**Assignment #8: 田忌赛马来了**

2024 fall, Complied by 吕金浩，物理学院

**1. 题目**

**12558: 岛屿周⻓**

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):

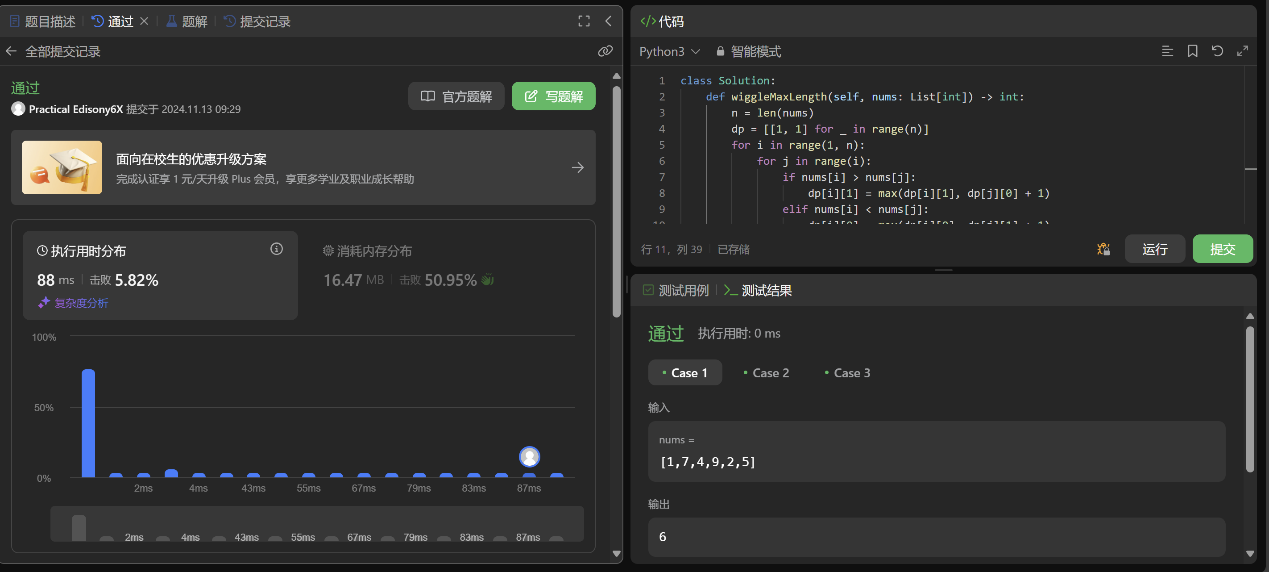
                if nums[i] > nums[j]:

                    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)

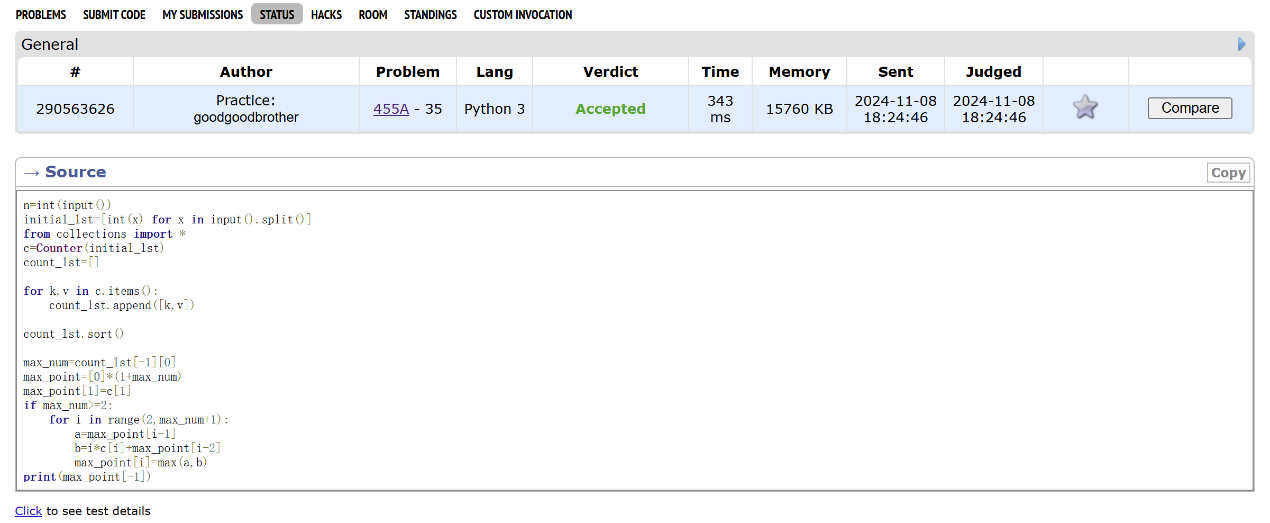


**CF455A: Boredom**

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))



**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。