HW#12

经济学院 刘安澜 1700015495

校门外的树， http://cs101.openjudge.cn/practice/1810

解题思路:只需要将树的列表在经过的路段改成0，最后统计存活的树即可。

1. ﻿﻿ L,M=map(int,input().split())
2. l=[1]\*(L+1)
3. **for** i **in** range(M):
4. a,b=map(int,input().split())
5. **for** j **in** range(a,b+1):
6. l[j]=0
7. **print**(sum(l))

约瑟夫问题， http://cs101.openjudge.cn/practice/1748

解题思路：建立参与者列表，存活记为1，如果出局状态变为0；每一次只对存活的参与者计数，然后对整个活动的存活者计数；由于需要考虑到循环的情况，所以每一次换人的时候采取i=（i+1）%n来构造循环的遍历。

1. n,m=map(int,input().split())
2. **while** m+n!=0:
3. l=[1]\*n
4. i=0
5. j=m
6. s=n
7. **while** s!=1:
8. **if** l[i]==1:
9. j-=1
10. **if** j==0:
11. l[i]=0
12. s-=1
13. j=m
14. i+=1
15. i=i%n
16. **else**:
17. i+=1
18. i=i%n
19. **print**(l.index(1)+1)
20. n,m=map(int,input().split())

brute force

假币问题， http://cs101.openjudge.cn/practice/1694

解题思路：这道题一开始我采取正向解法（没有理解brute force的内涵），但是总会碰到无法全部判断的情况，所以是wrong answer。这道题正常的思路应该是对12枚硬币是否是假币的24种情况遍历一遍，如果满足称重的情况，则输出结果。我用字典的写法有些复杂了，其实采取‘ABCD’字符串形式会更加简单。

1. #CS101 1694
2. **def** balance(l1,l2,l3):
3. s1=s2=0
4. **for** i **in** l1:
5. s1+=d[i]
6. **for** i **in** l2:
7. s2+=d[i]
8. **if** (s1>s2 **and** l3=='up') **or** (s1<s2 **and** l3=='down') **or** (s1==s2 **and** l3=='even'):
9. **return** True
10. **else**:
11. **return** False
13. n=int(input())
14. **for** i **in** range(n):
15. l=[input().split() **for** i **in** range(3)]
16. d={'A':0,'B':0,'C':0,'D':0,'E':0,'F':0,'G':0,'H':0,'I':0,'J':0,'K':0,'L':0}
17. status=[-1,1]
18. word=['light.','heavy.']
19. **for** j **in** range(24):
20. d[list(d.keys())[j//2]]=status[j%2]
21. **for** k **in** range(3):
22. **if** balance(l[k][0],l[k][1],l[k][2])==False:
23. **break**
24. **if** k==2 **and** balance(l[k][0],l[k][1],l[k][2])==True:
25. ans=j
26. d={'A':0,'B':0,'C':0,'D':0,'E':0,'F':0,'G':0,'H':0,'I':0,'J':0,'K':0,'L':0}
27. **print**(list(d.keys())[ans//2]+' is the counterfeit coin and it is '+word[ans%2])

greedy  
圣诞老人的礼物-Santa Clau’s Gifts， http://cs101.openjudge.cn/practice/7813

解题思路：对礼物的平均价值进行倒序排列，然后依次取用即可。一开始不AC的关键在于如果圣诞老人带的货物足够多，把全部礼物带走，就不会出现小数。所以一定要统一在最后输出一位小数的格式。

1. n,w=map(int,input().split())
2. l=[[int(x) **for** x **in** input().split()] **for** i **in** range(n)]
3. l.sort(key=**lambda** x: -x[0]/x[1])
4. ans=0
5. **for** i **in** range(n):
6. **if** w>l[i][1]:
7. ans+=l[i][0]
8. w-=l[i][1]
9. **else**:
10. ans+=w\*(l[i][0]/l[i][1])
11. **break**
12. **print**('%.1f' % ans)

binary search  
方程求解， http://cs101.openjudge.cn/practice/12065

解题思路：先找到一个大概区间（0-10），然后在区间内进行循环二分查找迭代，最终控制精度在10-10以内，输出控制在9位小数即可。

1. **def** f(x):
2. **return**(x\*\*3-5\*(x\*\*2)+10\*x-80)
3. i=0
4. j=10
5. **while** j-i>0.1\*\*10:
6. **if** f((i+j)/2)>0:
7. j=(i+j)/2
8. **else**:
9. i=(i+j)/2
10. **print**('%.9f' % i)

matrices

螺旋矩阵，<http://cs101.openjudge.cn/practice/18106/>

解题思路：我采取一种几何化的想法，就按照右-下-左-上的循环顺序写入数据。定义一个方向向量，如果写到头（即下一个位置不为0）就更换方向（这里的循环设置也参考之前猴子称大王问题的循环变量设计）。外加一层不为0的保护圈作为界的标志。

1. n=int(input())
2. l=[[-1]\*(n+2)]+[[-1]+[0]\*n+[-1] **for** j **in** range(n)]+[[-1]\*(n+2)]
3. turn=[[0,1],[1,0],[0,-1],[-1,0]]
4. i=1
5. j=1
6. s=1
7. t=0
8. **while** s<=n\*\*2:
9. l[i][j]=s
10. s+=1
11. **if** l[i+turn[t][0]][j+turn[t][1]]!=0:
12. t=(t+1)%4
13. i+=turn[t][0]
14. j+=turn[t][1]
15. **for** i **in** range(1,n+1):
16. **print**(' '.join([str(x) **for** x **in** l[i][1:n+1]]))

//选做：

熄灯问题，

brute force, <http://cs101.openjudge.cn/practice/1813>

解题思路：一开始我一直想使用dfs的方法，但是因为dfs方向比较散，操作起来确实复杂（还是没有理解本题brute force的思想）。最终采取的思路仍然是对第一行遍历（64组），这样后面的行只需要根据第一行的情况进行改变即可。注意需要第一行遍历的循环最里（而不是第一层）对原矩阵进行拷贝，否则输出的结果不正确。

1. **import** copy
2. dx=[0,0,0,-1,1]
3. dy=[0,1,-1,0,0]
4. mat=[[2]\*(8)]+[[2]+[int(x) **for** x **in** input().split()]+[2] **for** i **in** range(5)]+[[2]\*(8)]
5. output=[[0]\*(6) **for** i **in** range(5)]
7. **def** turn(x,y,k):
8. **if** k==1:
9. **for** i **in** range(5):
10. **if** board[x+dx[i]][y+dy[i]]==0:
11. board[x+dx[i]][y+dy[i]]=1
12. **elif** board[x+dx[i]][y+dy[i]]==1:
13. board[x+dx[i]][y+dy[i]]=0
14. **return**
15. **if** k==0:
16. **return**
18. **for** a **in** range(2):
19. **for** b **in** range(2):
20. **for** c **in** range(2):
21. **for** d **in** range(2):
22. **for** e **in** range(2):
23. **for** f **in** range(2):
24. board=copy.deepcopy(mat)
25. output[0]=[a,b,c,d,e,f]
26. **for** i **in** range(6):
27. turn(1,i+1,output[0][i])
28. **for** i **in** range(2,6):
29. **for** j **in** range(1,7):
30. **if** board[i-1][j]==0:
31. output[i-1][j-1]=0
32. **else**:
33. output[i-1][j-1]=1
34. turn(i,j,1)
35. **if** sum(board[5][1:7])==0:
36. **for** i **in** range(5):
37. **print**(' '.join([str(x) **for** x **in** output[i]]))

Tian Ji -The Horse Racing田忌赛马，

greedy, http://cs101.openjudge.cn/practice/1289

解题思路：我最初的思路是双指针的想法，找到比田忌最快的马稍慢的国王的马就赢一场，以此类推。但是这种想法在处理相等速度的竞赛中出现问题，因为如果优先让相等的马竞技，可能后面本来可赢的比赛就也只能平局或者输掉，因此在有多匹相同速度的马时答案就会出现问题（例如两人的马都是14，14，13，13，12，12，优先处理相等结果为0，但最优应该是净胜2局）。所以最后还是只能逐步遍历筛选剔除已有的最优配对组合，筛选的依据为，尽量让田忌的弱马同对方的强马竞技，让田忌略优的马与对方竞技：①如果田忌最快的马优于国外的马，那么win+1，将这两匹马去掉；②如果田忌最慢的马优于国外的马，那么win+1，将这两匹马去掉；③如果田忌最弱的马不强于对方最弱的马，，将田忌最弱的马和对方最强的马去掉一定是最优（包含平局的情况）；如果田忌最弱的马严格弱于对方最强的马，那么必然会输一场，win-1（考虑了平局的情况）。循环这个策略直到双方剩余马为0即可。

解题代码：

1. n=int(input())
2. **while** n!=0:
3. l1=[int(x) **for** x **in** input().split()]
4. l1.sort(reverse=True)
5. l2=[int(x) **for** x **in** input().split()]
6. l2.sort(reverse=True)
7. ans=0
8. **for** \_ **in** range(n):
9. **if** l2[0]<l1[0]:
10. ans+=1
11. **del** l2[0]
12. **del** l1[0]
13. **else**:
14. **if** l2[-1]<l1[-1]:
15. ans+=1
16. **del** l2[-1]
17. **del** l1[-1]
18. **else**:
19. **if** l2[0]>l1[-1]:
20. ans-=1
21. **del** l2[0]
22. **del** l1[-1]
23. **print**(200\*ans)
24. n=int(input())