

Task1-Q3

2019年10月12日 11:54

用递归算法完成如下问题：有52张牌，使它们全部正面朝上，第一轮是从第2张开始，凡是2的倍数位置上的牌翻成正面朝下；第二轮从第3张牌开始，凡是3的倍数位置上的牌，正面朝上的翻成正面朝下，正面朝下的翻成正面朝上；第三轮从第4张牌开始，凡是4的倍数位置上的牌按上面相同规则翻转，以此类推，直到第一张要翻的牌超过52为止。统计最后有几张牌正面朝上，以及它们的位置号。

- **递归**

在这里想要提到的是一些在别人博客里看到的，或许东大本课老师有讲到，可能我没认真听讲...

- **递归要的是以下几点要素**

1. 一个问题的解可以分解为几个子问题的解
2. 这个问题分解后的子问题，除了数据规模不同，求解思路完全一样
3. 一定存在终止递归的条件（找到递归出口）

- **关键带点**

- 写出递推公式
 - 找到终止条件
 - 转换为代码

- **递归代码警惕堆栈溢出**

函数调用会使用栈来保存临时变量，每调用一个函数，都会将临时变量封装为栈帧压入内存栈，等函数执行完成返回时，才出栈。系统栈或者虚拟机栈空间一般都不大，如果递归求解的数据规模很大，调用层次很深，一直压入栈，就会有堆栈溢出的风险。

- **如何避免堆栈溢出**

在代码中限制递归调用的最大深度，当递归调用超过一定深度，比如1000之后，就不再继续往下递归了，直接返回报错。但是如果递归深度比较大，这种方法就不太适用。

采取循环的方式来解决，将需要的数据在关键的调用点保存下来使用。简单的说，就是用自己的数据保存方法来代替系统递归调用产生的堆栈数据

- **思路：**

要求结束时的状态是直到有要翻过的牌超过52为止，也就是我们要找到的递归出口。在翻牌的过程中要明确，对于第*i*轮翻牌时，满足 $(j+1)\%(i+1)==0$ 的 *j* 位置上的牌将会翻转，**翻转的含义**：
当 $s[j]=0$ 时， $s[j]$ 将会变成1；当 $s[j]=1$ 时， $s[j]$ 将会变成0；
当所有牌不再变动时，所有背面朝上的牌即满足 $s[j]=0$ ，牌上的数字即为 $j+1$

- 首先利用**非递归**做法来验证思路，源代码如下：

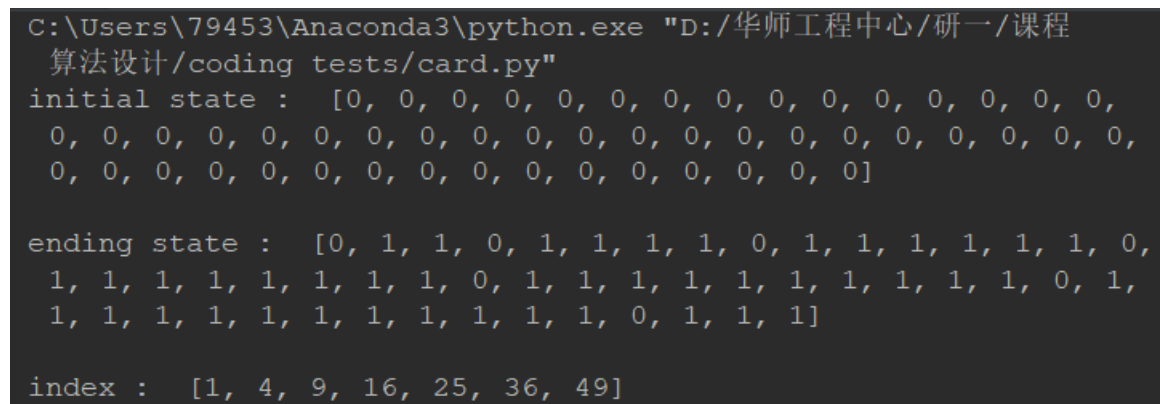
```
1  #non_recursion
2  s = [0]*52
3  print(s)
4
5  for i in range(1,52):
6      for j in range(i,52):
7          if ((j+1)%(i+1)==0):
8              if(s[j]==0):
9                  s[j]=1
10             else:
11                 s[j]=0
12
13  print(s)
14
15
16  vec = [x+1 for x in range(52) if s[x]==0]
17  print(vec)
```

运行截图：

第一行代表初始状态

第二行代表翻牌结束状态

第三行表示正面朝上的牌的位置号



```
C:\Users\79453\Anaconda3\python.exe "D:/华师工程中心/研一/课程
算法设计/coding tests/card.py"
initial state :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

ending state :  [0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]

index :  [1, 4, 9, 16, 25, 36, 49]
```

- **递归**

找到递归出口为最后一张翻得牌大于等于52时，每一次的逻辑重复就是题干所描述的。

源代码如下:

```
#recursion
s = [0]*52
print("initial state : ",s,"\n")

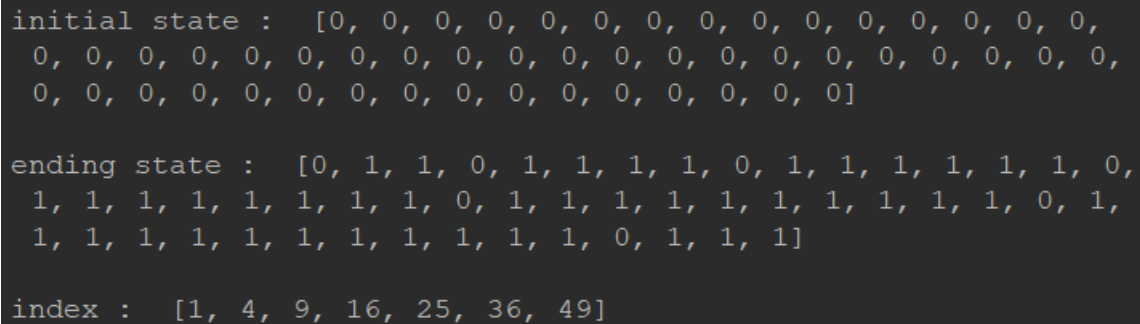
def turn_card(n):
    if(n>52):
        return
    else:
        for j in range(n,52):
            if ((j + 1) % (n + 1) == 0):
                if (s[j] == 0):
                    s[j] = 1
                else:
                    s[j] = 0
            turn_card(n+1)

turn_card(1)

print("ending state : ", s, "\n")

vec = [x+1 for x in range(52) if s[x]==0]
print("index : ",vec,"\n")
```

运行截图:



```
initial state :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

ending state :  [0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]

index :  [1, 4, 9, 16, 25, 36, 49]
```

完整代码如下:

```
# -*- coding: utf-8 -*-
# @Time : 2019/10/11 18:42
# @Author : BaoBao
# @Mail : baobaotql@163.com
# @File : card.py
# @Software: PyCharm

#non_recursion
s = [0]*52
print("initial state : ",s,"\n")

for i in range(1,52):
    for j in range(i,52):
        if((j+1)%(i+1)==0):
            if(s[j]==0):
                s[j]=1
            else:
                s[j]=0
```

```

print("ending state : ",s,"\n")

vec = [x+1 for x in range(52) if s[x]==0]
print("index : ",vec,"\n")


#recursion
s = [0]*52
print("initial state : ",s,"\n")

def turn_card(n):
    if(n>52):
        return
    else:
        for j in range(n,52):
            if ((j + 1) % (n + 1) == 0):
                if (s[j] == 0):
                    s[j] = 1
                else:
                    s[j] = 0
            turn_card(n+1)

turn_card(1)

print("ending state : ", s, "\n")

vec = [x+1 for x in range(52) if s[x]==0]
print("index : ",vec,"\n")

```

运行截图:

```

C:\Users\79453\Anaconda3\python.exe "D:/华师工程中心/研一/课程
算法设计/coding tests/card.py"
initial state :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
ending state :  [0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
index :  [1, 4, 9, 16, 25, 36, 49]

initial state :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
ending state :  [0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
index :  [1, 4, 9, 16, 25, 36, 49]

Process finished with exit code 0

```