EE3980 Algorithms

Homework 1. Quadratic Sorts

Due: Mar. 13, 2021

Sorting algorithms rearrange a set of items into an increasing or decreasing order. Many sorting algorithms have been proposed and have been used in various applications in our daily lives. In this homework, you are requested to implement four sorting algorithms, analyze and compare their performance. The four sorting algorithms are: **selection sort**, **insertion sort**, **bubble sort**, and **odd-even sort**.

The **selection sort** has been discussed in the class and also been shown as **Algorithm (1.1.3)** in the class handout.

The **insertion sort** is shown below.

```
// Sort A[1 : n] into nondecreasing order.
// Input: array A, int n
// Output: array A sorted.
1 Algorithm InsertionSort(A, n)
2 {
3     for j := 2 to n do {  // Assume A[1 : j − 1] already sorted.
4         item := A[j];  // Move A[j] to its proper place.
5         i := j − 1;  // Init i to be j − 1.
6         while i ≥ 1 and item < A[i] do {  // Find i such that A[i] ≤ A[j].
7             A[i + 1] := A[i];  // Move A[i] up by one position.
8             i := i − 1;
9         }
10        A[i + 1] = item;  // Move A[j] to A[i + 1].
11    }
12 }
```

The **bubble sort** is as follows.

```
// Sort A[1 : n] into nondecreasing order.
// Input: array A, int n
// Output: array A sorted.
1 Algorithm BubbleSort(A, n)
2 {
3     for i := 1 to n − 1 do {  // Find the smallest item for A[i].
4         for j := n to i + 1 step −1 do {
5             if A[j − 1] > A[j] then {  // Swap A[j] and A[j − 1].
```

```
6                   t = A[j];
7                   A[j] = A[j − 1];
8                   A[j − 1] = t;
9               }
10          }
11      }
12 }
```

And, the **odd-even sort** is as follows.

```
   // Sort A[1 : n] into nondecreasing order.
   // Input: array A, int n
   // Output: array A sorted.
 1 Algorithm OddEvenSort(A, n)
 2 {
 3     done := false ;
 4     while not done do {
 5         done := true ;
 6         for i := 1 to n − 1 step 2 do {
 7             if A[i] > A[i + 1] then {
 8                 done := false ;
 9                 t = A[i]; A[i] = A[i + 1]; A[i + 1] = t;
10             }
11         }
12         for i := 0 to n − 1 step 2 do {
13             if A[i] > A[i + 1] then {
14                 done := false ;
15                 t = A[i]; A[i] = A[i + 1]; A[i + 1] = t;
16             }
17         }
18     }
19 }
```


Your assignment is to write a C program that contains four functions:

```
void SelectionSort(char **list,int n);
void InsertionSort(char **list,int n);
void BubbleSort(char **list,int n);
void OddEvenSort(char **list,int n);
```

where the `list` is an array of `character pointers`, and the size of the array is defined by the other parameter `n`. All four function perform in-place ordering, that is, the `list` array is

rearranged to an increasing order.

To test these four functions, a `main` function should also be implemented as follows.

```
// main function for Homework 1.
// Input: read wordlist from stdin,
// Output: sorted wordlist and CPU time used.
Algorithm main(void)
{
    Read n and a list of n words and store into A array ;
    t0 := GetTime();
    repeat 500 times {
        Copy array A to array list;
        SelectionSort(list, n);
    }
    t1 := GetTime();
    t := (t1 − t0)/500;
    write (sorting method, n , CPU time) ;
    Repeat CPU time measurement for other sorts ;
    write (sorted list ) ;
}
```

Nine files are also provided for you to measure the performance of the algorithms. They are `s1.dat`, `s2.dat`, …, `s9.dat`. The first line of each file is an integer, n, which is the number of words contained in the file. It is followed by n lines with one word on each line.

Example of program execution is shown below.

```
$ a.out < s1.dat
Selection sort: N = 10 CPU = 2.98023e-07 seconds
Insertion sort: N = 10 CPU = 1.87874e-07 seconds
Bubble sort: N = 10 CPU = 2.49863e-07 seconds
OddEven sort: N = 10 CPU = 3.00407e-07 seconds
1 anemometry
2 cates
3 cincture
4 homebuilder
5 preestablish
6 roccellaceae
7 seedbed
8 speedboat
9 synclinal
10 unamusing
```

Please execute the program for each data file to get the CPU times and compare the performance using the nine wordlist files.

The time complexities of these four algorithms should be analyzed and compare that to the measured CPU time.

Notes.

1. One executable and error-free `C` source file should be turned in. This source file should be named as `hw01.c`.
2. A report file in `pdf` format is also needed. This file should be named as `hw01a.pdf`.
3. Submit your `hw01.c` and `hw01a.pdf` on EE workstations using the following command:

   ```
   ~ee3980/bin/submit hw01 hw01.c hw01a.pdf
   ```

   where `hw01` indicates homework 1.
4. Your report should be clearly written such that I can understand it. The writing, including English grammar, is part of the grading criteria.
5. In comparing two strings, the following library function in the `<string.h>` package can be used.

   ```
   int strcmp(const char *s1, const char *s2);
   ```