# 復習済み

A-F

# A Triple Four

正解

# B Card Pile

正解

# C Buy Balls

正解

場合分けが多く分かりづらい。解説のように累積和を使って解く。

## 正解コード

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

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

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

B.sort(reverse = True)

W.sort(reverse = True)

ans = 0

i = 0

j = 0

while i < N and j < M and B[i]>=0 and W[j] >= 0:

ans += B[i]+W[j]

i += 1

j += 1

if i == N:

print(ans)

exit()

if j == M:

while i < N and B[i] > 0:

ans += B[i]

i += 1

print(ans)

exit()

if B[i] < 0:

while i < N and j < M and B[i]+W[j] > 0:

ans += B[i]+W[j]

i += 1

j += 1

print(ans)

elif W[j] < 0:

while i < N and B[i] > 0:

ans += B[i]

i += 1

print(ans)

## 改善コード

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

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

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

B.sort(reverse = True)

W.sort(reverse = True)

if N > M:

W = W+[0]\*(N-M)

B\_sum = [0]\*(N+1)

W\_sum = [0]\*(N+1)

W\_max = [0]\*(N+1)

for i in range(N):

B\_sum[i+1] += B[i]+B\_sum[i]

for i in range(N):

W\_sum[i+1] += W[i]+W\_sum[i]

W\_max[i+1] = max(W\_sum[i+1], W\_max[i])

ans = 0

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

ans = max(ans, B\_sum[i]+W\_max[i])

print(ans)

# D Minimum XOR Path

正解

自分はキューを使って幅優先探索(bfs)を行った。これでも行けたが、メモリを使わず王道的な解き方はdfsである。

## 正解コード

def solve():

from collections import deque

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

ad = [[] for \_ in range(N)]

for \_ in range(M):

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

u -= 1

v -= 1

ad[u].append((v, w))

ad[v].append((u, w))

q = deque([[0, set([0]), 0]])

ans = set()

while q:

pre, visit, cost = q.popleft()

for nxt, w in ad[pre]:

if nxt not in visit:

if nxt == N-1:

ans.add(cost^w)

continue

q.append([nxt, visit|set([nxt]), cost^w])

m = float("inf")

for a in ans:

m = min(m, a)

print(m)

solve()

## 改善コード

def dfs(now, cost):

visit[now] = True

for nxt, w in ad[now]:

if visit[nxt]:

continue

if nxt == goal:

global ans

ans.add(cost^w)

continue

dfs(nxt, cost^w)

visit[now] = False

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

ad = [[] for \_ in range(N)]

for \_ in range(M):

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

u -= 1

v -= 1

ad[u].append((v, w))

ad[v].append((u, w))

ans = set()

visit = [False]\*N

goal = N-1

dfs(0, 0)

print(min(list(ans)))

# E Min of Restricted Sum

解いてない

解説と同じようなアルゴリズム

## 正解コード

from collections import deque

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

A = [None]\*N

visit = [False]\*N

adjacent = [[] for \_ in range(N)]

for i in range(M):

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

adjacent[x-1].append((y-1, z))

adjacent[y-1].append((x-1, z))

for i in range(N):

if visit[i]:

continue

visit[i] = True

A[i] = 0 #任意の値

q = deque([i])

component = [i]

while q:

now = q.popleft()

for nxt, z in adjacent[now]:

if visit[nxt]:

if A[now]^z != A[nxt]:

print(-1)

exit()

else:

A[nxt] = A[now]^z

visit[nxt] = True

q.append(nxt)

component.append(nxt)

border = (len(component)+1)//2

for i in range(30):

count = 0

search\_num = 1<<i

for node in component:

if A[node]&search\_num:

count += 1

if count >= border:

for node in component:

A[node] ^= search\_num

print(\*A)

# F Rotated Inversions

解いてない

解説と同じ

## 正解コード

from atcoder import fenwicktree

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

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

id\_by\_value = [[] for \_ in range(M)]

ft = fenwicktree.FenwickTree(M)

for i in range(N):

id\_by\_value[A[i]].append(i)

ans = 0

# k = 0

for a in A:

ans += ft.sum(a+1, M)

ft.add(a, 1)

print(ans)

# k = 1 ~ M-1

for k in range(1, M):

for sub\_id, main\_id in enumerate(id\_by\_value[M-k]):

ans += main\_id - sub\_id

ans -= (N-1-main\_id) - (len(id\_by\_value[M-k])-1-sub\_id)

print(ans)

## 自分で作ったfenwick

class Fenwicktree:

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

self.s = [0]\*n

self.n = n

def sum(self, l, r): #半開区間

if l > r:

print("error")

return False

sr = 0

id = r

while id > 0:

sr += self.s[id-1]

id -= id & (-id)

sl = 0

id = l

while id > 0:

sl += self.s[id-1]

id -= id & (-id)

return sr-sl

def add(self, i, x):

id = i + 1

while id <= self.n:

self.s[id-1] += x

id += id & (-id)

# G Flip Row or Col

解いてない