**컴퓨터 알고리즘과 실습**

* 2회차 실습 -

2017.03.15

컴퓨터공학과

2013112149 박지용

* **실습문제 1번. Recursive형태의 BruteForceChange알고리즘**

<유사코드>

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

for i=0 to d-1 do

k[i] ← M/c[i];

howMuch = sum of (c[]\*k[]);

sum = sum of k[];

if(howMuch == original\_M)

if(sum < smallest\_sum)

smallest\_sum = sum;

if(M/c[i] != 0)

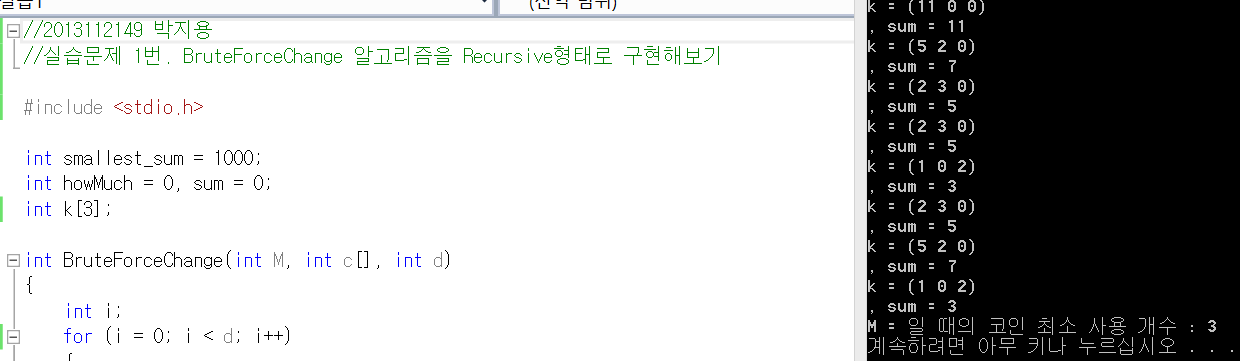
BruteForceChange(M-c[i], c, d);

return smallest\_sum;

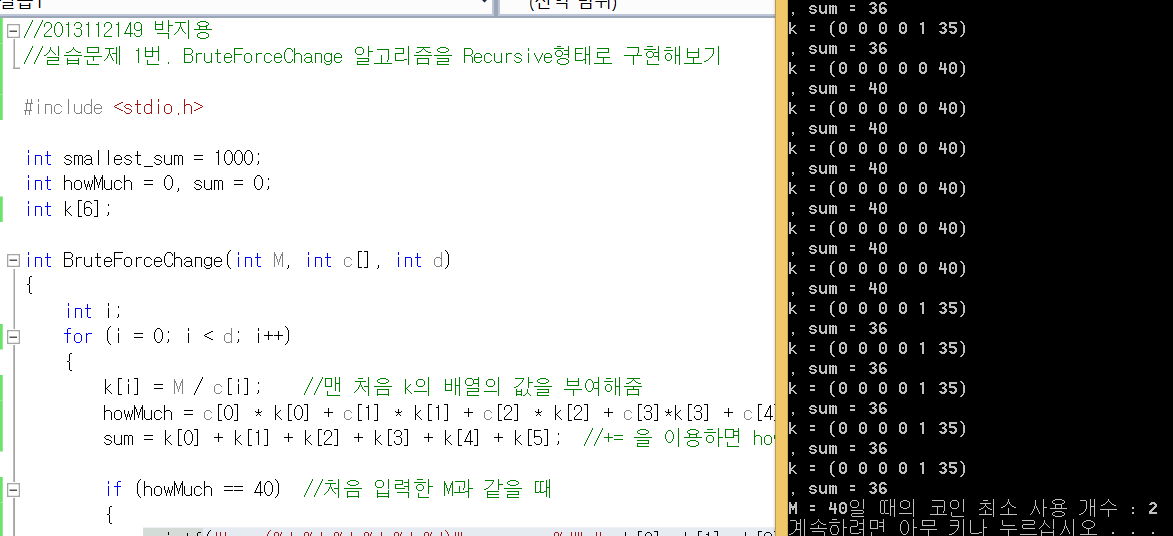
먼저 결과부터 말씀드리자면, 프로그램을 완성시키지 못했습니다. 완성이 안됐다는 얘기는, 기본적으로 전역으로 탐색을 하긴 하는데, c배열의 첫번째 원소(c[0] 값이 어떤 때는 되고 어떤 때는 탐색을 실행하지 않는다는 것을 말씀드리는 겁니다. 또, 계속 잘못된 것 같은 부분을 반복적으로 고치다보니 과정이 매우 복잡해졌습니다. 아직 순환에 대한 이해가 부족한 것 같습니다. 재귀를 하는 코드의 위치만 바꿔도 컴퓨터 안에서 도는 과정이 완전히 바뀌기 때문에 그걸 계산하는데 어려움이 있었습니다.

기본적인 아이디어는 M에서 반복적으로 c[i]를 줄여서 M이 작아진 상태로 탐색하고 k[i]의 index를 0~d-1까지 주면서 탐색하는데, 이를 위해서 k[i] = M/c[i]라는 줄을 넣었습니다. 하지만 k[0]을 구하는 과정에 뭔가 내부적인 오류가 있는 것 같습니다. k배열을 구한 후에는 k배열과 c배열의 값을 합한 것의 합(총 금액)이 처음에 입력한 M(유사코드에서 original\_M으로 표현)과 같을 때 k배열 원소들의 합인 sum이 smallest\_sum보다 작으면 smallest\_sum을 sum으로 바꿔주는 식으로 하였습니다. 코드의 부분적인 설명은 주석으로 달아두었습니다.

일단 input값을 문제에서 주어진 대로 넣었을 때의 결과는 이론적으로 계산할 수 있는 결과와 같습니다. 이 문제를 해결할 수 있는 방법을 계속해서 고민해봐야겠습니다.



M=11, c=(1, 3, 5), d=3 일때의 출력화면입니다.



M=40, c=(25,20,15,10,5,1), d=6 일때의 출력화면입니다.

<소스코드>

//2013112149 박지용

//실습문제 1번. BruteForceChange 알고리즘을 Recursive형태로 구현해보기

#include <stdio.h>

int smallest\_sum = 1000;

int howMuch = 0, sum = 0;

int k[6];

int BruteForceChange(int M, int c[], int d)

{

int i;

for (i = 0; i < d; i++)

{

k[i] = M / c[i]; //맨 처음 k의 배열의 값을 부여해줌

howMuch = c[0] \* k[0] + c[1] \* k[1] + c[2] \* k[2] + c[3]\*k[3] + c[4]\*k[4] + c[5]\*k[5]; //howMuch는 각 코인의 액수와 쓰인 횟수를 곱한 것의 합(총 금액)

sum = k[0] + k[1] + k[2] + k[3] + k[4] + k[5]; //+= 을 이용하면 howMuch와 sum의 값을 다시 초기화시켜야 하는 문제가 발생하므로 직접 입력해줌

if (howMuch == 40) //처음 입력한 M과 같을 때

{

printf("k = (%d %d %d %d %d %d)\n, sum = %d\n", k[0], k[1], k[2], k[3], k[4], k[5], sum);

if (sum < smallest\_sum)

smallest\_sum = sum;

}

if (M / c[i] != 0)

{

BruteForceChange(M - c[i], c, d);

}

}

return smallest\_sum;

}

int main(void)

{

int M = 40, d = 6;

int c[] = { 25, 20, 15, 10, 5, 1 };

printf("M = 40일 때의 코인 최소 사용 개수 : %d\n", BruteForceChange(M, c, d));

return 0;

}

* **실습문제 2번 – (1). 선택정렬 알고리즘**

<문제 2-1 유사코드>

**Selection\_sort(a, n)**

for i ← 0 to n-2 do

min ← a[i];

for(j ← i+1 to n-1 do

if a[j] < min

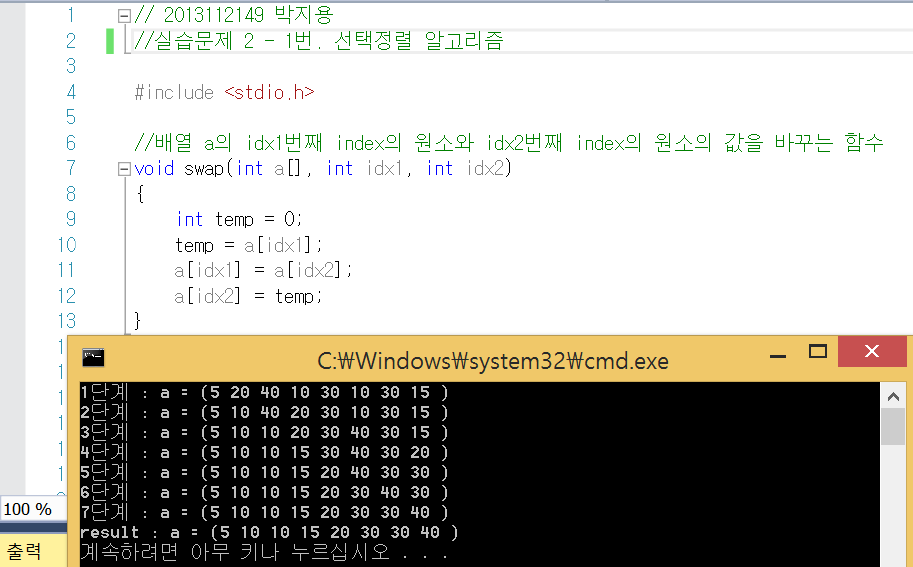
min ← a[j]; idx ← j;

if min < a[i]

swap(a, idx, i);

기본 아이디어는 배열에서 최솟값을 구하는 방식(min이라는 변수에 맨 앞의 원소를 초기값으로 넣어두고 배열의 끝까지 원소를 비교하면서 더 작은 것을 찾아서 min에 저장시키는 방식)을 이용하였는데, 2중 for문으로 첫번째 비교가 끝나면 index를 1 증가시켜서 두번째 원소부터 끝까지, 그 다음은 세번째 원소부터 끝까지 ... 를 비교하여 각각 단계에서의 최솟값을 첫번째 원소와 바꿔줘서 앞에서부터 하나하나 정렬해나가면서 마지막엔 오름차순으로 정렬된 배열이 생깁니다. 유사코드에서 idx(min원소의 index)와 min을 따로 둔 이유는 어느쪽이든 선택에 따라 반환할 수 있도록하기 위해서입니다. swap함수의 input은 배열이름과 int형 정수 두개인데, 이 정수들은 배열내에서 서로 위치를 바꾸고자 하는 원소의 index입니다.

C언어를 이용해 이 알고리즘을 구현해보았습니다. 다음은 출력화면과 소스코드입니다.



소스코드의 내용은 유사코드에 정렬이 실행되는 과정을 출력해주는 부분을 추가한 것 외에 유사코드의 내용과 거의 비슷합니다. 또한 세부 내용을 소스코드 안에 주석으로 자세히 달아놓았기 때문에 생략하겠습니다.

<문제 2-1 소스코드>

// 2013112149 박지용

//실습문제 2 - 1번. 선택정렬 알고리즘

#include <stdio.h>

//배열 a의 idx1번째 index의 원소와 idx2번째 index의 원소의 값을 바꾸는 함수

void swap(int a[], int idx1, int idx2)

{

int temp = 0;

temp = a[idx1];

a[idx1] = a[idx2];

a[idx2] = temp;

}

//선택정렬 알고리즘

void selection\_sort(int a[], int n)

{

int i, j, k, min = 0, idx = 0;

static int count = 0; //count는 출력을 위한 단계를 세주기 위해 만들었음

for (i = 0; i < n - 1; i++)

{

min = a[i]; // i번째 원소부터 배열의 끝까지 비교할 것이므로 a[i]에서 시작

for (j = i+1; j < n; j++)

{

if (a[j] < min) // a[i]보다 작은놈이 나타나면 그놈을 최솟값으로 지정... 반복

{

min = a[j];

idx = j; //idx는 배열 나머지 부분 중 최솟값인 원소의 index

}

}

if (min < a[i])

swap(a, idx, i); /\* 최솟값이 처음 index(= i)의 원소보다 작으면 두 수의 자리를 바꿈

이걸 i가 n-1이 될때까지 반복

\*/

printf("%d단계 : a = (", ++count);

for (k = 0; k < n; k++)

printf("%d ", a[k]); //단계별 출력

printf(")\n");

}

}

int main(void)

{

int a[] = { 30, 20, 40, 10, 5, 10, 30, 15 };

int n = 8, i;

selection\_sort(a, n);

printf("result : a = (");

for(i=0; i<n; i++)

printf("%d ", a[i]); //최종 결과 출력

printf(")\n");

return 0;

}

* **실습문제 2번 – (2). Recursive형태의 선택정렬**

<문제 2-2 유사코드>

**Selection\_sort(a, n)**

min ← a[0];

for i ← 1 to n-1 do

if a[i] < min

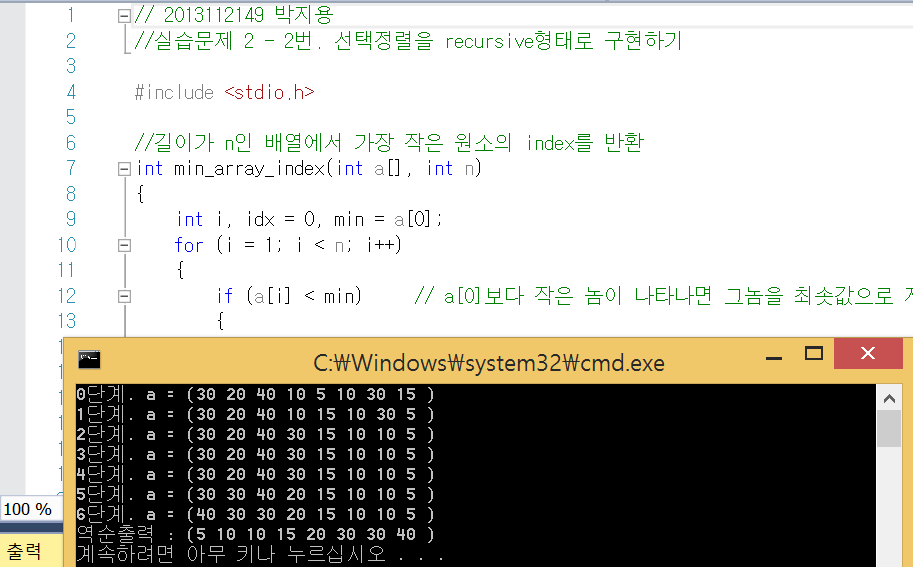
min ← a[i]; idx ← i;

min\_idx ← idx;

swap(min\_idx, n-1);

Selection\_sort(a, n-1)

소스코드내에선 유사코드의 윗부분을 크기가 계속 줄어드는 배열 안에서 가장 작은 원소를 구할 수 있는 min\_array\_index( ) 함수로 대신하였습니다. 전체 코드에 대한 설명은 출력화면 아래 쓰도록 하겠습니다.



문제 2-2번 출력화면입니다. 정렬이 일어나는 과정을 단계적으로 출력하였습니다. Parameter를 하나 더 추가하는 방식으로 하거나 새로운 배열을 만들어서 바뀐 결과를 저장하는 식으로 한다면 정방향으로 만드는 방법으로 생각해보다가 지금 가지고 있는 배열 안에서 정렬시키기 위해 역순으로 정렬하는 아이디어를 생각해봤습니다. 기본적인 선택정렬은 첫번째 원소와 나머지 원소 중 최솟값을 바꾸는데비해 저는 최솟값을 배열의 마지막 자리로 보내고 n을 하나 줄이는 순환호출을 통해 배열의 크기를 1씩 줄이면서 전체 배열 내에서 최솟값을 구하고 바꾸는 것을 반복하는 식으로 정렬해보았습니다.

출력화면의 0단계는 정렬을 시작하기 전의 배열이고 마지막 줄은 결과 배열을 역순으로 출력한 것입니다.

<문제 2-2 소스코드>

// 2013112149 박지용

//실습문제 2 - 2번. 선택정렬을 recursive형태로 구현하기

#include <stdio.h>

//길이가 n인 배열에서 가장 작은 원소의 index를 반환

int min\_array\_index(int a[], int n)

{

int i, idx = 0, min = a[0];

for (i = 1; i < n; i++)

{

if (a[i] < min) // a[0]보다 작은 놈이 나타나면 그놈을 최솟값으로 지정... 반복

{

min = a[i];

idx = i; // 최솟값일 때의 index

}

}

return idx;

}

//배열 a의 idx1번째 index의 원소와 idx2번째 index의 원소의 값을 바꾸는 함수

void swap(int a[], int idx1, int idx2)

{

int temp = 0;

temp = a[idx1];

a[idx1] = a[idx2];

a[idx2] = temp;

}

//선택정렬 알고리즘(재귀)

void selection\_sort(int a[], int n)

{

int i;

static int count = 0;

int min\_idx = 0; // 각 시점별로 가장 작은 원소의 index

if (n > 1)

{

min\_idx = min\_array\_index(a, n); // min\_array\_index 함수를 통해 구한 값을 대입

printf("%d단계. a = (", count++);

for (i = 0; i < 8; i++)

{

printf("%d ", a[i]);

}

printf(")\n");

swap(a, min\_idx, n - 1); // n-1번째 원소(가장 마지막 원소)와 가장 작은 원소를 바꿔줌

selection\_sort(a, n - 1); /\* 배열의 크기를 하나 줄여서 나머지 부분들에서 다시 최솟값을 구해

바꿔주는 식으로 반복 \*/

}

}

int main(void)

{

int a[] = { 30, 20, 40, 10, 5, 10, 30, 15 };

int n = 8, i;

selection\_sort(a, n);

/\* 현재 내가 정렬한 배열은 최솟값을 맨 앞자리와 바꾸는 식으로 반복하는

기존의 선택정렬의 역순으로 구현하였으니 역순으로 출력

원래 순서대로로 구할 수도 있으나 indexing이 복잡하여 이 방식으로 하였음 \*/

printf("역순출력 : (");

for (i = n - 1; i >= 0; i--)

{

printf("%d ", a[i]);

}

printf(")\n");

return 0;

}