Assignment #8: 田忌赛马来了

2024 fall, Complied by 吕金浩，物理学院

1. 题目

12558: 岛屿周长

matices, http://cs101.openjudge.cn/practice/12558/

思路：每个非零方块贡献的周长等于 4 减去其相邻非零方块数目

代码：

```
row,column=map(int,input().split())
matrix=[[0]*(column+2)]
for _ in range(row):
    matrix.append([0]+[int(x) for x in input().split()]+[0])
matrix.append([0]*(column+2))
perimeter=0
for i in range(1,row+1):
    for j in range(1,column+1):
        if matrix[i][j]:
            perimeter+=4-matrix[i-1][j]-matrix[i+1][j]-matrix[i][j-1]-matrix[i][j+1]
print(perimeter)
```

LeetCode54.螺旋矩阵

matrice, https://leetcode.cn/problems/spiral-matrix/

与 OJ 这个题目一样的 18106: 螺旋矩阵，http://cs101.openjudge.cn/practice/18106

思路：碰到边界/碰到用过的方格就转 90 度

代码：

```
class Solution:
    def spiralOrder(self, matrix: List[List[int]]) -> List[int]:
        if not matrix:
            return []
        row=len(matrix)
        column=len(matrix[0])
        step=0
        x,y=0,0
```

```
        moves=[(0,1),(1,0),(0,-1),(-1,0)]
        move=0
        ans=[matrix[0][0]]
        written=[[False]*column for _ in range(row)]
        written[0][0]=True
        while step<row*column-1:
            step+=1
            written[x][y]=True
            if x+moves[move][0] not in range(row) or y+moves[move][1] not in range(column)
or written[x+moves[move][0]][y+moves[move][1]]:
                move=(move+1)%4
            x,y=x+moves[move][0],y+moves[move][1]
            ans.append(matrix[x][y])
        return ans
```

04133:垃圾炸弹

matrices, http://cs101.openjudge.cn/practice/04133/

代码:

```
d=int(input())

all_rubbish=[]
all_locations=[]
n=int(input())
most,most_value=0,0
locations_set=set()
for _ in range(n):
    x,y,num =map(int,input().split())
    all_rubbish.append([x,y,num])
    all_locations.append((x,y))
    for i in range(x - d, x + d + 1):
        for j in range(y - d, y + d + 1):
            if 0<=i<=1024 and 0<=j<=1024:
                locations_set.add((i, j))
```

```
for x,y in locations_set:

    current=0
    for k in range(n):
        x0,y0,num0=map(int,all_rubbish[k])
        if x in range(x0-d,x0+d+1) and y in range(y0-d,y0+d+1):
            current+=num0
    if current==most_value:
        most+=1
    if current>most_value:
        most=1
        most_value=current

print('{} {}'.format(most,most_value))
```

LeetCode376.摆动序列

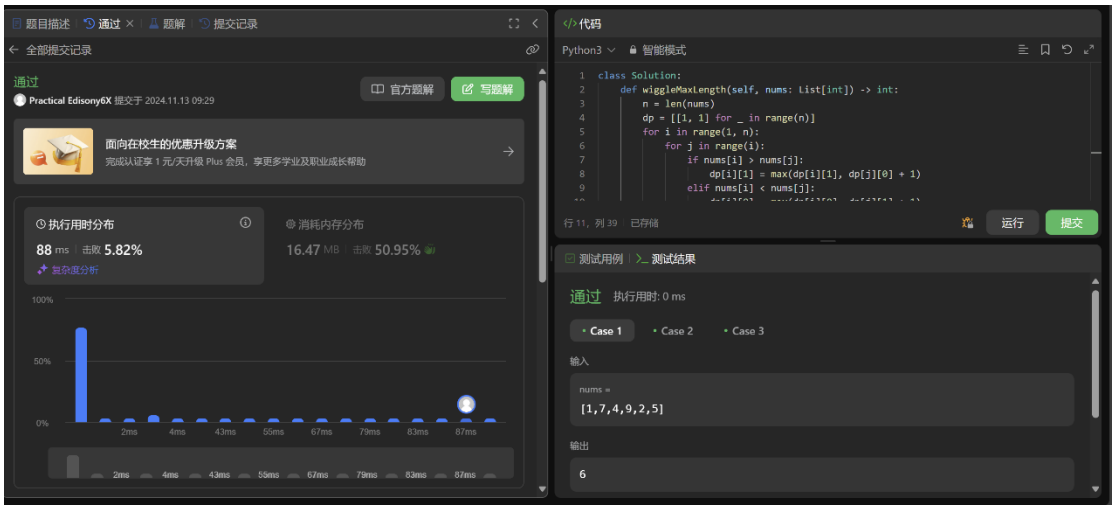
greedy, dp, https://leetcode.cn/problems/wiggle-subsequence/

与 OJ 这个题目一样的，26976:摆动序列, http://cs101.openjudge.cn/routine/26976/

代码:

```
class Solution:
    def wiggleMaxLength(self, nums: List[int]) -> int:
        n = len(nums)
        dp = [[1, 1] for _ in range(n)]
        for i in range(1, n):
            for j in range(i):
                if nums[i] > nums[j]:
```

```
                dp[i][1] = max(dp[i][1], dp[j][0] + 1)
            elif nums[i] < nums[j]:
                dp[i][0] = max(dp[i][0], dp[j][1] + 1)
    return max(max(x) for x in dp)
```



CF455A: Boredom

dp, 1500, https://codeforces.com/contest/455/problem/A

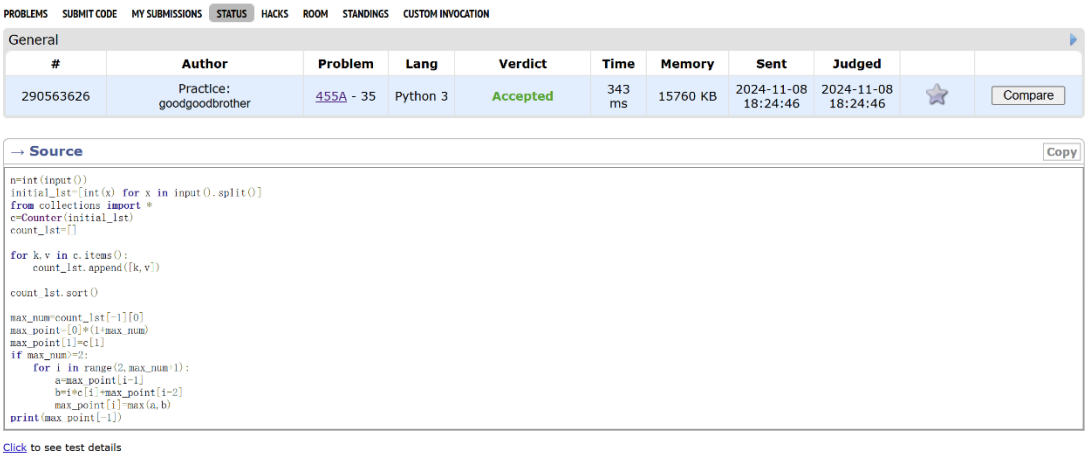
代码:

```
n=int(input())
initial_lst=[int(x) for x in input().split()]
from collections import *
c=Counter(initial_lst)
count_lst=[]

for k,v in c.items():
    count_lst.append([k,v])

count_lst.sort()

max_num=count_lst[-1][0]
max_point=[0]*(1+max_num)
max_point[1]=c[1]
if max_num>=2:
    for i in range(2,max_num+1):
        a=max_point[i-1]
        b=i*c[i]+max_point[i-2]
        max_point[i]=max(a,b)
print(max_point[-1])
```



02287: Tian Ji -- The Horse Racing

greedy, dfs http://cs101.openjudge.cn/practice/02287

思路：看了一个 dp 的做法才写出来（其实脑袋里有和那一 dp 做法差不多的想法，但不敢试），自己用递归写了一遍。

lru_cache 的 maxsize 的设置貌似充分体现了空间换时间的思想，经过尝试，设置成 None 会 MLE，（仿照另一个递归做法）设置成 1024 会 TLE，而设置成 2**11 到 2**16 均能 AC。

代码：

```
import sys
sys.setrecursionlimit(1<<30)
from functools import lru_cache
while True:
    n=int(input())
    if not n:
        break
    tian=[int(x) for x in input().split()]
    king=[int(x) for x in input().split()]
    tian.sort(reverse=True)
    king.sort(reverse=True)
    #前 i,前 j
    @lru_cache(maxsize=2**11)
    def max_weight(i,j):
        if i*j==0:
            return 0
        else:
            if tian[i-1]>king[j-1]:
                return max(max_weight(i,j-1),max_weight(i-1,j),max_weight(i-1,j-1)+2)
            elif tian[i-1]==king[j-1]:
                return max(max_weight(i,j-1),max_weight(i-1,j),max_weight(i-1,j-1)+1)
            else:
                return max(max_weight(i,j-1),max_weight(i-1,j),max_weight(i-1,j-1))
    print(200*(max_weight(n,n)-n))
```

#47140865提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import sys
sys.setrecursionlimit(1<<30)
from functools import lru_cache
while True:
    n=int(input())
    if not n:
        break
    tian=[int(x) for x in input().split()]
    king=[int(x) for x in input().split()]
    tian.sort(reverse=True)
    king.sort(reverse=True)
    #前i,前j
    @lru_cache(maxsize=2**11)
    def max_weight(i,j):
        if i*j==0:
            return 0
        else:
            if tian[i-1]>king[j-1]:
                return max(max_weight(i,j-1),max_weight(i-1,j),max_weigh
            elif tian[i-1]==king[j-1]:
                return max(max_weight(i,j-1),max_weight(i-1,j),max_weigh
            else:
                return max(max_weight(i,j-1),max_weight(i-1,j),max_weigh
    print(200*(max_weight(n,n)-n))
```

基本信息

#: 47140865
题目: 02287
提交人: 2400011490
内存: 7068kB
时间: 40957ms
语言: Python3
提交时间: 2024-11-13 19:03:10

English 帮助 关于

2. 学习总结和收获

感觉 dp 和递归做多了之后思路都比较公式化，基本都是二十分钟左右能做出来的程度（在没有令人难以注意到的坑的情况下），比如最近的每日选做。另外希望自己接下来能学到一些 dfs 和 bfs 的知识。