

정렬 알고리즘의 수행시간 비교

이다예 (2019-07)

수행시간 측정

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <time.h>

void insertion_sort(int *arr, int n);
void reverse_array(int *arr, int start, int end);

int main() {
    int N, *arr = NULL;
    int i;
    LARGE_INTEGER frequency;
    LARGE_INTEGER t1, t2;
    double elapsed_time;

    QueryPerformanceFrequency(&frequency);

    scanf("%d", &N);
    arr = (int *)malloc(N * sizeof(int));
    if (arr == NULL) return -1;
    srand(time(NULL));
    for (i = 0; i < N; i++)
        arr[i] = rand();
    //insertion_sort(arr, N); //<< 정렬된 데이터 삽입시 주석처리 제거
    //reverse_array(arr, 0, N - 1); //<< 역정렬 데이터 삽입시 주석처리 제거

    QueryPerformanceCounter(&t1);
    // Sort
    QueryPerformanceCounter(&t2);
    elapsed_time = (t2.QuadPart - t1.QuadPart) * 1000.0 / frequency.QuadPart;
    printf("Time by Sort: %lf ms\n", elapsed_time);
```

```
    free(arr);
}

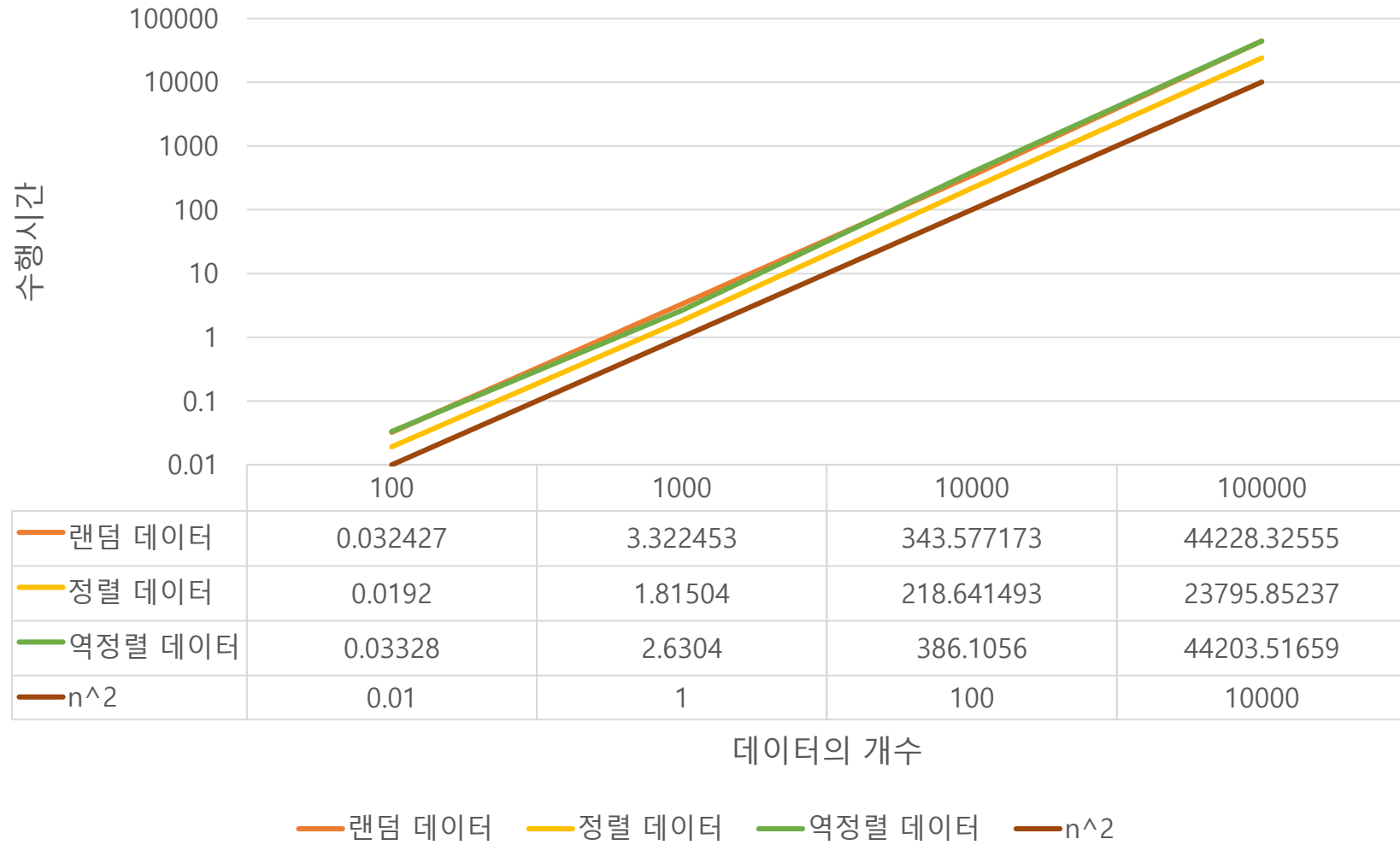
void insertion_sort(int *arr, int n) {
    int tmp;
    int *p, *q;

    for (p = arr + 1; p < arr + n; p++) {
        tmp = *p;
        for (q = p; q > arr; q--) {
            if (*(q - 1) < tmp) break;
            *q = *(q - 1);
        }
        *q = tmp;
    }
}

void reverse_array(int *arr, int start, int end) {
    int temp;
    while (start < end) {
        temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}
```

Bubble Sort

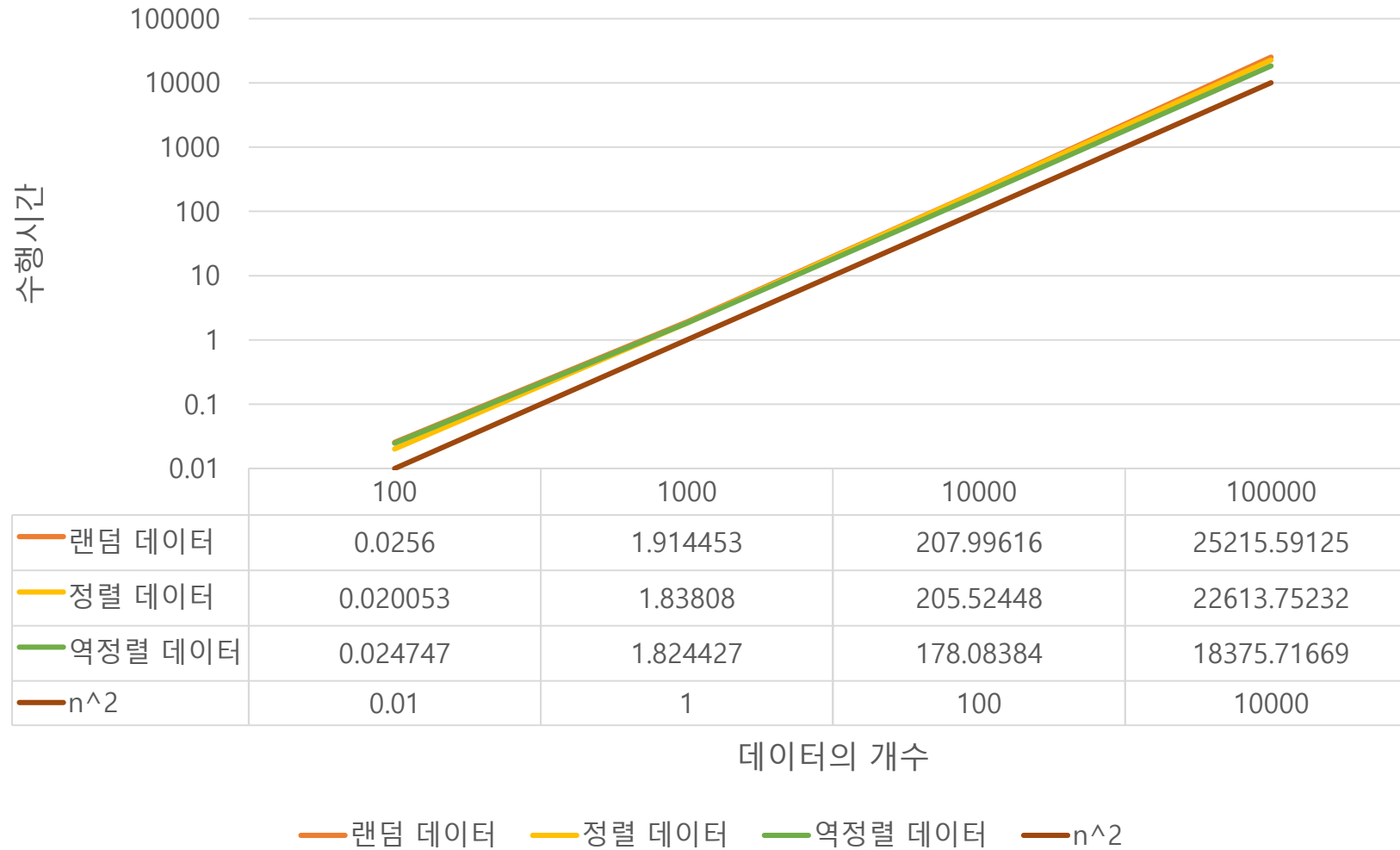
Bubble Sort



Bubble Sort	랜덤 데이터	정렬 데이터	역 정렬 데이터
시간 복잡도	$O(n^2)$	$O(n^2)$	$O(n^2)$

Selection Sort

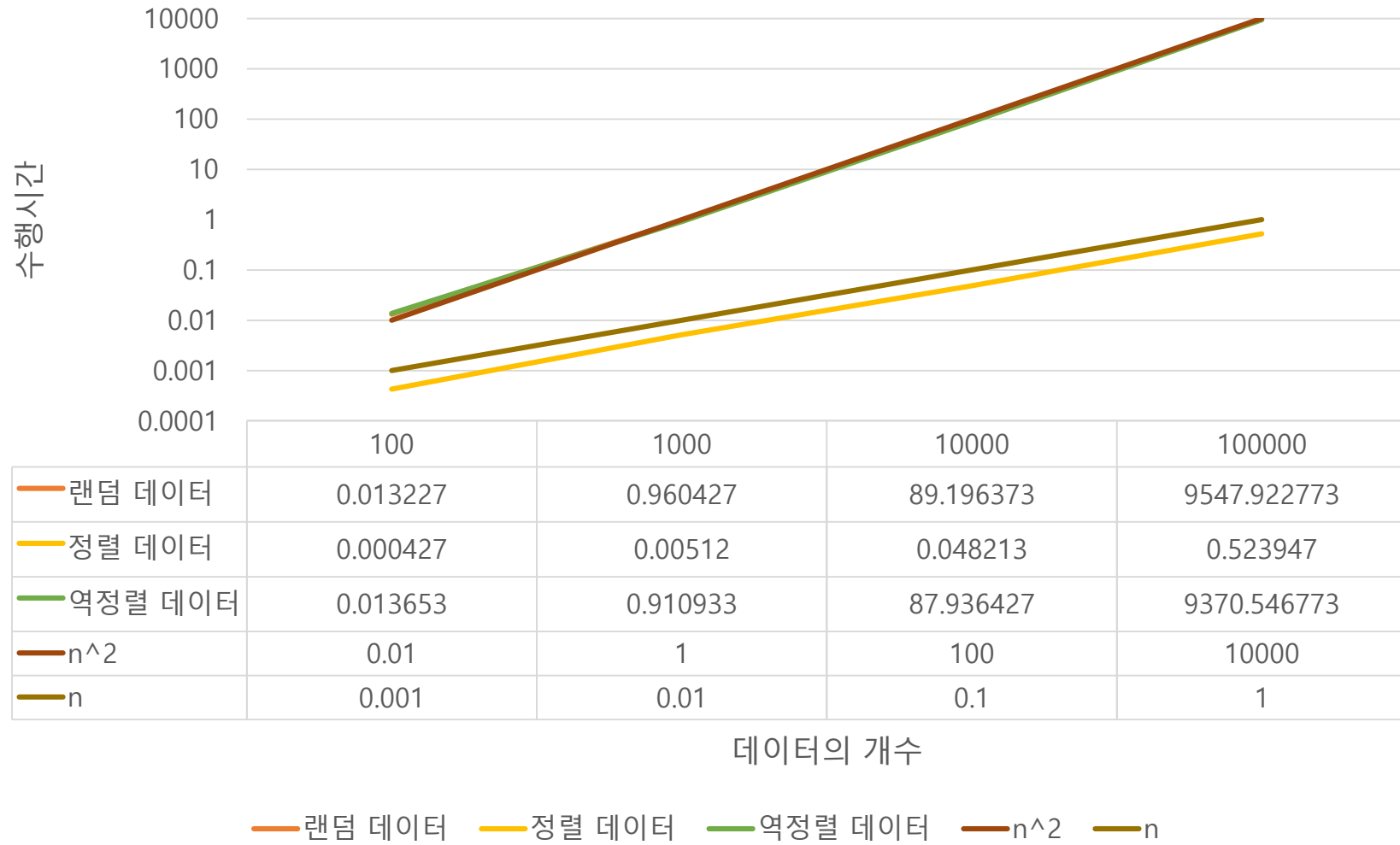
Selection Sort



Selection Sort	랜덤 데이터	정렬 데이터	역 정렬 데이터
시간 복잡도	$O(n^2)$	$O(n^2)$	$O(n^2)$

Insertion Sort

Insertion Sort



Insertion Sort	랜덤 데이터	정렬 데이터	역 정렬 데이터
시간 복잡도	$O(n^2)$	$O(n)$	$O(n^2)$

Merge Sort

Merge Sort



Merge Sort	랜덤 데이터	정렬 데이터	역 정렬 데이터
시간 복잡도	$O()$	$O()$	$O()$

Quick Sort

Quick Sort



Quick Sort	랜덤 데이터	정렬 데이터	역 정렬 데이터
시간 복잡도	$O()$	$O()$	$O()$