Homework Assignment 8

Due by November 15, 9:00 PM

We want to investigate the performances of the sorting algorithms we discussed in the class. Find `sorting-comparisons.c` in `hw8.zip`, in which the mergesort is implemented so that you can count

(a) the number C_1 of key comparisons,

(b) the number C_2 of writing operations on the array to be sorted,

while it sorts the first $N = 100, 200, \ldots, 900$ data read from the standard input. Also included in the zip file is `one-million.in` which contains one million random 32bit integers. So, if you run the following command

```
SHELL> sorting-comparisons < one-million.in
```

a table of C_1 and C_2 for N's is printed. For this homework,

1. In addition to mergesort, implement selection sort, insertion sort, and quicksort similarly so that you can make the table of C_1 and C_2 for each of these sorting algorithms. For example, with the input of `one-million.in`, you will have the output as follows:

```
** Selection Sort **
n = 100; C1 = 4950; C2 = 198.
n = 200; C1 = 19900; C2 = 398.
...

** Insertion Sort **
n = 100; C1 = 2643; C2 = 2742.
n = 200; C1 = 10507; C2 = 10706.
...

** Merge Sort **
n = 100; C1 = 535; C2 = 1344.
n = 200; C1 = 1290; C2 = 3088.
...

** Quicksort **
n = 100; C1 = 579; C2 = 352.
n = 200; C1 = 1621; C2 = 730.
...
```

   For a fair comparison, make sure that all the sorting algorithms get the same inputs.

2. Discuss about the result and performance characteristics of the algorithms. Your discussion must include

   (a) Insertion sort is better than selection sort in terms of the number of key comparisons. Is Insertion sort is always better than selection sort in overall performance?

   (b) For a random data, or rather usually, does quicksort really run in $O(n \log n)$ time, even though its worst-case running time is $O(n^2)$?

   (c) Why does quicksort usually run faster than mergesort?