# 復習済み

A-E

# A Daily Cookie

正解

# B Daily Cookie 2

正解

# C Kaiten Sushi

正解

# D Keep Distance

正解

# E Expansion Packs

解いてない

解説と同じ解法。ただ、解説の読解に苦しんだので自分で解説する。

２つの段階がある。

１段階目は、”1パック引いた時、レアカードを0~N枚引く確率”を求めること。これはdp[N-1], one\_packにあたる。ここは問題ない。

2段階目に問題がある。解説では

「i>0 のときは次に開封するパックに入っているレアカードの枚数で場合分けして足し合わせることを考えることにより、  
がわかります。」

と書いてある。これはあまり直感的でなかった。

例えば、4枚のパックとする。この時1パックに入っているレアカードの枚数別確率をg[j]とする。この時、1回目に引いたレアカードが1枚だとする。この時、合計がi枚以上に達するにはf[i-1]の回数だけパックを開けることが期待される。よって1回目に1枚レアカードを引いて、i枚以上レアカードを引くために開くパック数の期待値は、1回目の1と残りi-1枚以上レアカードを引かなくてはならないので となる。同様に考えると以下のような式が得られる。

ただ、iがN未満の時は、i-jがマイナスになってしまうので、上記のようなmax関数の工夫が必要である。

## 正解コード

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

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

P = [p/100 for p in P]

# probability of the number of rare cards in a pack

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

dp[0][0] = 1-P[0]

dp[0][1] = P[0]

for i in range(1, N):

for j in range(i+2):

if j != 0:

dp[i][j] = dp[i-1][j]\*(1-P[i]) + dp[i-1][j-1]\*P[i]

else:

dp[i][0] = dp[i-1][0]\*(1-P[i])

one\_pack = dp[N-1]

# n packs

f = [0]\*(X+1)

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

tmp = 1

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

id = max(0, i-j)

tmp += f[id]\*one\_pack[j]

f[i] = tmp/(1-one\_pack[0])

print(f[X])

# F Falling Bars

解いてない

# G Tile Distance 3

解いてない