

100

Python实验题样例及解析

cracked and reassembled by ar

经沈睿老师审核通过 仅供教学使用 严禁商用 for educational use only

30分钟内完成有效

7-1 求平方与倒数序列的近似和

题目描述

本题要求对两个正整数 m 和 n ($m \leq n$) 编写程序，计算序列近似和。近似和定义为小于序列和的最大整数。（提示：用 `floor` 函数）

即求出 $m^2 + \frac{1}{m} + (m+1)^2 + \frac{1}{m+1} + \dots + n^2 + \frac{1}{n}$ 的值。

输入格式

输入在一行中给出两个正整数 m 和 n ($m \leq n$)，其间以空格分开。

输出格式

在一行中按照 `sum ≈ S` 的格式输出近似和 S 。

样例输入

在这里给出一组输入。例如：

```
5 10
```

样例输出

```
sum ≈ 355
```

限制条件

代码长度限制：16 KB

时间限制：400 ms

内存限制：64 MB

参考代码及思路提示

这是一道典型的求和问题（来自平时作业）。通过观察不难发现数列的通项公式为 $(i^2 + \frac{1}{i})$ 且*i* ∈ [m, n]，因此可结合第I类列表推导式和内置函数sum即可快速实现求和。另外题目中提示的floor函数其实可以用内置函数int代替，为使代码尽可能简洁这里不再使用math库。

最后需要注意输出格式（格式错误也算答案错误，0分），因为存在不常见字符约等号≈且其两侧各存在1个空格，所以建议在OMS中点击【样例输出】框的右上角的【点击复制】按钮一键复制；或者也可以用鼠标选中整串输出再按下组合键【Ctrl+C】进行复制，然后将它粘贴到格式化字符串中即可。

方法一：

输入样例:

在这里给出一组输入。例如:

```
5 10
```

输出样例:

在这里给出相应的输出。例如:

```
sum ≈ 355
```

鼠标点击直接复制 

方法二：

输入样例:

在这里给出一组输入。例如:

```
5 10
```

输出样例:

在这里给出相应的输出。例如:

```
sum ≈ 355
```

用鼠标选中→Ctrl+C复制 

```
m,n=map(int,input().split())
print("sum ≈ {}".format(int(sum([i**2+1/i for i in range(m,n+1)])))) # 第I类列表推导式
# 将复制下来的样例输出粘贴到格式化字符串中，具体数值用占位符代替
```

7-2 逆时针蛇形矩阵

题目描述

所谓“蛇形矩阵”，是指对任意给定的正整数N（1 ≤ N ≤ 20），将1到N × N的数字从左上角第1个格子开始，按顺时针或逆时针螺旋方向顺序填入一个N × N的方阵里。本题要求构造逆时针蛇形矩阵。

输入格式

输入在一行中给出一个正整数 N ($1 \leq N \leq 20$)。

输出格式

输出 $N \times N$ 的逆时针蛇形矩阵。每行 N 个数字，每个数字占4位。

样例输入

```
5
```

样例输出

```
1 16 15 14 13
2 17 24 23 12
3 18 25 22 11
4 19 20 21 10
5  6   7   8   9
```

限制条件

代码长度限制：16 KB

时间限制：400 ms

内存限制：64 MB

参考代码及思路提示

这是一道典型的格式题，即按照一定的格式输出数据的题目（来自C语言习题集）。乍一看似乎很不好做，但想通了其实很简单！不过有些同学一看题目可能就会想当然地试着用递归做，这里要注意，能用递归解决的题目一定能将问题拆分为规模更小的子问题（也就是“老和尚给小和尚讲故事”中的这个“故事”除了规模差异外不能有其它任何不同点），且应具有一个非常明显的递归出口（例如 $0! = 1$ ），所以本题并不具备递归的条件，这个思路是行不通的。

那么，正确的思路是什么呢？我们不妨先来仔细观察规模为 5×5 的逆时针蛇形矩阵：

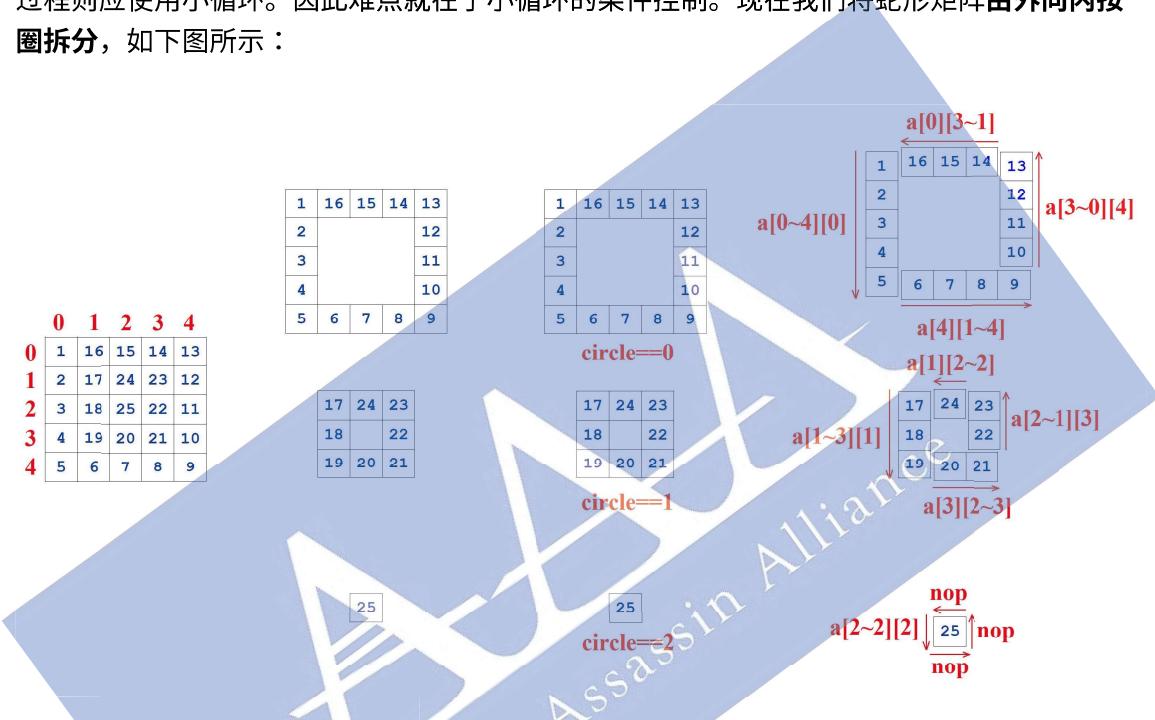
1	16	15	14	13
2	17	24	23	12
3	18	25	22	11
4	19	20	21	10
5	6	7	8	9

还是没有思路？好的，我们试着用不同颜色标记这个蛇形矩阵：

1	16	15	14	13
2	17	24	23	12
3	18	25	22	11
4	19	20	21	10
5	6	7	8	9

你发现了什么？是的，这个蛇形矩阵恰好可以由外到内分为3圈，就像“回”字那样，每“圈”的左上角是数字的起点，按照逆时针方向依次递增1。恭喜你！只要发现了这条性质，你距离最终的答案就只有一步之遥了，因为你发现了本程序中最重要的一个变量，该变量需要控制圈数，我们不妨将它记为 `circle`。

既然本题不需要使用递归，那么我们应该如何画出这个矩阵呢？我们知道，基本的程序设计结构只有3种：顺序结构、选择结构、循环结构。在这里我们显然需要循环结构。现在的问题就在于：如何控制循环条件？我们需要一个大循环，一个变量应从0开始累加，每加1就按照逆时针方向填入矩阵中，同时当一圈的填充结束后还应将变量 `circle` 的值加1；逆时针填充的过程则应使用小循环。因此难点就在于小循环的条件控制。现在我们将蛇形矩阵由外向内按圈拆分，如下图所示：



其中最小圈的 `nop` 表示仅画左侧1次，下、右、上3个方向均不执行操作。通过仔细观察每圈的画法，不难总结出4个方向的通项公式：左侧为 `range(circle, n-circle)`、下方为 `range(circle+1, n-circle)`、右侧为 `range(n-circle-2, circle-1, -1)`、上方为 `range(n-circle-2, circle, -1)`；在这里我们并不需要担心“越界”的问题，因为当 `circle` 足够大时，`range` 对象就会变为空集，对应的 `for` 循环就不再被执行。这样我们就能轻松写出正确代码：

```

n=int(input())
a,i,circle=[[0]*n for j in range(n)],0,0 # (等长) 序列赋值
# 变量circle控制圈数，变量i用于计数并填充
while i<n*n: # 这里使用while循环更方便
    for j in range(circle,n-circle): # 左
        i+=1
        a[j][circle]=i
    for j in range(circle+1,n-circle): # 下
        i+=1
        a[n-circle-1][j]=i
    for j in range(n-circle-2,circle-1,-1): # 右
        i+=1
        a[j][n-circle-1]=i
    
```

```

for j in range(n-circle-2,circle,-1): # 上
    i+=1
    a[circle][j]=i
circle+=1 # 按照左下右上的顺序依次填充完成后一圈就完成了，圈数累加
for i in range(n):
    for j in range(n):
        print("{:4d}".format(a[i][j]),end='') # 输出，每个数宽度为4
    print() # 换行

```

像这样，如果我们需要举一反三写出顺时针蛇形矩阵的代码也是不在话下的，只需要修正通项公式并改为按照上、右、下、左的顺序依次填充即可。

看到这里，有些同学可能会问：做题的时间这么紧张，如果没法想到这样的好办法怎么办呢？这里就要说到一个技巧，也就是“试”或“猜”测试点，通过打表的方法得分（学过OI的同学肯定知道这个方法）。例如，由题意 N 的取值范围为 $1 \leq N \leq 20$ ，题目显然不可能将20个测试点全部测一遍，而一定是抽样测试的。通过仔细分析和测试，不难测出本题的测试点为 N 分别取5（样例输入）、6、9和13，因此在草稿纸上演算出这4个测试点直接通过字符串输出也可得满分。一旦发现打表比思考正确算法节省时间，则不妨采用聪明的打表方法得分。

参考代码如下：

```

# 打表，这里使用字典比列表更方便
testcase={1:' 1 4\n 2 3  ',\n2:' 1 8 7\n 2 9 6\n 3 4 5  ',\n3:' 1 12 11 10\n 2 13 16 9\n 3 14 15 8\n 4 5 6 7  ',\n4:' 1 16 15 14 13\n 2 17 24 23 12\n 3 18 25 22 11\n 4 19 20 21 10\n 5 6 7 8 9  ',\n5:' 1 20 19 18 17 16\n 2 21 32 31 30 15\n 3 22 33 36 29 14\n 4 23 34 35 28 13\n 5 24 25 26 27 12\n 6 7 8 9 10 11  ',\n6:' 1 32 31 30 29 28 27 26 25\n 2 33 56 55 54 53 52 51 24\n 3 34 57 72 71 70 69 50 23\n 4 35 58 73 80 79 68 49 22\n 5 36 59 74 81 78 67 48 21\n 6 37 60 75 76 77 66 47 20\n 7 38 61 62 63 64 65 46 19\n 8 39 40 41 42 43 44 45 18\n 9 10 11 12 13 14 15 16 17  ',\n9:' 1 48 47 46 45 44 43 42 41 40 39 38 37\n 2 49 88 87 86 85 84 83 82 81 80 79 36\n 3 50 89 120 119 118 117 116 115 114 113 78 35\n 4 51 90 121 144 143 142 141 140 139 112 77 34\n 5 52 91 122 145 160 159 158 157 138 111 76 33

```

```
6 53 92 123 146 161 168 167 156 137 110 75 32
7 54 93 124 147 162 169 166 155 136 109 74 31
8 55 94 125 148 163 164 165 154 135 108 79 30
9 56 95 126 149 150 151 152 153 134 107 72 29
10 57 96 127 128 129 130 131 132 133 106 71 28
11 58 97 98 99 100 101 102 103 104 105 70 27
12 59 60 61 62 63 64 65 66 67 68 69 26
13 14 15 16 17 18 19 20 21 22 23 24 25''' ,}
n=int(input())
print(testcase[n])
```

