            return Type.INVALID;
        trian = 0;
        if (a == b)
            trian = trian + 1;
        if (a == c)
            trian = trian + 2;
        if (b == c)
            trian = trian + 3;
        if (trian == 0)
            if (a + b <= c || a + c <= b || b + c <= a)
                return Type.INVALID;
            else
                return Type.SCALENE;
        if (trian > 3)
            return Type.EQUILATERAL;
        if (trian == 1 && a + b > c)
            return Type.ISOSCELES;
        else if (trian == 2 && a + c > b)
            return Type.ISOSCELES;
        else if (trian == 3 && b + c > a)
            return Type.ISOSCELES;
        return Type.INVALID;
    }

    public static double area(int a, int b, int c) {
        if (a <= 0 || b <= 0 || c <= 0) {
            return 0;
        }
        if (a + b <= c || a + c <= b || b + c <= a) {
            return 0;
        }

        double p = (a + b + c) / 2.0;
        return Math.sqrt(p * (p - a) * (p - b) * (p - c));

    }
}

```

TestSuite2.java

```java
package triangle;

import junit.framework.TestCase;
```

```
4
5  /**
6   * @author Han Ruiqian
7   * @date 2022/3/30
8   */
9  public class TestSuite2 extends TestCase {
10     public void test1() {
11         assertEquals (triangle.UpgradedTriangle.area(0,1301,1), 0, 0.01);
12     }
13     public void test2() {
14         assertEquals (triangle.UpgradedTriangle.area(1108,1,1), 0, 0.01);
15     }
16     public void test3() {
17         assertEquals (triangle.UpgradedTriangle.area(1,0,-665), 0, 0.01);
18     }
19     public void test4() {
20         assertEquals (triangle.UpgradedTriangle.area(1,1,0), 0, 0.01);
21     }
22     public void test5() {
23         assertEquals (triangle.UpgradedTriangle.area(582,582,582),
   146671.79, 0.01);
24     }
25     public void test6() {
26         assertEquals (triangle.UpgradedTriangle.area(1,1088,15), 0, 0.00);
27     }
28     public void test7() {
29         assertEquals (triangle.UpgradedTriangle.area(1,2,450), 0, 0.00);
30     }
31     public void test8() {
32         assertEquals (triangle.UpgradedTriangle.area(1663,1088,823),
   386411.66, 0.01);
33     }
34     public void test9() {
35         assertEquals (triangle.UpgradedTriangle.area(1187,1146,1), 0, 0.00);
36     }
37     public void test10() {
38         assertEquals (triangle.UpgradedTriangle.area(1640,1640,1956),
   1287516.10, 0.01);
39     }
40     public void test11() {
41         assertEquals (triangle.UpgradedTriangle.area(784,784,1956), 0,
   0.00);
42     }
43     public void test12() {
44         assertEquals (triangle.UpgradedTriangle.area(1,450,1), 0, 0.00);
45     }
46     public void test13() {
47         assertEquals (triangle.UpgradedTriangle.area(1146,1,1146), 572.99,
   0.01);
48     }
49     public void test14() {
50         assertEquals (triangle.UpgradedTriangle.area(1640,1956,1956),
   1456172.24, 0.01);
51     }
52     public void test15() {
53         assertEquals (triangle.UpgradedTriangle.area(-1,1,1), 0, 0.01);
54     }
55     public void test16() {
```

```
        assertEquals (triangle.UpgradedTriangle.area(1,-1,1), 0, 0.01);
    }
    public void test17() {
        assertEquals (triangle.UpgradedTriangle.area(1,2,3), 0, 0.01);
    }
    public void test18() {
        assertEquals (triangle.UpgradedTriangle.area(2,3,1), 0, 0.01);
    }
    public void test19() {
        assertEquals (triangle.UpgradedTriangle.area(3,1,2), 0, 0.01);
    }
    public void test20() {
        assertEquals (triangle.UpgradedTriangle.area(1,1,2), 0, 0.01);
    }
    public void test21() {
        assertEquals (triangle.UpgradedTriangle.area(1,2,1), 0, 0.01);
    }
    public void test22() {
        assertEquals (triangle.UpgradedTriangle.area(2,1,1), 0, 0.01);
    }
    public void test23() {
        assertEquals (triangle.UpgradedTriangle.area(1,1,1), 0.43, 0.01);
    }
    public void test24() {
        assertEquals (triangle.UpgradedTriangle.area(0,1,1), 0, 0.01);
    }
    public void test25() {
        assertEquals (triangle.UpgradedTriangle.area(1,0,1), 0, 0.01);
    }
    public void test26() {
        assertEquals (triangle.UpgradedTriangle.area(1,2,-1), 0, 0.01);
    }
    public void test27() {
        assertEquals (triangle.UpgradedTriangle.area(1,1,-1), 0, 0.01);
    }
    public void test28() {
        assertEquals (triangle.UpgradedTriangle.area(0,0,0), 0, 0.01);
    }
    public void test29() {
        assertEquals (triangle.UpgradedTriangle.area(3,4,5), 6, 0.01);
    }

}
```