

# HW12 과제: 정렬 성능 비교

**제출 데드라인:** 과제 부여 후 2주 후 24시 까지

- 금요일 실습 시 2주 후 목요일 23:59:59까지

**질문**

pemds81718@gmail.com

**제출파일**

- sort.h
- hw12.cpp
- 학번.tex
- 학번.pdf
- + 보고서에 첨부한 이미지 파일들 및 컴파일에 필요한 모든 파일
- ! 파일명 유의

# 1. 문제

정렬 알고리즘의 구현 및 성능 분석 과제입니다. 책에 소개된 정렬 알고리즘들을 구현하고 임의의 리스트에 대한 실행시간을 측정하여 각 알고리즘의 복잡도를 분석하고 이해하는 것이 목표입니다. 구현할 정렬 알고리즘은 다음과 같습니다.

- (1) 삽입 정렬(Insertion Sort) (원서 8판 기준 14.4 장)
- (2) 빠른 정렬(Quick Sort) (원서 8판 기준 15.3 장)
- (3) 자연 합병 정렬(Natural Merge Sort) (원서 8판 기준 15.1 장)
- (4) 힙 정렬(Heap Sort) (원서 8판 기준 22.3 장)

# 2. 공지사항

- a. 소스파일(헤더파일 포함), 결과보고서를 함께 클래스룸에 submit
- b. 결과보고서는 과제의 이해도를 평가하는 중요한 척도입니다. 구현 과정과 분석결과를 최대한 자세히 작성해주세요.
- c. 라이브러리의 **sort 함수 이용할 수 없습니다**. 알고리즘을 직접 구현해야 합니다.

### 3. 요구사항

다음과 같이 작동하는 프로그램을 만들고자 합니다.

**hw12 <테스트케이스파일\_개수;T> 파일명 1 파일명 2 파일명 3 ...**

실행파일 뒤의 첫번째 인자는 정렬 테스트케이스 파일 개수 T 입니다.  
그 뒤로 T 개의 파일명이 나옵니다.

예시) T=10 의 경우 10 개의 파일명이 뒤따라옵니다. (partially\_sorted.txt)

```
[pem_ta@localhost hw11]$ ./hw12 10 t50.txt t100.txt t200.txt t300.txt t500.tx
t t1000.txt t3000.txt t5000.txt t10000.txt t30000.txt
T=10
--INS--|--QUICK--|--NATMG--|--HEAP--|
0.00000|0.00001|0.00001|0.00001|N=50
0.00000|0.00002|0.00000|0.00002|N=100
0.00001|0.00008|0.00000|0.00004|N=200
0.00002|0.00016|0.00000|0.00006|N=300
0.00003|0.00045|0.00001|0.00011|N=500
0.00006|0.00171|0.00001|0.00039|N=1000
0.00019|0.01537|0.00003|0.00080|N=3000
0.00031|0.04365|0.00004|0.00139|N=5000
0.00059|0.18234|0.00008|0.00295|N=10000
0.00183|0.98785|0.00013|0.00505|N=30000
```

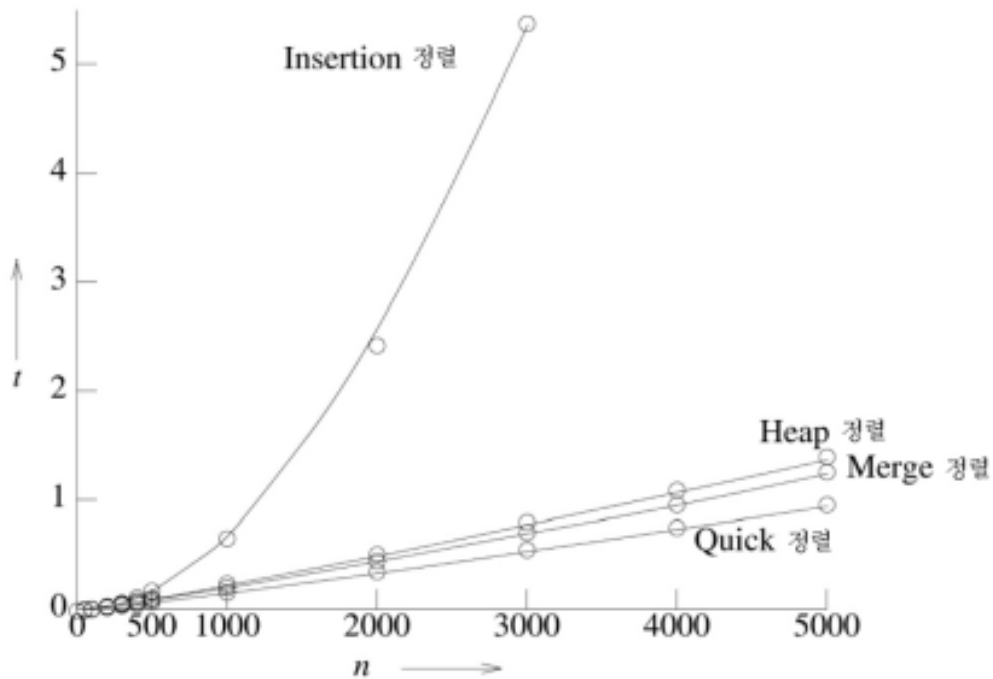
1) 실행 시 위와 같이 각 파일별로 실행시간을 측정하여 출력합니다. 단위는 초(s)입니다.(출력 부분은 이미 작성되어 있습니다.)

2) 총 4 종류의 데이터 폴더가 제공됩니다. 각 폴더는 50~100000 개 사이 데이터를 담은 txt 파일로 이루어져 있습니다. 폴더당 txt 파일은 총 15 개입니다.

- \* random\_t : 데이터가 랜덤으로 위치
- \* partially\_sorted\_t : 부분적으로 정렬된 데이터
- \* sorted\_t : 완전 정렬된 데이터
- \* decreasing\_t : 거꾸로 정렬된 데이터

각 txt 파일의 첫번째 값은 파일 내의 총 데이터 수를 의미합니다. 나머지 값은 데이터입니다.

3) 엑셀 또는 자신이 원하는 프로그램을 사용해서 아래와 같이 그래프를 그립니다. 직접 그려도 상관 없습니다. 이때 각 데이터 폴더당 하나의 그래프를 그립니다. 즉, **총 4 개의 결과 캡처와 4 개의 그래프가 결과보고서에 첨부**되어야 합니다.



4) 데이터 폴더의 모든 txt 파일을 이용할 필요는 없습니다. 단, **그래프를 그리기 위해 충분한 파일 수**를 이용해야 합니다.

## 4. 코드

### a. hw12.cpp

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <stdlib.h>
#include <sys/time.h>
#include "sort.h"
using namespace std ;
int main (int argc , char*argv []) {
    int T = atoi (argv [1]); // num of test case
    cout <<"T="<<T <<endl ;

    int N ; // 각 테스트케이스 별 레코드의 길이
    int i ; // iterator

    double result[4]; // result 배열에 각 알고리즘 별로 실행시간을 저장하게 됩니다.
    /*
    * result[0]: insertion sort
    * result[1]: quick sort
    * result[2]: natural merge sort
    * result[3]: heap sort
    */
    struct timeval start_t , end_t ;
    double diff_t ;
    if (argc <3 ) {
        cerr <<"wrong argument count"<<endl ;
        return 1 ;
    }
    cout <<"--INS--|--QUICK--|--NATMG--|--HEAP--|"<<endl ;
    for (i = 2 ; i <T + 2 ; i ++ ) {
        // i번째 인자의 파일을 읽습니다.

        // 각 정렬 알고리즘에 필요한 자료구조를 생성하고 데이터를 담습니다.

        // 여기부터 정렬 시간 측정을 시작합니다.
        /* example
        gettimeofday(&start_t, NULL);
        삽입정렬 수행
        gettimeofday(&end_t, NULL);
```

```

diff_t = (double)(end_t.tv_sec-start_t.tv_sec)+
((double)(end_t.tv_usec-start_t.tv_usec)/1000000);
result[0] = diff_t;
*/

/* 결과를 출력합니다. (이 부분은 수정하지 않습니다) */
cout.precision(5);
cout <<fixed ;
for (int j = 0 ; j <4 ; j ++ ) {
    cout <<result [j ] <<"|";
}
cout <<"N="<<N <<endl ;
}
}

```

b. sort.h // 자유롭게 구현

## 5. 결과보고서 내용

- 1) 과제를 진행한 환경의 OS 종류와 CPU, 메모리 사양을 작성해주세요.
- 2) 각 알고리즘별로 섹션을 만들어 작동방식을 자세히 설명하세요.
- 3) 테스트 결과(캡처 및 그래프)에 대해 분석하세요.
- 4) 만약 부분적으로 구현 못한 부분이 있다면, 어디서 막혔고 원인 등을 분석하여 작성하세요.

## 6. 주의사항

- a. 라이브러리의 **sort 함수 이용할 수 없습니다**. 알고리즘을 직접 구현해야 합니다.
- b. Cheating 은 F.
- c. 클래스룸이 12 시간 이상 오류 나지 않는다면 기간 연장은 없습니다.
- d. 기본적으로 메일로 제출된 과제는 읽지 않습니다.
- e. 제출 마감시간 기준으로 4 시간 전부터 마감까지 제출이 불가능하다면, 이메일로 제출도 받습니다.
- f. 하지만, 제출 마감시간 15 분 전부터 마감시간까지 제출이 가능했다면, 이메일로 제출된 것은 읽지 않습니다.
- g. 지각 제출은 제출일로부터 일주일까지 허용하고, 점수는 50%만 반영됩니다.