BOJ 4781.

사탕 가게

소프트웨어학과 201921017 이지우

Step1 – 접근

문제

상근이는 선영이와 걸어가다가 사탕 가게를 지나가게 되었다. 갑자기 상근이는 선영이에게 사탕이 얼마나 건강에 안 좋은지 설명하기 시작했다. 선영이는 매우 짜증이 났고, 상근이에게 누가 더 건강이 안 좋아질 수 있는지 내기를 하자고 했다. 상근이는 내기를 그 즉시 받아들였다.

두 사람은 같은 돈을 가지고 가게에 들어가서 사탕을 산다. 이때, 구매한 사탕의 칼로리가 더 큰 사람이 내기에서 이기게 된다.

상근이는 잠시 화장실에 갔다온다고 핑계를 댄 뒤에, 노트북을 열고 사탕 가게의 시스템을 해킹하기 시작했다. 이 시스템에는 현재 사탕 가게에 있는 사탕의 가격과 칼로리가 모두 등재되어 있다. 각 사탕의 개수는 매우 많기 때문에, 원하는 만큼 사탕을 구매할 수 있다. 또, 사탕은 쪼갤 수 없기 때문에, 일부만 구매할 수 없다.

사탕 가게에 있는 모든 사탕의 가격과 칼로리가 주어졌을 때, 어떻게 하면 칼로리의 합이 가장 크게 되는지를 구하는 프로그램을 작성하시오.

입력

각 테스트 케이스의 첫째 줄에는 가게에 있는 사탕 종류의 수 n과 상근이가 가지고 있는 돈의 양 m이 주어진다. ($1 \le n \le 5,000$, $0.01 \le m \le 100.00$) m은 항상 소수점 둘째자리까지 주어진다.

다음 n개 줄에는 각 사탕의 칼로리 c와 가격 p가 주어진다. ($1 \le c \le 5,000$, $0.01 \le p \le 100.00$) c는 항상 정수, p는 항상 소수점 둘째자리이다.

입력의 마지막 줄에는 '0 0.00'이 주어진다.

→ DP, Knapsack

출력

각 테스트 케이스에 대해서, 상근이가 돈 m을 가지고 구매할 수 있는 가장 높은 칼로리를 출력한다.

Step1 – 접근

dp[money] : 돈 money로 살 수 있는 가장 큰 칼로리의 합

dp[money] ← dp[money – candy1.price] + candy1.calorie

← dp[money – candy2.price] + candy2.calorie

← dp[money – candy3.price] + candy3.calorie

⋮

← dp[money – candyN.price] + candyN.calorie

0.01 ≤ money, price ≤ 100.00
* 배열의 인덱스는 실수가 될 수 없음

Step1 – 접근

① dp를 이차원 배열로 만들기

dp의 행: money의 정수부

dp의 열: money의 소수부

money = 8.23 일 때,

dp[8][23]에 답이 저장됨

*단점

integer, fraction = int(money // 1), int(money % 1)

dp[money - price] 인덱스 계산이 불편

→ diff = money – price

→ diff = (integer + fraction*0.01) – price

→ dp[diff // 1][diff % 1]

② money, price를 정수로 만들기

0.01 ≤ money, price ≤ 100.00

→ 1.00 ≤ money * 100 ≤ 10000.00

→ 1 ≤ int(money * 100) ≤ 10000

money = 8.23 일 때,

dp[823]에 답이 저장됨

Step2 – 구현

```
while True:
    n, m = map(float, stdin.readline().split())

    if n == 0 and m == 0.00:
        break

    m = int(m * 100 + 0.5)

    candies = list()
    for _ in range(int(n)):
        c, p = map(float, stdin.readline().split())
        candies.append((int(c), int(p * 100 + 0.5)))

    candies = sorted(candies, key = lambda x: x[1])

    print(get_calorie(m))
```

```
def get_calorie(money):

    dp = [0]*(money + 1)

    for i in range(candies[0][1], money + 1):
        for cal, price in candies:
            if i - price < 0:
                break
            dp[i] = max(dp[i], cal + dp[i - price])

    return dp[money]
```

Thank you~!@#$%^&*()_+