软件测试技术实验

03 测试自动化

玄跻峰

武汉大学 计算机学院

jxuan@whu.edu.cn

课件网站

http://cstar.whu.edu.cn/course/testing-and-practice/

本节内容讲解 - 测试自动化

- **测试需求提取的自动化**

  被测程序分析、覆盖准则编码

- **测试效果评估的自动化**

  测试评估、质量评估

- **测试用例编写的自动化**

  自动测试生成工具

- **测试过程执行的自动化**

  持续测试

自动化的测试准则提取 作业
（原理与理论课的白盒测试对应）

形式

- Java的代码文件；类名为自己姓名的拼音（如XiYangyang.java）
- 1-2页文档，描述程序编译运行步骤（写明步骤即可，不需太复杂）
- 以邮件附件形式发送至xuan_whu@sina.com
- 邮件题目为 [TP]_学号_姓名，

例如 [TP]_2012345678_喜羊羊

- 截止时间 12月23日（星期二）23:59

要求

- 已知源代码文件和覆盖准则，完成一个程序，功能为被测方法的控制流图构造、对应的测试需求提取、测试路径获取。

回顾：Java 8中的ArrayList类的remove方法的覆盖

背景简介

Java 预定义了基于数组实现的列表ArrayList（即java.util.ArrayList ），详见 https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

任务

ArrayList 包含若干方法，要求针对下面两个方法

- remove(int index)
- remove(Object o)

覆盖准则：主路径覆盖

回顾：Java的ArrayList类的方法

例如，对于 remove (Object o), 其定义位于

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#remove-java.lang.Object-

源代码位于 JDK安装路径下 src.zip 或

http://cstar.whu.edu.cn/course/testing-and-practice/exp03/jdk8-src.zip

```
523     public boolean remove(Object o) {
524         if (o == null) {
525             for (int index = 0; index < size; index++)
526                 if (elementData[index] == null) {
527                     fastRemove(index);
528                     return true;
529                 }
530         } else {
531             for (int index = 0; index < size; index++)
532                 if (o.equals(elementData[index])) {
533                     fastRemove(index);
534                     return true;
535                 }
536         }
537         return false;
538     }
```

从输入到输出的全自动流程

输入

- 一个被测程序源代码
- 一个被测方法名（对于重载方法，可以分别输出）

输出

- 被测方法的控制流图的边（如右图结果表示从结点3到5，5到7，5到6的3条边），输出边的集合
- 覆盖准则要求下的测试需求（如主路径覆盖下，输出所有需要测试的主路径），输出测试需求集合
- 可以覆盖主路径的测试路径，输出测试路径集合

3, 5
5, 7
5, 6

一些提示

为了降低难度，可以假定被测程序不含 switch-case，try-catch, throw, break, continue等控制结点。也就是说，
假定被测方法只含 if, while, for 引发的控制结点（含循环）、return 引发的返回结点、及其它语句。不需要考虑过多的条件和循环嵌套，如4层以上的嵌套。

获得控制流图时，只考虑被测方法内的代码，不用考虑被测方法外的代码。
不用考虑未知源代码的依赖库，如依赖的class文件、jar文件、native方法等。

可假设被测方法的复杂度与ArrayList.remove(Object o)大体相当。

另一些提示

对于一个被测Java类，下面程序已完成了遍历已知结点的代码（不是必须使用的）。

http://cstar.whu.edu.cn/course/testing-and-practice/exp03/cfgparser.jar

如运行

java -jar cfgparser.jar <路径>/ArrayList.java remove:readObject

将输出ArrayList.java文件内名为remove或名为readObject的方法的结点遍历结果。

上面java命令后面的最后可以加上重定向，如 > 1.txt，将屏幕输出定向到1.txt文件。

一些提示：遍历输出结点的说明

两个重载方法

parent = -1表示直接位于方法体

被测方法[起始行]	结点号	父结点号	嵌套深度	起始行	起始列	终止行	终止列	类型@代码首行
method[row]	node	parent	height	start.x	start.y	end.x	end.y	content
remove[486]	4	-1	0	496	8	496	26	first-statement@rangeCheck(index);
remove[486]	5	-1	0	505	8	505	35	after-branch@elementData[--size]=null;
remove[486]	6	-1	0	502	8	504	39	if-statement@if (numMoved > 0) System.a
remove[486]	7	6	1	503	12	504	39	then-body@System.arraycopy(elementData,
remove[486]	8	-1	0	507	8	507	24	return@return oldValue;
remove[510]	9	-1	0	537	8	537	21	after-branch@return false;
remove[510]	10	-1	0	524	8	536	9	if-statement@if (o == null) {
remove[510]	11	10	1	525	12	529	17	then-body@for (int index=0; index < siz
remove[510]	12	10	1	531	12	535	17	else-body@for (int index=0; index < siz
remove[510]	11	10	1	525	12	529	17	for-statement@for (int index=0; index <
remove[510]	13	11	2	525	32	525	44	for-condition@index < size
remove[510]	14	11	2	526	16	529	17	for-body@if (elementData[index] == null
remove[510]	15	11	2	525	46	525	53	for-update@index++
remove[510]	14	11	2	526	16	529	17	if-statement@if (elementData[index] ==
remove[510]	16	14	3	527	20	527	38	then-body@fastRemove(index);
remove[510]	17	14	3	528	20	528	32	return@return true;
remove[510]	12	10	1	531	12	535	17	for-statement@for (int index=0; index <
remove[510]	18	12	2	531	32	531	44	for-condition@index < size
remove[510]	19	12	2	532	16	535	17	for-body@if (o.equals(elementData[index
remove[510]	20	12	2	531	46	531	53	for-update@index++
remove[510]	19	12	2	532	16	535	17	if-statement@if (o.equals(elementData[i
remove[510]	21	19	3	533	20	533	38	then-body@fastRemove(index);
remove[510]	22	19	3	534	20	534	32	return@return true;
remove[510]	9	-1	0	537	8	537	21	return@return false;

评分标准

- 任何形式的抄袭都是0分

- 程序代码
  - 可编译、可执行
  - 代码风格
  - 代码质量
  - 按要求输出控制流图的边结构
  - 按要求输出给定覆盖准则的测试需求
  - 按要求输出给定覆盖准则的测试路径
  - 文档只要描述清楚编译和运行方式即可，文档占分数较少

测试需求提取

背景简介

理解、设计、实现测试需求提取。

任务

编写一个类文件（若需要，可包含其它文件），继承自给定的基类 extractbot.tool.BaseExtractor 。

重写基类的下面两个方法，

- public int[ ][ ] getControlFlowGraphInArray(String pathFile, String methodName)，其功能是，对于给定文件pathFile中名为methodName的方法，输出其控制流程图。
- public int[ ][ ] getTestRequirementsInArray(String pathFile, String methodName)，其功能是，对于给定文件pathFile中名为methodName的方法，输出其测试需求（如主路径）。
- public int[ ][ ] getTestPathsInArray(String pathFile, String methodName)，其功能是，对于给定文件pathFile中名为methodName的方法，输出其测试路径。

已知文件和背景知识

学生用于编程的文件

http://cstar.whu.edu.cn/course/testing-and-practice/exp03/extractbottest.zip

需要通过classpath引入的jar文件，即BaseExtractor的来源

http://cstar.whu.edu.cn/course/testing-and-practice/exp03/extractbot-base.jar

Junit运行库

http://cstar.whu.edu.cn/course/testing-and-practice/exp03/junit-4.13.jar

http://cstar.whu.edu.cn/course/testing-and-practice/exp03/hamcrest-core-1.3.jar

Example.java

```java
package extractbot;

public class Example {

    public static void example1(int[] numbers)
    {
        int length = numbers.length;
        double med, var, sd, mean, sum, varsum;

        sum = 0;
        for(int i = 0; i < length; i++)
        {
            sum += numbers[i];
        }
        med = numbers[length / 2];
        mean = sum / (double) length;

        varsum = 0;
        for(int i = 0; i < length; i++)
        {
            varsum = varsum + ((numbers[i] - mean) * (numbers
        }
        var = varsum / (length - 1.0);
        sd = Math.sqrt(var);

        System.out.println("length:             " + length);
        System.out.println("mean:               " + mean);
        System.out.println("median:             " + med);
        System.out.println("variance:           " + var);
        System.out.println("standard deviation: " + sd);
    }
}
```

ExampleTests.java

```
10
11  public class ExampleTests {
12
13      ////////////
14      /** TODO
15       *  Jifeng: Change the path of "pathFile" to your l
16       */
17      private String pathFile = "C:\\eclipse-workspace\\M
18      ////////////
19
20      private String methodName1 = "example1";
21      private String methodName2 = "example2";
22
23      /** TODO
24       *  Jifeng: Write down the source code of your own
25       */
26      private MyExtractor myExtractor = new MyExtractor()
27
28      /* A matrix (a 2D-array) of a control flow graph. E
29      private int[][] matrixCfg = new int[][] {
30          {0, 3}, {3, 4}, {4, 5}, {5, 4}, {4, 1}, {1, 8},
31      };
32
33      /* A matrix (a 2D-array) of prime paths. Each row i
34      private int[][] matrixPrimePath = new int[][] {
35          {4, 5, 4},
36          {5, 4, 5},
37          {8, 9, 8},
38          {9, 8, 9},
39          {9, 8, 2},
40          {0, 3, 4, 5},
41          {5, 4, 1, 8, 2},
42          {5, 4, 1, 8, 9},
43          {0, 3, 4, 1, 8, 2},
44          {0, 3, 4, 1, 8, 9}
45      };
46
47      /**
48       *  Jifeng: You do not need to update the following
49       *  This test method is used to check whether the c
50       */
51      @Test
52      public void testControlFlowGraph()
53      {
54          int[][] source = myExtractor.getControlFlowGrap
55          assertTrue(TestUtils.checkControlFlowGraph(sour
56      }
```

MyExtractor.java

```java
package extractbot;


import extractbot.tool.BaseExtractor;

public class MyExtractor extends BaseExtractor{

    @Override
    public int[][] getControlFlowGraphInArray(String pathFile
    {
        /** TODO
         *  Jifeng: Write down your source code here.
         */

        return null;
    }



    @Override
    public int[][] getTestRequirementsInArray(String pathFile
    {
        /** TODO
         *  Jifeng: Write down your source code here.
         */

        return null;
    }


    @Override
    public int[][] getTestPathsInArray(String pathFile, Strin
    {
        /** TODO
         *  Jifeng: Write down your source code here.
         */

        return null;
    }

}
```

下面是一个完整的运行例子

1. 输入被测代码Example.java，输出语法树结点列表；
2. 通过分析语法树结构，生成控制流程图；
3. 通过分析控制流程图，提取测试需求（主路径列表）；
4. 通过分析测试需求，提取测试路径。

其中，1可以按给定工具生成，2-4提供了测试用例。

Example.java 中 example1()

```
1   package extractbot;
2
3   public class Example {
4
5   public static void example1(int[] numbers)
6   {
7         int length = numbers.length;
8        double med, var, sd, mean, sum, varsum;
9
10       sum = 0;
11        for(int i = 0; i < length; i++)
12        {
13              sum += numbers[i];
14        }
15        med = numbers[length / 2];
16        mean = sum / (double) length;
17
18        varsum = 0;
19        for(int i = 0; i < length; i++)
20        {
21              varsum = varsum + ((numbers[i] - mean) * (numbers[i] - mean));
22        }
23        var = varsum / (length - 1.0);
24        sd = Math.sqrt(var);
25
26        System.out.println("length:" + length);
27        System.out.println("mean:" + mean);
28        System.out.println("median:" + med);
29        System.out.println("variance: " + var);
30        System.out.println("standard deviation:" + sd);
31        }
32  }
```

1.输入被测代码Example.java，输出语法树结点列表

对于一个被测Example.java类，下面程序已完成了遍历已知结点的代码（不是必须使用的）。

http://cstar.whu.edu.cn/course/testing-and-practice/exp03/cfgparser.jar

如运行

java -jar cfgparser.jar <路径>/Example.java example1

将输出Example.java文件内名为example1的方法的结点遍历结果。

上面java命令后面的最后可以加上重定向，如 > 1.txt，将屏幕输出定向到1.txt文件。

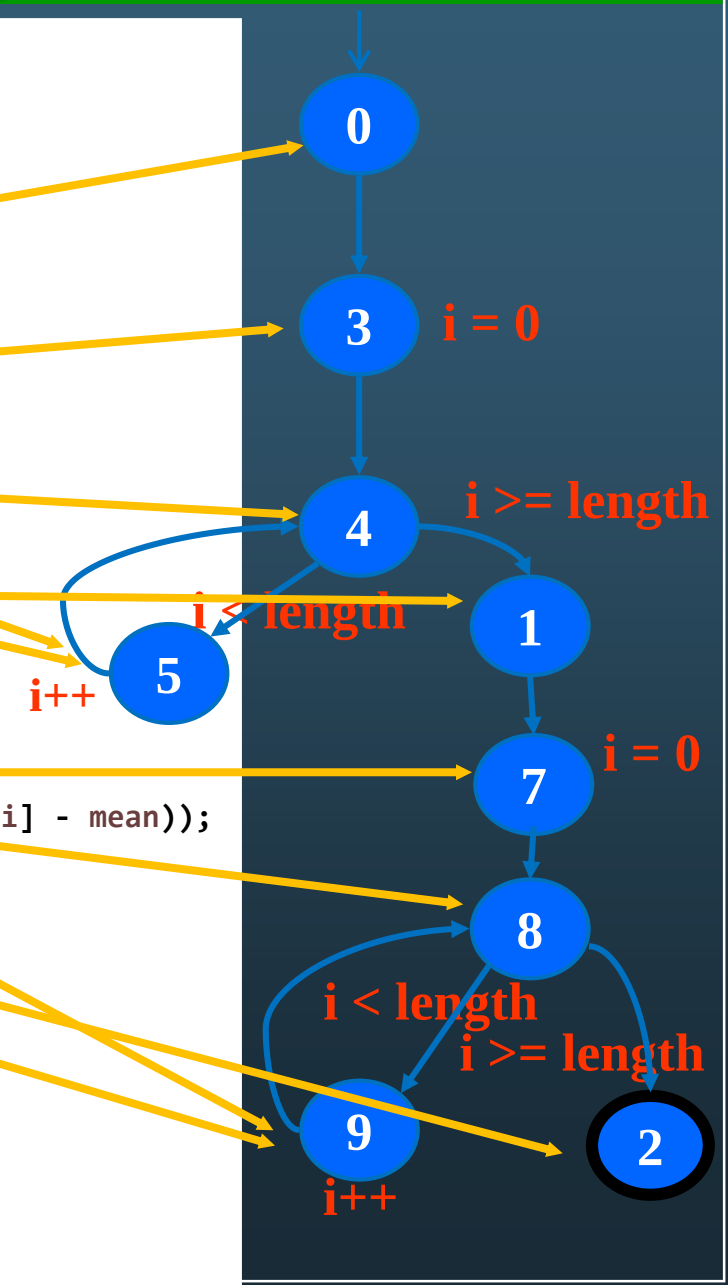
1.输入被测代码Example.java，输出语法树结点列表

通过cfgparser.jar输出的语法树结点列表。

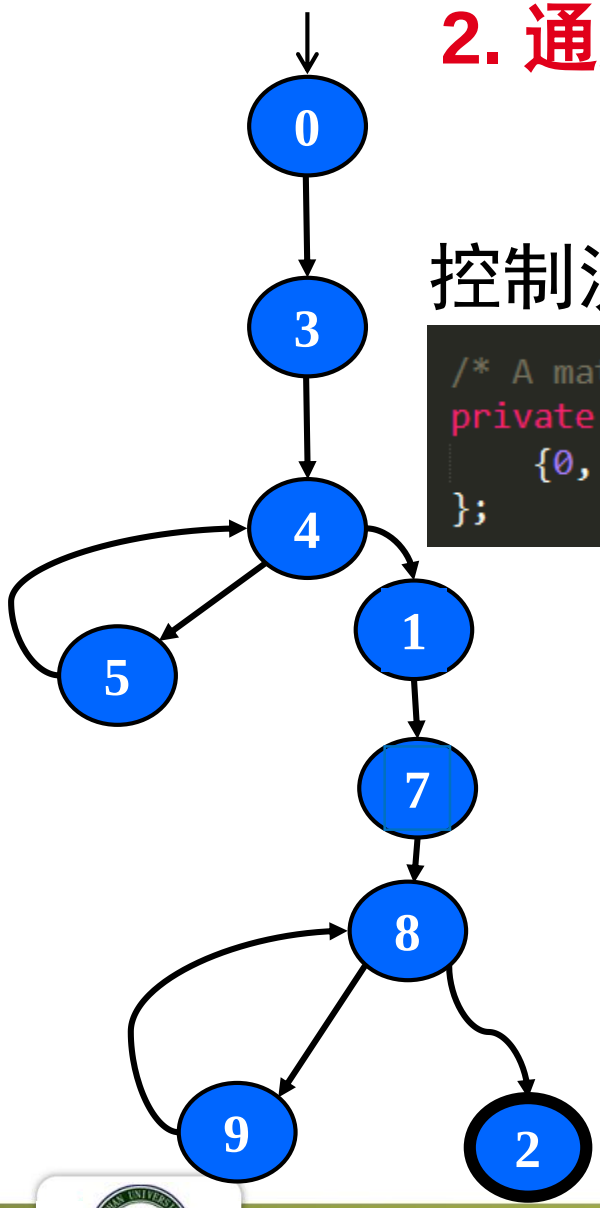
	A	B	C	D	E	F	G	H	I	J	K
1	method[row]	node	parent	height	start.x	start.y	end.x	end.y	content		
2	example1[5]	0	-1	0	7	8	7	36	first-statement@int length=nur		
3	example1[5]	1	-1	0	15	8	15	34	after-branch@med=numbers[l		
4	example1[5]	2	-1	0	23	8	23	38	after-branch@var=varsum / (le		
5	example1[5]	3	-1	0	11	8	14	9	for-statement@for (int i=0; i <		
6	example1[5]	4	3	1	11	23	11	33	for-condition@i < length		
7	example1[5]	5	3	1	13	12	13	30	for-body@sum+=numbers[i];		
8	example1[5]	6	3	1	11	35	11	38	for-update@i++		
9	example1[5]	7	-1	0	19	8	22	9	for-statement@for (int i=0; i <		
10	example1[5]	8	7	1	19	23	19	33	for-condition@i < length		
11	example1[5]	9	7	1	21	12	21	74	for-body@varsum=varsum + (		
12	example1[5]	10	7	1	19	35	19	38	for-update@i++		
13	example1[5]	11	-1	0	31	5	31	5	pseudo-return@pseudo-return		
14											

控制流程图

```
1   package extractbot;
2
3   public class Example {
4
5   public static void example1(int[] numbers)
6   {
7         int length = numbers.length;
8        double med, var, sd, mean, sum, varsum;
9
10       sum = 0;
11        for(int i = 0; i < length; i++)
12        {
13              sum += numbers[i];
14        }
15        med = numbers[length / 2];
16        mean = sum / (double) length;
17
18        varsum = 0;
19        for(int i = 0; i < length; i++)
20        {
21              varsum = varsum + ((numbers[i] - mean) * (numbers[i] - mean));
22        }
23        var = varsum / (length - 1.0);
24        sd = Math.sqrt(var);
25
26        System.out.println("Length:" + length);
27        System.out.println("mean:" + mean);
28        System.out.println("median:" + med);
29        System.out.println("variance: " + var);
30        System.out.println("standard deviation:" + sd);
31        }
32  }
```



2. 通过分析语法树结构，生成控制流程图



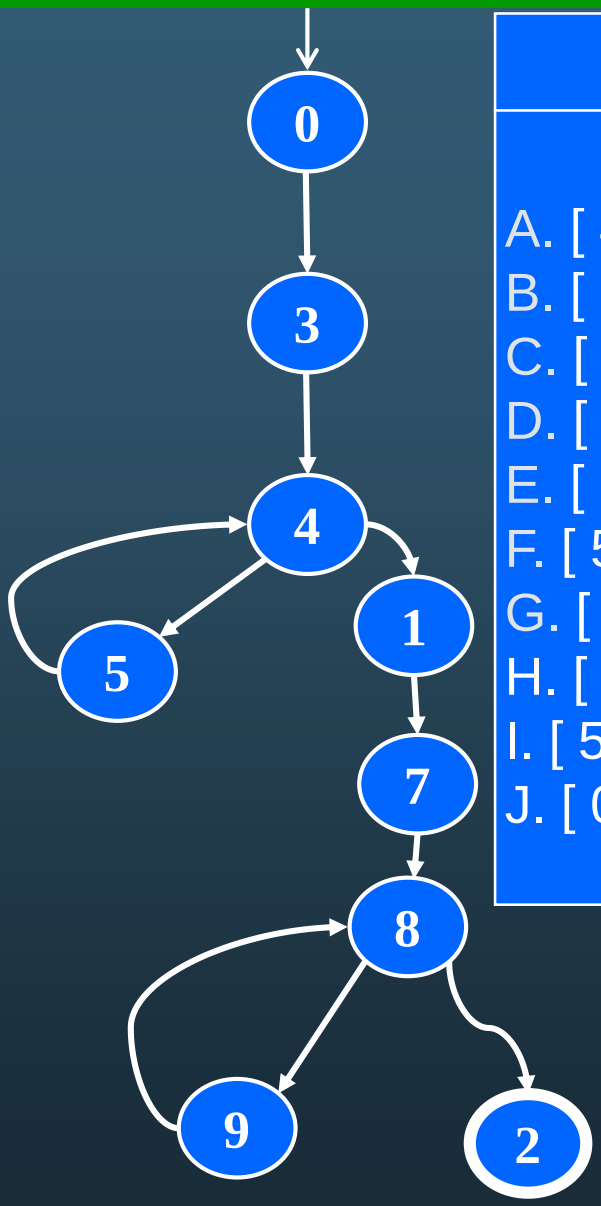
控制流程图的格式

```
/* A matrix (a 2D-array) of a control flow graph. Each row is an edge;
private int[][] matrixCfg = new int[][] {
    {0, 3}, {3, 4}, {4, 5}, {5, 4}, {4, 1}, {1, 7}, {7, 8}, {8, 9}, {9,
};
```

对应的测试用例

```
/**
 *  Jifeng: You do not need to update the following-up
 *  This test method is used to check whether the contr
 */
@Test
public void testControlFlowGraph()
{
    int[][] source = myExtractor.getControlFlowGraphInA
    assertTrue(TestUtils.checkControlFlowGraph(source,
}
```

主路径覆盖



主路径覆盖

测试需求

A. [4, 5, 4] *
B. [5, 4, 5] *
C. [8, 9, 8] *
D. [9, 8, 9] *
E. [9, 8, 2] !
F. [5, 4, 1, 7, 8, 2] !
G. [0, 3, 4, 1, 7, 8, 2] !
H. [0, 3, 4, 5]
I. [5, 4, 1, 7, 8, 9]
J. [0, 3, 4, 1, 7, 8, 9]

测试路径

i. [0,3,4,1,7,8,2] G
ii. [0,3,4,1,7,8,9,8,2] J
iii. [0,3,4,5,4,1,7,8,2] F,A,H
iiii. [0,3,4,5,4,5,4,1,7,8,9,8,9,8,2] E,D,C,B,I

在这个例子中，
测试需求是唯一的，
但测试路径不是唯一的。

3. 通过分析控制流程图，提取测试需求（主路径列表）

提取的主路径列表格式

```
/* A matrix (a 2D-array) of prime paths. Each row is one prime path; each node
private int[][] matrixPrimePath = new int[][] {
    {4, 5, 4},
    {5, 4, 5},
    {8, 9, 8},
    {9, 8, 9},
    {9, 8, 2},
    {5, 4, 1, 7, 8, 2},
```

对应的测试用例

```
/**
 *  Jifeng: You do not need to update the following-up code.
 *  This test method is used to check whether the prime paths are correctly extr
 */
@Test
public void testPrimePaths()
{
    int[][] source = myExtractor.getTestRequirementsInArray(pathFile, methodName
    assertTrue(TestUtils.checkTestRequirements(source, matrixPrimePath));
}
```

4. 通过分析测试需求，提取测试路径

提取的测试路径格式

```
private int[][] matrixTestPath = new int[][] {
        {0,3,4,1,8,2},
        {0,3,4,1,8,9,8,2}
```

对应的测试用例

```
/**
 *  Jifeng: You do not need to update the following-up code.
 *  This test method is used to check whether the test paths are correctly extracted.
 */
@Test
public void testTestPaths()
{
    int nodeStart = 0;
    int nodeEnd = 2;
    int[][] source = myExtractor.getTestPathsInArray(pathFile, methodName1);

    assertEquals(TestUtils.checkTestPaths(source, matrixPrimePath, matrixCfg, nodeSta
}
```

编码任务的简单概括：

重写MyExtractor中的
三个父类方法，
并通过测试用例。

```java
package extractbot;


import extractbot.tool.BaseExtractor;

public class MyExtractor extends BaseExtractor{

    @Override
    public int[][] getControlFlowGraphInArray(String pathFile
    {
        /** TODO
         *  Jifeng: Write down your source code here.
         */

        return null;
    }



    @Override
    public int[][] getTestRequirementsInArray(String pathFile
    {
        /** TODO
         *  Jifeng: Write down your source code here.
         */

        return null;
    }


    @Override
    public int[][] getTestPathsInArray(String pathFile, Strin
    {
        /** TODO
         *  Jifeng: Write down your source code here.
         */

        return null;
    }

}
```