# 알고리즘 4차 과제

그리디, 동적계획 알고리즘 - 거스름돈

2020. 12. 03. 목 컴퓨터공학과

2019305059

이현수

```
주제: 동전거스름돈 문제를 Greedy 방법론과 Dynamic Programming 방법론으로 구현
```

• 동전 액면가

```
규정 A: 500원, 100원, 50원, 10원, 5원, 1원
규정 B: 500원, 130원, 51원, 10원, 5원, 1원
```

### ■ 코드복사

```
#include<stdio.h>
#define INF 2147483647 #무한대를 의미. int형이 가질 수 있는 최대숫자
int CoinChange_greedy_A(int cash) #그리디 알고리즘 방식 _ 규정A
{
      int change = cash;
                         #거스름돈 저장
      int n500 = 0, n100 = 0, n50 = 0, n10 = 0, n5 = 0, n1 = 0;#액면가별 개수 저장 변수. 모두 0으로초기화
      while (change >= 500) #거스름돈이 500원 이상일 때 무한반복
             change -= 500; # change변수에서 500원 차감
             n500++;
                          # n500변수 1 증가
      }
      while (change >= 100) #거스름돈이 100원 이상일 때 무한반복
             change -= 100; # change변수에서 100원 차감
                          # n100변수 1 증가
             n100++;
      while (change >= 50) #거스름돈이 50원 이상일 때 무한반복
             change -= 50; # change변수에서 50원 차감
                        # n50변수 1 증가
             n50++;
      while (change >= 10) #거스름돈이 10원 이상일 때 무한반복
      {
             change -= 10; # change변수에서 10원 차감
                        # n10변수 1 증가
             n10++;
      while (change >= 5) #거스름돈이 5원 이상일 때 무한반복
             change -= 5; # change변수에서 5원 차감
             n5++;
                        # n5변수 1 증가
      }
      while (change >= 1) #거스름돈이 1원 이상일 때 무한반복
             change -= 1; # change변수에서 1원 차감
                        # n1변수 1 증가
      return n500 + n100 + n50 + n10 + n5 + n1; #액면가별 개수 저장변수 모두 더해서 반환
}
int CoinChange_greedy_B(int cash) #그리디 알고리즘 방식 _ 규정B
{
                        #거스름돈 저장
      int change = cash;
      int n500 = 0, n130 = 0, n51 = 0, n10 = 0, n5 = 0, n1 = 0; #액면가별 개수 저장 변수. 모두 0으로초기화
      while (change >= 500) #거스름돈이 500원 이상일 때 무한반복
      {
             change -= 500; # change변수에서 500원 차감
             n500++;
                          # n500변수 1 증가
      }
```

```
while (change >= 130) #거스름돈이 130원 이상일 때 무한반복
      {
             change -= 130; # change변수에서 130원 차감
                          # n130변수 1 증가
             n130++;
      }
      while (change >= 51) #거스름돈이 51원 이상일 때 무한반복
             change -= 51; # change변수에서 51원 차감
                         # n51변수 1 증가
             n51++:
      }
      while (change >= 10) #거스름돈이 10원 이상일 때 무한반복
             change -= 10; # change변수에서 10원 차감
             n10++;
                         # n10변수 1 증가
      }
      while (change >= 5) #거스름돈이 5원 이상일 때 무한반복
             change -= 5; # change변수에서 5원 차감
                        # n5변수 1 증가
             n5++;
      }
      while (change >= 1) #거스름돈이 1원 이상일 때 무한반복
             change -= 1; # change변수에서 1원 차감
                        # n1변수 1 증가
      return n500 + n130 + n51 + n10 + n5 + n1; #액면가별 개수 저장변수 모두 더해서 반환
}
int CoinChange_dynamic_A(int cash) #동적계획 알고리즘 _ 규정A
{
      int i, j, answer;
      int d[6] = { 500,100,50,10,5,1 }; #정수형 배열에 액면가 저장
      int* Cash = (int*)malloc((cash + 1) * sizeof(int)); # cash+1 크기로 int형 배열 Cash동적할당
      for (i = 1; i \le cash; i++)
      {
          Cash[i] = INF; # Cash배열 인덱스 1~cash까지 INF(무한대) 저장
      Cash[0] = 0; # Cash배열 인덱스 0에 0 저장
      for (j = 1; j <= cash; j++) # j=1부터 cash까지 i를 1씩 증가하면서 반복(j는 거스름돈 액수변수)
      {
          for (i = 0; i < 6; i++) # i=0부터 5까지 (액면가가 들어있는 d배열의 크기가 6이기 때문에)
               #액면가보다 j(금액)이 크거나 같으면서 j금액-d[i]의 금액값 개수+1이 현재 j원개수보다 작으면
             if (d[i] \leftarrow j \&\& (Cash[j - d[i]] + 1 < Cash[j]))
             {
                Cash[j] = Cash[j - d[i]] + 1; # j금액-d[i](액면가)의 개수 + 1 을저장
             }
          }
      answer = Cash[cash]; #Cash배열의 cash인덱스 값 answer 변수에 저장
      free(Cash); #동적할당 해체
      return answer; #answer변수 반환
}
```

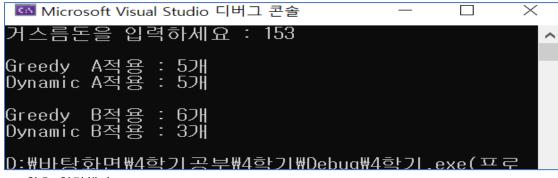
```
int CoinChange_dynamic_B(int cash) #동적계획 알고리즘 _ 규정B
       int i, j, answer;
       int d[6] = { 500,130, 51,10,5,1 }; #정수형 배열에 액면가 저장
       int* Cash = (int*)malloc((cash+1)*sizeof(int)); # cash+1 크기로 int형 배열 동적할당
      for (i = 1; i \le cash; i++)
      {
          Cash[i] = INF; # Cash배열 인덱스 1~cash까지 INF(무한대) 저장
      Cash[0] = 0; # Cash배열 인덱스 0에 0 저장
      for (j = 1; j <= cash; j++) # j=1부터 cash까지 j를 1씩 증가하면서 반복(j는 거스름돈 액수변수)
      {
          for (i = 0; i < 6; i++) # i=0부터 5까지 (액면가가 들어있는 d배열의 크기가 6이기 때문에)
               #액면가보다 j(금액)이 크거나 같으면서 j금액-d[i]의 금액값 개수+1이 현재 j원개수보다 작으면
              if (d[i] \le j \&\& (Cash[j - d[i]] + 1 < Cash[j]))
                 Cash[j] = Cash[j - d[i]] + 1; # j금액-d[i](액면가)의 개수 + 1 을저장
             }
          }
      }
      answer = Cash[cash]; #Cash배열의 cash인덱스 값 answer 변수에 저장
      free(Cash); #동적할당 해체
      return answer; #answer 변수 반환
}
int main(void)
{
       int cash; #거스름돈 입력 변수
      printf("거스름돈을 입력하세요: ");
      scanf_s("%d", &cash); #거스름돈 입력
      printf("₩nGreedy A적용 : %d개₩n", CoinChange_greedy_A(cash)); #규정A-그리디 알고리즘
      printf("Dynamic A적용 : %d개\n\n", CoinChange_dynamic_A(cash)); #규정A-동적계획 알고리즘
      printf("Greedy B적용 : %d개\n", CoinChange_greedy_B(cash));
                                                           #규정B-그리디 알고리즘
      printf("Dynamic B적용: %d개\n", CoinChange_dynamic_B(cash));
                                                          #규정B-동적계획 알고리즘
}
```

#### ■코드캡처

```
#include<stdio.h>
 2
       #define INF 2147483647
 3
      pint CoinChange_greedy_A(int cash) {
 4
 5
            int change = cash;
            int n500 = 0, n100 = 0, n50 = 0, n10 = 0, n5 = 0, n1 = 0;
 6
 8
            while (change >= 500) {
 9
                change -= 500;
                п500++;
10
11
12
            while (change >= 100) {
                change -= 100;
13
14
                n100++;
15
16
            while (change >= 50) {
17
               change -= 50;
18
                n50++;
19
20
            while (change >= 10) {
21
                change -= 10;
22
                n10++;
23
24
            while (change >= 5) {
25
               change -= 5;
                п5++;
26
27
            while (change >= 1) {
29
               change -= 1;
30
                n1++;
31
            return n500 + n100 + n50 + n10 + n5 + n1;
32
33
34
35
      pint CoinChange_greedy_B(int cash) {
36
            int change = cash;
37
            int n500 = 0, n130 = 0, n51 = 0, n10 = 0, n5 = 0, n1 = 0;
39
            while (change >= 500) {
40
                change -= 500;
41
                n500++;
42
43
            while (change >= 130) {
                change -= 130;
44
45
                n130++;
46
47
            while (change >= 51) {
                change -= 51;
48
                n51++;
49
50
51
            while (change >= 10) {
52
                change -= 10;
53
                n10++;
54
            }
55
            while (change >= 5) {
                change -= 5;
56
57
                n5++;
58
59
            while (change >= 1) {
                change -= 1;
60
                n1++;
61
62
63
            return n500 + n130 + n51 + n10 + n5 + n1;
64
65
```

```
□ int CoinChange_dynamic_A(int cash) {
66
             int i, j, answer;
67
             int d[6] = \{500,100,50,10,5,1\};
68
            int* Cash = (int*)malloc((cash + 1) * sizeof(int));
69
70
            for (i = 1; i \le cash; i++)
71
72
            {
73
                Cash[i] = INF;
74
75
            Cash[0] = 0;
76
            for (j = 1; j \le cash; j++)
77
                for (i = 0; i < 6; i++)
78
79
                    if (d[i] \le j \&\& (Cash[j - d[i]] + 1 < Cash[j]))
80
81
                        Cash[i] = Cash[i - d[i]] + 1;
82
                }
84
85
            answer = Cash[cash];
86
            free(Cash);
87
            return answer;
88
89
       }
90
91
       □ int CoinChange dynamic B(int cash) {
              int i, j, answer;
92
              int d[6] = \{500,130,51,10,5,1\};
93
              int* Cash = (int*)malloc((cash+1)*sizeof(int));
94
95
96
             for (i = 1; i \le cash; i++)
97
                  Cash[i] = INF;
98
99
             Cash[0] = 0;
100
              for (j = 1; j \le cash; j++)
101
102
                  for (i = 0; i < 6; i++)
103
104
                      if (d[i] \le j \&\& (Cash[j - d[i]] + 1 < Cash[j]))
105
106
                      {
                          Cash[j] = Cash[j - d[i]] + 1;
107
108
109
110
              answer = Cash[cash];
111
              free(Cash);
112
113
              return answer:
114
       | }
115
       □ int main(void) {
116
             int cash:
117
             printf("거스름돈을 입력하세요 : ");
118
             scanf_s("%d", &cash);
119
120
121
             printf("#nGreedy A적용 : %d개#n", CoinChange_greedy_A(cash));
            printf("Dynamic A적용 : %d개₩n₩n", CoinChange_dynamic_A(cash));
122
             printf("Greedy B적용 : %d개₩n", CoinChange_greedy_B(cash));
123
             printf("Dynamic B적용 : %d개#n", CoinChange_dynamic_B(cash));
124
125
```

#### ■실행



153원을 입력했다.

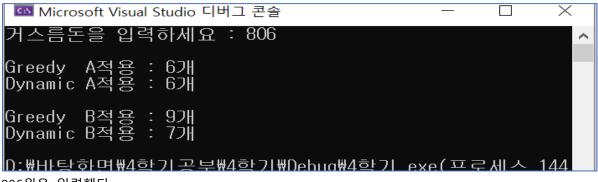
A규정(500, 100, 50, 10, 5, 1)을 적용한 그리디, 동적계획 알고리즘 모두 5개로 나온다.

100원-1개, 50원-1개, 1원-3개 = 총 5개

B규정(500, 130, 51, 10, 5, 1)을 적용한 그리디 알고리즘은 6개, 동적계획 알고리즘은 3개이다.

그리디 B적용 : 130원-1개, 10원-2개, 1원-3개 = 6개

동적계획 B적용 : 51원-3개 = 3개



806원을 입력했다.

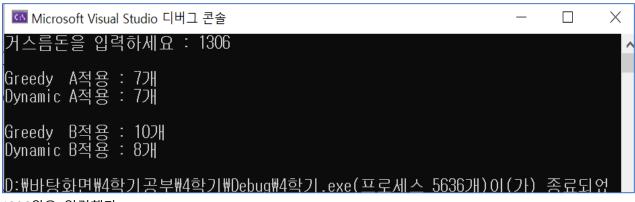
A규정(500, 100, 50, 10, 5, 1)을 적용한 그리디, 동적계획 알고리즘 모두 6개로 나온다.

500원-1개, 100원-3개, 5원-1개, 1원-1개 = 6개

B규정(500, 130, 51, 10, 5, 1)을 적용한 그리디 알고리즘은 9개, 동적계획 알고리즘은 7개이다.

그리디 B적용 : 500원-1개, 130원-2개, 10원-4개, 5원-1개, 1원-1개 = 9개

동적계획 B적용: 500원-1개, 51원-6개 = 7개



1306원을 입력했다.

A규정(500, 100, 50, 10, 5, 1)을 적용한 그리디, 동적계획 알고리즘 모두 7개로 나온다.

500원-2개, 100원-3개, 5원-1개, 1원-1개 = 7개

B규정(500, 130, 51, 10, 5, 1)을 적용한 그리디 알고리즘은 10개, 동적계획 알고리즘은 8개이다.

그리디 B적용: 500원-2개, 130원-2개, 10원-4개, ,5원-1개, 1원-1개 = 10개

동적계획 B적용: 500원-2개, 51원-6개 = 8개

## ■소감

이론으로 배운 내용을 코딩해보면서 동전거스름돈 문제를 통해 그리디 알고리즘과 동적계획 알고리즘의 차이를 확인할 수 있어서 유익했다.