

문제해결기법 - 201921725 안성현

평균 원소 구하기

<2021/09/18>

문제 소개 및 접근법

1-1] 평균 원소 구하기

① 문제

원소가 모두 양의 정수인 길이가 n 인 배열 A 가 있다. 이 배열의 i 번째부터 j 번째까지 원소들로 이루어진 배열 $A[i], \dots, A[j]$ 를 부분 배열이라고 하고 $A[i,j]$ 로 나타낸다. 이때 $i \leq j$, $0 \leq i$, $j \leq n-1$ 이 성립한다. 부분 배열 $A[i,j]$ 에 대하여, 이 부분 배열에 속한 $j-i+1$ 개의 원소의 평균값과 원소의 크기(값)가 정확히 같은 원소를 평균 원소라고 부른다. 양의 정수 n 과 배열 원소의 크기들이 입력으로 주어질 때, 이 배열의 부분 배열 중에서 평균 원소를 가지는 것의 개수를 구하라.

(배열은 $n \leq 2000$ 을 만족하고 각 원소들은 1,000 이하인 양의 정수이다.)

② 접근법 (소스 간략 설명)

▶ 이 문제를 풀기 위해서는 (1)부분 배열의 합을 효율적으로 구하고 (2)각 부분 배열 합에 대해서 평균값(average)을 구하고 (3)부분 배열 중에 평균값의 크기를 지닌 원소가 있는지 확인하면 된다. 부분 배열 합을 빨리 구하려면 누적 계산을 할 줄 알아야 한다. ThisSum 변수의 값이 현재 부분 배열의 합이라고 하겠다. 처음에는 $A[0,0]$ 의 합이 ThisSum에 들어간다. 이후에는 $A[0,1]$ 의 합이 들어간다. 즉 $A[i,j]$ 라고 생각하면 j 값을 $j+1$ 로 업데이트하고 ThisSum에 $A[j]$ 를 더하는 누적 과정이 필요하다. 이 작업을 j 가 $n-1$ 일 때까지 진행하면 ThisSum은 $A[i,n-1]$ 의 합을 구한 것이 된다. 이제 j 값이 더 커질 수 없으니 누적 계산도 불가능하다. 하지만 모든 부분 배열을 구한 것이 아니라 현재 i 를 기준으로 한 부분 배열을 구했을 뿐이다. 그렇기에 ThisSum을 0으로 업데이트하고 i 도 $i+1$ 로 업데이트해서 다음 부분 배열을 구하면 된다. 이 과정을 i 가 $n-1$ 일 때까지 진행하면 모든 부분 배열의 합을 구할 수 있다. 물론 이 문제는 ThisSum을 구하는 것에서 끝나지 않는다. 고로 ThisSum값을 누적할 때마다 평균값을 구하고 평균 원소가 존재하는지 찾아야 한다. 이 과정은 나눗셈 연산과 선형 탐색으로 해결하면 된다. 전체 과정에 대해서는 간단한 알고리즘으로 설명하겠다. [1. 사전식 순서로 부분 배열을 한 개 찾고 합을 구하기 -> 2. 해당 부분 배열의 평균값을 구하기 -> 3. average가 양수이면 해당 부분 배열에 average크기를 지니는 원소가 있는지 구하기 -> 4. (3)을 만족하면 평균 원소의 개수를 저장하는 num에 1을 더해 업데이트하기 -> (1~4)과정을 모든 부분 배열을 찾을 때까지 반복하기]

소스 코드와 실행 결과

2-1] 소스 코드와 실행 결과

① 코드

```
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h>
#define MAX 1001;

typedef struct subarray {
    int* list;
    int size;
}subarray;

// 순차 탐색
int sequentialSearch(int* list, int n, double x) {
    int idx = 0;
    while (idx < n && list[idx] != x)
        idx++;
    if (idx < n)
        return 1;
    else if (idx == n)
        return 0;
}

// subarray 초기화
void sub_init(subarray* s, int n) {
    s->size = 0;
    s->list = (int*)malloc(sizeof(int)*n);
    int i;
    for (i = 0; i < n; i++)
        s->list[i] = MAX;
}

// 평균 원소의 개수 찾기
int find_average_num(int A[], int n) {
    subarray* s = (subarray*)malloc(sizeof(subarray));

    int ThisSum;
    double ThisAverage;
    int average_num = 0;

    int i, j;
    for (i = 0; i < n; i++) {
        sub_init(s, n);
        ThisSum = 0;
        for (j = i; j < n; j++) {
```

```

        s->list[s->size++] = A[j];
        ThisSum = ThisSum + A[j];
        ThisAverage = (double)ThisSum / (double)(j - i + 1);

        if (ThisAverage / (int)ThisAverage == 1) {
            if (sequentialSearch(s->list, s->size, ThisAverage)) {
                average_num++;
            }
        }
        free(s->list);
    }
    return average_num++;
}

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

    int* array = (int*)malloc(sizeof(int) * n);

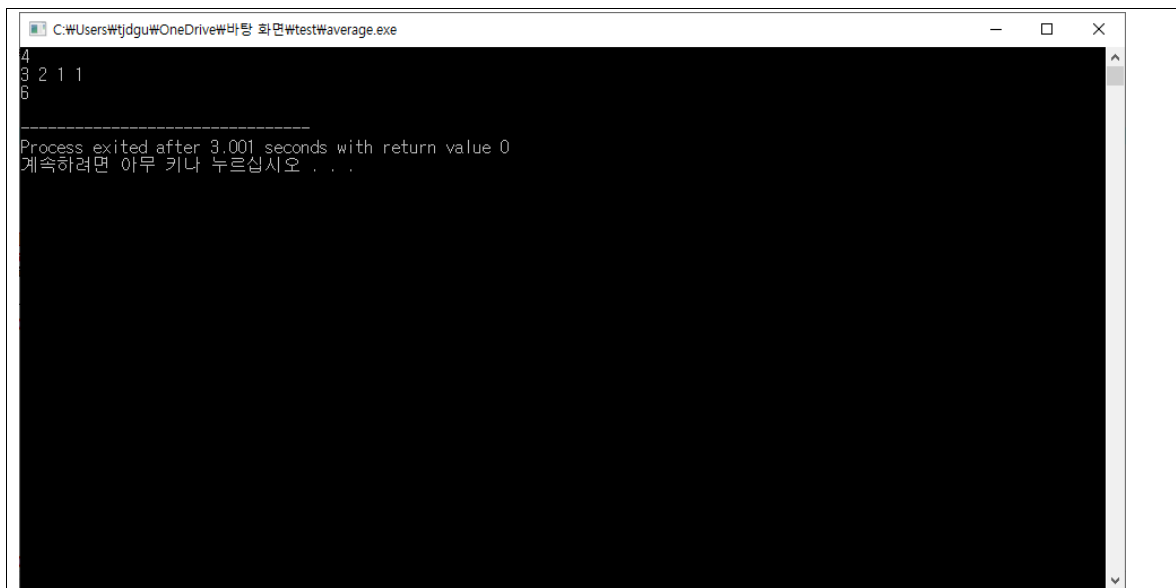
    int i;
    for (i = 0; i < n; i++) {
        scanf("%d", &array[i]);
    }

    printf("%d\n", find_average_num(array, n));

    return 0;
}

```

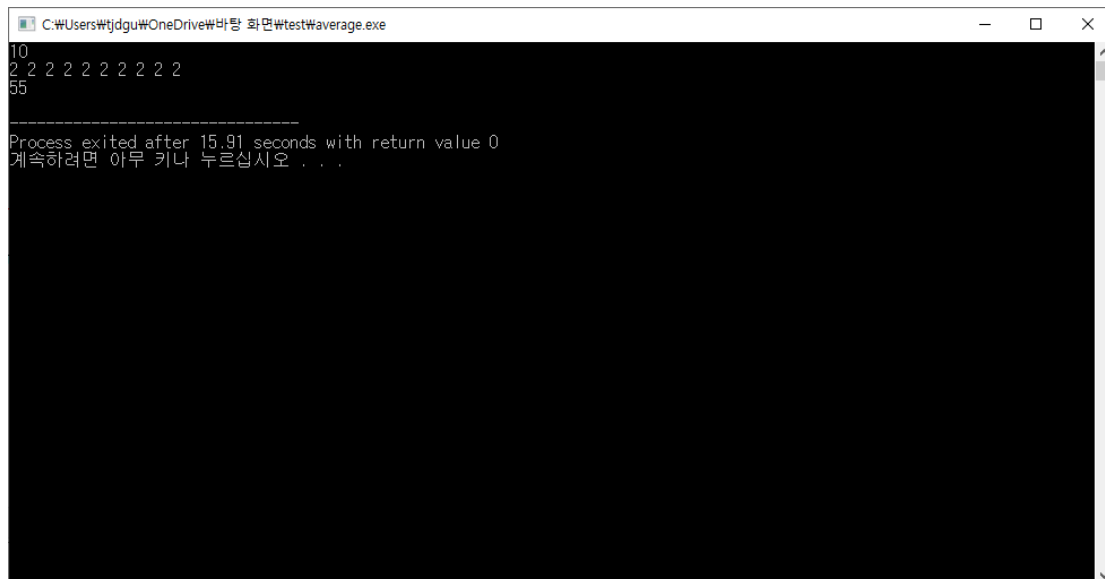
② DEV-C++ 컴파일러 이용 실행 결과



```
C:\Users\tjdgu\OneDrive\바탕 화면\test\average.exe
4
3 2 1 1
6
-----
Process exited after 3.001 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

입력: $n=4$, 원소={3,2,1,1} / 출력: 6

→ $A[0,0], A[0,2], A[1,1], A[2,2], A[2,3], A[3,3]$



```
C:\Users\tjdgu\OneDrive\바탕 화면\test\average.exe
10
2 2 2 2 2 2 2 2 2 2
55
-----
Process exited after 15.91 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

입력: $n=10$, 원소={2,2,2,2,2,2,2,2,2,2} / 출력: 55

→ 모든 경우 ($10+9+\dots+2+1=55$)

→ $(A[0,0], A[0,1], \dots, A[0,9]), (A[1,1], A[1,2], \dots, A[1,9]), \dots, (A[8,8], A[8,9]), (A[9,9])$