# 復習済み

A-E

# A Takahashi san 2

正解

# B Unvarnished Report

正解

# C. Separated Lunch

正解だが、chatGPTからヒントを得た

## 正解コード

n = int(input())

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

total = sum(k)

minp = total

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

ga = 0

gb = 0

buf = i

for j in range(n):

if buf&1 == 1:

ga += k[j]

else:

gb += k[j]

buf >>= 1

if minp > max(ga, gb):

minp = max(ga, gb)

print(minp)

## 正解コード(復習)

N, S, T = map(int, input().split())

edge = []

ans = float("inf")

for i in range(N):

A, B, C, D = map(int, input().split())

edge.append(((A, B), (C, D)))

def distance(x1, y1, x2, y2):

return ((x1-x2)\*\*2+(y1-y2)\*\*2)\*\*0.5

def dfs(x, y, rest, cost):

if not rest:

global ans

ans = min(ans, cost)

return

for one, another in rest:

cost\_one = distance(x, y, one[0], one[1])/S + distance(one[0], one[1], another[0], another[1])/T

cost\_another = distance(x, y, another[0], another[1])/S + distance(one[0], one[1], another[0], another[1])/T

rest\_nxt = rest.copy()

rest\_nxt.remove((one, another))

dfs(another[0], another[1], rest\_nxt, cost+cost\_one)

dfs(one[0], one[1], rest\_nxt, cost+cost\_another)

dfs(0, 0, edge, 0)

print(ans)

# D. Laser Marking

不正解。chatGPTや解説から順列とbit全探索だった。

## 正解コード

import itertools

import math

def distance(x1, y1, x2, y2):

return math.sqrt((x2 - x1) \*\* 2 + (y2 - y1) \*\* 2)

def time\_ans():

N, S, T = map(int, input().split())

segments = [tuple(map(int, input().split())) for \_ in range(N)]

min\_time = float('inf')

for perm in itertools.permutations(range(N)):

for bit in range(2\*\*N):

x, y = 0, 0

time = 0

buf = bit

for p in perm:

A, B, C, D = segments[p]

if buf & 1 == 1:

start\_x, start\_y, end\_x, end\_y = A, B, C, D

else:

start\_x, start\_y, end\_x, end\_y = C, D, A, B

time += distance(x, y, start\_x, start\_y) / S

time += distance(start\_x, start\_y, end\_x, end\_y) / T

buf >>= 1

x, y = end\_x, end\_y

min\_time = min(min\_time, time)

print(min\_time)

time\_ans()

## 正解コード(復習)

N, S, T = map(int, input().split())

edge = []

ans = float("inf")

for i in range(N):

A, B, C, D = map(int, input().split())

edge.append(((A, B), (C, D)))

def distance(x1, y1, x2, y2):

return ((x1-x2)\*\*2+(y1-y2)\*\*2)\*\*0.5

def dfs(x, y, rest, cost):

if not rest:

global ans

ans = min(ans, cost)

return

for one, another in rest:

cost\_one = distance(x, y, one[0], one[1])/S + distance(one[0], one[1], another[0], another[1])/T

cost\_another = distance(x, y, another[0], another[1])/S + distance(one[0], one[1], another[0], another[1])/T

rest\_nxt = rest.copy()

rest\_nxt.remove((one, another))

dfs(another[0], another[1], rest\_nxt, cost+cost\_one)

dfs(one[0], one[1], rest\_nxt, cost+cost\_another)

dfs(0, 0, edge, 0)

print(ans)

# E. Sensor Optimization Dilemma 2

不正解

復習コードもABの生産量は安い方という考え方は同じで残りの計算をdpを使ったが少し遅かった。

## 正解コード

import math

def judge\_w(w, st\_info, n, x):

allcost = 0

for A, P, B, Q in st\_info:

mincost\_i = float('inf')

NA = min(int(w/A), B)

NB = min(int(w/B), A)

for nb in range(NB+1):

na = math.ceil((w-B\*nb)/A)

mincost\_i = min(mincost\_i, na\*P+nb\*Q)

for na in range(NA+1):

nb = math.ceil((w-A\*na)/B)

mincost\_i = min(mincost\_i, na\*P+nb\*Q)

allcost += mincost\_i

if allcost <= x:

return True

else:

return False

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

st\_info = []

for i in range(n):

st\_info.append( list(map(int, input().split())) )

left = 0

right = 10\*\*9

while left < right:

mid = (left + right + 1)//2

if judge\_w(mid, st\_info, n, x):

left = mid

else:

right = mid - 1

print(left)

## 復習コード

import math

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

A = []

B = []

P = []

Q = []

R = []

for i in range(N):

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

A.append(a)

B.append(b)

P.append(p)

Q.append(q)

lcm = a\*b // math.gcd(a, b)

dp = [float("inf")]\*(2\*lcm+1)

dp[0] = 0

for d in range(a, 2\*lcm+1, a):

dp[d] = min(dp[d], dp[d-a]+p)

for i in range(2\*lcm-b+1):

dp[i+b] = min(dp[i+b], dp[i]+q)

for i in range(2\*lcm, 0, -1):

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

R.append(dp)

def judge(w):

amount = 0

for i in range(N):

lcm = len(R[i])//2

q, r = divmod(w, lcm)

if q < 1:

amount += R[i][r]

else:

amount += (q-1)\*R[i][lcm] + R[i][lcm+r]

return amount

ok = 0

ng = 10\*\*9+1

while ok + 1 < ng:

mid = (ok+ng)//2

if judge(mid) <= X:

ok = mid

else:

ng = mid

print(ok)

# F. Shipping

一旦断念

動的計画法らしい

# G. Only One Product Name

一旦、断念

最小道被覆、DAG、二部グラフの最大マッチングとかがキーワードらしい