

알고리즘과 문제해결 일반과제5

20181061 전하영

1. 문제기술

동전 거스름돈 알고리즘은 동전의 종류와 거스름돈이 주어졌을 때 동전의 개수를 최소화 하여 거스름돈을 거슬러주는 문제다. 같은 동전을 여러번 사용할 수 있다.

Greedy 알고리즘에서 배운 동전 거스름돈 알고리즘은 항상 최적의 해를 내놓지 않았다. 이를 보완하여 동적 프로그래밍을 이용한 최적의 해를 찾는 알고리즘을 구현하였다. (작은과제 11-1). 하지만 이 문제는 최소로 사용한 동전 개수 합만 알려줄 뿐, 어떤 동전을 몇 개 사용하였는지는 알려주지 않는다.

이번 과제는 이 부분을 보완하여 반환할 동전의 개수와 그 개수를 반환하도록 코드를 변경하는 것이다.

이 부분을 해결하기 위해, 크게 생각한 방법은 기존 코드를 작성하면서 사용한 DP 배열을 잘 응용하는 것이다. 단계마다 사용한 동전의 종류와 그 개수를 저장할 수 있는 2차원 리스트를 하나 만들어 해결해 보고자 하였다.

2. 세부 구현사항

hw5_DPChange.py

- 동전 거스름돈 알고리즘의 수행결과이다. 최소 동전 개수, 사용한 동전의 종류와 개수를 출력한다.

기존 DPChange 코드는 강의자료의 의사코드를 참고하여 쉽게 작성할 수 있었다.

동전의 개수를 출력하기 위해 우선 생각을 해 보았다. 기존의 2중 for문 이내에서 코드를 조금만 추가하면 결과를 볼 수 있을지, 아니면 또 다른 출력 함수를 작성해야 하는지 등 다양한 생각을 하였다. 가장 1차원적으로 생각한 방법은 '단계별로 사용한 동전과 그 개수를 저장' 하는 것이었다. 여기서 말하는 단계는 1부터 n까지 증가하는 거스름돈 금액이다.

이 방법은 기존의 DP 배열과 for문을 조금 응용하면 되었다. 추가로 단계마다 사용한 동전의 종류와 그 개수를 저장할 수 있는 2차원 배열을 사용해주면 되었다.

동전의 종류가 담긴 d 배열이다. $d = [16, 10, 5, 1]$ 여기에서 만약에

- 16원짜리 동전 1개가 있다면 [1,0,0,0]

- 16원짜리 동전 1개, 5원짜리 동전 1개가 있다면 [1,0,1,0]

으로 표현하기로 하였다. 이 표현법을 적용하여 생각한 구현방법을 손으로 적어보았다.

$d = [16, 10, 5, 1]$, $n = 25$ 일때
 $[0\ 0\ 0\ 0]$ $C[0]$
 $[0\ 0\ 0\ 1]$ $C[1]$
 $[0\ 0\ 0\ 2]$ $C[2] = C[1] + 1\text{원}$
 $\quad = [0\ 0\ 0\ 1] + [0\ 0\ 0\ 1]$
 $[0\ 0\ 0\ 3]$ $C[3] = C[2] + 1\text{원}$
 $\quad = [0\ 0\ 0\ 2] + [0\ 0\ 0\ 1]$
 $[0\ 0\ 0\ 4]$ $C[4] = C[3] + 1\text{원}$
 $\quad = [0\ 0\ 0\ 3] + [0\ 0\ 0\ 1]$
 $[0\ 0\ 1\ 0]$ $C[5] = C[0] + 5\text{원}$
 $\quad = [0\ 0\ 0\ 0] + [0\ 0\ 1\ 0]$
 $[0\ 0\ 1\ 1]$ $C[6] = C[1] + 5\text{원}$
 $\quad = [0\ 0\ 0\ 1] + [0\ 0\ 1\ 0]$

```

coin_result = [[0,0,0,0]]
for j in range(1,n+1):
    mini = 0
    for i in range(0,len(d)): # 동전 순회
        if (d[i]<=j) and (C[j-d[i]]+1 < C[j]):
            C[j]= C[j-d[i]]+1
            mini = i
    temp = [0,0,0,0]
    temp[mini]+=1 # 현재 사용한 동전 갯수 추가
    coin_result.append([x+y for x,y in zip(temp,coin_result[j-d[mini]])]) # 현재
    사용한 동전 개수와, j-d[i] 에서 사용한 동전 개수를 합침.

```

위의 코드에서, coin_result는 단계별 동전 개수를 저장할 2차원 리스트이다.

동전을 순회하는 구문에서 선택된 동전의 인덱스를 mini 변수에 저장한다.

그리고 temp라는 [0,0,0,0] 리스트를 만들고 현재 선택된 동전의 개수를 1개 추가한다.

그리고 $C[j-d[i]]$ 이 부분을 응용하여 현재 추가한 동전을 제외한 금액에서 사용한 동전 개수를 구해온다. 이때 구문은 $coin_result[j-d[mini]]$ 이 된다.

앞에서 구한 temp와 $coin_result[j-d[mini]]$ 를 각 요소끼리 덧셈을 한다. 그렇게 한 결과를 coin_result 리스트에 append한다.

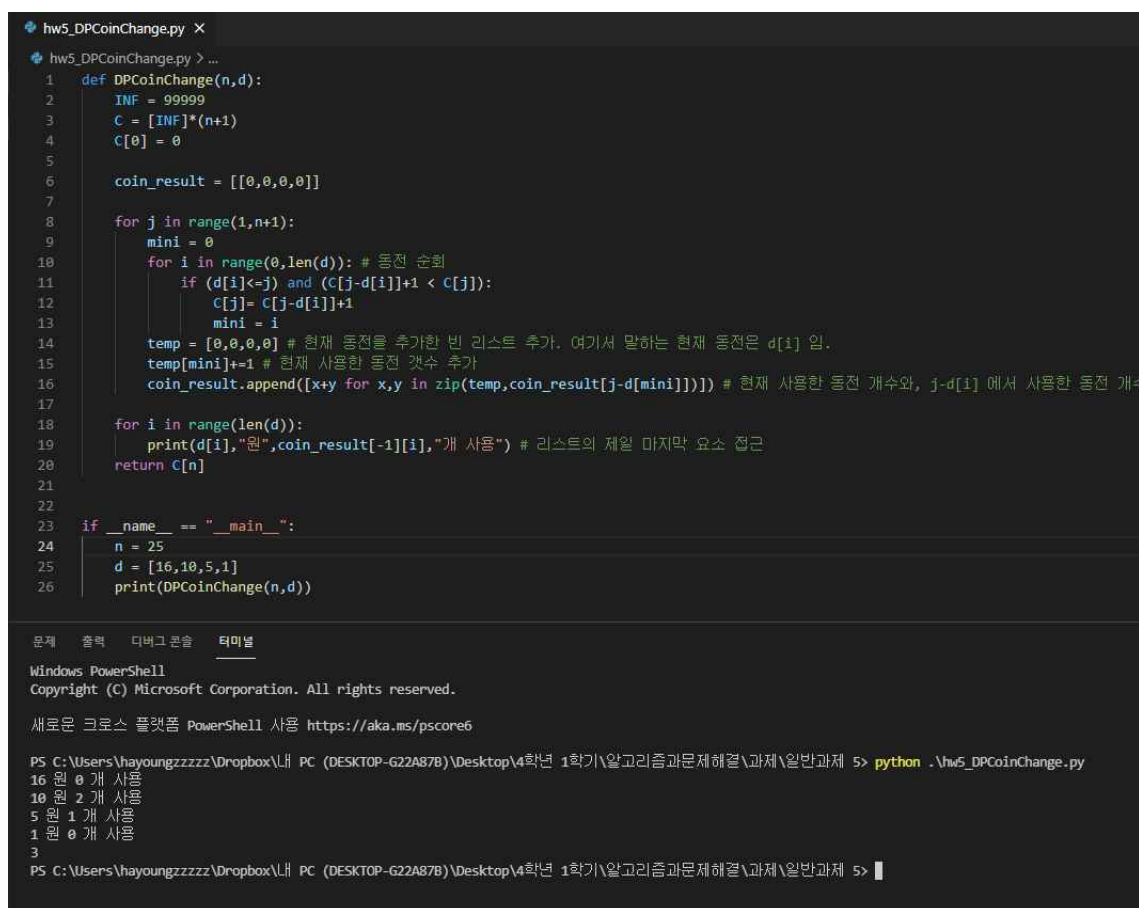
마지막에 출력할 때는

```
for i in range(len(d)):
    print(d[i],"원",coin_result[-1][i],"개 사용") # 리스트의 제일 마지막 요소 접근
return C[n]
```

coin_result의 가장 마지막 리스트를 출력해주면 된다.

3. 체크포인트

3-1. 동전 [16,10,5,1]으로 거스름돈 25이 입력일 때의 실행 스크린샷



The screenshot shows a Python IDE with a file named `hw5_DPCoinChange.py`. The code implements a dynamic programming solution for the coin change problem. It defines a function `DPCoinChange(n, d)` that takes the number of coins `n` and a list of denominations `d`. It initializes an array `C` of size `n+1` with a large value `INF` (99999) and sets `C[0] = 0`. It also initializes a 2D list `coin_result` of size `(n+1) x len(d)` with zeros. The algorithm iterates over the number of coins `j` from 1 to `n`, and for each `j`, it iterates over the denominations `i` from 0 to `len(d)-1`. It checks if the current denomination `d[i]` is less than or equal to the current coin value `j`, and if the current value `C[j]` is greater than `C[j-d[i]] + 1`. If so, it updates `C[j]` and `mini`. It also updates a temporary list `temp` and appends it to `coin_result`. Finally, it prints the result for `n=25` and `d=[16,10,5,1]`.

```
def DPCoinChange(n,d):
    INF = 99999
    C = [INF]*(n+1)
    C[0] = 0

    coin_result = [[0,0,0,0]]

    for j in range(1,n+1):
        mini = 0
        for i in range(0,len(d)): # 동전 순회
            if (d[i]<=j) and (C[j-d[i]]+1 < C[j]):
                C[j]= C[j-d[i]]+1
                mini = i
            temp = [0,0,0,0] # 현재 동전을 추가한 빈 리스트 추가. 여기서 말하는 현재 동전은 d[i] 임.
            temp[mini]+=1 # 현재 사용한 동전 갯수 추가
            coin_result.append([x+y for x,y in zip(temp,coin_result[j-d[mini]])]) # 현재 사용한 동전 개수와, j-d[i] 에서 사용한 동전 개수

    for i in range(len(d)):
        print(d[i],"원",coin_result[-1][i],"개 사용") # 리스트의 제일 마지막 요소 접근
    return C[n]

if __name__ == "__main__":
    n = 25
    d = [16,10,5,1]
    print(DPCoinChange(n,d))
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 <https://aka.ms/pscore6>

```
PS C:\Users\hayoungzzzzz\Dropbox\내 PC (DESKTOP-G22A87B)\Desktop\4학년 1학기\알고리즘과문제해결\과제\일반과제 5> python .\hw5_DPCoinChange.py
16 원 0 개 사용
10 원 2 개 사용
5 원 1 개 사용
1 원 0 개 사용
3
```

PS C:\Users\hayoungzzzzz\Dropbox\내 PC (DESKTOP-G22A87B)\Desktop\4학년 1학기\알고리즘과문제해결\과제\일반과제 5> █

3-2. 동전 종류와 개수를 반환하기 위한 자료구조와 관련 코드

```
def DPCoinChange(n,d):
    #일부 생략
    coin_result = [[0,0,0,0]]
    for j in range(1,n+1):
        mini = 0
        for i in range(0,len(d)): # 동전 순회
```

```
        if (d[i]<=j) and (C[j-d[i]]+1 < C[j]):
            C[j]= C[j-d[i]]+1
            mini = i
            temp = [0,0,0,0]
            temp[mini]+=1
            coin_result.append([x+y for x,y in zip(temp,coin_result[j-d[mini]])])
    for i in range(len(d)):
        print(d[i],"원",coin_result[-1][i],"개 사용")
    return C[n]
```

빨간색 부분이 추가된 코드임.

4. 결론

동적 계획 알고리즘을 배움으로써 항상 이해하기 어려웠던 동전 거스름돈 알고리즘을 완벽히 이해할 수 있게 되었다. 그리고 스스로 동전 개수까지 출력할 수 있게 하여 생각하는 부분도 확장된 것 같다. 처음에는 막막하여 인터넷의 도움을 받으려고 하였으나 아무래도 과제를 하는데 의미가 없을 것 같아 오랜 시간 고민하였다. 생각보다 답은 간단히 나왔으며 생각을 코드로 옮기는 것은 얼마 걸리지 않았다.

하지만 코드를 실행하면서 결과를 출력하는데 시간이 조금 걸리는 것을 확인하였다. 적어도 0.5초 이상은 걸렸다. 이보다 좋은 알고리즘이 있지 않을까 생각하여 주변에 물어보니 역으로 생각하여 동전 개수를 출력하는 방법도 있다고 하였다. 나중에 코드를 도움받아 시간복잡도를 개선하는 방법으로 새롭게 코드를 작성해 보고 싶다.