알고리즘 과제 3. 동전 거스름돈
(그리디 알고리즘)

제출일 : 2017/04/06

경영학과 201101328　박성환

목차 :

1. 문제 정의

동전 거스름돈을 그리디 알고리즘을 통해 구현한다.

2. 해결방안

그리디 알고리즘의 특징인 '욕심 내어' 최소 또는 최대값을 선택하는 특성에 맞게 동전을 **가장 큰 단위부터 최대한 택하면 동전 수를 최소화할 수 있다.** 본인이 동전 거스름돈을 위해 작성한 함수는 크게 세 파트로 이루어져 있다.

1. **coin_change 함수.**

```
// Main coin change algorithm.
int coin_change(int change){

   int i;
   int coin_unit_set[] = {500, 100, 50, 10, 10, 1};
   int coin_set_size = 6;
   int coin_count_set[6] = {0};
   int total_number = 0;

   for(i = 0; i < coin_set_size; i++)
   {
      count_coin(&change, coin_unit_set[i], &coin_count_set[i]);
      print_coin(coin_unit_set[i], &coin_count_set[i]);
   }

   for(i = 0; i < coin_set_size; i++)
      total_number += coin_count_set[i];

   printf("\nTotal # of coins = %d\n\n", total_number);
}
```

-> 동전 거스름돈 알고리즘의 진입점 함수로서 동전 금액을 인자로 받아 동전의 단위 배열과 개수 배열을 통해 알고리즘을 구현한다. 동전 단위 배열은 내림차순으로 정렬되어 있어 동전 개수를 최소화할 수 있도록 되어 있다.
이 함수는 동전을 세는 함수인 *count_coin* 함수와 과제 요구에 맞게 개수를 출력하는 *print_coin* 함수를 실행한다. 마지막으로는 동전의 총 개수를 출력한다.

2. **count_coin 함수**

```
// Count number of each coin units.(e.g. 500, 100, ..)
void count_coin(int *change, int coin_unit, int *count_unit){

   while(*change >= coin_unit)
   {
      *change -= coin_unit;
      (*count_unit)++;
   }
}
```

-> 그리디 알고리즘으로서의 동전 거스름돈 문제의 꽃인 부분이다. 이 함수는 잔돈 포인터 변수와 동전의 단위, 동전 개수를 세는 포인터 변수를 받아 **단위만큼 최대한 동전을 취한다**. 해당 단위로 더 이상 잔돈을 만들 수 없으면 보다 작은 단위로 넘어갈 것이다.

그리고 이 알고리즘은 개선의 여지가 있다. 유클리드의 두 자연수의 최대공약수를 구하는 알고리즘을 개선할 수 있는 것과 유사한데, 이 알고리즘도 동전 단위를 최대한 가감하는 방법 대신 잔돈을 동전 단위로 나누는 방법도 가능할 것이다. 가령 3400원을 500원으로 6번 빼서 500원 동전을 6개 만드는 것이 아니라 3400을 500으로 나눠 그 몫인 6개의 동전을 바로 만들 수도 있을 것이다. 일단은 교재의 내용대로 진행한다.

3. print_coin 함수

```
// Print coin numbers as symbols.(e.g. (500), (100), ..)
void print_coin(int coin_unit, int *count_unit){

   int i;
   int count = 0;
   for(i = 0;i < *count_unit; i++)
   {
      printf(" (%d) ", coin_unit);
   }
}
```

-> 이 함수는 과제의 내용대로 동전을 개수만큼 출력하는 함수이다. 동전 단위 변수와 동전 개수 포인터 변수를 받아서 개수만큼 단위를 심볼로 표현한다.
(500, 2)와 같이 인자를 받으면 '(500) (500)' 과 같이 동전을 출력한다.

3. 테스트

실행파일을 실행한 뒤 0보다 큰 정수를 입력하면 언제나 문제없이 작동한다. 0을 입력하면 종료되도록 설정했고 그보다 큰 모든 정수에서 정확히 작동한다.

Enter Change: 1
(1)
Total # of coins = 1

Enter Change: 11
(10) (1)
Total # of coins = 2

Enter Change: 21
(10) (10) (1)
Total # of coins = 3

Enter Change: 111
(100) (10) (1)
Total # of coins = 3

Enter Change: 1234
(500) (500) (100) (100) (10) (10) (10) (1) (1) (1) (1)
Total # of coins = 11

4. 결론

그리디 알고리즘을 통해 동전 거스름돈 문제를 쉽게 해결할 수 있지만 한계가 존재한다.

그리디 알고리즘을 통해 동전 거스름돈 알고리즘을 구현할 수 있었다. 그리디 알고리즘은 근시안적인 선택을 통해 부분해를 도출하고 그 부분해들을 취합해 최종해를 구하는 알고리즘 방식이다. 동전 거스름돈에서는 동전 단위마다의 근시안적으로 '가장' 많이 동전을 취해 부분해를 만들고(각각의 개수) 동전들의 개수를 취합해 최종해를 구할 수 있었다. 그리디 알고리즘은 비교적 쉬운 알고리즘으로 특히 이 과제에서는 동전의 개수를 최소화하기 위해 동전의 단위를 큰 단위부터 계산한다는 것만 유념하면 문제 없이 답을 도출할 수 있다.

그러나 난이도와 최적 성능은 trade off 관계에 있기 때문에 그리디 알고리즘을 통해 동전 거스름돈 문제를 해결하는 것은 한계가 있다. 주어진 동전 단위(500, 100, 50, 10, 1)들은 각각 개개 동전 양옆의 동전 단위들의 배수이거나 약수이다. 그래서 언제나 최적의 답을 도출할 수 있다. 그런데 만약 180원과 같은 단위가 등장하면 알고리즘은 먹통이 된다. 180원은 100원의 배수가 아니기 때문에 200원을 입력 받으면 최소의 동전 개수를 구할 수 없다.(정확히는 오답이 나온다.) 이때에는 동적 프로그래밍 방법이 대안이 될 수 있다. 동전 거스름돈을 동적 프로그래밍 방식으로 구현해봐야 할 것 같다.