HW#10

经济学院 刘安澜 1700015495

本周的作业中的几道题让我更加了解了赋值、浅拷贝和深拷贝的区别，这点在数组题目和循环内变量变量输出的题目中非常重要！

19943:图的拉普拉斯矩阵(matrix)

http://cs101.openjudge.cn/practice/19943/

解题思路：本题是比较中规中矩的二维数组题目，按照题目的意思一步一步写出即可，按照生存游戏提供的矩阵输出模板模版输出。需要注意的是我第一次写的时候直接初始同时赋值 D=A=L 为 0 矩阵，但是这样的赋值命令会使三个变量共用一个存储空间，导致后续的操作中矩阵会同步改变，得到错误的结果。

```
n,m=map(int,input().split())
D=[[0]*n for i in range(n)]
A=[[0]*n for i in range(n)]
L=[[0]*n for i in range(n)]
for i in range(m):
    a,b=map(int,input().split())
    D[a][a]+=1
    D[b][b]+=1
    A[a][b]=A[b][a]=1
for i in range(n):
    for j in range(n):
        L[i][j]=D[i][j]-A[i][j]
for i in range(n):
    print(' '.join([str(x) for x in L[i]]))
```

19942:二维矩阵上的卷积运算 v0.2 (matrix)

http://cs101.openjudge.cn/practice/19942/

解题思路：思路比较简单的二维数组题目，但是代码量较大；其实我的函数设计的有些不方便，这样每一次矩阵移动都要生成一个新的矩阵，但是也顺便学习了不用 numpy 的情况下提取二维数组部分行列的方法。

```
def con(A,B,p,q):
    c=0
    for i in range(p):
        for j in range(q):
            c+=A[i][j]*B[i][j]
    return c
m,n,p,q=map(int,input().split())
A=[]
B=[]
C=[[0]*(n+1-q) for i in range(m+1-p)]
for i in range(m):
    A.append([int(x) for x in input().split()])
for i in range(p):
```

```
    B.append([int(x) for x in input().split()])
for i in range(m+1-p):
    for j in range(n+1-q):
        D=[A[r][j:j+q] for r in range(i,i+p)]
        C[i][j]=con(D,B,p,q)
for i in range(m+1-p):
    print(' '.join([str(x) for x in C[i]]))
```

3532：最大上升子序列和

请用 dp 实现，http://cs101.openjudge.cn/practice/3532/

解题思路：dp 的思路是开一个数组，对应的值是以该点为最大子序列的和的最大值，这样递归时只需要对之前小于该点的子序列和再加上该点的值，对这些值再去最大值即可；其实我觉得本题的思路还有点卡，主要在于之前的 dp 都是有固定次数的递归找最大，但是这道题需要我们对列表进行两次循环，所以刚开始还有些担心超时问题，但是，本题必须完全遍历才能保证取到最大值（而不是我一开始想的只需要找最靠近的小于该点的子序列和）。

```
n=int(input())
l=[int(x) for x in input().split()]
s=l[:]
for i in range(1,n):
    for j in range(i):
        if l[i]>l[j]:
            s[i]=max(s[j]+l[i],s[i])
print(max(s))
```

16528:充实的寒假生活（cs10117 Final Exam）

请用 dp 实现，http://cs101.openjudge.cn/practice/16528/

解题思路：dp 的思路将无效的事件剔除，只留下同样截止时间的最短事件：开一个数组，对应的值是以该点为截止日期的最大开始日期，然后遍历列表找到开始时间晚于上一结束时间的事件就可以。

```
n=int(input())
l=[-1 for i in range(61)]
for i in range(n):
    a,b=map(int,input().split())
    if l[b]<a:
        l[b]=a
s=0
m=-1
for i in range(61):
    if l[i]>m:
        s+=1
        m=i
print(s)
```

//选做：

18211：军备竞赛

greedy/two pointer, http://cs101.openjudge.cn/practice/18211

解题思路：首先将所有的价格排序，从便宜的开始购买，从最贵的开始卖出；正序循环主要体现购买，如果现有的钱数可以购买物品，那么计数+1，如果不够的话就考虑从数额最大的开始卖，但要注意武器差小于 0 的话，程序退出，如果所有武器都使用过一遍，程序也退出。这里的指针主要是记录卖武器的位置，买武器由 for 循环实现。但是，对于本题 for 循环确实没有 while 循环简洁。

```
p=int(input())
l=[int(x) for x in input().split()]
l.sort()
j=1
n=len(l)
s=0
for i in range(n):
    if i+j>n:
        break
    else:
        if p>=l[i]:
            s+=1
            p-=l[i]
        else:
            p+=l[-j]
            s-=1
            if s<0:
                s=0
                break
            else:
                p-=l[i]
                j+=1
                s+=1
print(s)
```

1756：八皇后

dfs, http://cs101.openjudge.cn/practice/1756

解题思路：

这题是参考标准答案模仿写出来的，在看代码的时候画了图写了一下步骤捋顺了逻辑，也大概理解了回溯法。主要是建立一个 flag 代表程序是否退出的指示，函数设立为逐层推导的形式，如果达到了输出条件就将列表输出至已有的列表中。这种顺序的循环也满足八皇后问题需要的顺序输出。

这里需要注意，由于A始终是一个可变的数组，所以如果直接在列表里添加A的话，列表中所有元素都是可变的A；这时必须采取拷贝才能让列表中保存的数组不根据A的变化而变化。

解题代码：

```
def queen(A,cur=0):
    if cur==len(A):
        l.append(A[:])
        return 0
    for i in range(len(A)):
        A[cur],flag=i,True
        for j in range(cur):
            if A[j]==A[cur] or abs(A[j]-A[cur])==cur-j:
                flag=False
                break
        if flag:
            queen(A,cur+1)
l=[]
queen([None]*8)
n=int(input())
for i in range(n):
    print(''.join([str(x+1) for x in l[int(input())-1]]))
```

16532：上机考试（cs10117 Final Exam）

matrix, http://cs101.openjudge.cn/practice/16531/

解题思路：

这是一道思路简单，但是非常繁复，需要注意很多细节的题目。仍然类似于生存游戏，需要对第一个矩阵加一个保护圈（但是需要特别注意，复制代码的时候没有看清题目的要求是只和周围4个判断……）。不同的是这个题目需要对答题情况列表进行比较。但是仍然存在一个细节问题，保护圈圈层如果还是0的话会和本身矩阵中的0混淆，所以我将保护圈圈层设为-1，然后再在答题情况列表的最后加一个空数组（本来加的是0元素，但是存在只有一道题的考试……所以也WA），这样就能避免保护圈层的干扰。

计算优秀率的方式和之前CF上的题差不多，仍然是排序然后取正好卡在40%线外的同学成绩，只有高于这个成绩才能记为优秀，遍历求和即可。

最后，仍然需要注意输出格式，print(s,' '，t)会输出3个空格。

解题代码：

```
def same(l,board,x,y):
    if l[board[x][y]]==l[board[x-1][y]] or
l[board[x][y]]==l[board[x][y-1]] or
l[board[x][y]]==l[board[x][y+1]] \
    or l[board[x][y]]==l[board[x+1][y]]:
        return 1
    else:
        return 0
```

```
m,n=map(int,input().split())
board=[[-1]*(n+2)]+[[-1]+[int(x) for x in input().split()]+[-1]
for j in range(m)]+[[-1]*(n+2)]
l=[]
score=[]
for i in range(m*n):
    l1=[int(x) for x in input().split()]
    l.append(l1)
    score.append(sum(l1))
l.append([])
s=0
for i in range(m):
    for j in range(n):
        s+=same(l,board,i+1,j+1)
score.sort(reverse=True)
t=0
for i in range(m*n*2//5):
    if score[i]>score[m*n*2//5]:
        t+=1
print(s,t,sep=' ')
```