**实验4 – 变异测试**

学号: 2017141463145 姓名: 龙行超

分数：

# 引言（5分）

变异测试：在细节方面对源程序进行改动的软件测试方法。这些改动就被称之为变异操作，这些操作可以是：模拟典型的应用程序错误，例如大于等于少写了等于；还能是强制的产生有效测试，例如是每个表达式都等于0。变异操作的目的是：帮助测试者发现有效地测试或者定位测试数据的弱点。

变异算子：在符合语法规则前提下， 变异算子定义了从原有程序生成差别极小程序（即变异体） 的转换规则。例如一个典型的算子将“+”操作符号变异为“-”操作符，那么被测试的程序中表达式a+b>c执行该变异算子之后，将会变为a-b>c。

一阶变异体：原有程序p上执行单一变异算子并形成变异体p1，则称p1为p的一阶变异体。

高阶变异体：原有程序p上依次执行多次变异算子并形成变异体p1，则称p1为p的高阶变异体。

可杀除变异体：若存在测试用例t，在变异体p1和原有程序p上的执行结果不一致，则称该变异体p1相对于测试用例集T是可杀除变异体。

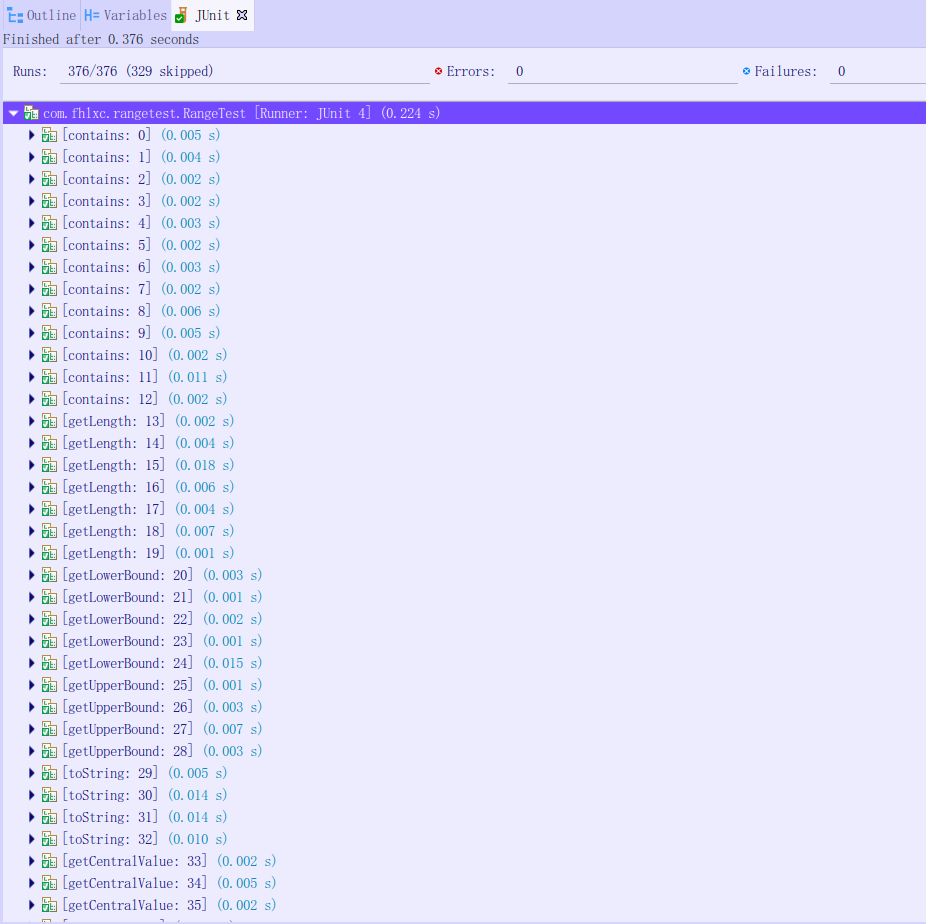
可存活变异体：若不存在任何测试用例t，使得变异体p1和原有程序p上的执行结果不一致，则称该变异体p1相当于测试用例集T是可存活变异体。一般通过设计新的测试用例可杀死一部分可存活测试用例，剩余的可存活变异体可能是等价变异体。

等价变异体：若变异体p1与原有程序p在语法上存在差异，但是在语义上与p一致，则称变异体p1是p的等价变异体。

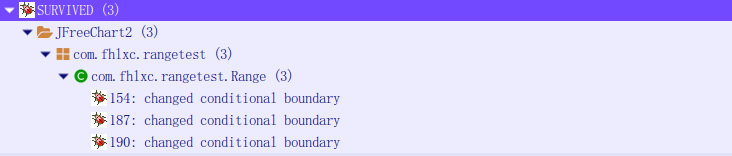
本实验通过eclipse的Pitest插件对上次的实验做变异测试，来完善测试用例。

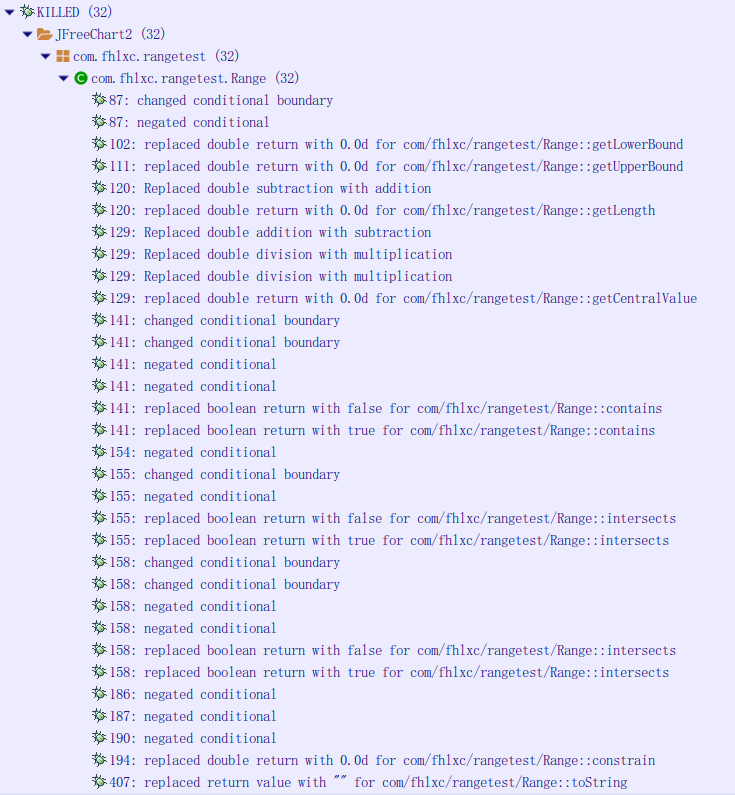
# 变异测试结果分析（90分）

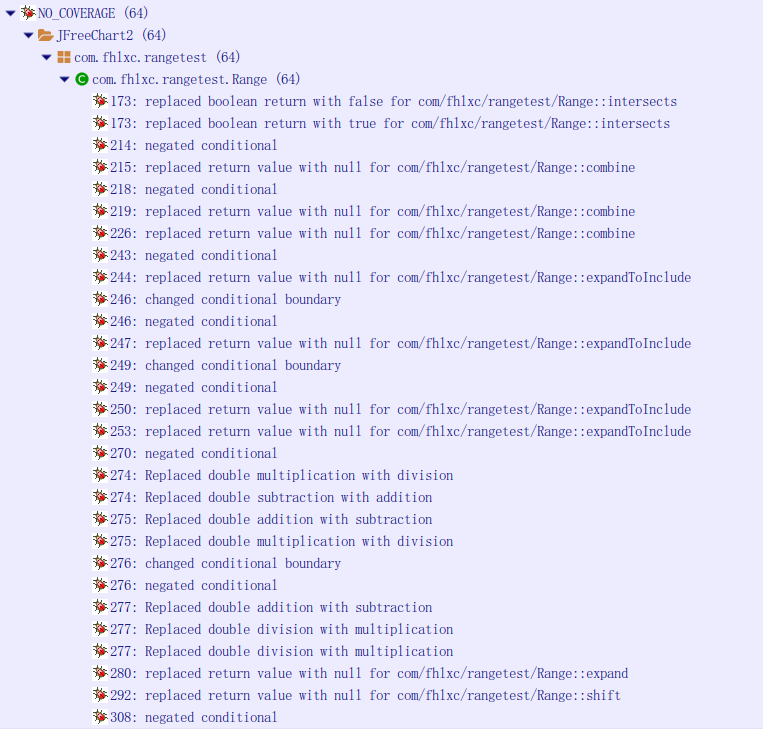
* 1. **测试用例运行结果**

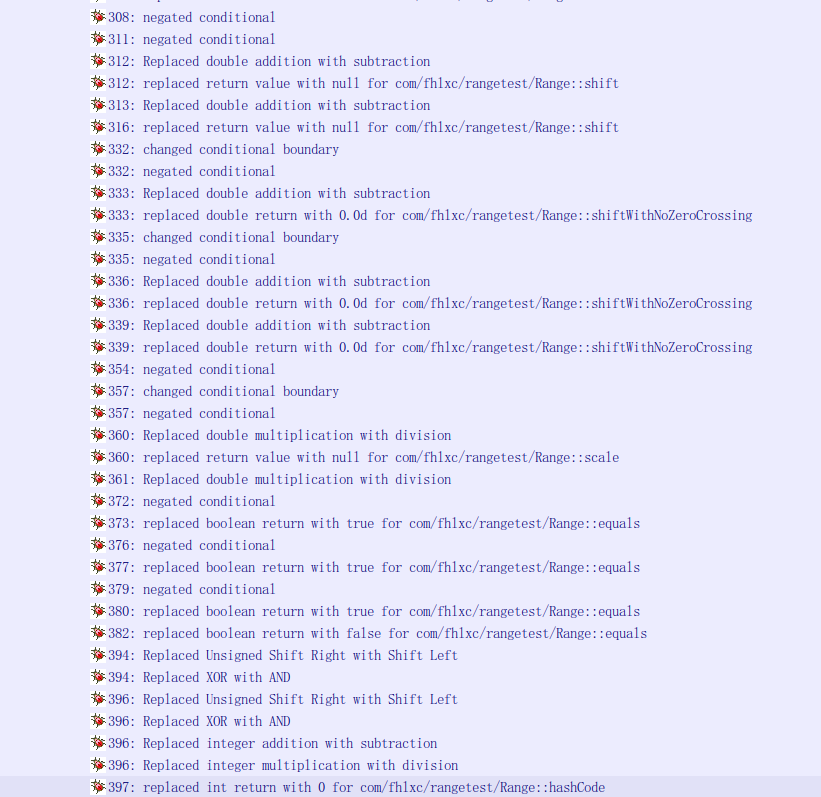


* 1. **变异测试运行结果**

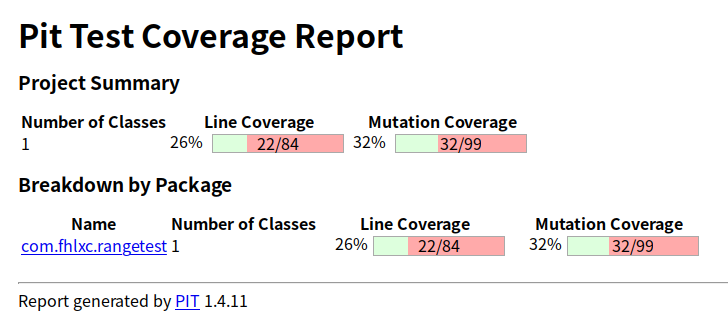




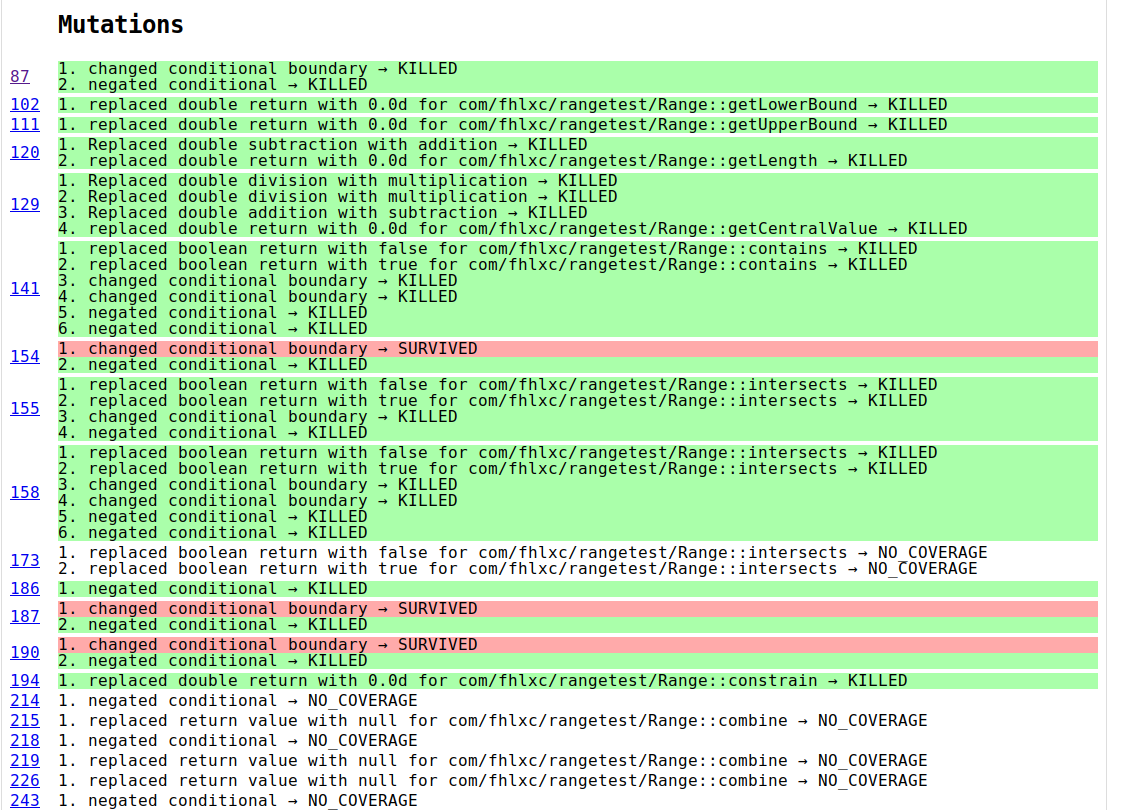


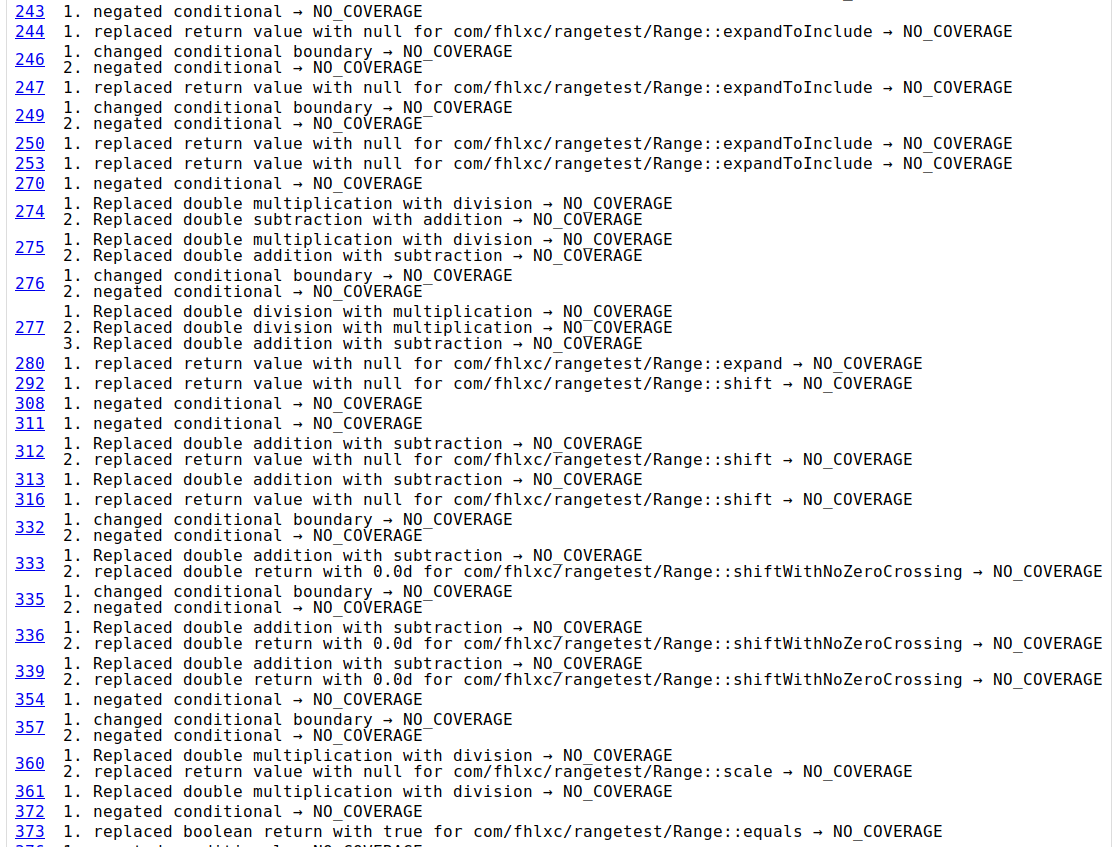


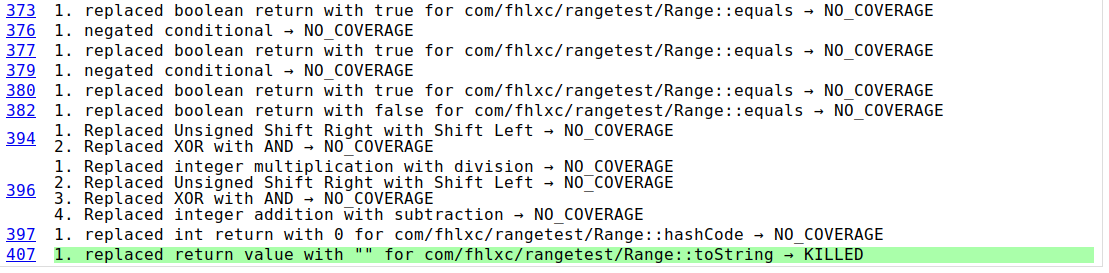
变异覆盖结果



变异覆盖分数截图（很重要）







变异操作截图

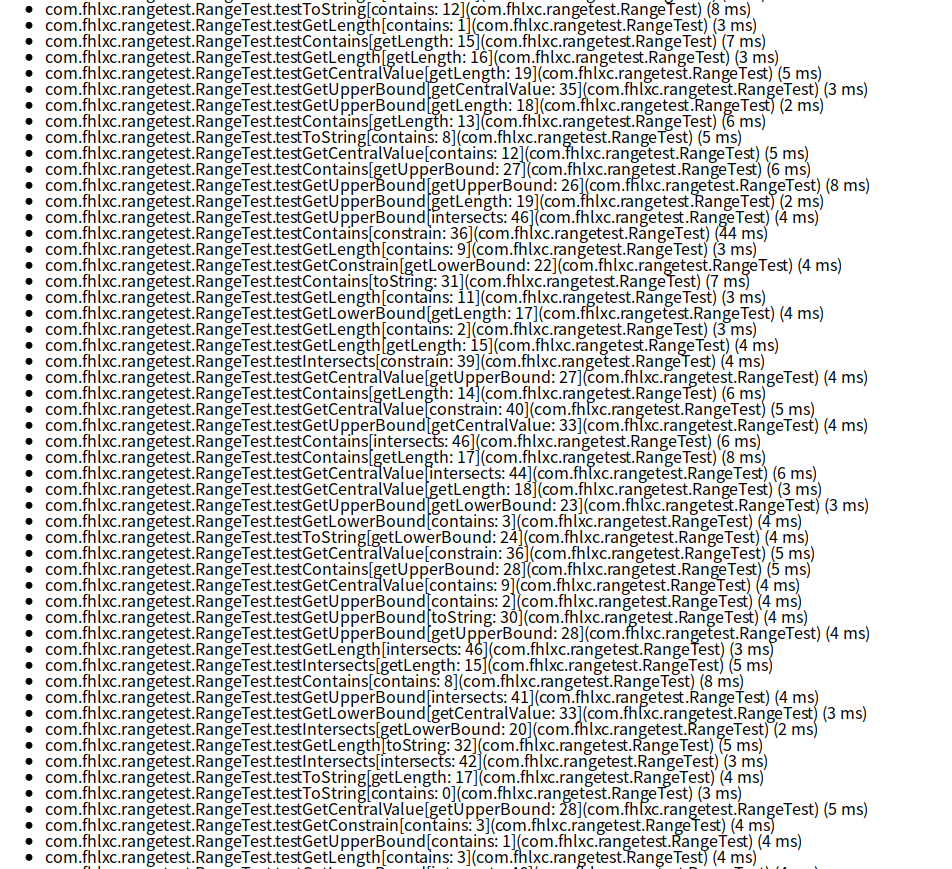
















激活算子和运行时间截图

* 1. **结果分析**

|  |  |
| --- | --- |
| 154行的第1个变异体的变异操作 | changed conditional boundary |
| 状态 | SURVIVED |
| 源代码 | **public** **boolean** intersects(**double** b0, **double** b1) {  **if** (b0 <= **this**.lower) {  **return** (b1 >= **this**.lower);  }  **else** {  **return** (b0 <= **this**.upper && b1 >= b0);  }  } |
| 分析 | 变异操作虽然将b0 <= this.lower变异为b0 < this.lower，这只改变了程序的语法,在语义上没有改变，仍然能够正确输出是否相交，所以这个变异体与原有程序是等价的，因此状态是已存活，并且没有一个测试用例能够杀死这个变异体。 |

|  |  |
| --- | --- |
| 187行的第1个变异体的变异操作 | changed conditional boundary |
| 状态 | SURVIVED |
| 源代码 | **public** **double** constrain(**double** value) {  **double** result = value;  **if** (!contains(value)) {  **if** (value > **this**.upper) {  result = **this**.upper;  }  **else** **if** (value < **this**.lower) {  result = **this**.lower;  }  }  **return** result;  } |
| 分析 | 变异操作虽然将value > this.upper变异为value >= this.upper，这只改变了程序的语法,在语义上没有改变，当值和upper相等时，无论取等与否都是返回的upper，所以变异体与原有程序是等价的，因此状态是已存活，并且没有一个测试用例能够杀死这个变异体。 |

|  |  |
| --- | --- |
| 190行的第1个变异体的变异操作 | changed conditional boundary |
| 状态 | SURVIVED |
| 源代码 | **public** **double** constrain(**double** value) {  **double** result = value;  **if** (!contains(value)) {  **if** (value > **this**.upper) {  result = **this**.upper;  }  **else** **if** (value < **this**.lower) {  result = **this**.lower;  }  }  **return** result;  } |
| 分析 | 变异操作虽然将value < this.lower变异为value <= this.lower，这只改变了程序的语法,在语义上没有改变，当值和lower相等时，无论取等与否都是返回的lower，所以变异体与原有程序是等价的，因此状态是已存活，并且没有一个测试用例能够杀死这个变异体。 |

|  |  |
| --- | --- |
| 173行的第1、2个变异体的变异操作 | 1. replaced boolean return with false for com/fhlxc/rangetest/Range::intersects  2. replaced boolean return with true for com/fhlxc/rangetest/Range::intersects |
| 状态 | NO\_COVERAGE |
| 源代码 | **public** **boolean** intersects(Range range) {  **return** intersects(range.getLowerBound(), range.getUpperBound());  } |
| 分析 | 只选了八个方法来测试，未包含这个，所以是未覆盖。 |

# 你的感受（5分）

【你的感受、想法、建议等。比如：在哪些条件下做变异测试比较好？哪些方面是变异测试所做不到的？等等。对于本实验，你有什么建议和意见？】