软件测试与分析-黑盒测试

软件测试　黑盒测试　脚本　`shell`

2018.11.24
张丽玮 2016K8009929029

测试要求

---

1. 输入"D M Y"返回当天是星期几（范围从1900年到2500年）
2. 对于输入的形式要有明确的提示
3. 当输入不合法的时候需要提醒用户
4. 程序不应该停止运行（卡住、关闭等）当输入不合法的时候
5. 黑盒方法设计测试用例

测试方法分析

---

1. 由于输入不合法时没有特定输出要求，因此暂时认为可以手动测试不合法输入，之后再用脚本进行压力测试。
2. 由于1900-2500年总体数据量过于庞大，因此不选择全部天数测试的全部遍历测试方法，但是可以考虑以下方法
   - 回溯算法
   - 随机测试方法
   - 组合测试方法
3. 数据量较大，考虑用脚本实现。而关于如何生成测试数据，我应用了现成的excel软件功能，只需要更改单元格格式，就可以输出对应日期是星期几，并且这样生成csv文件对于之后脚本处理较为方便。

测试算法分析

1. 全部遍历算法：数据量过大，不进行选择
2. 回溯算法：
   实际上考虑回溯算法，是因为日期和星期之间是有规律存在的，或者说是可以构成循环的。
   一周七天，一年365或者366天，由于4年一个闰年，因此以年来计算，28年将会产生一个"星期-日期"的循环。也就是说，我的"D M Y"和"D M Y+28"对应星期几应该是相同的。既然如此，我们就可以采用回溯算法，给定一个日期，然后在年份上加减28的倍数，从而计算是否相等。
3. 随机测试方法：
   非常显然，由于我之前分析过28年一个周期，所以这里可以只随机选择前28年中的任意一天，然后年份全部一直+28直到2500年。同样，这个也可以在excel中用随机生成日期的公式完成。
4. 组合测试方法：
   我认为这个针对这个黑盒测试，组合测试没有必要，因为其实它本身就是一个组合测试——日、月、年的三元素组合。因此我觉得这样的输入数据难以分组，分组形式只可能是无效输入和有效输入组合，这个手动测试就能发现问题。

具体实现

无效输入测试

- 整型无效

可以报错从年月日依次报错，并且会提示某月最多只有多少天，月份是1-12，不会出现卡死、关闭等现象。

但是如果年份月份符合要求时，判断日期会出错。当日期<=0时并不会报错，反而会输出结果。大体可以推断这是由于日期只进行了上界判断而没有判断下界。【bug1】

- 无输入

  无输入，一直空格回车时不会有任何输出（windows、linux都是）

- 输入类型无效

当输入非整型时，会进入死循环，只能强制退出。

无论是字母还是符号。【bug2】

- 压力测试

  只考虑整型无效的压力测试，测试结果并无异常。

有效输入测试

- 1900-1927范围内28年的所有日期情况

```
[10209]PASS!!!!!!
[10210]PASS!!!!!!
[10211]PASS!!!!!!
[10212]PASS!!!!!!
[10213]PASS!!!!!!
[10214]PASS!!!!!!
[10215]PASS!!!!!!
[10216]PASS!!!!!!
[10217]PASS!!!!!!
[10218]PASS!!!!!!
[10219]PASS!!!!!!
[10220]PASS!!!!!!
[10221]PASS!!!!!!
[10222]PASS!!!!!!
[10223]PASS!!!!!!
[10224]PASS!!!!!!
[10225]PASS!!!!!!
[10226]PASS!!!!!!
[10227]PASS!!!!!!
[10228]PASS!!!!!!
[10229]PASS!!!!!!
[10230]PASS!!!!!!
[10231]PASS!!!!!!
cod@cod-VirtualBox:/mnt/shared/shell$
```

10231个测试用例全部pass，没有生成diffline.txt文件

- 随机测试

  随机测试是在excel里面随机生成1000个1900-2500范围内的有效日期，然后用脚本进行测试。

·第一组1000个测试数据：

```
cod@cod-VirtualBox: /mnt/shared/shell
[978]PASS!!!!!!
[979]11 12 2223 reference:The Day is Thursday result:The Day is Saturday.
[980]PASS!!!!!!
[981]PASS!!!!!!
[982]PASS!!!!!!
[983]PASS!!!!!!
[984]PASS!!!!!!
[985]PASS!!!!!!
[986]PASS!!!!!!
[987]25 10 2158 reference:The Day is Wednesday result:The Day is Thursday.
[988]PASS!!!!!!
[989]PASS!!!!!!
[990]PASS!!!!!!
[991]PASS!!!!!!
[992]PASS!!!!!!
[993]PASS!!!!!!
[994]PASS!!!!!!
[995]25 4 2148 reference:The Day is Thursday result:The Day is Friday.
[996]PASS!!!!!!
[997]23 11 2467 reference:The Day is Wednesday result:The Day is Saturday.
[998]PASS!!!!!!
[999]PASS!!!!!!
[1000]PASS!!!!!!
cod@cod-VirtualBox:/mnt/shared/shell$
```

查看diffline.txt的错误输出文本，共有273个错误，即这次的错误率是273/1000

接下来开始分析错误日期：


```
251    [862]23 2 2185 reference:The Day is Wednesday result:The Day is Thursday.
252    [864]7 9 2215 reference:The Day is Thursday result:The Day is Saturday.
253    [868]28 1 2223 reference:The Day is Tuesday result:The Day is Thursday.
254    [872]28 11 2199 reference:The Day is Thursday result:The Day is Friday.
255    [880]11 8 2472 reference:The Day is Thursday result:The Day is Sunday.
256    [888]29 8 2205 reference:The Day is Thursday result:The Day is Saturday.
257    [893]9 6 2308 reference:The Day is Tuesday result:The Day is Friday.
258    [894]5 10 2269 reference:The Day is Tuesday result:The Day is Thursday.
259    [897]30 12 2421 reference:The Day is Thursday result:The Day is Sunday.
260    [903]10 2 2229 reference:The Day is Tuesday result:The Day is Thursday.
261    [907]30 9 2488 reference:The Day is Thursday result:The Day is Sunday.
262    [918]5 1 2185 reference:The Day is Wednesday result:The Day is Thursday.
263    [925]28 8 2341 reference:The Day is Thursday result:The Day is Sunday.
264    [945]31 10 2198 reference:The Day is Wednesday result:The Day is Thursday.
265    [947]5 1 2393 reference:The Day is Tuesday result:The Day is Friday.
266    [951]5 3 2336 reference:The Day is Thursday result:The Day is Sunday.
267    [973]31 7 2452 reference:The Day is Wednesday result:The Day is Saturday.
268    [975]30 5 2173 reference:The Day is Sunday result:The Day is Monday.
269    [977]22 5 2328 reference:The Day is Tuesday result:The Day is Friday.
270    [979]11 12 2223 reference:The Day is Thursday result:The Day is Saturday.
271    [987]25 10 2158 reference:The Day is Wednesday result:The Day is Thursday.
272    [995]25 4 2148 reference:The Day is Thursday result:The Day is Friday.
273    [997]23 11 2467 reference:The Day is Wednesday result:The Day is Saturday.
274
```

运用查询发现，所有错误日期都在2000年之后，不存在19xx的日期错误。那么初步猜测，1900-2000并没有错误。因此接下来进行分类测试：1900-2000年间和2000-2500年间

· 第二组1900-2000年间1000个测试数据：


```
[978]PASS!!!!!!
[979]PASS!!!!!!
[980]PASS!!!!!!
[981]PASS!!!!!!
[982]PASS!!!!!!
[983]PASS!!!!!!
[984]PASS!!!!!!
[985]PASS!!!!!!
[986]PASS!!!!!!
[987]PASS!!!!!!
[988]PASS!!!!!!
[989]PASS!!!!!!
[990]PASS!!!!!!
[991]PASS!!!!!!
[992]PASS!!!!!!
[993]PASS!!!!!!
[994]PASS!!!!!!
[995]PASS!!!!!!
[996]PASS!!!!!!
[997]PASS!!!!!!
[998]PASS!!!!!!
[999]PASS!!!!!!
[1000]PASS!!!!!!
cod@cod-VirtualBox:/mnt/shared/shell$
```

1000个测试数据全部pass，也就是100%的通过率。再结合前28年全部pass的情况，现在有充足理由认为1900-2000年间全部正确没有bug。

· 第三组2000-2500年间1000个测试数据：

可以发现这一次有相当多的错误情况。而查看diffline.txt的错误报告，共有378个错误，错误率是378/1000

继续分析错误日期：

发现并没有2000-2100年间的错误日期。按照同样的方法再次缩小范围。

· 第四组2000-2100年间1000个测试数据:

全部pass，认为这之间没有bug。

· 第五组2100-2500年间1000个测试数据：

```
cod@cod-VirtualBox: /mnt/shared/shell
[978]PASS!!!!!!
[979]8 11 2105 reference:The Day is Sunday result:The Day is Monday.
[980]PASS!!!!!!
[981]27 1 2366 reference:The Day is Thursday result:The Day is Sunday.
[982]2 11 2265 reference:The Day is Thursday result:The Day is Saturday.
[983]PASS!!!!!!
[984]PASS!!!!!!
[985]21 5 2498 reference:The Day is Wednesday result:The Day is Saturday.
[986]9 11 2129 reference:The Day is Wednesday result:The Day is Thursday.
[987]7 12 2293 reference:The Day is Thursday result:The Day is Saturday.
[988]PASS!!!!!!
[989]PASS!!!!!!
[990]11 1 2176 reference:The Day is Thursday result:The Day is Friday.
[991]15 11 2388 reference:The Day is Tuesday result:The Day is Friday.
[992]11 12 2334 reference:The Day is Tuesday result:The Day is Friday.
[993]PASS!!!!!!
[994]PASS!!!!!!
[995]25 10 2281 reference:The Day is Tuesday result:The Day is Thursday.
[996]5 7 2293 reference:The Day is Wednesday result:The Day is Friday.
[997]PASS!!!!!!
[998]PASS!!!!!!
[999]PASS!!!!!!
[1000]PASS!!!!!!
cod@cod-VirtualBox:/mnt/shared/shell$
```

明显错误更多了，这时候再查看diffline.txt，共有468个错误，错误率468/1000

继续分析错误日期：


```
450    [964]1 8 2201 reference:The Day is Saturday result:The Day is Monday.
451    [967]31 3 2304 reference:The Day is Thursday result:The Day is Sunday.
452    [968]24 7 2148 reference:The Day is Wednesday result:The Day is Thursday.
453    [971]20 1 2255 reference:The Day is Saturday result:The Day is Monday.
454    [972]24 4 2369 reference:The Day is Thursday result:The Day is Sunday.
455    [973]1 7 2192 reference:The Day is Sunday result:The Day is Monday.
456    [974]19 5 2466 reference:The Day is Wednesday result:The Day is Saturday.
457    [977]21 10 2483 reference:The Day is Thursday result:The Day is Sunday.
458    [979]8 11 2105 reference:The Day is Sunday result:The Day is Monday.
459    [981]27 1 2366 reference:The Day is Thursday result:The Day is Sunday.
460    [982]2 11 2265 reference:The Day is Thursday result:The Day is Saturday.
461    [985]21 5 2498 reference:The Day is Wednesday result:The Day is Saturday.
462    [986]9 11 2129 reference:The Day is Wednesday result:The Day is Thursday.
463    [987]7 12 2293 reference:The Day is Thursday result:The Day is Saturday.
464    [990]11 1 2176 reference:The Day is Thursday result:The Day is Friday.
465    [991]15 11 2388 reference:The Day is Tuesday result:The Day is Friday.
466    [992]11 12 2334 reference:The Day is Tuesday result:The Day is Friday.
467    [995]25 10 2281 reference:The Day is Tuesday result:The Day is Thursday.
468    [996]5 7 2293 reference:The Day is Wednesday result:The Day is Friday.
469
```

然而目前肉眼已经难以判断出其中的规律。为了判断主要错误从哪里开始发生，查看
samelin.txt，这时发现有这样的情况：


```
[1]18 6 2350 reference:The Day is Sunday
[2]18 6 2310 reference:The Day is Saturday
```

那么此时猜测一下，现在的bug和年份无关，而是和月、日有关，因此接下来选择一年的366天进行遍历，各选择两年。

· 第六组2325年的全部365个测试数据


```
cod@cod-VirtualBox: /mnt/shared/shell
 result:The Day is Saturday. Day is Wednesday
 result:The Day is Sunday.The Day is Thursday
[345]PASS!!!!!!
[346]PASS!!!!!!
[347]PASS!!!!!!
[348]PASS!!!!!!
 result:The Day is Friday.The Day is Tuesday
 result:The Day is Saturday.e Day is Wednesday
 result:The Day is Sunday.The Day is Thursday
[352]PASS!!!!!!
[353]PASS!!!!!!
[354]PASS!!!!!!
[355]PASS!!!!!!
 result:The Day is Friday.The Day is Tuesday
 result:The Day is Saturday.e Day is Wednesday
 result:The Day is Sunday.The Day is Thursday
[359]PASS!!!!!!
[360]PASS!!!!!!
[361]PASS!!!!!!
[362]PASS!!!!!!
 result:The Day is Friday.The Day is Tuesday
 result:The Day is Saturday.e Day is Wednesday
 result:The Day is Sunday.The Day is Thursday
cod@cod-VirtualBox:/mnt/shared/shell$
```

一眼就能看出报错非常有规律，然后去查看diffline.txt，


```
 [1]1 1 2325 reference:The Day is Thursday
  result:The Day is Sunday.
 [6]6 1 2325 reference:The Day is Tuesday
  result:The Day is Friday.
 [7]7 1 2325 reference:The Day is Wednesday
  result:The Day is Saturday.
 [8]8 1 2325 reference:The Day is Thursday
  result:The Day is Sunday.
 [13]13 1 2325 reference:The Day is Tuesday
  result:The Day is Friday.
 [14]14 1 2325 reference:The Day is Wednesday
  result:The Day is Saturday.
 [15]15 1 2325 reference:The Day is Thursday
```

非常明显，它所有的周二周三周四都判断错误，都错判成了周五周六周日。那么其他年份呢？

· 第七组2202年的全部365个测试数据

同样非常有规律，甚至比2325年还多了一个周六的错误。去查看diffline.txt也发现这一年所有的周二周三周四和周六都是错的。

【bug】

现在初步得到猜测：1900-2100年间的日期对应星期都是正确的，但是2100-2500年间有错误，并且错误有规律。是每一周的固定几天出错，后来猜测每一百年改变一次规律，又进行了几个单独年份的测试，最后得到的结果如下：

从2100年3月3日开始出错

2100-2199年——所有周三周四周日都为周四周五周一

2200-2299年——所有周二周三周四周六都为周四周五周六周一

2300-2499年——所有周二周三周四都为周五周六周日

2500年从3月1日开始所有周一周二周三周四都为周五周六周日周一

测试脚本

实验过程中均采用我自己用shell编写的简易脚本，在linux上运行

· 一键完成脚本test0

```
1.
2.    #! /bin/sh
3.    cat $1.csv | awk -F, '{ print $1; }' > test_in.txt
4.    cat $1.csv | awk -F, '{ print $2; }' > test_out.txt
5.    ##生成对应的测试文件输入和输出
6.    #i=0;
```

```
7.    cat test_in.txt | while read str_in #逐行获取输入
8.    do
9.        output=`./test.out $str_in >> test_temp.txt 2>&1`
10.   done
11.   cat test_temp.txt | grep "The" >test_result.txt
12.
13.       if [  -f "test_temp.txt" ];
14.       then
15.           rm test_temp.txt
16.       fi
17.       if [  -f "diffline.txt" ];
18.       then
19.           rm diffline.txt
20.       fi
21.       if [  -f "sameline.txt" ];
22.       then
23.           rm sameline.txt
24.       fi
25.
26.
27.       i=0
28.       cat test_result.txt | while read result
29.       do
30.       ((i=i+1))
31.         sed -n ${i}p test_out.txt | while read str_out
32.       do
33.       reference="The Day is $str_out"
34.       sed -n ${i}p test_in.txt | while read str_in
35.       do
36.       #echo $output
37.       if [ "$reference" \< "$result" ]
38.           then
39.           echo "[$i]PASS!!!!!!"
40.           echo "[$i]$str_in reference:$reference" >> sameline.txt
41.           else
42.           echo "[$i]$str_in reference:$reference result:$result" >> diffl
      ine.txt
43.           echo "[$i]$str_in reference:$reference result:$result"
44.         fi
45.       done
46.       done
47.   done
```

使用方法是输入 ./test0.sh 测试数据csv表名

注：不要写csv后缀

生成的diffline.txt会标注所有错误的行数、日期、以及正确输出和错误输出

生成的sameline.txt则是所有正确的行数、日期、正确输出

· 分步式防卡死脚本test+test2

```sh
#! /bin/sh
cat $1.csv | awk -F, '{ print $1; }' > test_in.txt
cat $1.csv | awk -F, '{ print $2; }' > test_out.txt
##生成对应的测试文件输入和输出
#i=0;
cat test_in.txt | while read str_in #逐行获取输入
do
    #echo "$str_in"
    output=`./test.out $str_in`
done
```

```sh
#! /bin/sh
    cat terminal.txt | grep "The" >test_result.txt
    if [  -f "diffline.txt" ];
    then
        rm diffline.txt
    fi
    if [  -f "sameline.txt" ];
    then
        rm sameline.txt
    fi


    i=0
    cat test_result.txt | while read result
    do
    ((i=i+1))
      sed -n ${i}p test_out.txt | while read str_out
    do
    reference="The Day is $str_out"
    sed -n ${i}p test_in.txt | while read str_in
        do
```

```
23.        #echo $output
24.        if [ "$reference" \< "$result" ]
25.            then
26.            echo "[$i]PASS!!!!!!"
27.            else
28.            echo "[$i]$str_in reference:$reference result:$result" >> diffl
    ine.txt
29.            echo "[$i]$str_in reference:$reference result:$result"
30.         fi
31.       done
32.       done
33.    done
```

使用方法：

script terminal.txt先打开终端日志记录

./test.sh 测试数据表名

exet退出终端日志

然后执行./test2.sh

最后生成的文件结果和test0一样。

这两个脚本只是防止一键完成时中途卡死，通过终端日志中转完成测试。

结果总结

发现的bug

1. 非整型类型的无效输入会导致死循环
2. 日期<=0 但是月年正确时不会报错
3. 从2100-2500年间规律性出错

   从2100年3月3日开始出错

   2100-2199年——所有周三周四周日都为周四周五周一

   2200-2299年——所有周二周三周四周六都为周四周五周六周一

   2300-2499年——所有周二周三周四都为周五周六周日

   2500年从3月1日开始所有周一周二周三周四都为周五周六周日周一

猜测bug原因

1. 没有进行类型判断，并没有处理整型以外的类型，导致读入错误进入死循环
2. 没有进行日期的下界判断，导致没有报错正常输出
3. 这个我觉得不是代码功能性错误或者逻辑错误导致的，应该属于人为控制，故意更改了一些年份的输出（比如每隔100年换一下错误输出），而且这个太具有规律性，是按周出错，猜测是直接写了if判断条件强行改变了输出。

不足之处

1. 初始的28年计算显得多此一举。实际上这个实验，回溯算法并没有很好的效果
2. 随机算法随机性比较大，还是不能完全排除遗漏的错误可能（比如再1900-2100年间）
3. 最后单独分析年份的时候是取的每一百年的头尾，因为猜测是百年计更改的输出，但是不排除可能这中间的年份并不是这个规律。
4. 测试工程量还是比较大，实际上我是在手动进行数据的筛选工作，而更理想的情况是我们把这个也编入脚本，由脚本来进行筛选，这样效率更高，准确率也更高。
5. 实际上查资料发现一般万年历是从1900年1月1日为周一开始计算的，但是excel实际上是周日，而且1900年被认为是闰年，巧合的是这个软件也这么认为了。所以说我这一段是全pass的，但如果是从按照一般万年历，应该从1900年1月1日到1900年2月28日的结果都是有误的，直到3月1日恢复正常。

补充说明

除了文档外所有的包括脚本、数据、错误输出等文件全部打包
文件架构如下：

```
1.    SoftwareTest_findday
2.    ├─reveal.js
3.    ├─test_data
4.    │   ├─test_2100
5.    │   ├─test_2199
6.    │   ├─test_2202
7.    │   ├─test_2299
```

```
  8.    │    ├─test_2300
  9.    │    ├─test_2325
 10.    │    ├─test_2400
 11.    │    ├─test_2499
 12.    │    ├─test_2500
 13.    │    ├─test_all
 14.    │    ├─test_random1
 15.    │    ├─test_random2
 16.    │    ├─test_random3
 17.    │    ├─test_random4
 18.    │    └─test_random5
 19.    ├─test_result
 20.    │    ├─diffline_2100
 21.    │    ├─diffline_2199
 22.    │    ├─diffline_2202
 23.    │    ├─diffline_2299
 24.    │    ├─diffline_2300
 25.    │    ├─diffline_2325
 26.    │    ├─diffline_2400
 27.    │    ├─diffline_2499
 28.    │    ├─diffline_2500
 29.    │    ├─diffline_random1
 30.    │    ├─diffline_random3
 31.    │    └─diffline_random5
 32.    ├─test.exe
 33.    ├─test.out
 34.    ├─test0.sh
 35.    ├─test.sh
 36.    ├─test2.sh
 37.    ├─test_in.txt
 38.    ├─test_out.txt
 39.    ├─test_result.txt
 40.    ├─my_presentation.md
 41.    ├─my_presentation.html
 42.    ├─软件测试分析-黑盒测试.pdf
 43.    └─README.md
```

使用说明：由于测试的时候并没有分文件夹全都在一个目录下，因此如果要测试数据需要在写表名时加上path。

my_presentation和reveal.js必须在同一目录下才能正常播放