

# HW01

PB19071405 王昊元

2022 年 3 月 29 日

1. 等价类表如下：（其中认为接种者数字编号从 00000 开始）

功能项	有效等价类	编号	无效等价类	编号
接种号	10 位	1	少于 10 位（包括空）	2
			多于 10 位	3
	首位为 A、B 或 C	4	D-Z 的大写字母	5
			小写字母	6
			其他字符	7
	2~3 位为 01~12 的数字	8	00	9
			13~99	10
			包含其它字符	11
	4~5 位为 07、13 或 23	12	00~06	13
			08~12	14
			14~22	15
			24~99	16
			包含其它字符	17
	6~10 位为数字	18	包含其它字符	19

覆盖数据如下：

序号	输入数据	覆盖等价类	预期输出	备注
1	C030700000	1、4、8、12、18	合法输入	
2	C03070000	2、4、8、12	非法输入	因最后不足 5 位，故不认为覆盖第 18 号等价类
3	C0307000000	3、4、8、12、18	非法输入	
4	F030700000	1、5、8、12、18	非法输入	
5	c030700000	1、6、8、12、18	非法输入	
6	>030700000	1、7、8、12、18	非法输入	
7	C000700000	1、4、9、12、18	非法输入	
8	C130700000	1、4、10、12、18	非法输入	
9	CAB0700000	1、4、11、12、18	非法输入	
10	C030400000	1、4、8、13、18	非法输入	
11	C031100000	1、4、8、14、18	非法输入	
12	C032100000	1、4、8、15、18	非法输入	
13	C039900000	1、4、8、16、18	非法输入	
14	C03AB00000	1、4、8、17、18	非法输入	
15	C03070000A	1、4、8、12、19	非法输入	

2.

代码 1 流图如下所示：

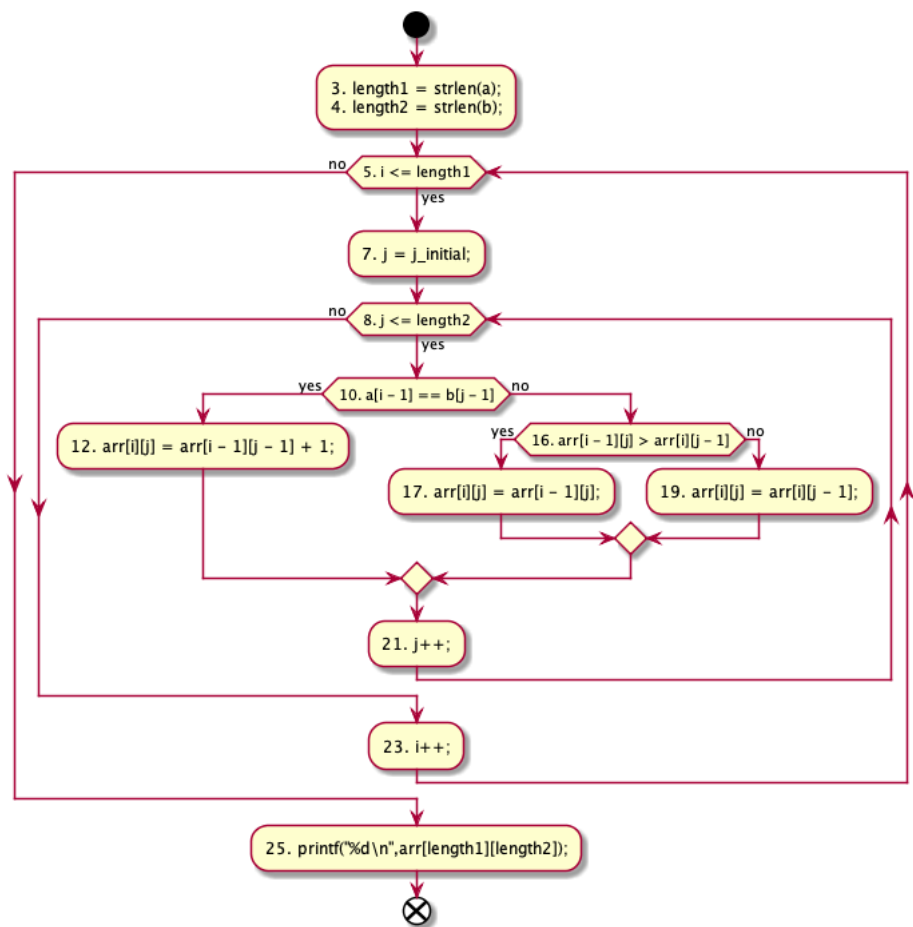


图 1: 代码 1 流图

环路复杂度为：4 + 1 = 5

测试用例如下：

独立路径	测试用例	结果
3,4-5-25	i = 10, j_initial = 1, a = "ABCD", b = "BAD"	打印 0
3,4-5-7-8-23-5-25	i = 1, j_initial = 10, a = "ABCD", b = "BAD"	打印 0
3,4-5-7-8-10-12-21-8-23-5-25	i = 1, j_initial = 1, a = "A", b = "A"	打印 1
3,4-5-7-8-10-16-17-21-8-23-5-25	该路径实际不存在，无测试样例	
3,4-5-7-8-10-16-19-21-8-23-5-25	i = 1, j_initial = 1, a = "A", b = "B"	打印 0

代码 2 流图如下所示：

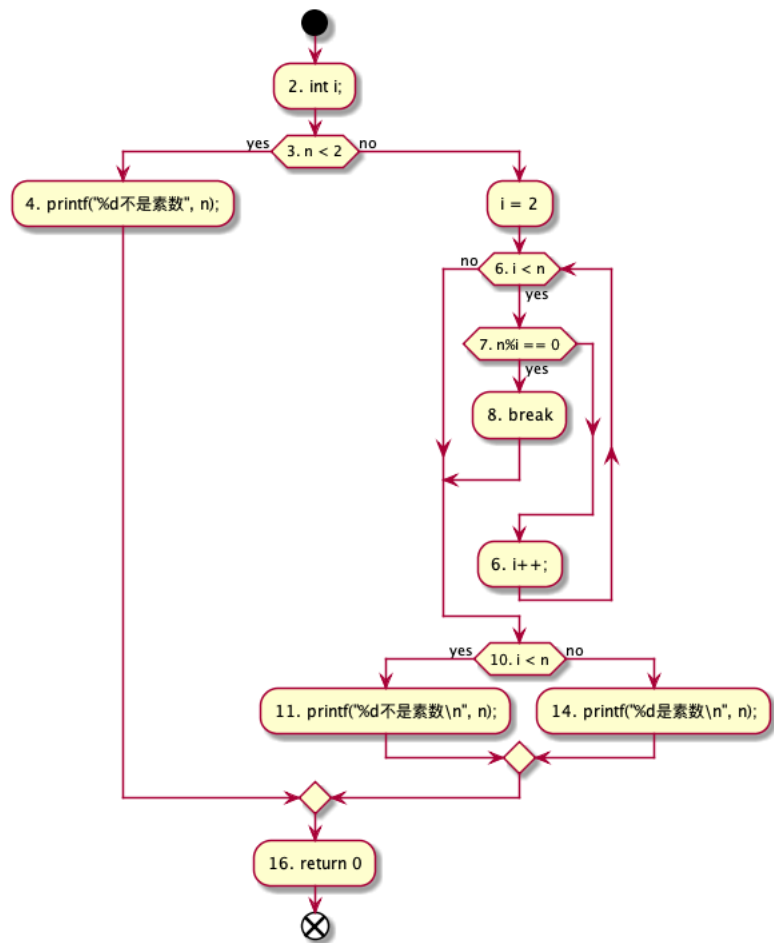


图 2: 代码 2 流图

环路复杂度为:  $4 + 1 = 5$

测试用例如下:

独立路径	测试用例	结果
2,3,4,16	$n = 1$	打印 1 不是素数
2,3,6,10,11,16	该路径实际不存在, 无测试样例	
2,3,6,10,14,16	$n = 2$	打印 2 是素数
2,3,6,7,6,10,14,16	$n = 3$	打印 3 是素数
2,3,6,7,8,10,11,16	$n = 4$	打印 4 不是素数

### 3. 包含不同严重程度 bug 的 python 程序

```

1 class MyException:
2     def __init__(self, msg):
3         print(msg)
4         return self
5
6 def foo(s: str, l: int):
7     l = 10
  
```

```
8      try:
9          if l == 0:
10             print("no string?")
11          elif l > 0:
12             print("The length of string \"{}\": {}".format(s, l))
13          elif l == 0:
14             print("?")
15          else:
16             raise MyException("length is negative.")
17      except Exception as e:
18          pass
19      msg = s + l
20      print(msg)
21      return
22      ++l
23      print("The result of code \"++l\": {}".format(l))
24      break
25      print("function foo done.")
26
27 s = "PB19071405"
28 foo(s, 5)
```

SonarQube 检测结果如下:

Bug00

test.py

29

0.0%

6

0

1

0

```
1 -- class MyException:
2 --     def __init__(self, msg):
3 --         print(msg)
4 --         return self
5 --
6 -- def foo(s: str, l: int):
7 --     l = 10
8 --     try:
9 --         if l == 0:
10 --             print("no string")
11 --         elif l > 4:
12 --             print("The length of string {}".format(s, l))
13 --         elif l == 0:
14 --             print("")
15 --         else:
16 --             raise MyException("length is negative.")
17 --     except Exception as e:
18 --         pass
19 --         msg = s + l
20 --         print(msg)
21 --         return
22 --         ++l
23 --         print("The result of code {}".format(l))
24 --         break
25 --         print("function foo done.")
26 --
27 -- s = "P819871485"
28 -- foo(s, 5)
29 --
```

test.py

Remove this return value. Why is this an issue?

19 minutes ago ▾ L4 🔗 📄 ▾

🐛 Bug ▾

🔴 Blocker ▾

🔵 Open ▾

Not assigned ▾

5min effort

Comment

🔗 No tags ▾

This branch duplicates the one on line 9. Why is this an issue?

27 minutes ago ▾ L13 🔗 📄 ▾

🐛 Bug ▾

🔴 Major ▾

🔵 Open ▾

Not assigned ▾

10min effort

Comment

🔗 piltall, unused ▾

Change this code so that it raises an object deriving from BaseException. Why is this an issue?

19 minutes ago ▾ L16 🔗 📄 ▾

🐛 Bug ▾

🔴 Blocker ▾

🔵 Open ▾

Not assigned ▾

5min effort

Comment

🔗 python3 ▾

Remove or refactor this statement; it has no side effects. Why is this an issue?

35 minutes ago ▾ L22 🔗 📄 ▾

🐛 Bug ▾

🔴 Major ▾

🔵 Open ▾

Not assigned ▾

10min effort

Comment

🔗 cwe, unused ▾

This statement doesn't produce the expected result, replace use of non-existent pre-increment operator

Why is this an issue?

35 minutes ago ▾ L22 🔗 📄 ▾

🐛 Bug ▾

🔴 Major ▾

🔵 Open ▾

Not assigned ▾

5min effort

Comment

🔗 No tags ▾

Remove this "break" statement Why is this an issue?

35 minutes ago ▾ L24 🔗 📄 ▾

🐛 Bug ▾

🔴 Critical ▾

🔵 Open ▾

Not assigned ▾

10min effort

Comment

🔗 No tags ▾

图 3: 完整代码及检测出的 bug



(a) Function parameters initial values should not be ignored



(b) All code should be reachable

图 4: 一些在完整代码中没有被检测出的 bug

修复后的代码如下（修改方案及原因由对应注释说明）：

```

1 # Raised Exceptions must derive from BaseException
2 # 修复方案：将 MyException 类继承于 BaseException
3 class MyException(Exception):
4     def __init__(self, msg):
5         print(msg)
6         # "__init__" should not return a value
7         # __init__()函数应返回 None
8         # 修复方案：不 return 或 return None
9
10        # return self
11        return None
12
13 def foo(s: str, l: int):

```

```

14 # Function parameters initial values should not be ignored
15 # 根据官方说明，这种行为只是像一个bug，
16 # 因为参数传进来的值并没有用，设置的参数也就失去了意义
17 # 但实际coding过程中可能很少会出现这个bug
18 # 修复方案：直接注释/删除掉
19 # 注：本来打算通过修改变量名的方式来修复这个bug，
20 # 但考虑到修改变量名会产生 Unused local variables should be removed 的 Code
    Smell，
21 # 最后决定通过注释/删除来修复
22
23 # l = 10
24 try:
25     if l == 0:
26         print("no string?")
27     elif l > 0:
28         print("The length of string \"{}\": {}".format(s, l))
29     # Related "if/else if" statements should not have the same condition
30     # 实际coding过程中也很少遇到，
31     # 但是不排除太多情况用if/else if实现时脑袋混乱最后导致该bug的情况
32     # 修复方案：注释/删除一个分支即可
33
34     # elif l == 0:
35     #     print("?")
36 else:
37     # Raised Exceptions must derive from BaseException
38     # 该bug实际报错在此处，因为只有解释该语句时才会发现这个没有继承自
        BaseException 的类被 raise 了
39     # 修复方案：修改 MyException 继承自 BaseException
        raise MyException("length is negative.")
40 except Exception as e:
41     # 其实这个地方应该有一个 Unused local variables should be removed 的 Code
        Smell 来着（以我个人的理解），
42     # 但是没有出现。事实上我个人的习惯是将 Exception 打印出来，也方便 debug
43     # 修复方案：使用变量 e （如：print(e)）
44
45     # pass
46     print(e)
47
48 # Operators should be used on compatible types
49 # 不过应该是因为在定义函数的时候其实并不知道 s 和 l 的类型，所以没检测出来这个
    Blocker 的 bug
50 # 修复方案：可以通过try - except 解决
51 # 但其实平时也只会是在有可能产生 Exception 的地方加 try，或者在函数刚开始的时候进
    行变量类型的检查之类的
52 try:
53     msg = s + l
54     print(msg)
55 except Exception as e:
56     print(e)

```

```

57     # All code should be reachable
58     # 修复方案：删除/注释 return
59
60     # return
61
62     # Increment and decrement operators should not be used
63     # 修复方案：删除/注释
64     # ++l
65     print("The result of code \"++l\": {}".format(l))
66     # "break" and "continue" should not be used outside a loop
67     # 修复方案：删除 break
68     # break
69     print("function foo done.")
70
71 s = "PB19071405"
72 foo(s, 5)

```

修复后进行测试，结果如下：

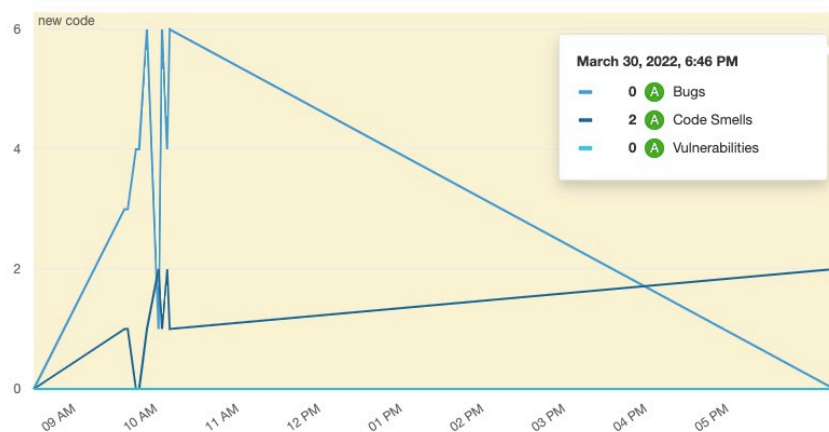


图 5: 修复后的代码及修复方案

图 5 中有两个 Code Smells，是由于写修复方案的注释与空格导致的。