Homework 5: Comparison of Sorting Algorithms

Data Structures, Konkuk Univ.

수업 시간에 배운, 여러 가지 sorting algorithm (오름차순)들을 구현하여 수행시간을 분석하시오. 즉, ①bubble sort (adaptability를 위한 수정 버전), ② selection sort, ③ insertion sort, ④ merge sort, ⑤ quick sort를 수업 슬라이드에 나오는 코드를 기반으로 구현하고 아래와 같은 실험을 진행합니다.

(1) array의 크기가 증가함에 따라서, 평균의 경우 (random data) 각 sorting algorithm의 time complexity를 측정해 보자.

- **n=10,000**인 array에 정수 data를 random하게 대입한다.
- 각 sorting algorithm이 같은 array를 sort해야 하기 때문에, 위의 array를 5개의 set로 만든다.
- 각 sorting algorithm이 sorting을 수행한 시간을 순차적으로 측정한다.
- **n=20,000, 30,000, ..., n=100,000**일 때, 같은 실험을 반복 실행하여, **각 알고리즘의 time complexity를 비교하여 결론을 제시**하고, **알고리즘들의 수행 속도 순위에 대한 결론**도 제시하시오.

(0) (1)번 실험을 진행하기 전에, 각자 구현한 sorting algorithm들이 정확히 동작하는지를 보이시오. (n=30 정도)

(2) (1)번 실험은 데이터들이 random하게 섞여 있는 경우를 가정하였다. 그러나, 몇몇 알고리즘들은 데이터의 상태에 따라 알고리즘의 수행속도가 달라질 수 있다. 이러한 점을 실험하여 제시하시오.

- Bubble sort (adaptability를 위한 수정 버전)의 경우, sorting이 된 데이터를 입력함으로써, O(n)의 time complexity를 보이시오 (n=10,000 ~ n=100,000). 실험 결과는 머신의 성능에 따라서, 실행시간이 0 milisec으로 나와서 O(n)인지 정확히 확인이 되지 않을 수도 있는데, 이 경우는 그대로 제출하시오. (대신, (1)번 실험에서 같은 크기의 random data보다는 훨씬 빠른 시간에 sorting이 끝나는 것을 확인하시오.)
- Insertion sort도 bubble sort와 동일.
- Quick sort의 경우는 이미 sorting된 데이터를 입력받으면, 성능이 안 좋아져서 $O(n^2)$의 성능을 보임을 제시하시오 (n=10,000 ~ n=100,000).
- Randomized quick sort를 구현하여, sorting된 데이터를 입력받아도 O(n*logn)의 성능을 보임을 확인하시오 (n=10,000 ~ n=100,000). 이 경우, pivot을 맨 앞의 데이터로 사용하는 것이 아니라, random index를 발생시켜서, 그 index의 데이터와 맨 앞의 데이터를 swap

한 후, 일반 quick sort를 진행하는 방법으로 구현한다.

제출방법:

(1) **소스 파일**(***.c 또는 ***.cpp)과 **보고서**를 ecampus에 제출하시오. 소스 파일 맨 윗쪽에 코멘트로 학번과 이름, 실행환경(Windows, Mac, Linux)을 명시할 것. **작성한 function들이 몇 번 문제에 해당하는 function인지 대한 설명을 코멘트를 반드시 남기고, 수행시간을 printf할 때도 어떤 문제의 수행시간인지를 구별할 수 있도록 하시오.**