東南大學

课程实验报告

课 程 软件测试与质量保证

计算机科学与工程 院（系） 软件工程 专业

学 号 171694

姓 名 吴江恒

指导教师 戚晓芳

设计地点 计算机楼

实验一 黑盒测试

一、实验要求

编写程序实现命令“grep”的功能，利用类别划分方法(category-partition method)设计黑盒测试用例，测试并完善程序。

命令 grep 在文件中搜索一个 pattern，打印文件中包含该 pattern 的行，且单行只打印一次。以下是对程序提出的功能要求：

1) pattern 可以是任意的字符序列；
2) 若 pattern 中包含空格，则 pattern 必须用单引号括起来；
3) 在 pattern 中包含单引号，则必须进行转义；
4) 文件中某行出现多次 pattern，该行只被打印一次。

二、测试方法

如实验要求所言，实验一采用 Category-Partition Method 设计测试用例。Category-Partition Method 是一种生成功能测试用例的方法，测试人员分析系统的规格说明书，编写一系列的测试规格说明书，并利用生成工具生成对测试的描述，然后根据上述文档编写具体测试用例。

这种方法的优点是，在必要时测试人员可以较为容易地修改测试规格说明书，并且，在说明书的限制下，能够控制测试的复杂度及测试用例的数量。

三、实验

本部分包含了实验环境，测试用例设计和测试过程。

1. 实验环境

本次实验代码均使用 python3.6 编写，编译环境为 ubuntu 16.04 LTS 系统的命令行以及 pycharm-community-2018.1.1.

2. 测试用例设计

根据第二部分所言，Category-Partition Method 的一个重要过程是编写测试说明书。正如在课堂上提到的，如果不对测试条件加以限制，那么将会组合产生近 2000 种测试用例。一份好的说明书可以在覆盖所有功能的同时，控制测试用例的数量和复杂度。本报告借鉴了 Thomas 等人(1988)设计的规格说明书(图 1)，设计了 40 种测试用例，如表 1 所示。

表 1 中的第 2~8 列为规格说明书中的 7 个变量，表中的第 2 行，序号值代表了每个标量不同的取值，序号的顺序与规格说明书中的顺序相同。如 Pattern size 的“1”表示“empty”，“2”表示“single character”，以此类推。

表 1 的第 9~10 列为设计的测试用例，第 11 列为测试用例在程序上的运行结果。

表 1 测试用例设计

	Pattern Size				Quoting			Embeded Blanks			Embeded Quotes			Num of Pattern in File			No of Pat in Line		File Name			*Pattern*	*File*	*Result*
No	1	2	3	4	1	2	3	1	2	3	1	2	3	1	2	3	1	2	1	2	3			
1	√																					Pattern为空		Empty
2				√																		pattern长度大于所有行		Error
3																				√		文件不存在		Error
4																					√	文件名缺失		Error
5																√						文件中不包含pattern		Not match
6													√									a\'\'aaa	pattern出现	Match
7																		√				文件一行出现多次pattern		Match
8							√															错误引号		Error
9		√				√		√			√			√			√		√			a	一个a	Match
10		√				√		√			√				√		√		√			a	多个a	Match
11		√				√		√				√		√			√		√			a\‘	一个’	Match
12		√				√		√				√			√		√		√			a\‘	多个’	Match
13		√			√			√			√			√			√		√			‘a’	一个a	Match
14		√			√			√			√				√		√		√			‘a’	多个a	Match
15		√			√			√				√		√			√		√			‘a”	一个’	Match
16		√			√			√				√			√		√		√			‘a”	多个’	Match
17		√			√				√		√			√			√		√			‘a ‘	一个空格	Match
18		√			√				√		√				√		√		√			‘a ‘	多个空格	Match

19		√			√				√			√		√			√		√			‘’a ’	出现一次	Match
20		√			√				√			√			√		√		√			‘’a ’	出现多次	Match
21		√			√					√	√			√			√		√			‘ a ‘	出现一次	Match
22		√			√					√	√				√		√		√			‘ a’ ‘	出现多次	Match
23		√			√					√		√		√			√		√			‘ a’ ‘	出现一次	Match
24		√			√					√		√			√		√		√			‘ a’ ‘	出现多次	Match
25			√			√		√			√			√			√					ab	一个ab	Match
26			√			√		√			√				√		√		√			ab	多个ab	Match
27			√			√		√				√		√			√		√			ab \‘	一个ab’	Match
28			√			√		√				√			√		√		√			ab \‘	多个ab’	Match
29			√		√			√			√			√			√		√			‘ab’	一个ab	Match
30			√		√			√			√				√		√		√			‘ab’	多个ab	Match
31			√		√			√				√		√			√		√			‘ab”	一个ab’	Match
32			√		√			√				√			√		√		√			‘ab”	多个ab’	Match
33			√		√				√		√			√			√		√			‘ab c‘	一个ab c	Match
34			√		√				√		√				√		√		√			‘ab c‘	多个ab c	Match
35			√		√				√			√		√			√		√			‘a b\’c’	一个a b’c	Match
36			√		√				√			√			√		√		√			‘a b\’c’	多个a b’c	Match
37			√		√					√	√			√			√		√			‘a b c’	一个a b c	Match
38			√		√					√	√				√		√		√			‘a b c’	多个a b c	Match
39			√		√					√		√		√			√		√			‘a b c\”	一a b c’	Match
40			√		√					√		√			√		√		√			‘a b c\”	多a b c’	Match

```
# Test Specification for find command
# Modified: property lists and selector expressions added
# Modified: error and single annotations added

Parameters:
    Pattern size:
        empty                                   [property Empty]
        single character                        [property NonEmpty]
        many character                          [property NonEmpty]
        longer than any line in the file        [error]

    Quoting:
        pattern is quoted                       [property Quoted]
        pattern is not quoted                   [if NonEmpty]
        pattern is improperly quoted            [error]

    Embedded blanks:
        no embedded blank                       [if NonEmpty]
        one embedded blank                      [if NonEmpty and Quoted]
        several embedded blanks                 [if NonEmpty and Quoted]

    Embedded quotes:
        no embedded quotes                      [if NonEmpty]
        one embedded quote                      [if NonEmpty]
        several embedded quotes                 [if NonEmpty] [single]

    File name:
        good file name
        no file with this name                  [error]
        omitted                                 [error]

Environments:
    Number of occurrences of pattern in file:
        none                                    [if NonEmpty] [single]
        exactly one                             [if NonEmpty] [property Match]
        more than one                           [if NonEmpty] [property Match]

    Pattern occurrences on target line:
    #  assumes line contains the pattern
        one                                     [if Match]
        more than one                           [if Match] [single]
```

图 1 测试说明书

实验二 白盒测试

一、实现代码以及控制流图

1. 实现代码以及代码对应标号如表 2 所示，其中一些变量的默认值为：
   - is_enclosed: False
   - is_escaped: False
   - is_line_too_long: True
   - pattern_valid: 空字符串

表 2 实现代码以及代码对应标号

序号	代码
1	def grep(pattern, filename):
2	if len(pattern) < 1:
3	return
4	if pattern[0] is “\’”:
5	is_enclosed = True
6	for i in range(len(pattern)):
7	if pattern[i] is ‘\\’ and is_escaped:
8	pattern_valid += pattern[i]
9	is_escaped = False
10	continue
11	if pattern[i] is ‘\\’ and not is_escaped:
12	is_escaped = True
13	continue
14	if i is len(pattern) – 1 and is_enclosed and (is_escaped or pattern[i] is not “\’”):
15	return
16	if i is len(pattern) – 1 and not is_enclosed and not is_escaped and pattern[i] is \’”:
17	return
18	if i is not 0 and pattern[i] is “\’” and is_escaped:
19	pattern_valid += pattern[i]
20	is_escaped = False
21	continue
22	if pattern[i] is “\’” and i not in [0, len(pattern) - 1] and not is_escaped:
23	return
24	if pattern[i] is ‘ ’ and is_enclosed:
25	pattern_valid += pattern[i]
26	continue
27	if pattern[i] is ‘ ’ and not is_enclosed:

```
28                  return

29              if pattern[i] not in [‘\”, ‘ ‘, ‘\\’] and is_escaped:
30                  is_escaped = False
31                  continue
32              if pattern[i] not in [‘\”, ‘ ‘, ‘\\’] and not is_escaped:
33                  pattern_valid += pattern[i]
34                  continue

35          if len(pattern) < 1:
36              return

37          with open(filename) as f:
38              for line in f.readlines():
39                  if line.find(pattern_valid) >=0:
40                      print(line)
41                  if len(line) >= len(pattern_valid) and is_line_too_long:
42                      is_line_too_long = False

43          if is_line_too_long:
44              return
```

2. 程序控制流图

基于表2对程序代码的标号，为程序绘制控制流图，如图2所示。

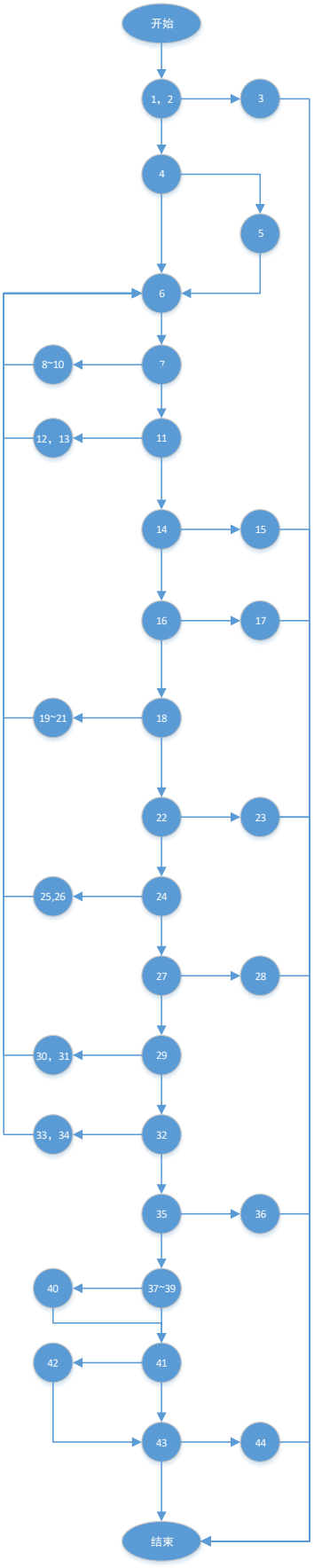

图 2 表 2 对应的程序控制流图

二、语句覆盖

1. 测试集设计

语句覆盖力求利用尽可能少的训练样本覆盖所有的语句。表 3 展示了为 grep 设计的语句覆盖测试用例，覆盖率由 python 扩展组件“coverage”生成。设计的测试用例最终达到了100%的语句覆盖率。

表 3 语句覆盖

序号	测试用例(pattern)	覆盖语句	覆盖率	作用
1	空	1, 2, 3	7%	覆盖 1~3
2	‘I\’m’	1, 2, 4~7, 11~14, 16, 18~22, 24, 27, 29, 32~35, 37~43	68%	覆盖 5, 12~13, 19~21, 33~34, 37~42
3	‘\ ’	1, 2, 4~14, 16, 18, 22, 24~27, 29, 32, 35, 37~39, 41~43	66%	覆盖 8~10, 25~26
4	\\	1, 2, 4, 6~13, 35, 37~39, 41~43	40%	覆盖 12~13
5	‘a	1, 2, 4~7, 11, 14~16, 18, 22, 24, 27, 32	36%	覆盖 15
6	a’	1, 2, 4, 6, 7, 11, 14, 16~18, 22, 24, 27, 29, 32~34	39%	覆盖 17
7	a’b	1, 2, 4, 6, 7, 11, 14, 16, 18, 22~24, 27, 29, 32~34	39%	覆盖 23
9	‘ ’	1, 2, 4, 6~7, 11, 14, 16, 18, 22, 24, 27~28	30%	覆盖 28
9	‘\m’	1, 2, 4, 6, 7, 11~14, 16, 18, 22, 24, 27, 29~31, 35~36	43%	覆盖 30~31，36
10	长度大于 File 每一行	1, 2, 4 , 6, 7, 11, 14, 16, 18, 22, 24, 27, 29, 32~35, 37~39, 41, 43, 44	52%	覆盖 43~44

2. 遇到的困难

在设计语句覆盖测试用例的过程中没有遇到特别多的困难，但在这个过程中，我了解到自己的程序中存在着许多冗余的地方，如判断pattern引号错误时，我将“首引尾无引”和“首无引尾引”当作两种错误来处理，这导致我分别设计了两个测试用例(表3中4, 5)才将这种错误覆盖。

三、分支覆盖

1. 分支条件统计

经统计，grep程序共有16个分支，32条边。现统计分支分别取真值、假值的条件如表4。表中第一列，代码行数字标红代表该分支取真值时，程序判定pattern出错。

表 4 分支取真\假值时的条件

代码行	真值	假值
2	空	不为空
4	首字符为引号	首字符不为引号
7	某字符为\ 转义	不为\ OR 非转义
11	某\ 非转义	不为\ OR 转义
14	尾字符 被括 转义OR非引	非尾 OR 不括 OR (不转义 AND 引)
16	尾字符 不括 不转义 引号	非尾 OR 被括 OR 转义 OR 非引
18	非首字符 引号 转义	首字符 OR 非引号 OR 不转义
22	非首尾字符 引号 不转义	首 OR 尾 OR 非引号 OR 转义
24	空格 被括	非空格 OR 不括
27	空格 不括	非空格 OR 括
29	非(引、空格、\) 转义	引号 OR 空格 OR \ OR 不转义
32	非(引、空格、\) 不转义	引号 OR 空格 OR \ OR 转义
35	无有效字符	含有效字符
39	文件中找到pattern	文件中未找到pattern
41	文件某行长度大于pattern长度 is_line_too_long为True	文件所有行长度小于pattern长度 OR is_line_too_long 为False
43	文件所有行长度小于pattern长度	文件某行长度大于pattern长度

2. 分支覆盖测试用例设计

根据表4，设计出分支覆盖测试用例，如表5所示。表5第一列表示设计的pattern值，第一列代表了分支所在代码行，标红的测试用例代表对应的pattern格式错误。从表中可以清楚看到，对于每一个分支，该测试集都会对其所有的边覆盖至少一次。最终，设计出的9个测试用例达到了100%分支覆盖率。

表 5 分支覆盖测试用例设计

pattern \ branch	2	4	7	11	14	16	18	22	24	27	29	32	35	39	41	43
null	T															
‘a					T											
a’						T										
a’a								T								
a a										T						
\													T			
longer than all lines																T
‘\\ \’’	F	T	T\F	T\F	F	F	T\F	F	T\F	F	F	F	F	F	T\F	F
m\m	F	F	F	T\F	F	F	F	F	F	F	T\F	T\F	F	T\F	T\F	F

四、条件覆盖

1. 条件覆盖测试用例设计

根据表4设计条件覆盖测试用例。其中，若有一个分支包含多个条件，则按表4中顺序为每个条件编号。例如，代码行为7的分支包含了两种条件：“某字符为\”和“转移状态”，则按照表中顺序，“某字符为\”标号为1，“转义状态”标号为2，以此类推。

条件覆盖测试用例如表6所示，由表知，所有条件的真/假值都被覆盖了一次，故这是个测试用例达到100%条件覆盖率。

表 6 条件覆盖测试用例设计

代码行	2	4	7		11		14				16				18			22			24		27		29		32		35	39	41	43
condition / pattern			1	2	1	2	1	2	3	4	1	2	3	4	1	2	3	1	2	3	1	2	1	2	1	2	1	2				
null	T																															
longer than all lines																															F	T
‘a\’	F	T	F		F		T\F	T	T																							
‘a	F	T	F		F		T\F	T	F	T																						
a’	F	F	F		F		T\F	F			T\F	T	T	T																		
a’a	F	F	F		F		T\F	F			T\F	T	T	F	T\F	T\F	F	T\F	T	T												
\	F	F	T	F	T	T	T	F			T	T	T	F	F			F			F		F		F		F		T			
空格	F	F	F		F		T	F			T	T	T	F	F			F			T	F	T	T								
‘\\\’a\a’	F	T	T\F	T\F	T\F	T\F	T\F	T	F	F	T\F	F			T\F	T\F	T	T\F	T\F	F	T\F	T	T\F	F	T\F	T\F	T\F	T\F	F	F	T	F
m\m	F	F	T	F	T	T	T\F	F			T\F	T	F		T	F		T\F	F		F		F		T\F	T\F	T\F	T\F	F	T	T	F

2. 遇到的困难

设计条件覆盖是一件细活，但绝不是一件难事。设计的难点在于python中并没有直接计算条件覆盖率的工具，因此结果的可靠性还需要进一步验证。

五、条件分支覆盖

1. 条件分支覆盖测试用例设计

条件分支覆盖要求测试用例覆盖所有条件的可能组合，延续第四部分条件覆盖的设计方法，设计条件分支覆盖测试用例如表7。由表7得，设计出的测试用例覆盖了所有分支条件的组合情况，条件分支覆盖率为100%.

表 7 条件分支覆盖测试用例设计

代码行	2	4	7		11		14				16				18			22			24		27		29		32		35	39	41	43
condition / pattern			1	2	1	2	1	2	3	4	1	2	3	4	1	2	3	1	2	3	1	2	1	2	1	2	1	2				
null	T																															
longer than all lines																															F	T
‘a\’	F	T	F		F		T\F	T	T																							
‘a	F	T	F		F		T\F	T	F	T																						
a’	F	F	F		F		T\F	F			T\F	T	T	T																		
a’a	F	F	F		F		T\F	F			T\F	T	T	F	T\F	T\F	F	T\F	T	T												
\	F	F	T	F	T	T	T	F			T	T	T	F	F			F			F		F		F		F		T			
空格	F	F	F		F		T	F			T	T	T	F	F			F			T	F	T	T								
‘\\\’a\a’	F	T	T\F	T\F	T\F	T\F	T\F	T	F	F	T\F	F			T\F	T\F	T	T\F	T\F	F	T\F	T	T\F	F	T\F	T\F	T\F	T\F	F	F	T	F
m\m	F	F	T	F	T	T	T\F	F			T\F	T	F		T	F		T\F	F		F		F		T\F	T\F	T\F	T\F	F	T	T	F

2. 遇到的困难

设计完条件覆盖测试用例之后，条件分支覆盖的设计就是一个查漏补缺的过程。其中比较繁琐的部分是对每个分支的所有条件进行组合。

实验小结

软件测试是软件开发中的重要过程之一。

程序员总认为自己写的代码是对的，我也不例外，虽然我有着这份自信，但是代码里总会让我有意想不到的情况发生，导致程序崩溃。如果按照编写程序的思维惯性去寻找程序中的错误，那可太难了——毕竟自己总是对的嘛。所以，需要利用科学的方法论来指导纠错的过程，避免低效无谓的“随机检测”，这就是软件测试的目的。

在实验的过程中，我确实发现了自己的程序存在的一些错误和缺陷。比如我没有按照规格说明书中的要求报错，又或者是多写了一些不必要的功能等等。这些错误有的会让程序直接崩溃，有的让我在设计测试用例时感叹，程序竟有如此多的冗余！

每个部分遇到的困难我都写在了相应章节的末尾，在这里就不详细说明了。本次实验的内容都是最基础的白盒黑盒测试，概念都直白易懂，设计测试用例的过程也并不麻烦。细心是最关键的。

附录 白盒测试文件内容

Need You Now
Picture perfect memories,
Scattered all around the floor,
Reaching for the phone cause, I can't fight it any more.
And I wonder if I ever cross your mind.
For me it happens all the time.
It's a quarter after one, I'm all alone and I need you now.
Said I wouldn't call but I lost all control and I need you now.
And I don't know how I can do without, I just need you now.
Oh...
Another shot of whiskey, can't stop looking at the door.
Wishing you'd come sweepin' in the way you did before.
And I wonder if I ever cross your mind.
For me it happens all the time.
It's a quarter after one, I'm a little drunk, and I need you now.
Said I wouldn't call but I lost all control and I need you now.
And I don't know how I can do without, I just need you now.
Guess I'd rather hurt than feel nothing at all.
It's a quarter after one, I'm all alone and I need you now.
And I said I wouldn't call but I'm a little drunk and I need you now.
And I don't know how I can do without, I just need you now.
I just need you now.
Oh baby I need you now.