# 復習済み

A-E

# A Equally

正解

# B Santa Claus 1

正解

# C Illuminate Buildings

正解

色々TLEして正解コードを出した。この問題のコードは基本的にHの高さが全て同じ時最悪計算量になる。その計算量を以下に示す。

(正解コード本番)

二重ループで、あるH[i]を見た時、それまで見た0<=j<iのH[j]と比較するので

O(N^2)

(正解コード復習)

これは二重ループがO(N^2)で、ループのたびに間隔dでのチェックがあり、その最悪計算量はO(N)なのでO(N^3)くらい。

それなのに計算時間が復習の方が早いのは本番のコードではdefaultdictを使っており、新しくメモリ領域を作るのに時間がかかったからと思われる。そこでCを配列にしたら復習と同じような時間になった。しかしまだ復習の方が速かった。

(不正解コード)

不正解コードは復習も本番も同じアルゴリズムで、高さによってHのインデックスを保存して、それぞれの高さごとに連続数を計算する方法である。このアルゴリズムでは同じ高さのものが多くあればあるほど時間がかかる。例えば全てが2だった時、

H[2]にはN個のインデックスが保持される。２つの組み合わせを全て検査するので、 組が最大であり、その組(l, r)ごとにrより後のものを一個ずつ検査していく。よって最悪N個計算し得るのでO(N^2)×O(N)=O(N^3)

## 正解コード(本番 1549ms)

from collections import defaultdict

N = int(input())

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

C = [defaultdict(int) for \_ in range(N)]

ans = 1

for i in range(N):

for j in range(i):

if H[i] == H[j]:

d = i-j

C[i][d] = C[j][d] + 1

ans = max(ans, C[i][d]+1)

print(ans)

## 正解コード(復習時 158ms)

N = int(input())

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

ans = 1

for i in range(N-1):

for j in range(i+1, N):

if H[i] == H[j]:

d = j-i

count = 2

k = j+d

while k < N and H[k] == H[i]:

k += d

count += 1

ans = max(ans, count)

print(ans)

## 不正解コード(本番)

N = int(input())

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

C = [[] for \_ in range(3001)]

ans = 1

for i, h in enumerate(H):

C[h].append(i+1)

for i in range(1, 3001):

if C[i] == [] or len(C[i])==1:

continue

len\_ci = len(C[i])

for l in range(len\_ci-1):

for r in range(l+1, len\_ci):

d = C[i][r]-C[i][l]

pre = r

count = 2

for j in range(r+1, len\_ci):

if C[i][j]-C[i][pre] == d:

count += 1

pre = j

elif C[i][j] - C[i][pre] > d:

break

else:

continue

ans = max(ans, count)

print(ans)

## 不正解コード(復習時)

from collections import defaultdict

N = int(input())

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

ans = 1

building = [[] for \_ in range(3001)]

for i in range(N):

building[H[i]].append(i)

for v in building:

lb = len(v)

if lb == 1:

continue

if lb == 2:

ans = max(ans, 2)

for i in range(lb-2):

for j in range(i+1, lb-1):

d = v[j]-v[i]

nxt = v[j] + d

count = 2

max\_id = v[-1]

while nxt <= max\_id and nxt in v:

count += 1

nxt += d

ans = max(ans, count)

print(ans)

# D Santa Claus 2

不正解

正解コードは頑張って自分が解いたコード。まず家はxとyの値ごとに分類する。またそれぞれの家のidをつける。その後、M回移動方法を読み取って処理する。移動するときはxかyが固定され、移動が線分をして現れる。そのため、その線分に含まれる最初の家に1をつけ、ギリギリ範囲外になる家に-1をつけ、最終的にいもす法で線分に含まれていれば1以上という状況を作り出す。

解説より速かった。解説はsortedsetを使っている。同じようにxとyで家を管理して、移動の範囲にある家を削除するようにしている。

## 正解コード

from collections import defaultdict

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

house\_x = defaultdict(list)

house\_y = defaultdict(list)

house\_flag = [0]\*N

for i in range(N):

x, y = map(int, input().split())

house\_x[x].append([y, i, 0])

house\_y[y].append([x, i, 0])

for key in house\_x:

house\_x[key].sort()

for key in house\_y:

house\_y[key].sort()

def bs(low, high, L):

if L == []:

return

n = len(L)

l = 0

r = n

while l < r:

mid = (l+r)//2

if L[mid][0] < low:

l = mid+1

elif L[mid][0] == low:

l = mid

break

else:

r = mid

if l == n:

return

L[l][2] += 1

l = 0

r = n

while l < r:

mid = (l+r)//2

if L[mid][0] < high:

l = mid+1

elif L[mid][0] == high:

l = mid+1

break

else:

r = mid

if l != n:

L[l][2] -= 1

for j in range(M):

d, c = input().split()

c = int(c)

if d == "U":

nxt\_y = ny+c

if nx in house\_x:

bs(ny, nxt\_y, house\_x[nx])

ny = nxt\_y

elif d == "D":

nxt\_y = ny-c

if nx in house\_x:

bs(nxt\_y, ny, house\_x[nx])

ny = nxt\_y

elif d == "L":

nxt\_x = nx-c

if ny in house\_y:

bs(nxt\_x, nx, house\_y[ny])

nx = nxt\_x

elif d == "R":

nxt\_x = nx+c

if ny in house\_y:

bs(nx, nxt\_x, house\_y[ny])

nx = nxt\_x

for key in house\_x:

s = 0

for \_, id, num in house\_x[key]:

s += num

if s > 0:

house\_flag[id] = 1

for key in house\_y:

s = 0

for \_, id, num in house\_y[key]:

s += num

if s > 0:

house\_flag[id] = 1

print(nx, ny, sum(house\_flag))

# E Snowflake Tree

解いてない

書いたら１発ACした。各頂点をユ木の頂点とした時、作れる最大のユ木を求め、最大のユ木からの頂点数をNから引いた数が答えである。ユ木の頂点につながるノードを次数ごとに数え、どのくらいのy個の葉をもったノードを選ぶかによって色々なユ木を作ってそのノード数を数える。

## 正解コード

from collections import defaultdict

N = int(input())

adjacent = [[] for i in range(N)]

degree = [0]\*N

ans = 0

for \_ in range(N-1):

u, v = map(int, input().split())

u -= 1

v -= 1

adjacent[u].append(v)

adjacent[v].append(u)

degree[u] += 1

degree[v] += 1

for i in range(N):

tmp = defaultdict(int)

for node in adjacent[i]:

tmp[degree[node]] += 1

tmp = sorted([[k, v] for k, v in tmp.items()], reverse = True)

s = 0

for k, v in tmp:

s += v

ans = max(ans, k\*s)

print(N-ans-1)

# F Visible Buildings

解いてない

# G Counting Buildings

解いてない