## 实验源码

下面三个函数分别对应实现的三个问题的代码求解：

1. **package** org.example.code;
3. /\*\*
4. \* @program: part2
5. \* @description:
6. \* @author: CuiJieXiang\_1023
7. \* @create: 2023-11-20 12:18
8. \*\*/
9. **public** **class** code {
11. **public** **int** findDuplicate(**int**[] nums) {
12. // 初始化快慢指针
13. **int** slow = nums[0];
14. **int** fast = nums[0];
16. // 快指针每次前进两步，慢指针每次前进一步，直到两指针相遇
17. **do** {
18. slow = nums[slow];
19. fast = nums[nums[·fast]];
20. } **while** (slow != fast);
22. // 将慢指针重置为起始位置，并继续前进，直到与快指针相遇
23. slow = nums[0];
24. **while** (slow != fast) {
25. slow = nums[slow];
26. fast = nums[fast];
27. }
29. // 返回重复的整数
30. **return** slow;
31. }
33. **public** **int** findOddOccurrence(**int**[] nums) {
34. **int** result = 0;
35. **for** (**int** num : nums) {
36. result ^= num;  // 使用异或运算找到出现奇数次的数字
37. }
38. **return** result;
39. }
41. **public** **int** maxProfit(**int**[] prices) {
42. **int** maxProfit = 0;
44. **for** (**int** i = 1; i < prices.length; i++) {
45. // 如果当前价格高于前一天的价格，说明可以获利，累加到总利润中
46. **if** (prices[i] > prices[i - 1]) {
47. maxProfit += prices[i] - prices[i - 1];
48. }
49. }
51. **return** maxProfit;
52. }
53. }

## 测试程序

题目一测试代码：testFindDuplicate.class

1. **package** org.example.test;

4. **import** org.example.code.code;
5. **import** org.junit.Test;
7. **import** **static** org.junit.Assert.assertEquals;
9. /\*\*
10. \* @program: part2
11. \* @description:
12. \* @author: CuiJieXiang\_1023
13. \* @create: 2023-11-20 12:43
14. \*\*/
15. **public** **class** test\_findDuplicate {
17. @Test
18. **public** **void** testAdd() {
19. code c = **new** code();
20. // 测试案例
21. assertEquals(2, c.findDuplicate(**new** **int**[]{1, 3, 4, 2, 2}));
22. assertEquals(3, c.findDuplicate(**new** **int**[]{5, 6, 3, 2, 4, 1, 7, 3}));
23. assertEquals(3, c.findDuplicate(**new** **int**[]{5, 6, 3, 2, 4, 1, 8, 7, 3}));
24. assertEquals(1, c.findDuplicate(**new** **int**[]{1, 1, 2, 3, 4, 5}));
25. assertEquals(4, c.findDuplicate(**new** **int**[]{1, 2, 3, 4, 4, 5}));
26. assertEquals(9, c.findDuplicate(**new** **int**[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 10}));
27. assertEquals(5, c.findDuplicate(**new** **int**[]{5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5}));
28. assertEquals(3, c.findDuplicate(**new** **int**[]{1, 2, 3, 3, 4, 5, 6, 7, 8, 9}));
29. assertEquals(1, c.findDuplicate(**new** **int**[]{1, 1}));
30. }
31. }

题目二测试代码：testFindOdd.class:

1. **package** org.example.test;

4. **import** org.example.code.code;
5. **import** org.junit.Test;
7. **import** **static** org.junit.Assert.assertEquals;
9. /\*\*
10. \* @program: part2
11. \* @description:
12. \* @author: CuiJieXiang\_1023
13. \* @create: 2023-11-20 13:41
14. \*\*/
15. **public** **class** test\_findOdd {

18. @Test
19. **public** **void** test\_findOddOccurrence() {
20. code c = **new** code();
22. // 测试案例
23. assertEquals(8, c.findOddOccurrence(**new** **int**[]{2, 3, 5, 7, 8, 7, 3, 2, 5, 9, 9}));
24. assertEquals(3, c.findOddOccurrence(**new** **int**[]{3, 5, 4, 8, 3, 5, 3, 8, 4}));
25. assertEquals(5, c.findOddOccurrence(**new** **int**[]{2, 3, 2, 3, 4, 4, 5}));
26. assertEquals(1, c.findOddOccurrence(**new** **int**[]{1}));
27. assertEquals(1, c.findOddOccurrence(**new** **int**[]{1, 2, 2}));
28. assertEquals(0, c.findOddOccurrence(**new** **int**[]{0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8}));
29. assertEquals(7, c.findOddOccurrence(**new** **int**[]{6, 7, 8, 6, 8}));
30. assertEquals(100, c.findOddOccurrence(**new** **int**[]{100, 200, 300, 400, 500, 200, 300, 400, 500}));
31. assertEquals(9, c.findOddOccurrence(**new** **int**[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8}));
32. }
33. }

题目三测试代码：testMaxProfit.class:

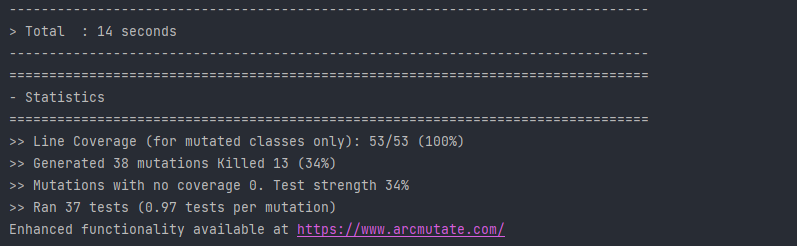
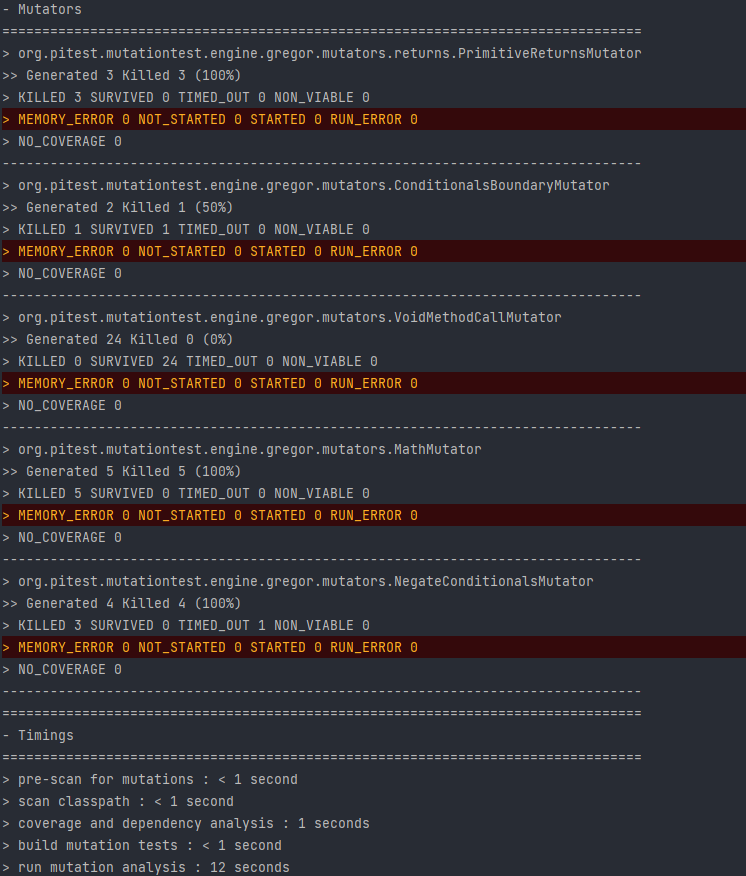
1. **package** org.example.test;
3. **import** org.example.code.code;
4. **import** org.junit.Test;
6. **import** **static** org.junit.Assert.assertEquals;
8. /\*\*
9. \* @program: part2
10. \* @description:
11. \* @author: CuiJieXiang\_1023
12. \* @create: 2023-11-20 13:43
13. \*\*/
14. **public** **class** test\_maxProfit {
15. @Test
16. **public** **void** test\_maxProfit() {
17. code c = **new** code();
19. // 测试案例
20. assertEquals(7, c.maxProfit(**new** **int**[]{7, 1, 5, 3, 6, 4}));
21. assertEquals(0, c.maxProfit(**new** **int**[]{7, 6, 4, 3, 1}));
22. assertEquals(4, c.maxProfit(**new** **int**[]{1, 2, 3, 4, 5}));
23. assertEquals(0, c.maxProfit(**new** **int**[]{5, 4, 3, 2, 1}));
24. assertEquals(0, c.maxProfit(**new** **int**[]{1}));
25. assertEquals(0, c.maxProfit(**new** **int**[]{}));
26. }
27. }

## 使用工具得到的变异测试结果

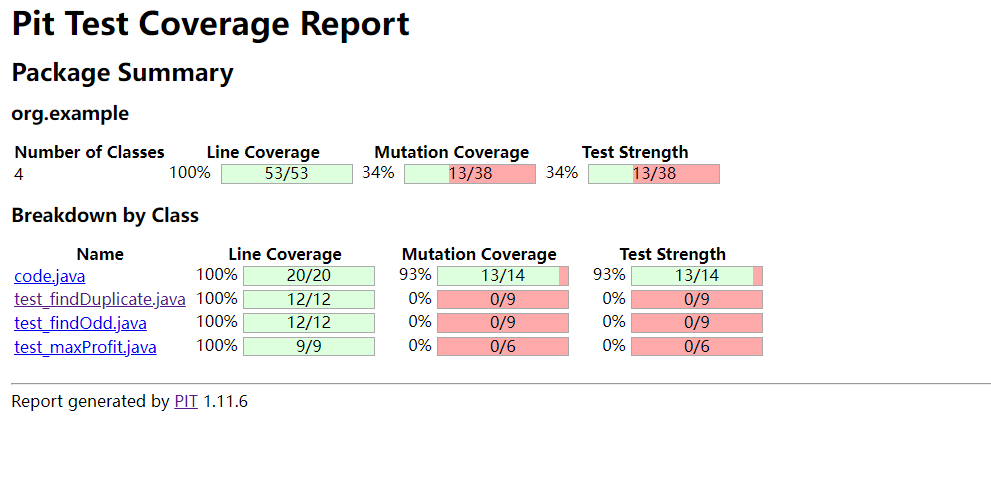
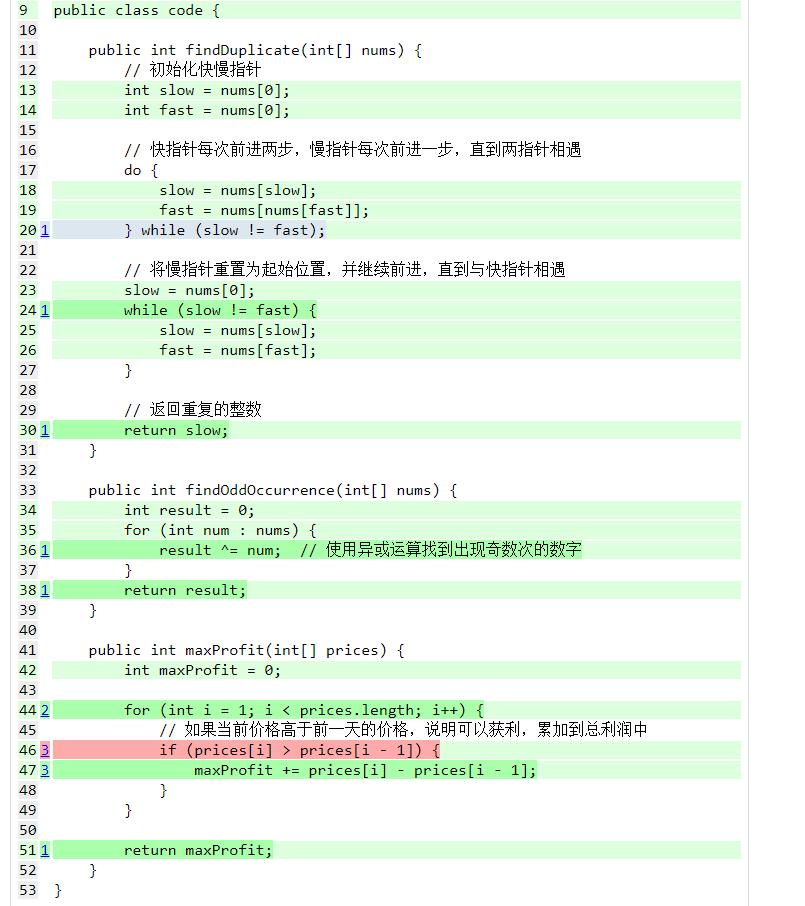
使用Maven构造Java项目后，在pom.xml 中写入下面代码添加PiTest插件：

1. <build>
2. <finalName>maven-mutation-testing</finalName>
3. <plugins>
4. <plugin>
5. <groupId>org.apache.maven.plugins</groupId>
6. <artifactId>maven-surefire-plugin</artifactId>
7. <version>3.0.0-M1</version>
8. </plugin>
10. <plugin>
11. <groupId>org.pitest</groupId>
12. <artifactId>pitest-maven</artifactId>
13. <version>${pitest.version}</version>
15. <executions>
16. <execution>
17. <id>pit-report</id>
18. <phase>test</phase>
19. <goals>
20. <goal>mutationCoverage</goal>
21. </goals>
22. </execution>
23. </executions>
25. <!-- https://github.com/hcoles/pitest/issues/284 -->
26. <!-- Need **this** to support JUnit 5 -->
27. <dependencies>
28. <dependency>
29. <groupId>org.pitest</groupId>
30. <artifactId>pitest-junit5-plugin</artifactId>
31. <version>0.8</version>
32. </dependency>
33. </dependencies>
34. <configuration>
35. <targetClasses>
36. <param>org.example.code.\*</param>
37. </targetClasses>
38. <targetTests>
39. <param>org.example.test.\*</param>
40. </targetTests>
41. </configuration>
42. </plugin>
44. </plugins>
45. </build>

随后使用maven命令 mvn org.pitest:pitest-maven:mutationCoverage 或是使用Idea 的 PITEST插件执行变异测试，得到如下结果：



同时在执行完后就会在**target/pit-reports/YYYYMMDDHHMI**目录生成一个html的报告，html报告的内容如下：

代码整体覆盖报告：  
code 代码覆盖、变异情况报告：

