# 復習済み

A―E

# A - September

正解

# B - 1D Keyboard

正解

# C - Max Ai+Bj

正解

# D - Hidden Weights

# 入力を読み込む

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

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

for \_ in range(M):

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

u -= 1

v -= 1

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

G[v].append((u, -w))

visited = [False] \* N # 値が確定しているかどうか

ans = [0] \* N # 書き込む値

for i in range(N):

# すでに値が確定しているならばスキップ

if visited[i]:

continue

# 頂点 i の値を 0 に確定し，探索を始める

st = [i]

visited[i] = True

while st:

# いま頂点 u にいる

u = st.pop()

# 頂点 u に隣接する頂点 v を調べる

for v, w in G[u]:

if not visited[v]:

# まだ頂点 v の値が確定していないならば，頂点 u の値と整合的になるように，頂点 v の値を確定させる

visited[v] = True

ans[v] = ans[u] + w

st.append(v)

print(\*ans)

# E - How to Win the Election

## プロ

from bisect import bisect\_left

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

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

V = K - sum(A)

if M == N:

print(\*[0 for \_ in range(N)])

exit()

a = [(A[i], i) for i in range(N)]

a.sort()

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

for i in range(N):

s[i + 1] = s[i] + a[i][0]

ans = [0] \* N

def extra\_votes(l, r, x):

pos = bisect\_left(a, (x, -1), lo=l, hi=r)

return (pos - l) \* x - (s[pos] - s[l])

for i in range(N):

ng, ok = -1, V + 1

while ng + 1 < ok:

mi = (ng + ok) // 2

x = a[i][0] + mi + 1

if i < N - M:

v = extra\_votes(N - M, N, x)

else:

v = (i - (N - M - 1)) \* x - (s[i] - s[N - M - 1])

v += extra\_votes(i + 1, N, x)

if v > V - mi:

ok = mi

else:

ng = mi

ans[a[i][1]] = ok if ok <= V else -1

print(\*ans)

## 自分

n, m, k = map(int, input().split()) # n人, 上位m位 当選, k票

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

new\_a = [[a[i], i] for i in range(n)]

new\_a\_sort = sorted(new\_a)

sum\_new\_a\_sort = [0]\*(n+1)

ans = [0]\*n

if n == m:

print(\*ans)

exit()

for j in range(n):

sum\_new\_a\_sort[j+1] += sum\_new\_a\_sort[j] + new\_a\_sort[j][0]

rest\_votes = k - sum(a)

for j in range(n):

lv = 0

rv = rest\_votes + 1

while lv != rv: #考える

mv = (lv+rv)//2

Aj\_vote = new\_a\_sort[j][0] + mv

border = Aj\_vote + 1

#graceを求める

if n-m <= j <= n-1:

grace = border\*(j-(n-m-1)) - (sum\_new\_a\_sort[j]-sum\_new\_a\_sort[n-m-1])

#posを求める

lpos = j+1

rpos = n

while lpos != rpos:

mpos = (lpos+rpos)//2

if new\_a\_sort[mpos][0] >= border:

rpos = mpos

else:

lpos = mpos+1

pos = lpos

grace += border\*(pos-(j+1)) - (sum\_new\_a\_sort[pos]-sum\_new\_a\_sort[j+1])

else:

#posを求める

lpos = n-m

rpos = n

while lpos != rpos:

mpos = (lpos+rpos)//2

if new\_a\_sort[mpos][0] >= border:

rpos = mpos

else:

lpos = mpos+1

pos = lpos

grace = border\*(pos-(n-m)) - (sum\_new\_a\_sort[pos]-sum\_new\_a\_sort[n-m])

if grace > rest\_votes-mv:

rv = mv

else:

lv = mv + 1

if lv == rest\_votes+1:

ans[ new\_a\_sort[j][1] ] = -1

else:

ans[ new\_a\_sort[j][1] ] = lv

print(\*ans)

# F - Knapsack with Diminishing Values

## メモ

引っかかった部分(２つ)

・jは1からWではなく、0からW

0+i\*kが出来なくなるから

・dp[i][j] = max(dp[i][j], dp[i-1][j-k\*i] + w\_k\_values[i][k]) ではなく、

dp[i][j+i\*k] = max(dp[i][j+i\*k], dp[i-1][j] + w\_k\_values[i][k])

j-k\*iがマイナスになるから(エラーは起こらないが、マイナスになるということは配列を後ろから辿るため、目的の要素ではない。

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

for j in range(W+1):

for k in range((W-j)//i+1):

dp[i][j+i\*k] = max(dp[i][j+i\*k], dp[i-1][j] + w\_k\_values[i][k])

## 自分の正解コード

from heapq import heappop, heappush, heapify

# heapq.heappop() heapq.heappush() heappq.heapify()

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

#　w\_values : wの重さの品物の価値を格納する

w\_values = [[] for \_ in range(W+1)]

for \_ in range(N):

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

w\_values[w].append(-(v-1))

# w\_k\_values : wの重さの品物からk個選んだ時の価値

w\_k\_values = [[0]\*(W+1) for \_ in range(W+1)]

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

if w\_values[i] != []:

heapify(w\_values[i])

for k in range(W//i):

x = - heappop(w\_values[i])

w\_k\_values[i][k+1] = w\_k\_values[i][k] + x

heappush(w\_values[i], -(x-2))

dp = [[0]\*(W+1) for \_ in range(W+1)]

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

for j in range(W+1):

for k in range((W-j)//i+1):

dp[i][j+i\*k] = max(dp[i][j+i\*k], dp[i-1][j] + w\_k\_values[i][k])

print(max(dp[W]))

# G - No Cross Matching

一旦諦め