

문제해결기법 - 201921725 안성현

할인 행사

<2021/10/31>

문제 소개 및 접근법

1-1] 할인 행사

① 문제

학교에서 할인 행사를 열고 있다. 할인 행사는 손흥민 할인, 류현진 할인을 진행한다. 먼저 손흥민 할인은 모든 상품을 $x\%$ 할인하여 판매하는 방식이다. 류현진 할인은 상품 3개 이상을 구매할 경우 제일 값싼 것은 공짜로 주는 방식이다. 위 두 할인이 동시에 적용되지는 않는다고 한다. 한 학생이 구입할 상품 n 개의 가격과 할인율 x 가 주어질 때, 학생이 상품 n 개를 모두 구입할 때 지불해야할 최소 비용을 구하는 프로그램을 작성하라.

($n \leq 100,000$ 이고 $x \leq 100$ 이다. 첫째 줄에 x 와 n 이 입력되고 둘째 줄에 상품의 가격을 나타내는 양의 정수 n 개가 입력된다. 상품의 가격은 100,000원 이하이고 항상 100으로 나누어진다.)

② 설명

▶ 류현진 할인 방식을 보고 처음 떠오른 것은 상품을 3개씩 묶어야 한다는 것이다. $3+a$ 개씩 묶어도 한 개만 공짜로 주는 것은 동일하기 때문에 굳이 3개보다 많이 묶을 필요가 없다. 그리고 정렬을 할 필요가 있다. 그래야 가장 싼 값을 선택하기가 용이하기 때문이다. 또한 3개씩 묶을 때는 연속된 값들로 묶어야 한다. 상품 가격이 $(a, a+50, a+100, a+150)$ 이라고 할 때, 연속적으로 묶지 않아서 굳이 50 이상 손해를 볼 필요는 없다. [예시: $(a, a+50, a+100 \rightarrow 2a+150)$, $(a, a+50, a+150 \rightarrow 2a+200)$] 손흥민 할인 방식에 대해서는 크게 고려할 사항이 없다. 한 개를 보든지 여러 개를 보든지 특별하게 묶든지 $x\%$ 할인되는 것은 변하지 않기 때문에 특정 방식이 더 이득이라고 볼 수는 없다.

이 문제는 동적계획법의 대표적인 문제이다. 동적계획법은 큰 문제를 풀기 위해 작은 문제의 해를 구하고 저장하는 방식이다. 우리는 n 개의 상품에 대한 최소 비용을 구할 때 $n-1$ 개의 상품에 대한 최소 비용을 이용할 수 있다. ($n-1$ 개의 상품의 최소 비용 + 손흥민 할인된 n 번째 상품 가격)을 구하면 (n 개의 상품의 최소 비용) 후보가 된다. 그리고 n 번째 상품이 오면서 $(n-2, n-1, n)$ 개로 묶은 케이스도 고려해야한다. 예를 들어 상품 가격이 $n=4$, $x=10$, $(300, 100, 100, 100)$ 이라고 하면 $(300, 100, 100, | 100)$ 인 케이스는 [$n-1$ 개의 상품의 최소 비용 + $100 * 0.9$]를 구하는 것이고, $(300, | 100, 100, 100)$ 인 케이스는 $\min([n-3$ 개의 상품의 최소 비용 + $100 + 100], [n-3$ 개의 상품의 최소 비용 + $(100$

소스 코드와 실행 결과

2-1] 소스 코드와 실행 결과

① 코드

```
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h>
#define ull unsigned long long

// 배열의 두 요소의 위치를 바꿔주는 함수
void swap_elements(int* list, int index1, int index2) {
    int temp = list[index1];
    list[index1] = list[index2];
    list[index2] = temp;
}

// 퀵 정렬에서 사용되는 partition 함수
int partition(int* list, int start, int end) {

    int i, b;
    i = b = start;
    int p = end;

    while (i < p) {
        if (list[i] >= list[p]) {
            swap_elements(list, i, b);
            b += 1;
        }
        i += 1;
    }
    swap_elements(list, b, p);
    p = b;

    return p;
}

// 퀵 정렬
int quicksort(int* list, int start, int end) {
    if (end - start < 1)
        return 0;

    int p = partition(list, start, end);
    quicksort(list, start, p - 1);
    quicksort(list, p + 1, end);
}

// 최소 가격 찾기
```

```

ull find_best(int* array, int n, int x) {
    ull* dptable = (ull*)malloc(sizeof(ull)*n); // i개의 상품을 살 때 최소 가격 모음
    (i=1,2,3,...n)

    // dptable 초기 세팅
    dptable[0] = array[0] * (100 - x) / 100;
    dptable[1] = (array[0] + array[1]) * (100 - x) / 100;
    ull candi_1 = array[0] + array[1];
    ull candi_2 = (array[0] + array[1] + array[2]) * (100 - x) / 100;
    dptable[2] = (candi_1 < candi_2) ? candi_1 : candi_2;

    // dptable 활용해서 최소 가격 찾기
    if (n > 3) {
        ull candi1, candi2, candi2_1, candi2_2;
        for (int i = 3; i < n; i++) {
            candi1 = dptable[i - 1] + (array[i] * (100 - x) / 100);
            candi2_1 = dptable[i - 3] + array[i - 2] + array[i - 1];
            candi2_2 = dptable[i - 3] + (array[i-2] + array[i-1] + array[i])
* (100 - x) / 100;
            candi2 = (candi2_1 < candi2_2) ? candi2_1 : candi2_2;
            dptable[i] = (candi1 < candi2) ? candi1 : candi2;
        }
        return dptable[n - 1]; // n개의 상품을 살 때 최소 가격
    }
    else
        return dptable[2]; // 3개의 상품을 살 때 최소 가격
}

int main(void) {
    int n, x; // 상품의 개수, 할인을
    scanf("%d %d", &n, &x);
    int* array = (int*)malloc(sizeof(int)*n); // 상품의 가격 모음

    // 상품의 가격 입력
    for (int i = 0; i < n; i++) {
        scanf("%d", &(array[i]));
    }

    // 케이스에 따라 처리
    if (n <= 0)
        printf("%d\n", 0);

    else if (n == 1)
        printf("%d\n", array[0] * (100 - x) / 100);

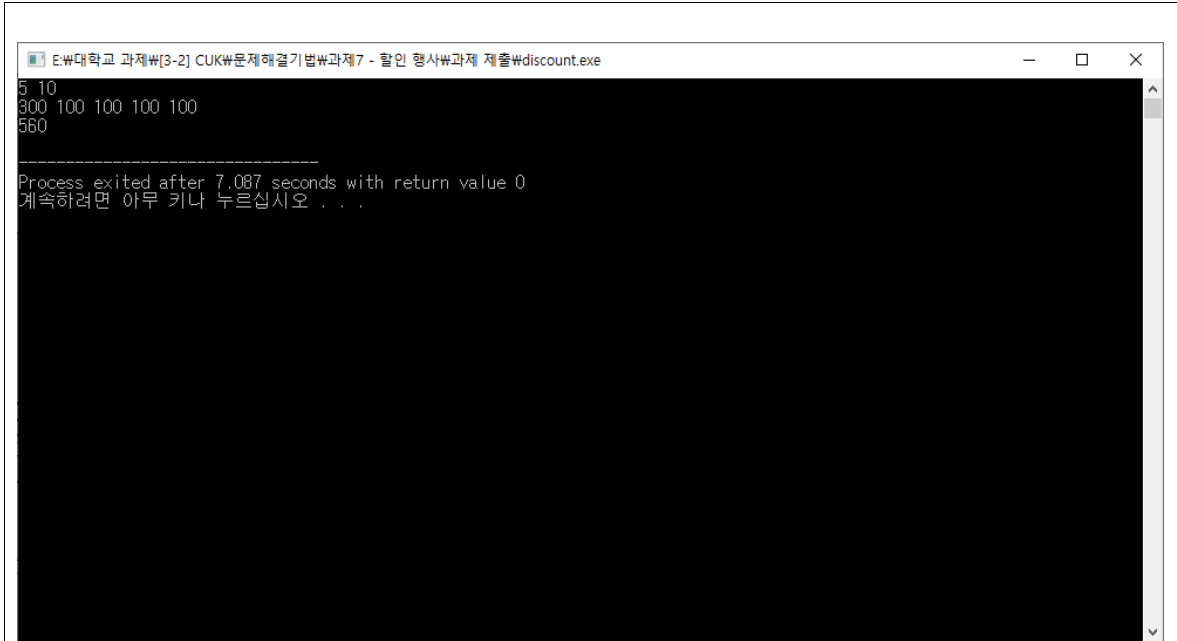
    else if (n==2)
        printf("%d\n", (array[0] + array[1]) * (100 - x) / 100);

    else {
        quicksort(array, 0, n - 1); // 내림차순
        ull sum_price = find_best(array, n, x); // n개의 상품을 살 때 최소 가격
        printf("%llu\n", sum_price);
    }
}

```

```
    return 0;  
}
```

② DEV-C++ 컴파일러 이용 실행 결과



```
E:\#대학교 과제#[3-2] CUK#문제해결기법#과제7 - 할인 행사#과제 제출#discount.exe  
5 10  
300 100 100 100 100  
560  
-----  
Process exited after 7.087 seconds with return value 0  
계속하려면 아무 키나 누르십시오 . . .
```

입력: n(상품의 개수): 5개, x(할인율): 10%, 상품의 가격 n개

출력: 560 (최소 비용)