2018年8月4日 星期六

数据结构第三次实验报告

3-1.py

```
'''先将范围内的所有素数写出啦，再判断是否是素数对'''
import math
# 判断是不是素数
def isPrime(number):
    bound = math.floor(math.sqrt(number)) + 1
    flag = True
    for i in range(2, bound):
        if number%i == 0:
            flag = False
            break
    return flag

# 求所有素数对
def getAllPrime(n):
    plist = []
    for i in range(2, n+1):
        if isPrime(i):
            plist.append(i)
    for i in range(len(plist)-1):
        if plist[i+1] - plist[i] == 2:
            print(str(plist[i]) + " " + str(plist[i+1]), end=", ")

if __name__ == "__main__":
    n = eval(input())
    getAllPrime(n)
```

```
In [14]: runfile('/Users/zhujun/Downloads/USTC/专业补课/DS/DS-experiment/3-1.py', wdir='/Users/
zhujun/Downloads/USTC/专业补课/DS/DS-experiment')

100
3 5, 5 7, 11 13, 17 19, 29 31, 41 43, 59 61, 71 73,
```

3-2.py

```
'''递归求解，注意排除重复情况'''
import math
def getA(num, st): # num和st分别是分解的两个因子且num>st
    if num == 1: # 递归出口
        return 1
    else:
        cnt = 0
        # 下面的循环就可以消除重复项
        for i in range(st, num+1):
            if num % i == 0:
                cnt += getA(num//i, i)
        return cnt

n = eval(input())
print(getA(n, 2))
```

```
In [14]: runfile('/Users/zhujun/Downloads/USTC/专业补课/DS/DS-experiment/3-1.py', wdir='/Users/
zhujun/Downloads/USTC/专业补课/DS/DS-experiment')

100
3 5, 5 7, 11 13, 17 19, 29 31, 41 43, 59 61, 71 73,
```

3-3.py

```
'''先进行插入排序，再进行奇偶排序'''
string = input()
nlist = string.split()
# 插入排序
for i in range(len(nlist)):
    nlist[i] = eval(nlist[i])
for i in range(1, len(nlist)):
    temp = nlist[i]
    insertindex = i
    for j in range(i):
        if nlist[j] > temp:
            insertindex = j
            break
    if insertindex != i:
        for j in range(i, insertindex, -1):
            nlist[j] = nlist[j-1]
    nlist[insertindex] = temp
print("第一次排序后的队列：", end="")
print(nlist)
# 实现奇偶分流
# 借助队列存储偶数项，先将奇数项排好，再直接顺序排列偶数项
from queue import Queue
queue = Queue(maxsize=len(nlist))
k = 0
for i in range(len(nlist)):
    if nlist[i] % 2 != 0:
        nlist[k] = nlist[i]
        k += 1
    else:
        queue.put(nlist[i])
while not queue.empty():
    nlist[k] = queue.get()
    k += 1
print("第二次排序后的队列：", end="")
print(nlist)
```

```
In [14]: runfile('/Users/zhujun/Downloads/USTC/专业补课/DS/DS-experiment/3-1.py', wdir='/Users/
zhujun/Downloads/USTC/专业补课/DS/DS-experiment')

100
3 5, 5 7, 11 13, 17 19, 29 31, 41 43, 59 61, 71 73,
```

3-4.py

```
'''利用数组存储，全部初始化为1，已经访问过的就置为0'''
# 寻找下一个未出列的人
def findNext(nlist, index):
    i = (index + 1) % len(nlist)
    while True:
        if nlist[i] == 1:
            break
```

```
        else:
            i = (i + 1) % len(nlist)
    return i

size = eval(input("n: "))
# 假设 m < n
m = eval(input("m: "))
nlist = [ 1 for i in range(size)]
norder = [ -1 for i in range(size)]
begin = 0
k = 0
# 寻找第m个人
while size > 0:
    for i in range(m-1): # 第一次寻找只需要调用m-1次函数
        begin = findNext(nlist, begin)
    if size < len(nlist): # 如果是第二次寻找以及以后的情况，则需要调用m次函数
        begin = findNext(nlist, begin)
    norder[k] = begin
    k += 1
    nlist[begin] = 0
    size -= 1
print(norder)
```

```
In [18]: runfile('/Users/zhujun/Downloads/USTC/专业补课/DS/DS-experiment/3-4.py', wdir='/Users/
zhujun/Downloads/USTC/专业补课/DS/DS-experiment')

n: 10

m: 2
[1, 3, 5, 7, 9, 2, 6, 0, 8, 4]
```