软件测试技术上机报告

--字符串匹配

序　号 _____

学　号 _____

姓　名 _____

教　师 _____

目录

1 问题描述

1.1 题目描述

给出两个非空字符串 s 和 p，若 s 的区间[l ,r]子串和 p 完全相同，则称 p 在 s 中出现了,其出现位置为 l。(字符串 s 的长度不小于字符串 p 的长度)

1.2 输入格式

第一行为一个字符串，即为 s。
第二行为一个字符串，即为 p。

1.3 输出格式

若 p 在 s 中出现了，则输出出现位置 l，否则，输出-1。

2 被测代码

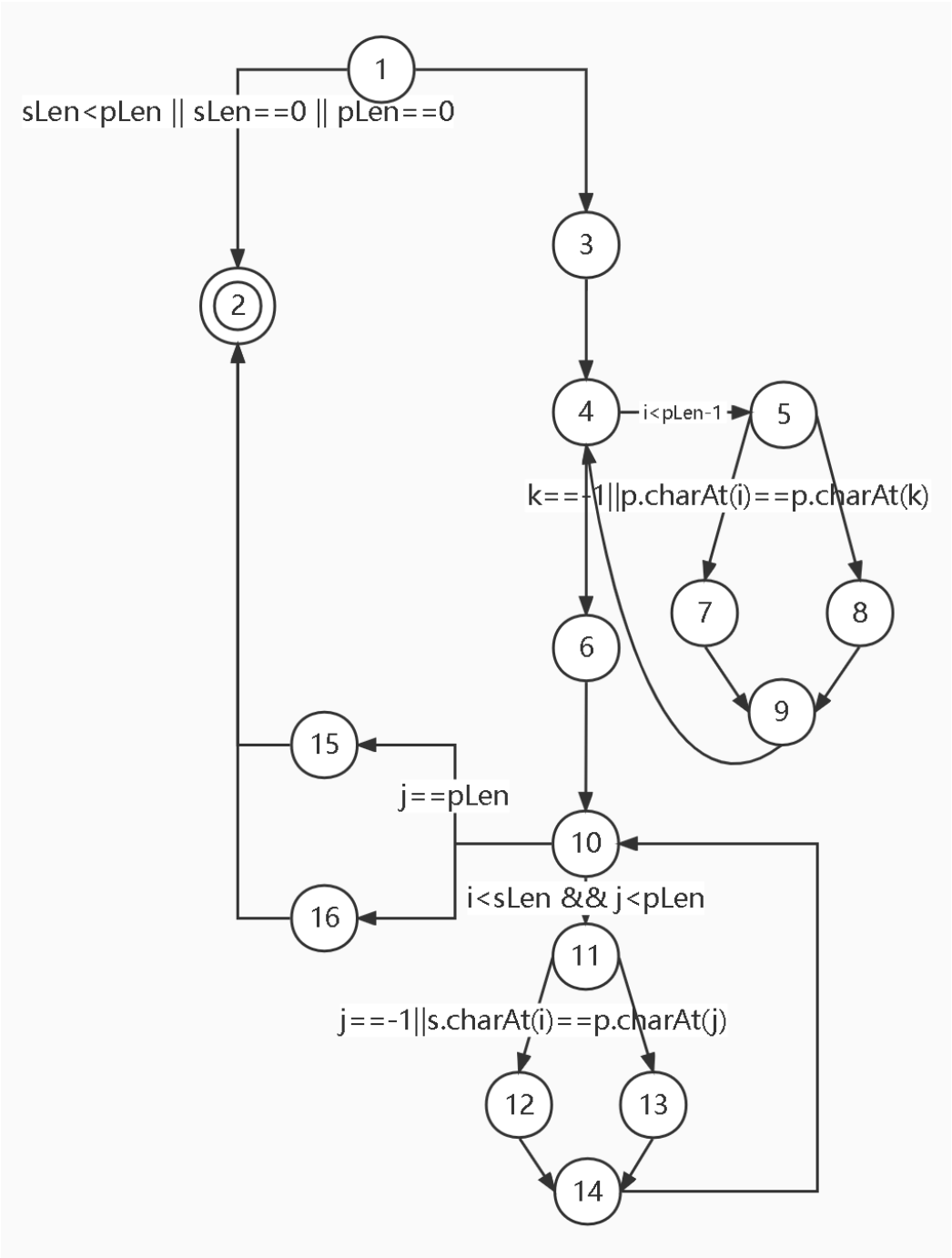
2.1 kmpSearch.java

```
package org;
public class kmpSearch {
    /**
     * O(m+n)
     *
     * @param s 目标串
     * @param p 模式串
     * @return 如果匹配成功，返回下标，否则返回-1
     */
    public static int kmp(String s, String p) {
        int sLen = s.length();
        int pLen = p.length();
        if (sLen < pLen || sLen==0 ||pLen==0) {
            return -1;
        }
```

```
        int[] next = new int[pLen];
        next[0] = -1;
        int i = 0, k = -1;
        while (i < pLen - 1) {
            // p[k]表示前缀，p[i]表示后缀
            if (k == -1 || p.charAt(i) == p.charAt(k)) {
                ++k;
                ++i;
                next[i] = k;
            } else {
                k = next[k];
            }
        }
        // matching: O(n)
        i = 0;
        int j = 0;
        while (i < sLen && j < pLen) {
            //①如果j = -1，或者当前字符匹配成功（即S[i] == P[j]），都令
i++, j++
            if (j == -1 || s.charAt(i) == p.charAt(j)) {
                i++;
                j++;
            } else {
                //②如果j != -1，且当前字符匹配失败（即S[i] != P[j]），则
令 i 不变，j = next[j]
                //next[j]即为j所对应的next值
                j = next[j];
            }
        }
        if (j == pLen) {
            return i - j;
        } else {
            return -1;
        }
    }
}
```

---

3 控制流图



图表 1 控制流图

4 黑盒测试用例设计

4.1 因果图

C1：S 为空字符串
C2：P 为空字符串
C3：S 的长度不小于 P 的长度
C4：S 可以匹配到 P
E1：输出-1
E2：输出 S 匹配到 P 的下标

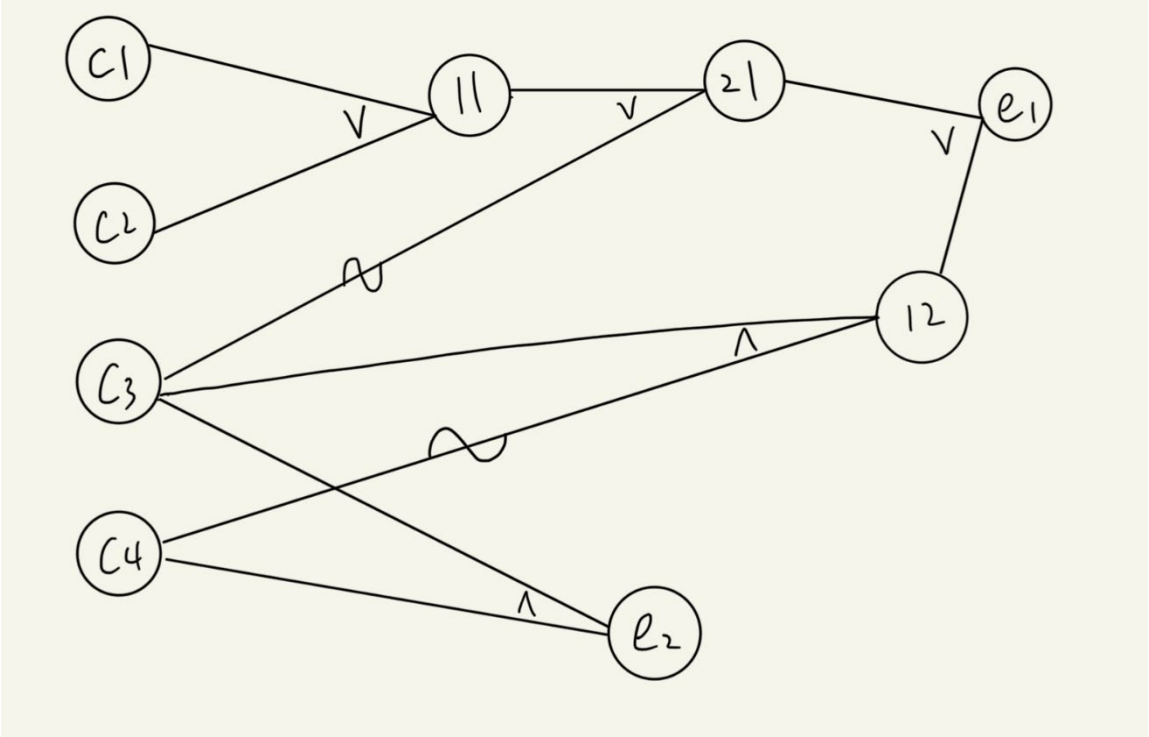


图 1 因果图

4.2 决策表

表格 1 决策表

	1	2	3	4	5
C1:S 为空字符串?	T	F	F	F	F
C2:P 为空字符串?	-	T	F	F	F
C3:S 的长度不小于 P 的长度?	-	-	F	T	T

C4:S 可以匹配到 P?	**-**	**-**	**-**	**T**	**F**
输出-1	✓	✓	✓		✓
输出 S 匹配到 P 的下标				✓	

表格 2 测试用例

测试用例	S	P	预期输出
TC1	**“abc”**	**“abcd“**	**-1**
TC2	**“abcde“**	**“cde“**	**2**
TC3	**“abcde“**	**“ace“**	**-1**
TC4	**“”**	**“ace“**	**-1**
TC5	**“abc“**	**“”**	**-1**

5 白盒测试用例设计

5.1 基于控制流（考虑覆盖准则）

5.1.1 Vertex Coverage

TP

1) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 5 7 9 4 5 8 9 4 5 8 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 15 2]
2) [1 3 4 5 7 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 13 14 10 11 13 14 10 11 12 14 10 16 2]

测试用例	S	P	预期输出
TC1	**“qabaca“**	**“abaca“**	**1**
TC2	**“abc”**	**“bbc”**	**-1**

5.1.2 Edge Coverage

TR

1) [1 2]
2) [1 3]
3) [3 4]
4) [4 5]
5) [4 6]
6) [5 7]

7) [5 8]
8) [7 9]
9) [8 9]
10) [9 4]
11) [6 10]
12) [10 11]
13) [10 15]
14) [10 16]
15) [11 12]
16) [11 13]
17) [12 14]
18) [13 14]
19) [14 10]
20) [15 2]
21) [16 2]

TP

1) [1 2]
2) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 5 7 9 4 5 8 9 4 5 8 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 15 2]
3) [1 3 4 5 7 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 13 14 10 11 13 14 10 11 12 14 10 16 2]

测试用例	S	P	预期输出
TC1	**“abc”**	**“abcd“**	**-1**
TC2	**“qabaca“**	**“abaca“**	**1**
TC3	**“abc”**	**“bbc”**	**-1**

5.1.3 Edge-Pair Coverage

TR

1) [1 3 4]
2) [3 4 5]
3) [3 4 6]
4) [4 5 7]
5) [4 5 8]
6) [4 6 10]
7) [5 7 9]
8) [5 8 9]
9) [7 9 4]
10) [8 9 4]

11) [9 4 6]
12) [9 4 5]
13) [6 10 11]
14) [6 10 15]
15) [6 10 16]
16) [10 11 12]
17) [10 11 13]
18) [10 15 2]
19) [10 16 2]
20) [11 12 14]
21) [11 13 14]
22) [12 14 10]
23) [13 14 10]
24) [14 10 11]
25) [14 10 15]
26) [14 10 16]

TP

1) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 5 7 9 4 5 8 9 4 5 8 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 15 2]
2) [1 3 4 5 7 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 13 14 10 11 13 14 10 11 12 14 10 16 2]
3) [1 3 4 6 10 11 12 14 10 15 2]

由于代码逻辑性：下述路径均不可能出现
4->5->8
6->10->15
6->10->16

测试用例	S	P	预期输出
TC1	**“qabaca“**	**“abaca“**	**1**
TC2	**“abc”**	**“bbc”**	**-1**
TC3	**“a”**	**“a”**	**0**

5.1.4 Prime Path Coverage

Prime Path

1) [1 3 4 5 7 9]
2) [1 3 4 5 8 9]
3) [1 3 4 6 10 15 2]

4) [1 3 4 6 10 16 2]
5) [1 3 4 6 10 11 12 14]
6) [1 3 4 6 10 11 13 14]
7) [4 5 7 9 4]
8) [4 5 8 9 4]
9) [5 7 9 4 5]
10) [5 8 9 4 5]
11) [5 7 9 4 6 10 15 2]
12) [5 7 9 4 6 10 16 2]
13) [5 8 9 4 6 10 15 2]
14) [5 8 9 4 6 10 16 2]
15) [5 7 9 4 6 10 11 12 14]
16) [5 7 9 4 6 10 11 13 14]
17) [5 8 9 4 6 10 11 12 14]
18) [5 8 9 4 6 10 11 13 14]
19) [7 9 4 5 7]
20) [7 9 4 5 8]
21) [8 9 4 5 8]
22) [8 9 4 5 7]
23) [9 4 5 7 9]
24) [9 4 5 8 9]
25) [10 11 12 14 10]
26) [10 11 13 14 10]
27) [11 12 14 10 11]
28) [11 13 14 10 11]
29) [11 12 14 10 15 2]
30) [11 12 14 10 16 2]
31) [11 13 14 10 15 2]
32) [11 13 14 10 16 2]
33) [12 14 10 11 12]
34) [12 14 10 11 13]
35) [13 14 10 11 13]
36) [13 14 10 11 12]
37) [14 10 11 12 14]
38) [14 10 11 13 14]
39) [1 2]

TP

1) [1 2]
2) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 5 7 9 4 5 8 9 4 5 8 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 15 2]
3) [1 3 4 5 7 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 13 14 10 11 13 14 10 11 12 14 10 16 2]
4) [1 3 4 6 10 11 12 14 10 15 2]

由于代码逻辑性：下述路径均不可能出现
4->5->8
8->9->4->6
6->10->15
6->10->16
13->14->10->15
13->14->10->16

测试用例	S	P	预期输出
TC1	**“abc”**	**“abcd“**	**-1**
TC2	**“qabaca“**	**“abaca“**	**1**
TC3	**“abc”**	**“bbc”**	**-1**
TC4	**“a”**	**“a”**	**0**

5.1.5 Complete Path Coverage

因为程序存在循环，所以无法做到全路径覆盖。

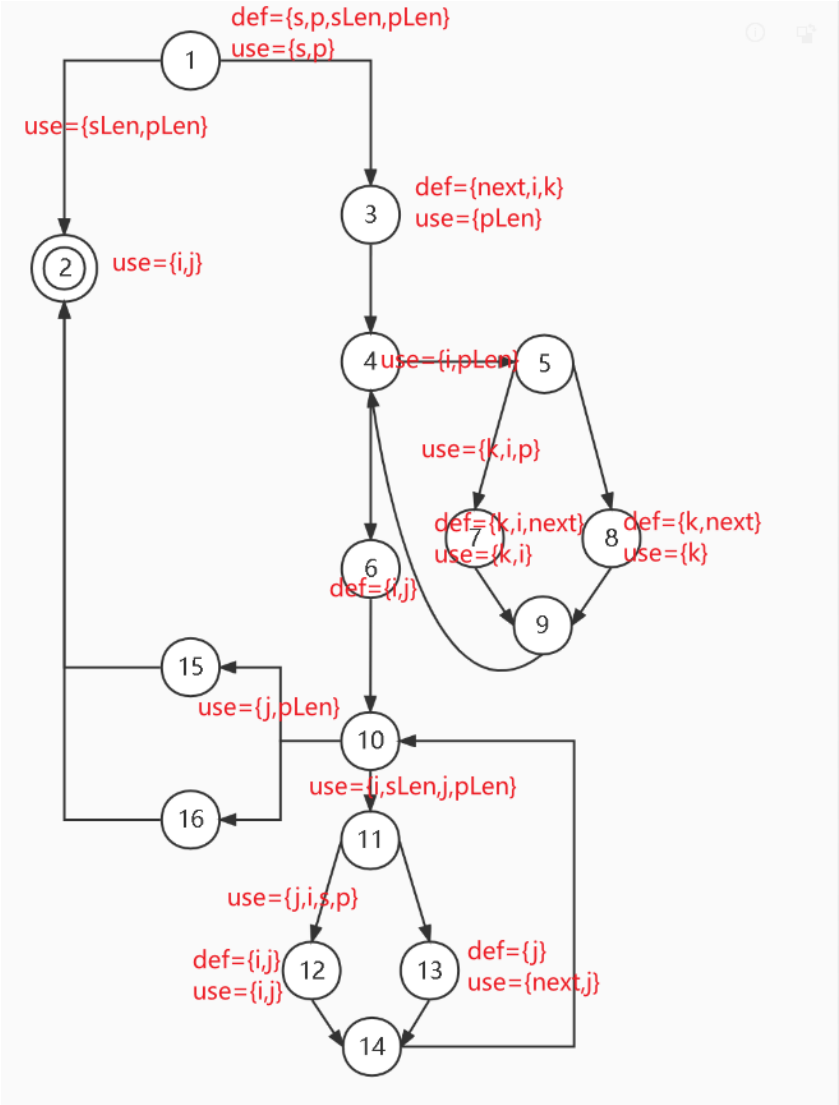
1) [1 2]
2) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 5 7 9 4 5 8 9 4 5 8 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 15 2]
3) [1 3 4 5 7 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 13 14 10 11 13 14 10 11 12 14 10 16 2]
4) [1 3 4 6 10 11 12 14 10 15 2]
5) …

由于代码逻辑性：下述路径均不可能出现
4->5->8
8->9->4->6
6->10->15
6->10->16
13->14->10->15
13->14->10->16

测试用例	S	P	预期输出
TC1	**“abc”**	**“abcd“**	**-1**
TC2	**“qabaca“**	**“abaca“**	**1**
TC3	**“abc”**	**“bbc”**	**-1**

TC4	“a”	“a”	0

5.2 基于数据流（考虑覆盖准则）



变量	def	use	du-pairs
s	1	1, (11,12),(11,13)	(1,1),(1,(11,12)) ,(1,(11,13))
p	1	1,(5,7) ,(5,8), (11,12) ,(11,13)	(1,1),(1,(5,7)),(1, (11,12)) ,(1,(5,8)),(1, (11,13))
sLen	1	(1,2), (10,11)	(1,(1,2)),(1,(10,11))
pLen	1	(1,2),3,(4,5),(4,6), (10,11) ,(10,15),(10,16)	(1,(1,2)) ,(1,3) ,(1,(4,5)) ,(1,(4,6)),(1,(10,11)) ,(1,(10,15)),(1,(10,16))
next	3,7,8	13	(3,7),(7,13),(8,13)

i	3,7,6,12	2,(4,5),(5,7),(4,6),(5,8),7,(10,11),(11,12),(11,13),12	(3,(4,5)),(3,(4,6)),(3,(5,7)),(3,(5,8)),(3,7),(7,(4,5)),(7,(5,7)),(7,(4,6)),(7,(5,8)),(7,7),(6,2),(6,(10,11)),(6,(11,12)),(6,(11,13)),(6,12),(12,2),(12,(10,11)),(12,(11,12)),(12,(11,13)),(12,12)
k	3,7,8	(5,7),(5,8),7,8	(3,(5,7)),(3,(5,8)),(3,7) ,(3,8) ,(7,(5,7)),(7,(5,8)),(7,7),(7,8),(8,(5,7)), (8,(5,8)),(8,7) ,(8,8)
j	6,12,13	2,(10,11),(11,12),(11,13),12,13,(10,15),(10,16)	(6,2) ,(6,(10,11)) ,(6,(11,12)),(6,(11,13)),(6,12) ,(6,13) ,(6,(10,15)),(6,(10,16)),(12,2) ,(12,(10,11)),(12,(11,12)),(12,(11,13)) ,(12,12) ,(12,13) ,(12,(10,15)),(12,(10,16)),(13,2) ,(13,(10,11)) ,(13,(11,12)), (13,(11,13)),(13,12) ,(13,13) ,(13,(10,15)),(13,(10,16))

5.2.1 全定义覆盖

TP

1) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 13 14 10 11 12 14 10 11 13 14 10 11 12 14 10 16 2]

测试用例	S	P	预期输出
TC1	**“bqq”**	**“aba“**	**-1**

5.2.2 全使用覆盖

TP

1) [1 2]
2) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 5 7 9 4 5 8 9 4 5 8 9 4 5 7 9 4 6 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 15 2]
3) [1 3 4 5 7 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 13 14 10 11 13 14 10 11 12 14 10 16 2]
4) [1 3 4 6 10 11 12 14 10 15 2]

由于代码逻辑性：下述路径均不可能出现
4->5->8
8->9->4->6
6->10->15
6->10->16
13->14->10->15
13->14->10->16

测试用例	S	P	预期输出
TC1	**“abc”**	**“abcd“**	**-1**
TC2	**“abaca“**	**“abaca“**	**0**
TC3	**“abc”**	**“bbc”**	**-1**
TC4	**“a”**	**“a”**	**0**

5.2.3 全定义-使用路径覆盖

TP

1) [1 2]
2) [1 3 4 5 7 9 4 5 8 9 4 5 7 9 4 5 7 9 4 5 8 9 4 5 8 9 4 5 7 9 4 6 10 11 12 14 10 11 12 14 10 11 12 14 10 11 12 14 10 15 2]
3) [1 3 4 5 7 9 4 5 7 9 4 6 10 11 13 14 10 11 12 14 10 11 12 14 10 11 13 14 10 11 13 14 10 11 12 14 10 16 2]
4) [1 3 4 6 10 11 12 14 10 15 2]

由于代码逻辑性：下述路径均不可能出现
4->5->8
8->9->4->6
6->10->15
6->10->16
13->14->10->15
13->14->10->16

测试用例	S	P	预期输出
TC1	**“abc”**	**“abcd“**	**-1**
TC2	**“abaca“**	**“abaca“**	**0**
TC3	**“abc”**	**“bbc”**	**-1**
TC4	**“a”**	**“a”**	**0**

6 基于 JUnit 的单元测试实践（黑盒测试）

6.1 TestByJunit.java

```
package org;
import org.junit.*;
import static org.junit.Assert.*;
public class TestByJunit {
    @BeforeClass
    public static void setUpBeforeClass() throws Exception{
        System.out.println("@BeforeClass\n");
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception{
        System.out.println("@AfterClass\n");
    }
    @Before
    public void setUp() throws Exception {
        System.out.println("测试开始");
    }

    @After
    public void tearDown() throws Exception {
        System.out.println("测试结束");
    }

    @Test
    public void kmp1() {
        String s="abc";
        String p="abcd";
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result = kmpSearch.kmp(s,p);
        System.out.println("result: "+result);
        assertEquals(-1,result);
    }

    @Test
    public void kmp2() {
        String s="abcde";
```

```
        String p="cde";
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result = kmpSearch.kmp(s,p);
        System.out.println("result: "+result);
        assertEquals(2,result);
    }

    @Test
    public void kmp3() {
        String s="abcde";
        String p="ace";
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result = kmpSearch.kmp(s,p);
        System.out.println("result: "+result);
        assertEquals(-1,result);
    }

    @Test
    public void kmp4() {
        String s="";
        String p="ace";
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result = kmpSearch.kmp(s,p);
        System.out.println("result: "+result);
        assertEquals(-1,result);
    }

    @Test
    public void kmp5() {
        String s="abc";
        String p="";
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result = kmpSearch.kmp(s,p);
        System.out.println("result: "+result);
        assertEquals(-1,result);
    }
}
```

6.2 运行截图

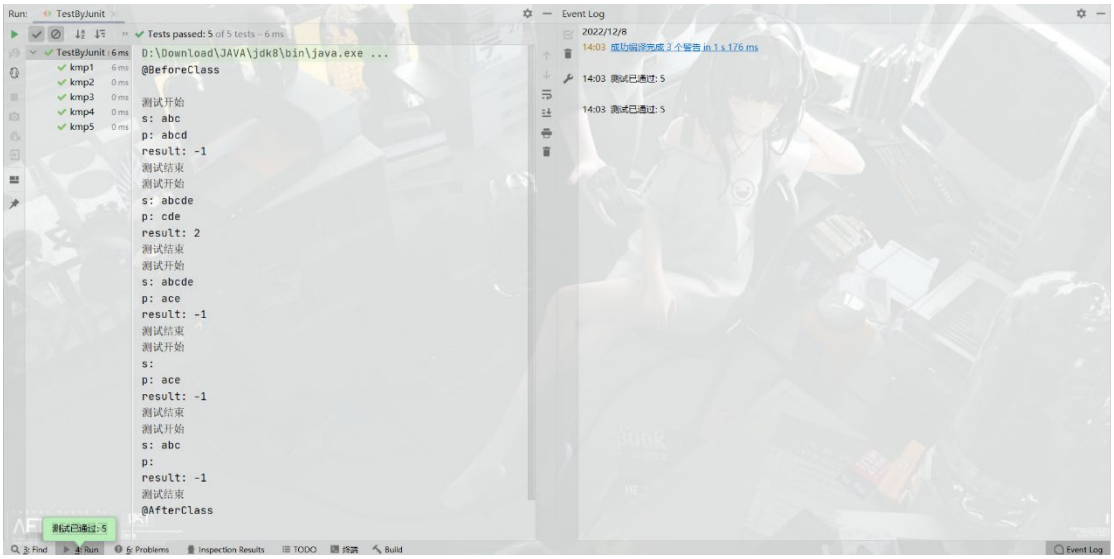


图 2Juint 运行截图

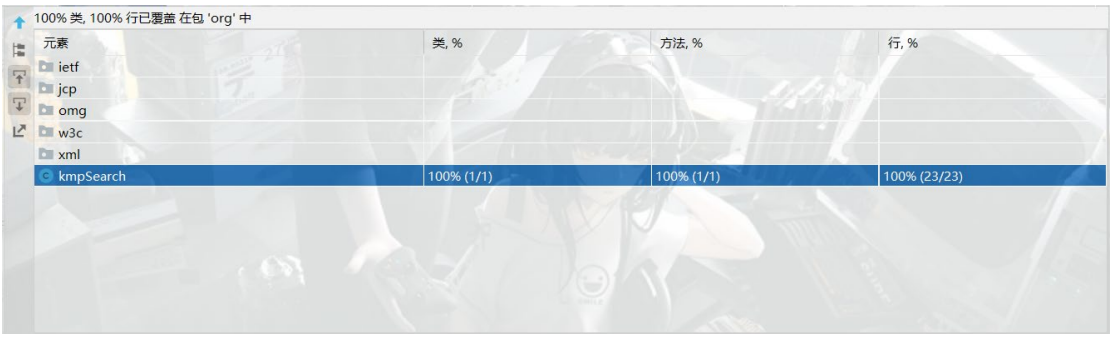


图 3 覆盖率

7 基于 TestNG 的单元测试实践（白盒测试）

7.1 java 文件

7.1.1 CsvUtil.java

```
package util;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class CsvUtil {
    public static Object[][] getTestData(String filename)throws
IOException{
        List<Object[]> records = new ArrayList<Object[]>();
        BufferedReader file = new BufferedReader(new
InputStreamReader(new FileInputStream(filename),"UTF-8"));
        String record;
        while((record = file.readLine())!=null){
            String[] fields =record.split(",");
            records.add(fields);
        }
        file.close();
        Object[][] results = new Object[records.size()][];
        for (int i = 0; i < records.size(); i++) {
            results[i] = records.get(i);
        }
        return results;
    }
}
```

7.1.2 TestByTestNG.java

```
package org;
import org.testng.annotations.DataProvider;
import util.CsvUtil;

import java.io.IOException;

import static org.testng.Assert.assertEquals;

public class TestByTestNG {

    private static final String filePathAllDef =
"src/main/resources/data_all_def.csv";
    private static final String filePathAllUse =
"src/main/resources/data_all_use.csv";
    private static final String filePathAllDu =
```

```
"src/main/resources/data_all_du.csv";

    private static final String filePathNC =
"src/main/resources/data_NC.csv";
    private static final String filePathEC =
"src/main/resources/data_EC.csv";
    private static final String filePathEPC =
"src/main/resources/data_EPC.csv";
    private static final String filePathPPC =
"src/main/resources/data_PPC.csv";
    private static final String filePathCPC =
"src/main/resources/data_CPC.csv";


    @org.testng.annotations.BeforeSuite(alwaysRun=true)
    public void setUpSuite() {
        System.out.print("BeforeSuite");
    }

    @org.testng.annotations.AfterSuite(alwaysRun=true)
    public void tearDownSuite() {
        System.out.print("AfterSuite");
    }

    @org.testng.annotations.BeforeGroups(groups =
{"group_all_def"})
    public void setUpGroups1() {
        System.out.print("开始测试全定义覆盖");
    }

    @org.testng.annotations.AfterGroups(groups =
{"group_all_def"})
    public void tearDownGroups1() {
        System.out.print("结束测试全定义覆盖");
    }
    @org.testng.annotations.BeforeGroups(groups =
{"group_all_use"})
    public void setUpGroups2() {
        System.out.print("开始测试全使用覆盖");
    }

    @org.testng.annotations.AfterGroups(groups =
{"group_all_use"})
    public void tearDownGroups2() {
```

```
        System.out.print("结束测试全使用覆盖");
    }

    @org.testng.annotations.BeforeGroups(groups =
{"group_all_du"})
    public void setUpGroups3() {
        System.out.print("开始测试全定义-使用覆盖");
    }

    @org.testng.annotations.AfterGroups(groups = {"group_all_du"})
    public void tearDownGroups3() {
        System.out.print("结束测试全定义-使用覆盖");
    }

    @org.testng.annotations.BeforeGroups(groups = {"group_NC"})
    public void setUpGroups4() {
        System.out.print("开始测试 Node Coverage");
    }

    @org.testng.annotations.AfterGroups(groups = {"group_NC"})
    public void tearDownGroups4() {
        System.out.print("结束测试 Node Coverage");
    }

    @org.testng.annotations.BeforeGroups(groups = {"group_EC"})
    public void setUpGroups5() {
        System.out.print("开始测试 Edge Coverage");
    }

    @org.testng.annotations.AfterGroups(groups = {"group_EC"})
    public void tearDownGroups5() {
        System.out.print("结束测试 Edge Coverage");
    }

    @org.testng.annotations.BeforeGroups(groups = {"group_EPC"})
    public void setUpGroups6() {
        System.out.print("开始测试 Edge Pair Coverage");
    }

    @org.testng.annotations.AfterGroups(groups = {"group_EPC"})
    public void tearDownGroups6() {
        System.out.print("结束测试 Edge Pair Coverage");
    }
```

```
    @org.testng.annotations.BeforeGroups(groups = {"group_PPC"})
    public void setUpGroups7() {
        System.out.print("开始测试 Prime Path Coverage");
    }

    @org.testng.annotations.AfterGroups(groups = {"group_PPC"})
    public void tearDownGroups7() {
        System.out.print("结束测试 Prime Path Coverage");
    }

    @org.testng.annotations.BeforeGroups(groups = {"group_CPC"})
    public void setUpGroups8() {
        System.out.print("开始测试 Complete Path Coverage");
    }

    @org.testng.annotations.AfterGroups(groups = {"group_CPC"})
    public void tearDownGroups8() {
        System.out.print("结束测试 Complete Path Coverage");
    }
//-------------------------------------------------------------------
---
    @org.testng.annotations.BeforeClass(alwaysRun=true)
    public void setUpClass() {
        System.out.print("BeforeClass");
    }
    @org.testng.annotations.AfterClass(alwaysRun=true)
    public void tearDownClass() {
        System.out.print("AfterClass");
    }
    @org.testng.annotations.BeforeTest(groups = {"DataFlow"})
    public void setUpTest1() {
        System.out.print("开始数据流测试");
    }
    @org.testng.annotations.AfterTest(groups = {"DataFlow"})
    public void tearDownTest1() {
        System.out.print("结束数据流测试");
    }

    @org.testng.annotations.BeforeTest(groups = {"ControlFlow"})
    public void setUpTest2() {
        System.out.print("开始控制流测试");
    }
    @org.testng.annotations.AfterTest(groups = {"ControlFlow"})
    public void tearDownTest2() {
```

```
        System.out.print("结束控制流测试");
    }

    @org.testng.annotations.BeforeMethod()
    public void setUp() {
        System.out.print("BeforeMethod");
    }

    @org.testng.annotations.AfterMethod()
    public void tearDown() {
        System.out.print("AfterMethod");
    }
//-----------------------------------------------------------------

    @DataProvider(name="DataAllDef")
    public static Object[][] getDataAllDef() throws IOException{
        return CsvUtil.getTestData(filePathAllDef);
    }

    @DataProvider(name="DataAllUse")
    public static Object[][] getDataAllUse() throws IOException{
        return CsvUtil.getTestData(filePathAllUse);
    }
    @DataProvider(name="DataAllDu")
    public static Object[][] getDataAllDu() throws IOException{
        return CsvUtil.getTestData(filePathAllDu);
    }
    @DataProvider(name = "DataNC")
    public static Object[][] getDataNC() throws IOException{
        return CsvUtil.getTestData(filePathNC);
    }
    @DataProvider(name = "DataEC")
    public static Object[][] getDataEC() throws IOException{
        return CsvUtil.getTestData(filePathEC);
    }
    @DataProvider(name = "DataEPC")
    public static Object[][] getDataEPC() throws IOException{
        return CsvUtil.getTestData(filePathEPC);
    }
    @DataProvider(name = "DataPPC")
    public static Object[][] getDataPPC() throws IOException{
        return CsvUtil.getTestData(filePathPPC);
    }
    @DataProvider(name = "DataCPC")
```

```
    public static Object[][] getDataCPC() throws IOException{
        return CsvUtil.getTestData(filePathCPC);
    }
//-------------------------------------------------------------
    @org.testng.annotations.Test(groups =
{"group_all_def"},dataProvider="DataAllDef")
    public void testKmpAllDef(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
    }

    @org.testng.annotations.Test(groups =
{"group_all_use"},dataProvider="DataAllUse")
    public void testKmpAllUse(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
    }
    @org.testng.annotations.Test(groups =
{"group_all_du"},dataProvider="DataAllDu")
    public void testKmpAllDu(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
    }
    @org.testng.annotations.Test(groups =
{"group_NC"},dataProvider="DataNC")
    public void testKmpNC(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
```

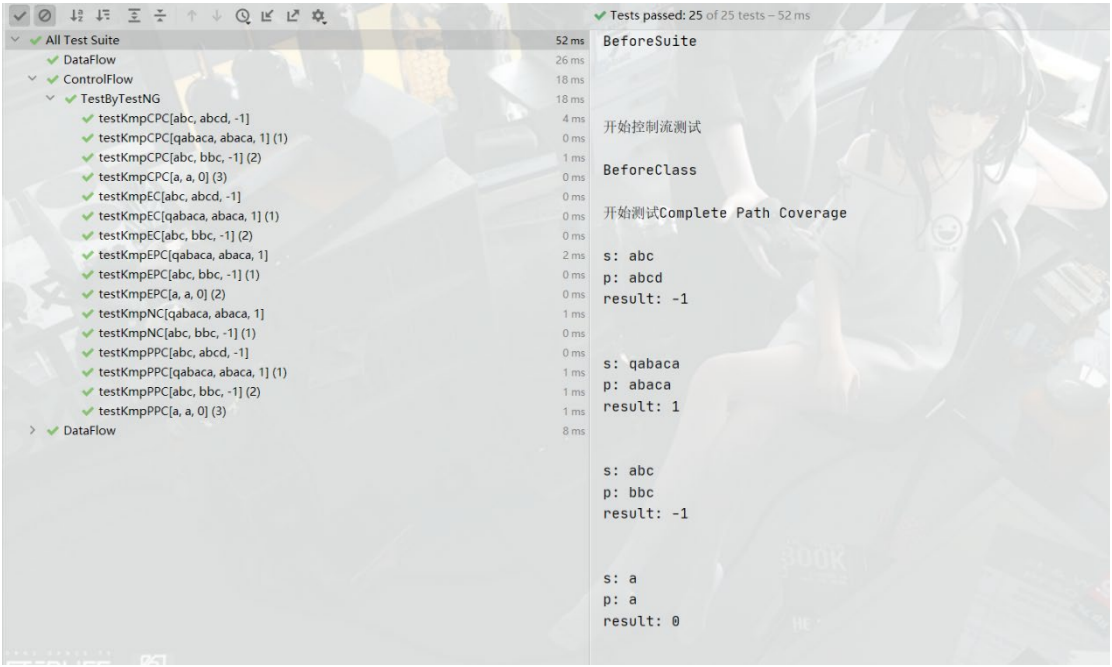
```
    }
    @org.testng.annotations.Test(groups =
{"group_EC"},dataProvider="DataEC")
    public void testKmpEC(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
    }
    @org.testng.annotations.Test(groups =
{"group_EPC"},dataProvider="DataEPC")
    public void testKmpEPC(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
    }
    @org.testng.annotations.Test(groups =
{"group_PPC"},dataProvider="DataPPC")
    public void testKmpPPC(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
    }
    @org.testng.annotations.Test(groups =
{"group_CPC"},dataProvider="DataCPC")
    public void testKmpCPC(String s,String p,String res) {
        int result = Integer.parseInt(res);
        System.out.println("s: "+s);
        System.out.println("p: "+p);
        int result_ = kmpSearch.kmp(s,p);
        System.out.println("result: "+result_);
        assertEquals(result,result_);
    }

}
```

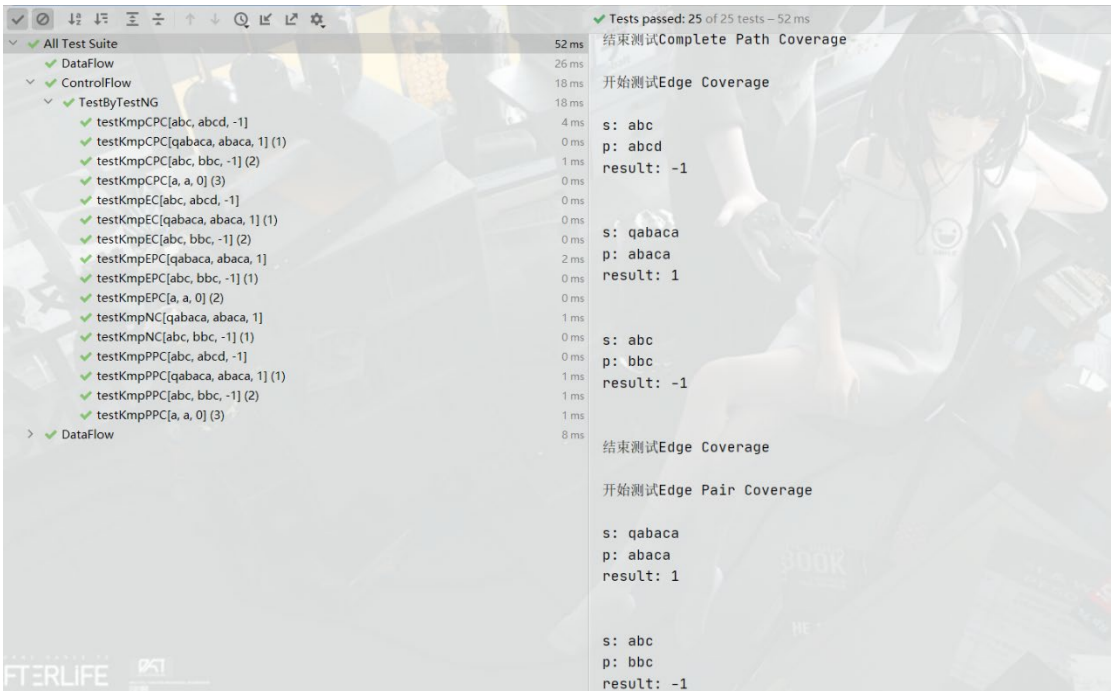
7.2 xml 文件

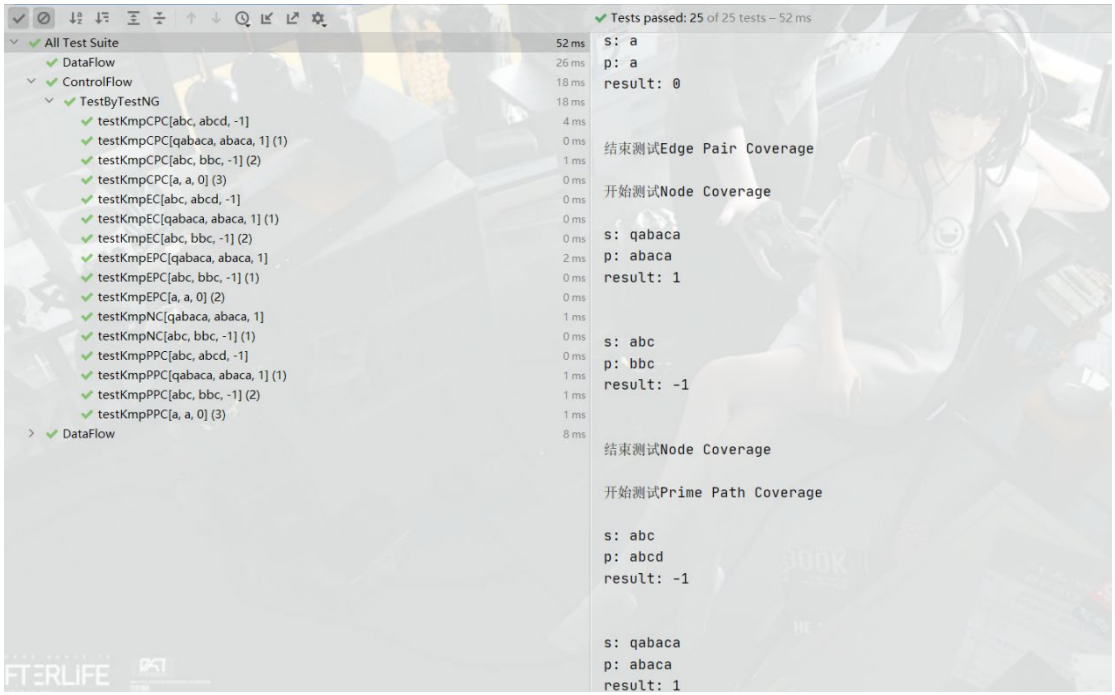
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="All Test Suite">
    <test verbose="2" preserve-order="true" name="ControlFlow">
        <groups>
            <run>
                <include name ="group_NC"/>
                <include name="group_EC"/>
                <include name="group_EPC"/>
                <include name="group_PPC"/>
                <include name="group_CPC"/>
                <include name="ControlFlow"/>
            </run>
        </groups>
        <classes>
            <class name="org.TestByTestNG">
                <methods><include name="testKmp"/>
                </methods>
            </class>
        </classes>
    </test>
    <test verbose="2" preserve-order="true" name="DataFlow">
        <groups>
            <run>
                <include name ="group_all_def"/>
                <include name="group_all_use"/>
                <include name="group_all_du"/>
                <include name="DataFlow"/>
            </run>
        </groups>
        <classes>
            <class name="org.TestByTestNG">
                <methods><include name="testKmp"/>
                </methods>
            </class>
        </classes>
    </test>
</suite>
```

7.3 运行截图

7.3.1 控制流测试

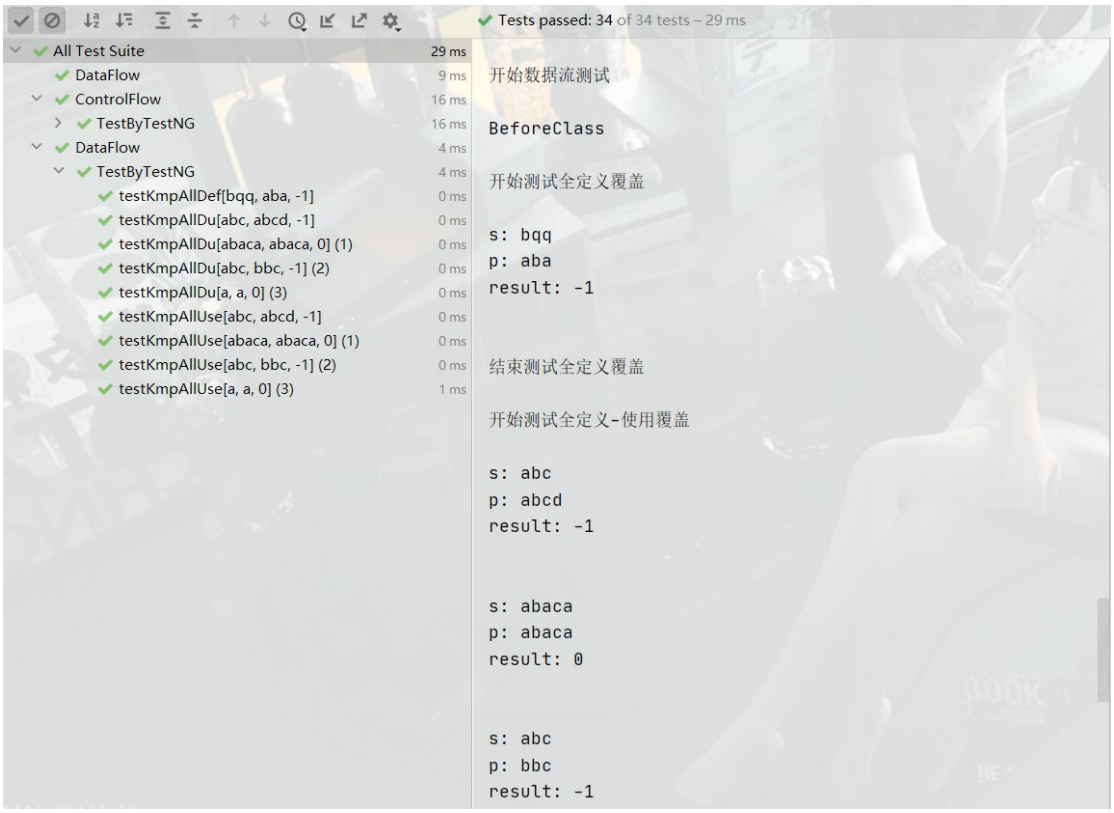


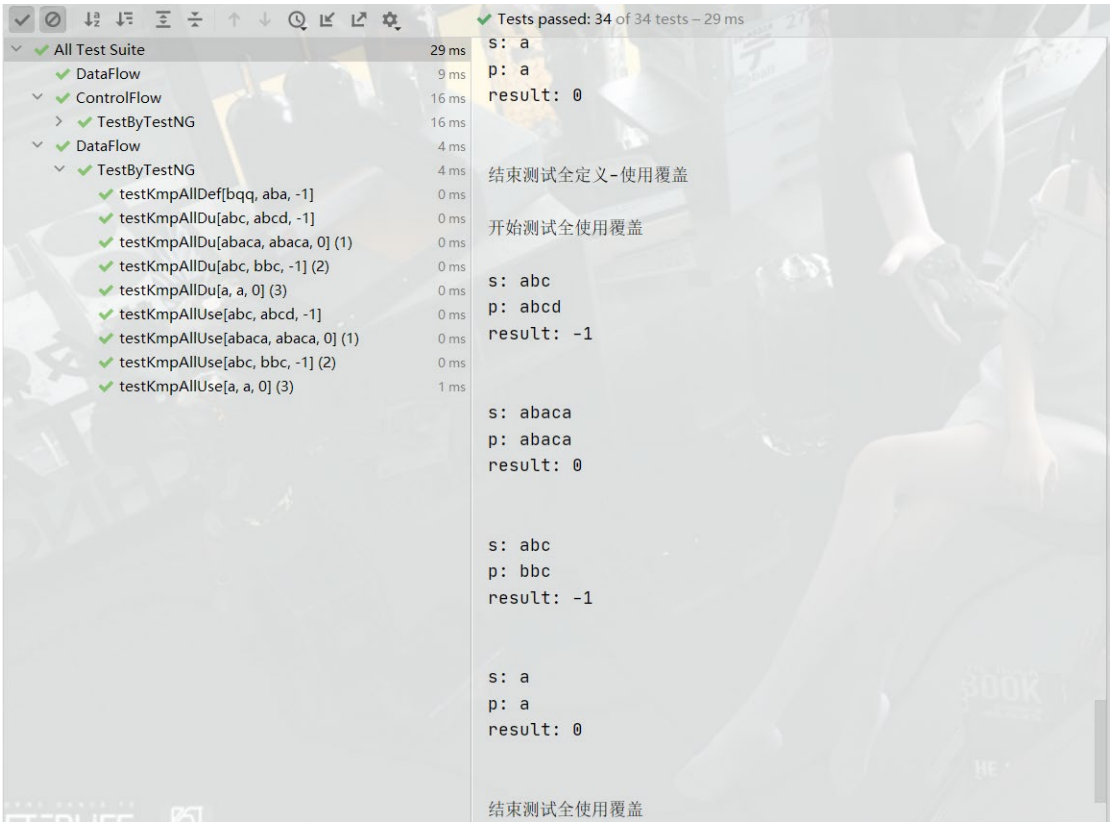


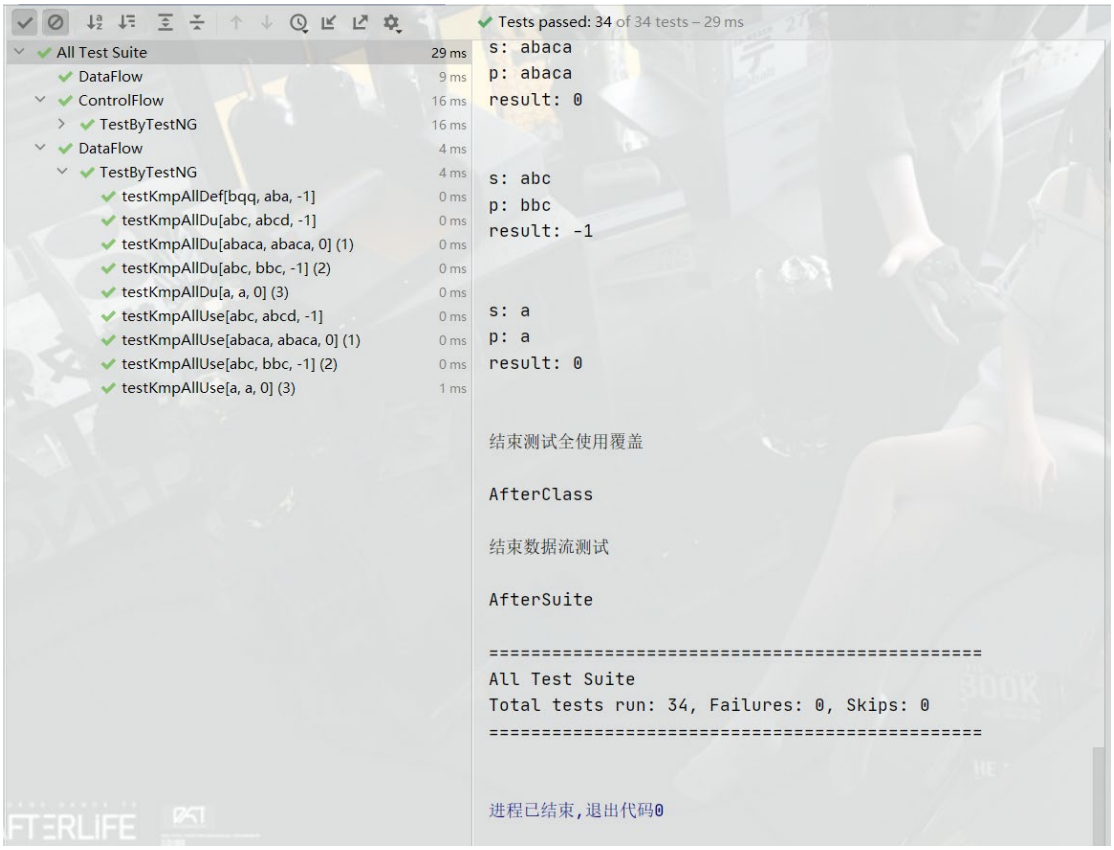

Tests passed: 25 of 25 tests – 52 ms
All Test Suite 52 ms
DataFlow 26 ms
ControlFlow 18 ms
TestByTestNG 18 ms
testKmpCPC[abc, abcd, -1] 4 ms
testKmpCPC[qabaca, abaca, 1] (1) 0 ms
testKmpCPC[abc, bbc, -1] (2) 1 ms
testKmpCPC[a, a, 0] (3) 0 ms
testKmpEC[abc, abcd, -1] 0 ms
testKmpEC[qabaca, abaca, 1] (1) 0 ms
testKmpEC[abc, bbc, -1] (2) 0 ms
testKmpEPC[qabaca, abaca, 1] 2 ms
testKmpEPC[abc, bbc, -1] (1) 0 ms
testKmpEPC[a, a, 0] (2) 0 ms
testKmpNC[qabaca, abaca, 1] 1 ms
testKmpNC[abc, bbc, -1] (1) 0 ms
testKmpPPC[abc, abcd, -1] 0 ms
testKmpPPC[qabaca, abaca, 1] (1) 1 ms
testKmpPPC[abc, bbc, -1] (2) 1 ms
testKmpPPC[a, a, 0] (3) 1 ms
DataFlow 8 ms
s: a
p: a
result: 0
结束测试Edge Pair Coverage
开始测试Node Coverage
s: qabaca
p: abaca
result: 1
s: abc
p: bbc
result: -1
结束测试Node Coverage
开始测试Prime Path Coverage
s: abc
p: abcd
result: -1
s: qabaca
p: abaca
result: 1


Tests passed: 25 of 25 tests – 52 ms
All Test Suite 52 ms
DataFlow 26 ms
ControlFlow 18 ms
TestByTestNG 18 ms
testKmpCPC[abc, abcd, -1] 4 ms
testKmpCPC[qabaca, abaca, 1] (1) 0 ms
testKmpCPC[abc, bbc, -1] (2) 1 ms
testKmpCPC[a, a, 0] (3) 0 ms
testKmpEC[abc, abcd, -1] 0 ms
testKmpEC[qabaca, abaca, 1] (1) 0 ms
testKmpEC[abc, bbc, -1] (2) 0 ms
testKmpEPC[qabaca, abaca, 1] 2 ms
testKmpEPC[abc, bbc, -1] (1) 0 ms
testKmpEPC[a, a, 0] (2) 0 ms
testKmpNC[qabaca, abaca, 1] 1 ms
testKmpNC[abc, bbc, -1] (1) 0 ms
testKmpPPC[abc, abcd, -1] 0 ms
testKmpPPC[qabaca, abaca, 1] (1) 1 ms
testKmpPPC[abc, bbc, -1] (2) 1 ms
testKmpPPC[a, a, 0] (3) 1 ms
DataFlow 8 ms
s: abc
p: bbc
result: -1
s: a
p: a
result: 0
结束测试Prime Path Coverage
AfterClass
结束控制流测试
开始数据流测试
BeforeClass
开始测试全定义覆盖
s: bqq
p: aba
result: -1
结束测试全定义覆盖

7.3.2 数据流测试


Tests passed: 34 of 34 tests – 29 ms
All Test Suite 29 ms
DataFlow 9 ms
ControlFlow 16 ms
TestByTestNG 16 ms
DataFlow 4 ms
TestByTestNG 4 ms
testKmpAllDef[bqq, aba, -1] 0 ms
testKmpAllDu[abc, abcd, -1] 0 ms
testKmpAllDu[abaca, abaca, 0] (1) 0 ms
testKmpAllDu[abc, bbc, -1] (2) 0 ms
testKmpAllDu[a, a, 0] (3) 0 ms
testKmpAllUse[abc, abcd, -1] 0 ms
testKmpAllUse[abaca, abaca, 0] (1) 0 ms
testKmpAllUse[abc, bbc, -1] (2) 0 ms
testKmpAllUse[a, a, 0] (3) 1 ms
开始数据流测试
BeforeClass
开始测试全定义覆盖
s: bqq
p: aba
result: -1
结束测试全定义覆盖
开始测试全定义-使用覆盖
s: abc
p: abcd
result: -1
s: abaca
p: abaca
result: 0
s: abc
p: bbc
result: -1


Tests passed: 34 of 34 tests – 29 ms
All Test Suite 29 ms
DataFlow 9 ms
ControlFlow 16 ms
TestByTestNG 16 ms
DataFlow 4 ms
TestByTestNG 4 ms
testKmpAllDef[bqq, aba, -1] 0 ms
testKmpAllDu[abc, abcd, -1] 0 ms
testKmpAllDu[abaca, abaca, 0] (1) 0 ms
testKmpAllDu[abc, bbc, -1] (2) 0 ms
testKmpAllDu[a, a, 0] (3) 0 ms
testKmpAllUse[abc, abcd, -1] 0 ms
testKmpAllUse[abaca, abaca, 0] (1) 0 ms
testKmpAllUse[abc, bbc, -1] (2) 0 ms
testKmpAllUse[a, a, 0] (3) 1 ms
s: a
p: a
result: 0
结束测试全定义-使用覆盖
开始测试全使用覆盖
s: abc
p: abcd
result: -1
s: abaca
p: abaca
result: 0
s: abc
p: bbc
result: -1
s: a
p: a
result: 0
结束测试全使用覆盖


Tests passed: 34 of 34 tests – 29 ms
All Test Suite 29 ms
DataFlow 9 ms
ControlFlow 16 ms
TestByTestNG 16 ms
DataFlow 4 ms
TestByTestNG 4 ms
testKmpAllDef[bqq, aba, -1] 0 ms
testKmpAllDu[abc, abcd, -1] 0 ms
testKmpAllDu[abaca, abaca, 0] (1) 0 ms
testKmpAllDu[abc, bbc, -1] (2) 0 ms
testKmpAllDu[a, a, 0] (3) 0 ms
testKmpAllUse[abc, abcd, -1] 0 ms
testKmpAllUse[abaca, abaca, 0] (1) 0 ms
testKmpAllUse[abc, bbc, -1] (2) 0 ms
testKmpAllUse[a, a, 0] (3) 1 ms
s: abaca
p: abaca
result: 0
s: abc
p: bbc
result: -1
s: a
p: a
result: 0
结束测试全使用覆盖
AfterClass
结束数据流测试
AfterSuite
==========
All Test Suite
Total tests run: 34, Failures: 0, Skips: 0
==========
进程已结束,退出代码0

7.3.3 总结

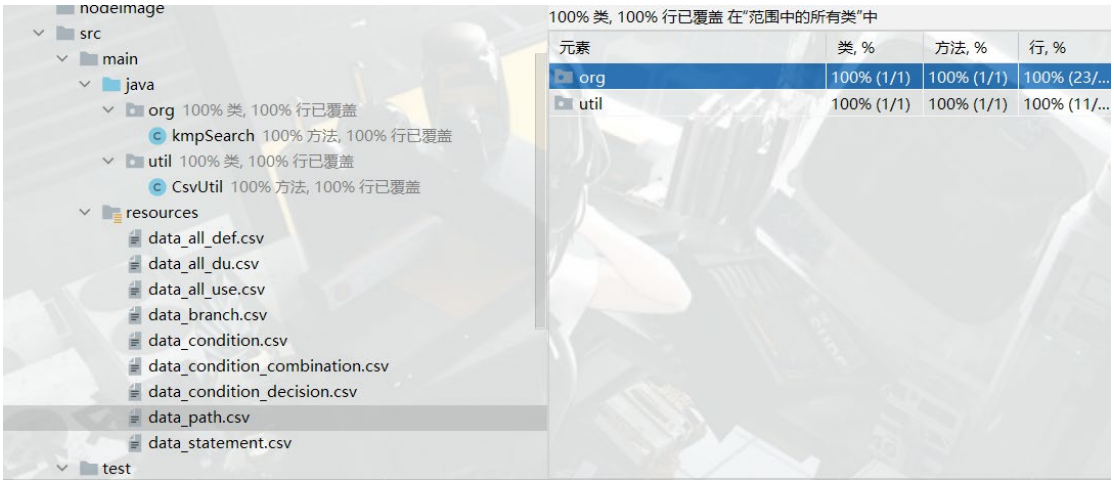

nodeImage
src
main
java
org 100% 类, 100% 行已覆盖
kmpSearch 100% 方法, 100% 行已覆盖
util 100% 类, 100% 行已覆盖
CsvUtil 100% 方法, 100% 行已覆盖
resources
data_all_def.csv
data_all_du.csv
data_all_use.csv
data_branch.csv
data_condition.csv
data_condition_combination.csv
data_condition_decision.csv
data_path.csv
data_statement.csv
test
100% 类, 100% 行已覆盖 在"范围中的所有类"中
元素
类, %
方法, %
行, %
org
100% (1/1)
100% (1/1)
100% (23/...
util
100% (1/1)
100% (1/1)
100% (11/...

图 4 覆盖率

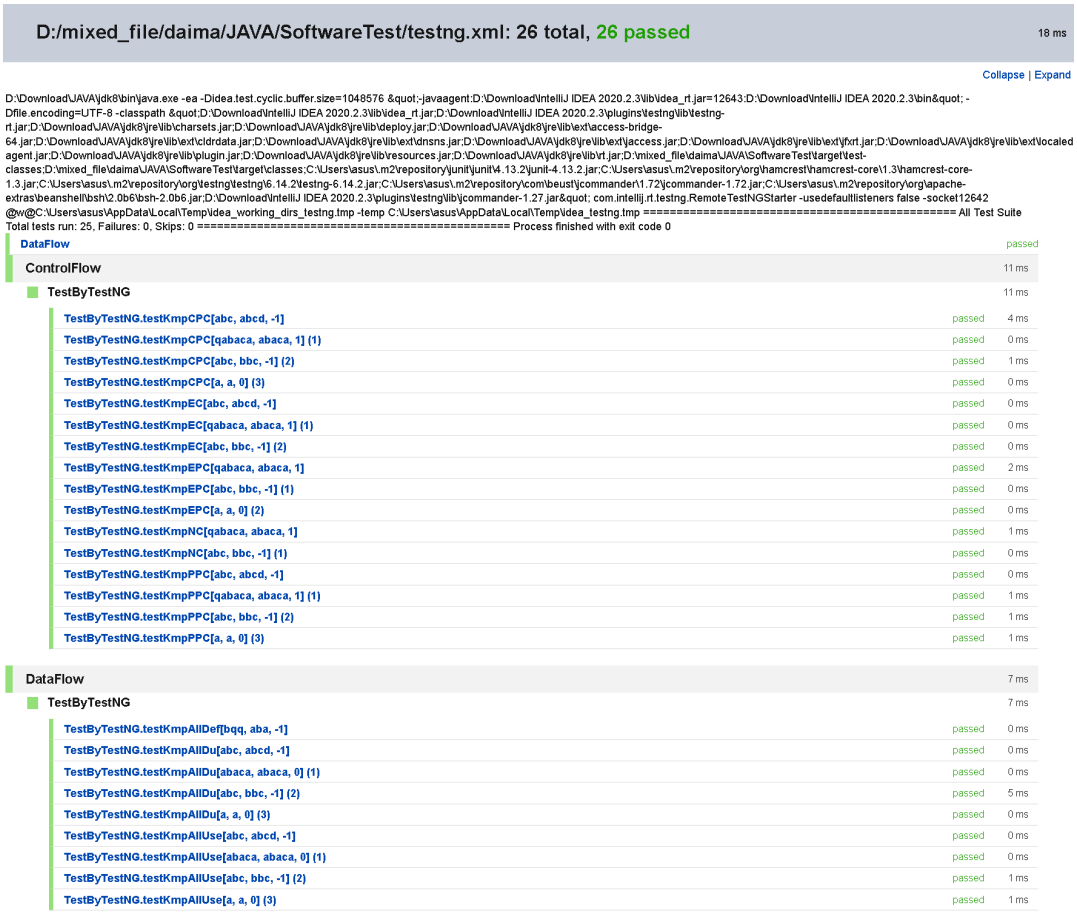


图 5 测试结果

8 测试体会

在此次上机测试之前，从没想过软件测试这么复杂，不比编写程序轻松。

从一开始的绘制控制流图，我就改了三四遍，这次最终报告的控制流图和第一次报告的控制流图相比有很大的改动，既把之前错误的地方修正了，又加上了条件判断。黑盒测试中，我选择了使用因果图和决策表进行测试，将建立的因果图转换成决策表，并生成测试用例，然后在 IDEA 上用 Juint 进行单元测试，由图 2 和图 3 可得，测试全部通过，且行覆盖、类覆盖、方法覆盖均为 100%。在白盒测试方面，因为需要针对程序的代码逻辑，还要考虑许多覆盖准则，总体来说比黑盒测试更加复杂，也是这次我上机测试花费时间最多的部分。控制流测试时，在进行 Edge-Pair 覆盖测试时，我发现边对由于程序代码的逻辑性根本不可能出现。在进行基本路径覆盖测试时，由于代码较为复杂，算 Prime Path 时算了好久，最后共找到了 39 条，但同样由于代码的逻辑性，部分 Prime Path 不具有可能性，因此在给出测试用例时将其排除。因为程序中存在循环，所以无法做到全路径覆盖。白盒测试部分，我使用了 TestNG 进行测试，TestNG 比 Juint 功能更多，可以进行数据驱动，所以在使用 TestNG 测试时，我选择用 csv 文件存储测试用例，方便进行

大量测试用例测试。由于 TestNG 还支持用 xml 文件运行，所以我编写了一个 xml 文件，对测试进行分组，这样极大地加大测试代码的可修改性和可读性，最后结果见 7.3。

事实上，我的测试代码并不是非常复杂，但此次测试却花费了我很多时间，可见软件测试确实是一门学问，需要深入研究。