# 復習済み

A-E

# A Humidifier 1

正解

# B Humidifier 2

正解

# C Humidifier 3

正解

復習した時のコードの方が遅くなってしまった(復習時 1586ms, 本番 279ms)。復習時のコードをヒープではなくキューで管理したら速くなった(484ms, 258ms)。これはグリード上の最短経路でヒープを使うとノードを入れるのにO(NlogN)かかるのに対して、キューならO(1)しかかからないからである。

## 正解コード

from collections import deque

def count\_humidified\_cells(H, W, D, grid):

directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

visited = [[-1] \* W for \_ in range(H)]

queue = deque()

for i in range(H):

for j in range(W):

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

queue.append((i, j, 0))

visited[i][j] = 0

humidified\_count = 0

while queue:

x, y, dist = queue.popleft()

if grid[x][y] == '.' or grid[x][y] == 'H':

humidified\_count += 1

if dist == D:

continue

for dx, dy in directions:

nx, ny = x + dx, y + dy

if 0 <= nx < H and 0 <= ny < W and visited[nx][ny] == -1:

if grid[nx][ny] != '#':

visited[nx][ny] = dist + 1

queue.append((nx, ny, dist + 1))

return humidified\_count

H, W, D = map(int, input().split())

grid = [input().strip() for \_ in range(H)]

print(count\_humidified\_cells(H, W, D, grid))

## 復習時コード

from heapq import heapify, heappop, heappush

H, W, D = map(int, input().split())

S = [list(input()) for \_ in range(H)]

check = [[float("inf")]\*W for \_ in range(H)]

q = []

heapify(q)

for i in range(H):

for j in range(W):

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

heappush(q, (0, i, j))

check[i][j] = 0

while q:

cost, y, x = heappop(q)

if cost == D:

break

nc = cost+1

for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:

nx = x+dx

ny = y+dy

if 0<=nx<W and 0<=ny<H and S[ny][nx] == "." and check[ny][nx] == float("inf") and cost < D:

check[ny][nx] = nc

q.append((nc, ny, nx))

ans = 0

for i in range(H):

for j in range(W):

if check[i][j] != float("inf"):

ans += 1

print(ans)

# D 9 Divisors

解いてない

１発でACして95msだった。考え方としてはN は a^8という形か b^2 c^2という形しか取れない(a, b, cは素数)。そのため２つのパターンで考えられるaの個数と(b, c)の個数を数え上げる。

## 正解コード

def prime\_list(n):

rst = []

sieve = [True]\*(n+1)

for i in range(2, n+1):

if sieve[i]:

rst.append(i)

for j in range(2\*i, n+1, i):

sieve[j] = False

return rst

N = int(input())

prime = prime\_list(10\*\*6)

ans = 0

p = len(prime)

# N = a\*\*8

a = int(N\*\*(1/8))

ok = -1

ng = p

while ok + 1 < ng:

mid = (ok+ng)//2

if prime[mid] <= a:

ok = mid

else:

ng = mid

ans += ok+1

# N = b\*\*2 \* c\*\*2 st. b < c

l = 0

while True:

c = int((N/(prime[l]\*\*2))\*\*(1/2))

ok = -1

ng = p

while ok + 1 < ng:

mid = (ok+ng)//2

if prime[mid] <= c:

ok = mid

else:

ng = mid

if l < ok:

ans += ok-l

l += 1

else:

break

print(ans)

# E Sum of Max Matching

解いてない

解説と同じようなコード。しかし他の正解を見たら遅い方だったので改善した。

## 正解コード(1150ms)

class Unionfind:

def \_\_init\_\_(self, n, A, B):

self.L = [[-1, 0, 0] for \_ in range(n)]

self.ans = 0

for a in A:

self.L[a-1][1] += 1

for b in B:

self.L[b-1][2] += 1

def find(self, x):

if isinstance(self.L[x], list):

return x

else:

self.L[x] = self.find(self.L[x])

return self.L[x]

def union(self, a, b, w):

ra = self.find(a)

rb = self.find(b)

if ra == rb:

return

tmp = [self.L[ra][1]+self.L[rb][1], self.L[ra][2]+self.L[rb][2]]

num = min(tmp)

self.ans += num\*w

if self.L[ra][0] < self.L[rb][0]:

self.L[rb] = ra

self.L[ra][1] = tmp[0]-num

self.L[ra][2] = tmp[1]-num

elif self.L[ra][0] > self.L[rb][0]:

self.L[ra] = rb

self.L[rb][1] = tmp[0]-num

self.L[rb][2] = tmp[1]-num

else:

self.L[rb] = ra

self.L[ra][1] = tmp[0]-num

self.L[ra][2] = tmp[1]-num

self.L[ra][0] -= 1

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

edge = []

for i in range(M):

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

edge.append((w, u-1, v-1))

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

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

uf = Unionfind(N, A, B)

edge.sort()

for w, u, v in edge:

uf.union(u, v, w)

print(uf.ans)

## 改善コード

def find(x):

if L[x] < 0:

return x

else:

L[x] = find(L[x])

return L[x]

def union(u, v, w):

ra = find(u)

rb = find(v)

if ra == rb:

return

if L[ra] > L[rb]:

ra, rb = rb, ra

L[ra] += L[rb]

L[rb] = ra

a[ra] += a[rb]

b[ra] += b[rb]

c = min(a[ra], b[ra])

global ans

ans += c\*w

a[ra] -= c

b[ra] -= c

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

edge = []

for \_ in range(M):

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

edge.append((w, u, v))

edge.sort()

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

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

L = [-1]\*(N+1)

ans = 0

a = [0]\*(N+1)

b = [0]\*(N+1)

for i in A:

a[i] += 1

for i in B:

b[i] += 1

for w, u, v in edge:

union(u, v, w)

print(ans)

# F Diversity

解いてない

# G Bar Cover

解いてない