201201871 서현택

1.목표

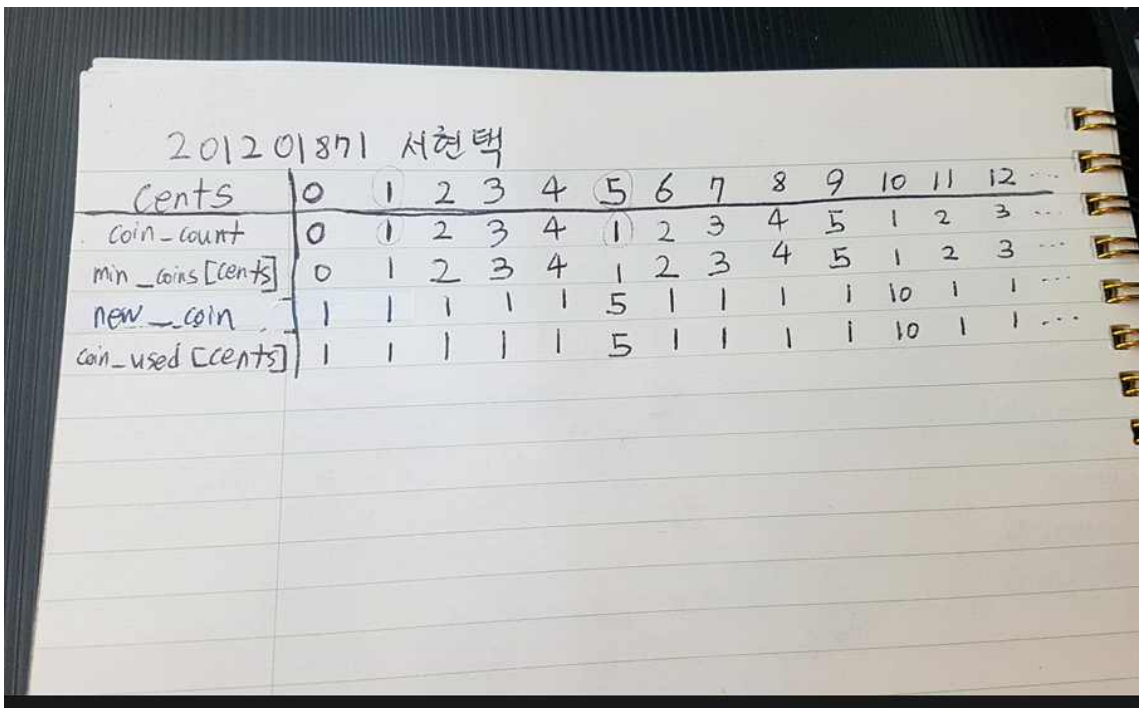
동적 프로그래밍을 활용해서 money_change_dp_list를 완성한다.

2.과제를 해결하는 방법

1. 동적 프로그래밍의 이해
2. known_result에 결과를 저장
3. 재귀에 대한 이해
4. 리스트형 자료를 합치는 방법의 이해

3.과제를 해결한 방법

1) 그림을 통한 make_change 이해



2) change_money_dp_list

```
def make_change(coin_value_list, coin_change, known_result):
    min_coins = [0] * (coin_change + 1)
    for cents in range(0, coin_change+1):
        coin_count = cents
        new_coin = 1
        coins = list()
        for coin in coin_value_list:
            if coin <= cents:
                coins.append(coin)
        for coin in coins:
            if min_coins[cents - coin] + 1 < coin_count:
                coin_count = min_coins[cents - coin] + 1
                new_coin = coin
            min_coins[cents] = coin_count
            known_result[cents] = [new_coin] + known_result[cents - new_coin]
    return min_coins[coin_change]
```

전체적인 흐름은 change_money_list와 비슷하나, 여기서는 min_coins와 used_coins 대신 known_result가 지역 변수로 들어갔다. min_coins를 list로 선언하고, list의 크기는 입력된 코인의 크기+1로 한다. 그 후, cents를 0부터 coin_change까지 반복하면서 진행하고, coins에는 coin_value_list에 저장된 값 중 해당 반복에서 cents보다 작은 coin을 저장한다.
coins를 반복하면서 min_coins[cents-coin]+1의 값이 coin_count보다 작으면 coin_count에 min_coins[cents-coin]+1을 저장하고, new_coin에 coin을 저장한다.
if문 이후에는 지금 저장된 coin_count의 값을 min_coins[cents]에 저장하고, known_result[cents]에 [new_coin]과 known_result[cents - new_coin] 더한 값을 저장하여 해당 동전에서 최소 잔돈을 반환한다.

3) draw_graphic_practice

```
import change_money_list as cml
import change_money_saved_list as cmsl
import change_money_dp_list as dp
```

import를 통해 change_money_list, change_money_saved_list, change_money_dp_list를 불러온 후, cml, cmsl, cmdl 라는 이름으로 저장한다.

```
# Y1
start_time = time.time()

cml.make_change(coin_value_list, x)

end_time = time.time()
execution_time = end_time - start_time
y_data1.append(execution_time)

# Y2
start_time = time.time()

known_result = list()
for index in range(0, x + 1):
    known_result.append(None)
cmsl.make_change(coin_value_list, x, known_result)

end_time = time.time()
execution_time = end_time - start_time
y_data2.append(execution_time)

# Y3
start_time = time.time()

known_result = [[]] * (x + 1)
dp.make_change(coin_value_list, x, known_result)
```

(1) Y1

cml.make_change에 coin_value_list와 x를 넣어 change_make_list에서 실행 속도를 구한다.

(2) Y2

0부터 x까지 반복하면서 known_result를 None으로 채운 뒤, cmsl.make_change에 coin_value_list와 x, known_result를 넣어 change_make_saved_list에서의 실행 속도를 구한다.

(3) Y3

coin_change에 x를 저장하고, coin_change+1 크기의 이중 list인 known_result를 선언해서 dp.make_change에 넣어서 동적 프로그래밍에서의 실행 속도를 알아본다.

4.결과화면

1) change_money_dp_list

```
C:\Users\HyunTaek\PycharmProjects\week05\venv\Scripts\python.exe C:/Users/HyunTaek/PycharmProjects/week05/change_money_dp_list.py
Change: 63
Making change for 63 requires
3 coins
They are:
[21, 21, 21]
The used list is as follows:
[[], [1], [1, 1], [1, 1, 1], [1, 1, 1, 1], [5], [1, 5], [1, 1, 5], [1, 1, 1, 5], [1, 1, 1, 1, 5], [10], [1, 10], [1, 1, 10], [1, 1, 1, 10],

Process finished with exit code 0
```

2) change_money_dp_list_test

```
Tests passed: 3 of 3 tests – 109 ms
True [25, 25, 21, 21] 92
True [5, 25, 21, 21, 21] 93
True [10, 21, 21, 21, 21] 94
True [25, 25, 25, 10, 10] 95
True [25, 25, 25, 21] 96
True [21, 21, 5, 25, 25] 97
True [10, 25, 21, 21, 21] 98


Ran 3 tests in 0.109s

OK
True [5, 21, 21, 21, 21, 10] 99
True [25, 25, 25, 25] 100

Process finished with exit code 0
```

3) draw_graphic_practice