자료구조 과제 #1

경제통상학과 18012248 신민기

1) 실험에 사용한 실행 코드

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <Windows.h>

double *prefixAvg1(int *arr, int N);    //O(N^2)
double *prefixAvg2(int *arr, int N);    //O(N)
```

```c
int main()
{
    srand(time(NULL));
    int n, i, j, arrSize;
    int *arr;
    printf("Enter size of array N : ");
    scanf("%d", &n);    // 배열 사이즈 n 입력
    for (i = 1; i <= 10; i++) {
        arrSize = i * n;    // 테스트할 배열 사이즈는 n, 2n, ,,, 10n
        printf("Size of Array : %d\n", arrSize);

        arr = (int *)malloc(sizeof(int) * (arrSize));   // 테스트 배열 할당
        for (j = 0; j < arrSize; j++) {
            arr[j] = rand() % 10000 + 1;    // 테스트 배열 원소에 난수 할당
        }
        LARGE_INTEGER ticksPerSec;
        LARGE_INTEGER start, end, diff;

        // 느린 함수 호출 시간 측정
        QueryPerformanceFrequency(&ticksPerSec);    //타이머 주파수 저장
        QueryPerformanceCounter(&start);    //start 시간
        prefixAvg1(arr, arrSize);           //함수 호출
        QueryPerformanceCounter(&end);      //end 시간

        diff.QuadPart = end.QuadPart - start.QuadPart;  //함수 호출 시간
        printf("Time of prefixAvg1 : %.12lf sec\n", ((double)diff.QuadPart / (double)ticksPerSec.QuadPart));
```

```c
        // 빠른 함수 호출 시간 측정
        QueryPerformanceFrequency(&ticksPerSec);
        QueryPerformanceCounter(&start);
        prefixAvg2(arr, arrSize);
        QueryPerformanceCounter(&end);

        diff.QuadPart = end.QuadPart - start.QuadPart;
        printf("Time of prefixAvg2 : %.12lf sec\n\n", ((double)diff.QuadPart / (double)ticksPerSec.QuadPart));

        free(arr);
    }
    return 0;
}
```

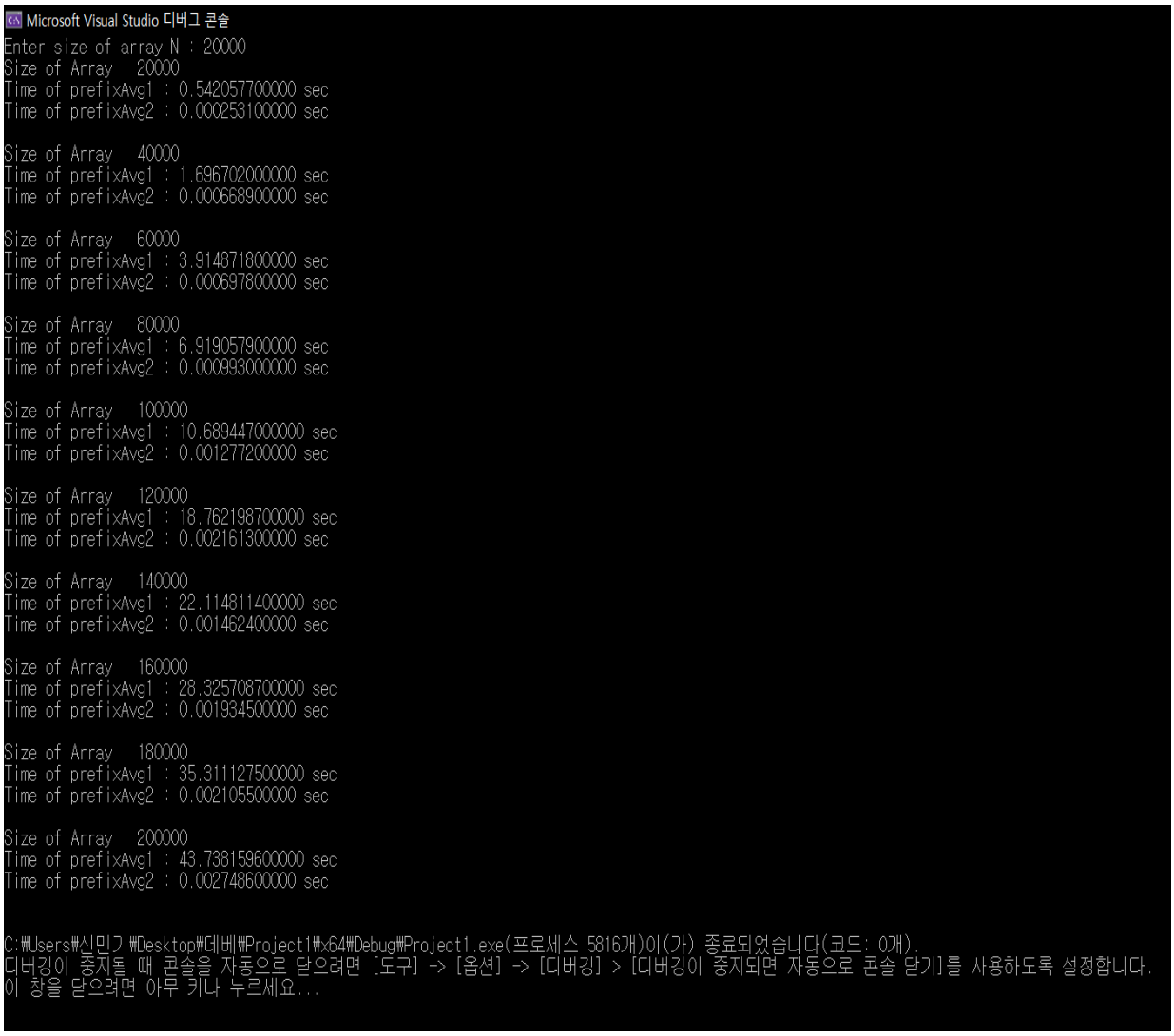
```c
double *prefixAvg1(int *arr, int N) {   //느린 함수
    int i, j, sum;
    double *arrAvg;
    arrAvg = (double *)malloc(sizeof(double) * N);

    for (i = 0; i < N; i++) {
        sum = 0;
        for (j = 0; j <= i; j++) {
            sum += arr[j];
        }
        arrAvg[i] = sum / (double)(i + 1);
    }
    return arrAvg;
}
double *prefixAvg2(int *arr, int N) {   //빠른 함수
    int i;
    double *arrAvg;
    int *sum = (int *)malloc(sizeof(int) * N);
    arrAvg = (double *)malloc(sizeof(double) * N);

    for (i = 0; i < N; i++) {
        if (i == 0)
            sum[i] = arr[i];

        else {
            sum[i] = arr[i] + sum[i - 1];
        }
        arrAvg[i] = sum[i] / (double)(i + 1);
    }
    return arrAvg;
}
```

2) 코드 실행한 화면 캡쳐


Microsoft Visual Studio 디버그 콘솔
Enter size of array N : 20000
Size of Array : 20000
Time of prefixAvg1 : 0.542057700000 sec
Time of prefixAvg2 : 0.000253100000 sec
Size of Array : 40000
Time of prefixAvg1 : 1.696702000000 sec
Time of prefixAvg2 : 0.000668900000 sec
Size of Array : 60000
Time of prefixAvg1 : 3.914871800000 sec
Time of prefixAvg2 : 0.000697800000 sec
Size of Array : 80000
Time of prefixAvg1 : 6.919057900000 sec
Time of prefixAvg2 : 0.000993000000 sec
Size of Array : 100000
Time of prefixAvg1 : 10.689447000000 sec
Time of prefixAvg2 : 0.001277200000 sec
Size of Array : 120000
Time of prefixAvg1 : 18.762198700000 sec
Time of prefixAvg2 : 0.002161300000 sec
Size of Array : 140000
Time of prefixAvg1 : 22.114811400000 sec
Time of prefixAvg2 : 0.001462400000 sec
Size of Array : 160000
Time of prefixAvg1 : 28.325708700000 sec
Time of prefixAvg2 : 0.001934500000 sec
Size of Array : 180000
Time of prefixAvg1 : 35.311127500000 sec
Time of prefixAvg2 : 0.002105500000 sec
Size of Array : 200000
Time of prefixAvg1 : 43.738159600000 sec
Time of prefixAvg2 : 0.002748600000 sec
C:₩Users₩신민기₩Desktop₩데베₩Project1₩x64₩Debug₩Project1.exe(프로세스 5816개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...

3) 표, 그래프, 수행 시간 분석에 대한 설명

1. $O(N^2)$ 함수 분석 (prefixAverage 1)

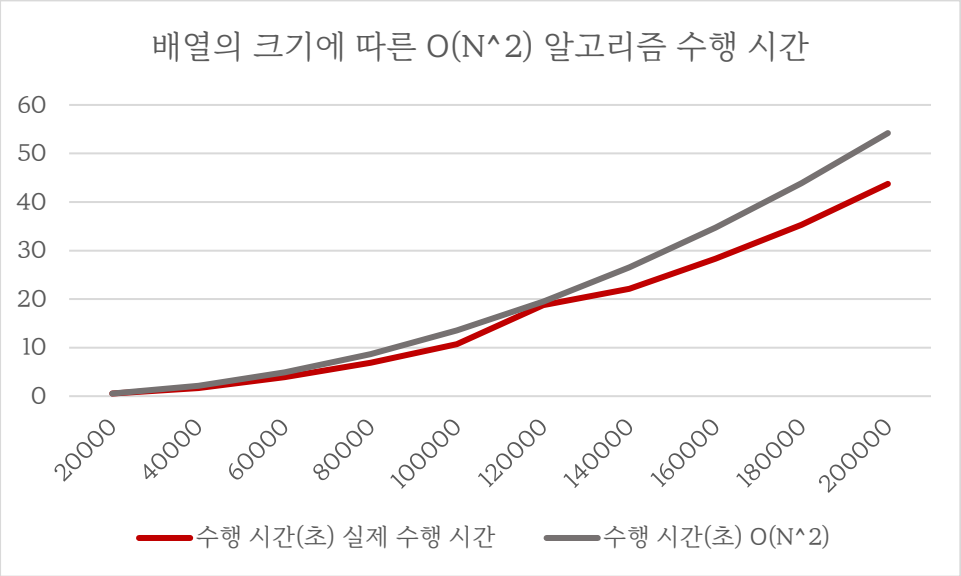
배열 크기	실제 수행 시간(초)
20000	0.542
40000	1.697
60000	3.915
80000	6.919
100000	10.689
120000	18.762
140000	22.115
160000	28.326
180000	35.311
200000	43.738

시간복잡도 $O(N^2)$ 함수는 배열의 크기가 0일 때, 수행 시간이 0일 것이므로 원점을 지난다. 이론적인 $y = \alpha x^2$ 의 형태를 하고, (20000, 0.542)의 점을 지난다고 가정한다. $\alpha = \frac{0.542}{20000^2} = 0.000000001355$ 가 나온다. 이론적인 함수의 형태는 $y = 0.000000001355x^2$가 된다. 이 함수에 배열의 크기 20000, 40000, ... ,200000을 대입해 이론적 수행 시간을 구한다.

	수행 시간(초)	
배열 크기	실제 수행 시간	O(N^2)
20000	0.542	0.542
40000	1.697	2.168
60000	3.915	4.878
80000	6.919	8.672
100000	10.689	13.55
120000	18.762	19.512
140000	22.115	26.558
160000	28.326	34.688
180000	35.311	43.902
200000	43.738	54.2

실제 수행 시간과 이론적 수행 시간에 대한 그래프를 그려 본다.

- x축 : (개)
- y축 : (초)



실제와 이론적 수행 시간의 오차가 커지는 모습이 있지만 두 그래프 모두 배열의 크기가 커짐에 따라 수행 시간 소요가 가파르게 커지는 것을 확인할 수 있었다.

2. O(N) 함수 분석 (prefixAverage 2)

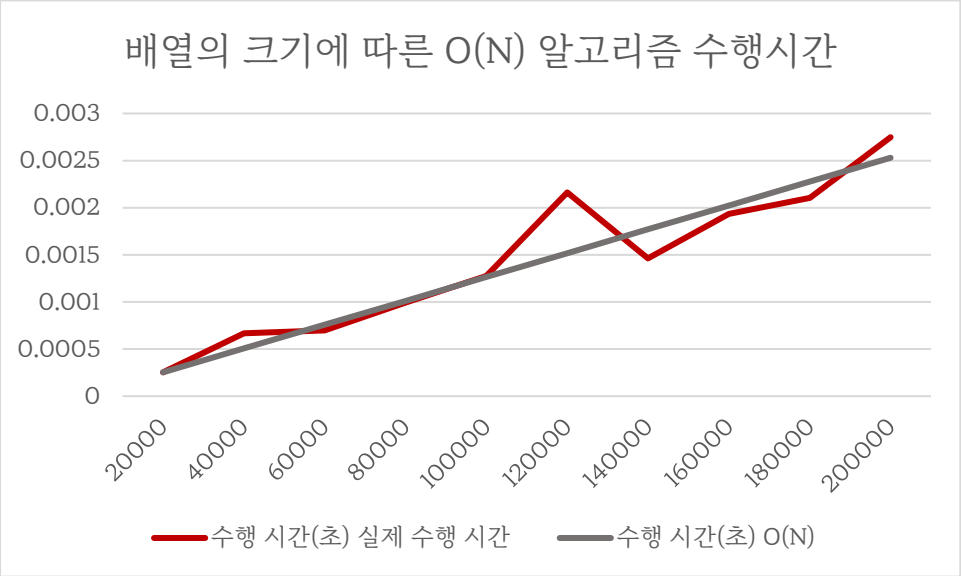
배열 크기	실제 수행 시간
20000	0.000253
40000	0.000668
60000	0.000697
80000	0.000993
100000	0.001277
120000	0.002161
140000	0.001462
160000	0.001934
180000	0.002105
200000	0.002748

시간복잡도 O(N) 함수는 배열의 크기가 0일 때, 수행 시간이 0일 것이므로 원점을 지난다. 이론적인 $y = \alpha x$ 의 형태를 하고, (20000, 0.000253)의 점을 지난다고 가정한다. $\alpha = \frac{0.000253}{20000} = 0.00000001265$ 가 나온다. 이론적인 함수의 형태는 $y = 0.00000001265x$ 가 된다. 이 함수에 배열의 크기 20000, 40000, ... ,200000을 대입해 이론적 수행 시간을 구한다.

	수행 시간(초)	
배열 크기	실제 수행 시간	O(N)
20000	0.000253	0.000253
40000	0.000668	0.000506
60000	0.000697	0.000759
80000	0.000993	0.001012
100000	0.001277	0.001265
120000	0.002161	0.001518
140000	0.001462	0.001771
160000	0.001934	0.002024
180000	0.002105	0.002277
200000	0.002748	0.00253

이 역시 실제 수행 시간과 이론적 수행 시간에 대한 그래프를 그려 본다.

- x축 : (개)
- y축 : (초)



마찬가지로 실제와 이론적 수행 시간의 오차가 있다. 특히 배열의 크기가 120,000일 경우가 배열의 크기가 140,000일 경우보다 알고리즘 수행 시간이 더 길었다. 하지만 전반적으로 실제 수행 시간은 이론적 수행 시간을 따라 간다.