# 復習済み

A-E

# A Rearranging ABC

正解

s = set(list(input()))

ans = set(["A", "B", "C"])

if ans == s:

print("Yes")

else:

print("No")

# B Avoid Rook Attack

正解

ans = [[0]\*8 for \_ in range(8)]

for i in range(8):

s = input()

for j in range(8):

if s[j] == "#":

for k in range(8):

ans[k][j] = 1

ans[i][k] = 1

print(64-sum(sum(ans[i]) for i in range(8)))

# C Avoid Knight Attack

正解

n, m = map(int, input().split())

ans = set()

dydx = [[2,1],[1, 2],[-1,2],[-2,1],[-2,-1],[-1,-2],[1,-2],[2,-1]]

for i in range(m):

a, b = map(int, input().split())

if (a-1,b-1) not in ans:

ans.add((a-1,b-1))

for d in dydx:

if 0<=a-1+d[0]<=n-1 and 0<=b-1+d[1]<=n-1:

if (a-1+d[0],b-1+d[1]) not in ans:

ans.add((a-1+d[0],b-1+d[1]))

print(n\*n-len(ans))

# D Many Segments 2

不正解

しゃくとり法の応用的考え方ができる。

LかRを固定するが、解説ではRを固定している。

以下はLを固定したもの。

## 正解コード

n, m = map(int, input().split())

minr = [m]\*(m+1)

ans = 0

for \_ in range(n):

L, R = map(int, input().split())

minr[L] = min(minr[L], R-1)

for i in range(m, 1, -1):

minr[i-1] = min(minr[i-1], minr[i])

for l in range(m, 0, -1):

ans += minr[l]-l+1

print(ans)

## 正解コード(復習)

N, M = map(int, input().split())

ans = 0

maxl = [0]\*(M+1)

for \_ in range(N):

L, R = map(int, input().split())

maxl[R] = max(maxl[R], L)

for i in range(M):

maxl[i+1] = max(maxl[i], maxl[i+1])

for r in range(M, 0, -1):

ans += r-maxl[r]

print(ans)

# E Permute K times 2

不正解

ダブリング

## 正解コード

n, k = map(int, input().split())

P = list(map(int, input().split()))

P = [p-1 for p in P]

ans = [None]\*n

used = [False]\*n

for i in range(n):

if used[i] == False:

cycle = [i]

used[i] = True

nxt = P[i]

while nxt != i:

used[nxt] = True

cycle.append(nxt)

nxt = P[nxt]

cycle\_len = len(cycle)

for i, node in enumerate(cycle):

dist = pow(2, k, cycle\_len)

ans[node] = cycle[(i+dist)%cycle\_len]

print(\*[a+1 for a in ans])

## 正解コード(復習)

N, K = map(int, input().split())

P = list(map(lambda x: int(x)-1, input().split()))

connected\_components = []

visit = [False]\*N

for i in range(N):

if visit[i]:

continue

visit[i] = True

pre = i

com = [i]

while True:

nxt = P[pre]

if visit[nxt]:

break

else:

visit[nxt] = True

com.append(nxt)

pre = nxt

connected\_components.append(com)

def power(base, K, length):

if K == 0:

return 1 % length

base2 = base\*\*2

if K % 2 == 0:

return power(base2%length, K//2, length)

else:

return (base%length)\*power(base2%length, K//2, length)

ans = [None]\*N

for com in connected\_components:

length = len(com)

di = power(2, K, length)

for i in range(length):

ans[com[i]] = com[(i+di)%length] + 1

print(\*ans)

# F Avoid Queen Attack

不正解

十字とクロスのマスの数を引いてから、交点の重複を修正する。

## 正解コード

from collections import defaultdict

n, m = map(int, input().split())

ans = n\*n

I = set() # i = a

J = set() # j = b

ImJ = set() # i-j = a-b

IpJ = set() # i+j = a+b

for \_ in range(m):

a, b = map(int, input().split())

I.add(a)

J.add(b)

ImJ.add(a-b)

IpJ.add(a+b)

#ラインのマス数(交点重複)

ans -= n\*(len(I)+len(J))

for imj in ImJ:

ans -= n - abs(imj)

for ipj in IpJ:

ans -= n - abs(ipj-(n+1))

#交点抽出

insec = defaultdict(int)

for i in I:

for j in J:

insec[(i,j)] += 1

for i in I:

for imj in ImJ:

if 1<=i-imj<=n:

insec[(i,i-imj)] += 1

for i in I:

for ipj in IpJ:

if 1<=ipj-i<=n:

insec[(i,ipj-i)] += 1

for j in J:

for imj in ImJ:

if 1<=j+imj<=n:

insec[(j+imj,j)] += 1

for j in J:

for ipj in IpJ:

if 1<=ipj-j<=n:

insec[(ipj-j, j)] += 1

for imj in ImJ:

for ipj in IpJ:

if (imj+ipj)%2 == 0:

if 1<=(imj+ipj)//2<=n and 1<=(ipj-imj)//2<=n:

insec[((imj+ipj)//2, (ipj-imj)//2)] += 1

#重複分修正

for v in insec.values():

if v == 1:

ans += 1

elif v == 3:

ans += 2

else:

ans += 3

print(ans)

# G Edit to Match

不正解