가톨릭대학교 THE CATHOLIC UNIVERSITY OF KOREA Computer SCIENCE

문제해결기법 - 201921725 안성현

# 스포츠 클라이밍

<2021/10/23>

## 문제 소개 및 접근법

### 1-1] 스포츠 클라이밍

#### ① 문제

스포츠 클라이밍에는 리드, 스피드, 볼더링이 있다. 금, 은, 동메달을 결정하기 위하여 선수들은 세 종목에서 겨루어 종합 순위를 매긴다. 종합 순위는 세 종목에서 거둔 순위를 곱한 점수로 결정된다. 곱한 점수가 낮은 선수가 종합 순위에서 앞선다. 선수 n명의 등번호와 이들이 세 종목에서 거둔 순위가 주어질 때 금, 은, 동메달을 받을 선수를 결정하는 프로그램을 작성하라. 두 선수의 곱한 점수가 같으면 세 종목 순위의 합산 점수가 낮은 선수가 이긴다. 두 선수의 곱한 점수와 합한 점수가 모두 같으면 등번호가 낮은 선수가 이긴다.
(선수의 명수:n은 100이하이고, 등번호:b는 999이하인 서로 다른 양의 정수이다. 종목에서 선수의 순위는 1~n 범위의 정수이다. [중복 가능])

#### ② 설명

▶ 안정정렬(stable sort)을 이용하면 문제를 쉽게 풀 수 있다. 안정정렬은 중복된 값을 입력 순서와 동일하게 정렬한다. 예를 들어, A를 기준으로 오름차순하는데 값이 같으면 B를 기준으로 오름차순하는 문제가 있다고 하겠다. 그러면 B로 안정정렬(오름차순)한 배열을 입력으로 줘서 A로 안정정렬(오름차순)을 하면 된다. A값이 같으면 입력 순서(B기준으로 정렬한 것)와 동일하게 정렬이 된다. 제시된 문제는 '순위의 곱'을 기준으로 오름차순하는데 값이 같으면 '순위의 합'을 기준으로 오름차순하고, '순위의 합'이 값이 같으면 '등번호'로 오름차순하라는 것과 같다. 일단 '순위의 곱'을 mul, '순위의 합'을 sum, '등번호'를 num으로 생각하였다. 이후 [num으로 안정정렬-> sum으로 안정정렬 -> mul로 안정정렬]하였다. 그럼 mul로 오름차순하다가 값이 같으면 sum 기준으로 오름차순이 된다. 이 sum으로 오름차순한 값들 중 값이 같은 것은 num 기준으로 오름차순이 된다. 이렇게 정렬이 완성되면 1,2,3등만 출력하면 된다. 참고로 안정정렬에 해당하는 정렬은 (삽입,버블,병합)이 있다. 필자는 평균 시간복잡도가 O(nlogn)으로 가장 우수한 병합정렬을 채택하였다. 보다 나은 이해를 위해 다음 페이지에 안정정렬의 예시를 작성하겠다.

I) A를 기준으로 오름차순하는데 값이 같으면 B를 기준으로 오름차순하는 문제

1) A:12, B:11
2) A:12, B:13
3) A:15, B:11

-> B 기준으로 안정정렬

1) A:12, B:11
3) A:15, B:11
2) A:12, B:13

-> A 기준으로 안정정렬

1) A:12, B:11
2) A:12, B:13
3) A:15, B:11

* 안정정렬을 이용하지 않으면 A값이 같아도 (1->2)순서를 보장하지 않아 (2-1-3)으로 정렬될 수 있다.

II) mul을 기준으로 오름차순하는데 값이 같으면 sum을 기준으로 오름차순하고, sum의 값이 같으면 num을 기준으로 오름차순하는 문제

1) mul:24, sum:9, num: 1
2) mul:6, sum:5, num: 4
3) mul:6, sum:5, num: 3
4) mul:24, sum:9, num: 2
5) mul:5, sum:7, num:5

-> num 기준으로 안정정렬

1) mul:24, sum:9, num: 1
4) mul:24, sum:9, num: 2
3) mul:6, sum:5, num: 3
2) mul:6, sum:5, num: 4
5) mul:5, sum:7, num:5

-> sum 기준으로 안정정렬

3) mul:6, sum:5, num: 3
2) mul:6, sum:5, num: 4
5) mul:5, sum:7, num:5
1) mul:24, sum:9, num: 1
4) mul:24, sum:9, num: 2

-> mul 기준으로 안정정렬

5) mul:5, sum:7, num:5
3) mul:6, sum:5, num: 3
2) mul:6, sum:5, num: 4
1) mul:24, sum:9, num: 1
4) mul:24, sum:9, num: 2

## 소스 코드와 실행 결과

### 2-1] 소스 코드와 실행 결과

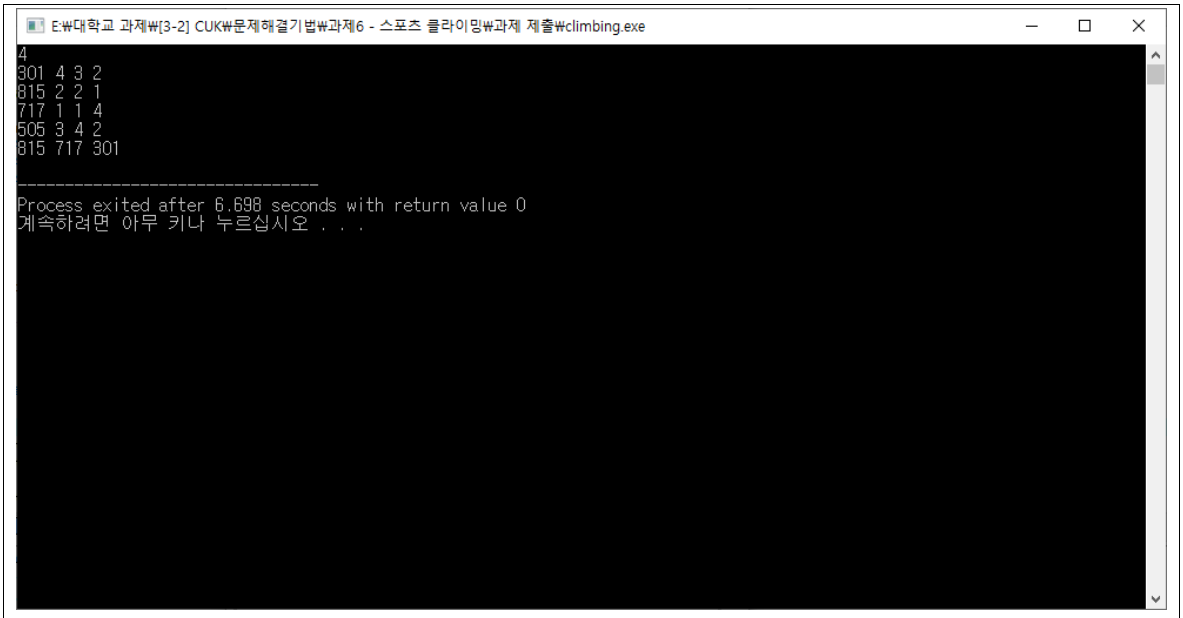
#### ① 코드

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
#pragma warning(disable:4996)

typedef struct Infor {
        int mul; // 점수의 곱
        int sum; // 점수의 합
        int num; // 선수의 등번호
}Infor;

Infor sorted[MAX];

// 합병정렬 (안정정렬에 속함)
void merge(Infor* list, int left, int mid, int right, int code) {
        int i, j, k;
        i = left; // 오른쪽 Part의 시작 위치
        j = mid + 1; // 왼쪽 Part의 시작 위치
        k = left; // 정렬될 리스트의 시작 위치

        // 작은 순서대로 합병 (오른쪽 or 왼쪽 Part가 전부 복사될 때까지)
        while (i <= mid && j <= right) {
                if (code == 0) { // mul을 기준으로 합병
                        if (list[i].mul <= list[j].mul)
                                sorted[k++] = list[i++];
                        else
                                sorted[k++] = list[j++];
                }
                else if (code == 1) { // sum을 기준으로 합병
                        if (list[i].sum <= list[j].sum)
                                sorted[k++] = list[i++];
                        else
                                sorted[k++] = list[j++];
                }
                else { // num을 기준으로 합병
                        if (list[i].num <= list[j].num)
                                sorted[k++] = list[i++];
                        else
                                sorted[k++] = list[j++];
                }
        }
        // 남은 요소 일괄 복사 (오른쪽 Part)
        if (i > mid) {
```

```
                for (int idx = j; idx <= right; idx++)
                        sorted[k++] = list[idx];
        }
        // 남은 요소 일괄 복사 (왼쪽 Part)
        else {
                for (int idx = i; idx <= mid; idx++)
                        sorted[k++] = list[idx];
        }
        // sorted 배열 내용을 list로 복사
        for (int idx = left; idx <= right; idx++)
                list[idx] = sorted[idx];
}

void merge_sort(Infor* list, int left, int right, int code) {
        int mid;
        // 배열의 크기가 2이상일 때 동작
        if (left < right) {
                mid = (left + right) / 2;
                merge_sort(list, left, mid, code);
                merge_sort(list, mid + 1, right, code);
                merge(list, left, mid, right, code);
        }
}

int main() {
        // 선수의 명수
        int n = 0;
        scanf("%d", &n);

        int* b = (int*)malloc(sizeof(int)*n); // 선수의 등번호
        int* p = (int*)malloc(sizeof(int)*n); // 리드 종목 점수
        int* q = (int*)malloc(sizeof(int)*n); // 스피드 종목 점수
        int* r = (int*)malloc(sizeof(int)*n); // 볼더링 종목 점수

        // 입력
        for (int i = 0; i < n; i++)
                scanf("%d %d %d %d", &(b[i]), &(p[i]), &(q[i]), &(r[i]));

        // infor 배열 생성
        Infor* infor = (Infor*)malloc(sizeof(Infor)*n);
        for (int i = 0; i < n; i++) {
                infor[i].mul = p[i] * q[i] * r[i];
                infor[i].sum = p[i] + q[i] + r[i];
                infor[i].num = b[i];
        }

        // 등번호를 기준으로 infor배열 정렬
        merge_sort(infor, 0, n - 1, 2);

        // 점수의 합을 기준으로 infor배열 정렬
        merge_sort(infor, 0, n - 1, 1);

        // 점수의 곱을 기준으로 infor배열 정렬
```

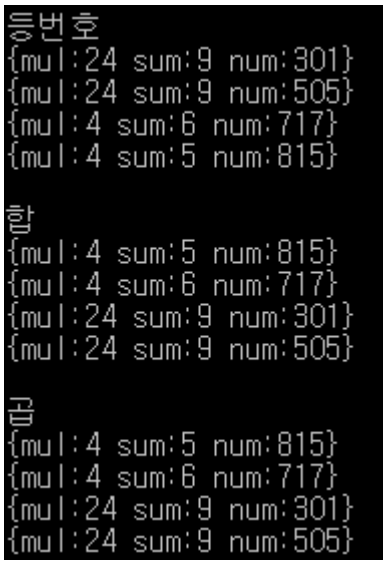
```
        merge_sort(infor, 0, n - 1, 0);

        // 금,은,동 선수 출력
        printf("%d %d %d₩n", infor[0].num, infor[1].num, infor[2].num);
}
```

## ② DEV-C++ 컴파일러 이용 실행 결과




**입력:** 선수의 명수(n), 각 선수의 등번호와 세 종목의 순위

**과정:**




(등번호->순위 합->순위 곱으로 정렬)

**출력:** 금(815), 은(717), 동(301)