Programming Assignment 2
CMPE 250, Data Structures and Algorithms, Fall 2010

Instructor: A. T. Cemgil
TA’s: Ali Haydar Özer, Umut Şimşekli, Cihat İmamoğlu

Due: 20 January 2011, 14:00 Sharp

1. Problem Definition
In this programming assignment, you will write a C program to create random strings and then sort them. First, your program will get two parameters: number of strings, \( N \), and maximum length of any string, \( K \), from the command line. Then, it will create \( N \) strings that will be composed of no more than \( K \) English lowercase characters. Consequently, it will sort the strings lexicographically (i.e smaller first, see important points for a definition) using heapsort. Then, it will sort the strings in reverse-lexicographically using quicksort. Finally, it will print these out into a sorting.out file as unsorted, lexicographically sorted, reverse lexicographically sorted, respectively.

Assignment
- Write an ANSI C program that does what’s described above.

Input Format(from command line)
- Two parameters: number of strings, \( N \), and maximum length of any string, \( K \).

Output Format(sorting.out)
- There will be \( 3 \times N + 1 \) lines in your output. In the the first line, the will be \( N \) and \( K \).
- The next \( N \) lines will have the random strings your program created.
- The next \( N \) lines will be the lexicographically sorted list of your random strings.
- The next \( N \) lines will be the reverse-lexicographically sorted list of your random strings.

Example Input

3 5

Example Output

3 5
xyzkk
ab
abc
Important Points

- It is very important that you follow exactly the given input/output format.
- You have to use both heapsort and quicksort. You may not use just one of them to sort, and then simply output the reversed version.
- You have to use ANSI C. You may not use C++ features such as classes, templates, Standard Template Library, new/delete, references, operator overloading, etc. Your program must compile with gcc, the gnu C compiler.
- You have to dynamically allocate memory for your strings and arrays, i.e you may not make a preliminary assumption about the size of your arrays and strings and define them statically.
- A string $S$ is greater than a string $T$ lexicographically if $T$ is a proper prefix of $S$, or if $S$ has a greater character at the first position where the strings differ.

Hints and Aims

- The goal of this project is to introduce basic string, I/O, dynamic memory allocation, pointers, command line parameter parsing, random number generation operations in C. Besides this programmatical aim, you are expected to get familiar with implementing heaps, heapsort and quicksort algorithms. Beware, though, although the main logic in both of these may be simple, it is actually notoriously hard to be able to write a perfect quicksort.
- You need to use malloc, free, rand, srand, fopen, fprintf, time standard library calls, defined in stdlib.h, time.h, stdio.h.
- Be careful when dealing with pointers and dynamic memory allocation. Both may create errors that can be hard to reproduce, find and debug.

2. Bonuses

- Make your quicksort more robust and efficient by shuffling the input first (Why?). Make sure that your shuffling algorithm perfectly and fairly distributes all permutations (look for Knuth-Fisher-Yates shuffle, for instance).
- Use a practically faster sorting algorithm (such as insertion sort for little inputs) joint with quicksort, by applying the former under a number of threshold of elements.

Submission & Grading

- What to submit
  - You should submit your project in electronic form.
  - You should run your program using there different inputs.
You should compress your source code (.cpp and .h files) and three output files in a zip file, name it as [hw] [#] [student ID] (e.g. hw 2 2000700803.zip), and email to bucmpe250@gmail.com.

Your zip file should NOT contain any executable file (.exe) or any folder. If you sent multiple e-mails, only the last one will be taken into account.

Projects that do not contain any output files will not be run even if they work correctly.

- **Grading**
  - As this programming project is significantly more involved, we will grade it over 6 rather than 3 as announced earlier.
  - Your program will be graded based on the quality of your source code and correctness of your output. Correctness of your output will be tested automatically so make sure you stick with the format described above.
  - Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not overcomment, or make your variable names unnecessarily long.