정렬 알고리즘

Practice 2

시작하기 전에

- 프로그램을 짤때 원하는 어떤 언어 (C/C++, Java, R, python 등)도 사용 가능.
- 프로그램 작성하는 문제는 프로그램도 함께 제출하여야 함.
- 코드를 짤때 코드의 맨위에 자신의 학번 이름을 주석으로 적으시오.
- 코드 문제는 꼭 수업시간에 주어진 그대로 짤 필요 없습니다. 같은 알고리즘도 여러가지 방식으로, 때로는 더 효율적으로 구현도 가능하니 그 알고리즘 내에서 자기가 원하는 방식으로 짜면 됩니다.
- 코드를 돌려서 결과물을 제출하라는 문제는 결과물을 제출할때 화면 캡쳐를 사용할 것. 이는 자신의 코드를 돌렸을때 나온 결과임을 보이기 위함으로 사용하는 언어나 에디터 등등에 따라 다를 수 있으므로 방법은 알아서 제출할 것. 어떤 방식이든 자신의 코드를 돌려서 나온 결과라는 것만 보여주면 됨. 예를들어, 자바 이클립스를 사용하면 이때 자신의 코드의 윗부분(학번 이름과 앞에 코드 5줄정도 포함)이 아래 실행 결과와 같이 캡쳐되도록 하시오.
- **다음 실습전날 저녁 9시까지 eclass에 업로드하고 출력물을 제출하시오.**

순환 알고리즘 VS 비순환 알고리즘

- 순환 알고리즘 (recursive algorithm)
  - A **recursive algorithm** is an **algorithm** which calls itself with "smaller (or simpler)" input values, and which obtains the result for the current input by applying simple operations to the returned value for the smaller (or simpler) input.
  - 쉽게 말해, 자기자신을 부르는 알고리즘.

순환 알고리즘 VS 비순환 알고리즘

- 순환 알고리즘 (recursive algorithm)
  - 예로 수업시간에 배웠던 QuickSort. 함수 내에서 자기자신을 부르고 있음

```
void QuickSort(int A[ ], int Left, int Right) //recursive algorithm
   /* 입력: A[Left:Right + 1], A[Right + 1]:  최대값보다 큰 값.
      출력: A[Left:Right],  A[Right + 1]: dummy */
    {     int k;                                  // position of partition
  1       if (Right  >  Left)
  2         k =  Partition(A[ ], Left, Right + 1);
  3          QuickSort(A[ ], Left, k  - 1);    //분할원소의 왼쪽 퀵정렬
  4          QuickSort(A[ ], k  +  1, Right); //분할원소의 오른쪽 퀵정렬
  5    }
```

순환 알고리즘 VS 비순환 알고리즘

- 비순환 알고리즘 (non-recursive/iterative algorithm)
  - A **iterative algorithm** is an **algorithm** that loops to repeat some part of the code to get the result.
  - 쉽게 말해, 순환(recursion)을 사용하지 않고 결과를 얻는 알고리즘. 종종 for/while과 같은 루프를 사용함.
- 알고리즘의 명확성, 메모리, 시간, 데이타의 종류, 구현의 편의성등에 따라서 종종 순환알고리즘과 비순환 알고리즘으로, 비순환 알고리즘은 순환 알고리즘으로 다시 쓰일 수 있다.

문제 1. 동전 교환문제

- 지난시간에 동전 교환문제의 순환알고리즘인 전역탐색 알고리즘을 배워보았다. 이를 순환알고리즘으로 바꾸어 보자.
- 동전 교환문제 – **필요한 총 동전 갯수만 출력하면 됨**

Coin Change Problem

Problem: Convert some amount of money ***M*** into given denominations, using the smallest possible number of coins.

Input: An amount of money, ***M***, and an array of ***d*** denominations ***c***=(c_1, c_2, … , c_d), in decreasing order of value ($c_1 > c_2 > \ldots > c_d$).

Output: The number of minimum coins needed to change M with c.

Let's define ***k***=(k_1, k_2, … , k_d) as the list of minimum number of coins needed to change M with c. Print $k_1 + k_2 + \ldots, + k_d$

문제 1. 동전 교환 문제

전역탐색 알고리즘(Exhaustive search/Brute force Algorithm)

Examines every possible alternative to find one particular solution

BruteForceChange (***M, c, d***)

smallestNumberOfCoins = ∞

for each (k_1 , k_2 , … , k_d) **from** (0,…,0) **to** (***M***/c_1,…,***M***/c_d)

valueOfCoins = $\sum_{i=1}^{d} c_i k_i$

if *valueOfCoins* = ***M***

numberOfCoins = $\sum_{i=1}^{d} k_i$

if *numberOfCoins* < *smallestNumberOfCoins*

smallestNumberOfCoins = *numberOfCoins*

return (*smallestNumberOfCoins*)

문제 1. 동전 교환 문제 (50)

- 문제 1앞에서 설명한 동전교환문제를 순환 알고리즘으로 구현해 보시오 (부분 점수 없음).

  - 유사코드를 작성하시오.
  - 프로그램을 작성하고 M=11, c=(1,3,5), d=3 의 입력을 넣었을 때와 M=40, c=(25,20,15,10,5,1)의 출력(필요한 동전 갯수출력. 필요한 동전 조합은 출력할 필요 없음)을 제출 하시오. (두경우다 답이 맞았을 경우만 점수 인정)

문제 2. 선택정렬 (50 점)

- 수업시간에 비순환적(iterative) 선택정렬을 구현해 보았다. 이를 순환적(recursive) 선택 정렬로 바꾸어 보자.

1. 비순환적 선택 정렬을 구현해 보아라. (20점, 부분 점수 없음)
   - 유사코드를작성하시오
   - 프로그램을 작성하고 입력 A=[30 20 40 10 5 10 30 15]에 대한 정렬 결과를 출력하여라. 단계별로 A 값이 어떻게 변화되는지 (뭔가 배열에 값이 바뀌거나 각단계마다 A 를 출력하면 됨) 와 최종 결과물을 출력할것.

2. 이를 순환적 선택 정렬로 구현해 보아라. (30점, 부분 점수 없음)
   - 유사코드를작성하시오
   - 프로그램을 작성하고 입력 A=[30 20 40 10 5 10 30 15]에 대한 정렬 결과를 출력하여라. 단계별로 A 값이 어떻게 변화되는지(뭔가 배열에 값이 바뀌거나 각단계마다 A 를 출력하면 됨) 와 최종 결과물을 출력할 것