SCSSE, University of Wollongong
CSCI203 Data Structures and Algorithms
Spring 2014

Assignment 2 (Due 11:59 PM Thurs 11th Sept)

Objectives

explore how different sorting algorithms are better/worse with particular types of data
research and/or design an appropriate sorting algorithm for the given data and implementation timeframe
implement a sorting algorithm
consider actual performance rather than asymptotic(big-Oh) performance

Important

Your program should compile to produce a program named `sortbook` on the linux labs in 3.127 using the command:
```
make
```

Your program must run using the command:
```
sortbook <inputfile> <outputfile> <timesfile> <ntests>
```

So `sortbook A.txt B.txt times.txt 100` should cause your program to read the text `A.txt`, write the sorted words to `B.txt` and output the sorted times for each of 100 tests in `times.txt`.

As well as the source code, there are a couple of other things to submit:
You should provide your makefile called makefile and you should provide your algorithm design choices in design.pdf

The data

The data you will be sorting is english text, stripped to remove punctuation. An example text 308-0.txt is provided, but is not the same as the text that will be used for testing. The text is composed entirely of words, separated by whitespace. A **word** is defined as a contiguous sequence of non-whitespace characters (so `it's` is one word, but `J W Arrowsmith` is three words).

Sorting

The primary sort order should be the number of characters in the word. The secondary sort order should be lexicographical order as you get using strcmp.

So sorting the words: `This is a list of unsorted words`
should give: `a is of This list words unsorted`

Sorting Algorithm

You can use any sorting algorithm, but you must implement it yourself, no code from elsewhere. No calls to standard library sorting algorithms for this part. No need for multithreading; if for some reason you want to write it multithreaded, then the time must be the sum of times for all threads.[1] Local machine only—no distributing work to other machines. Focus on designing/researching an efficient algorithm and coding it clearly and well.

Write a short description explaining what algorithm you chose and your reasoning behind it. If you researched anything, include references (I'm not fussy on referencing style, as long as we can find them—links are ok). If you tested more than one algorithm, talk about that. If you considered particular properties of the data or the algorithms, talk about that. This should not be more than a page.

Timing

To ensure consitency of timing, a skeleton cpp file containing timing code has been provided as `skel.cpp`. The sort times are in nanoseconds.

Not timed

You are allowed to make an initial pass through the input file, and gather a constant amount of data (e.g. the number of characters in the largest word), before reading in the words and storing it in an array. You may strip extraneous whitespace if you wish, but you may *not* partially sort the data while reading it in. Once the data is read in, that array should remain unchanged for the rest of the program.

You should also define (but not copy data into) the temporary data structure(s) which you will use for sorting. This may be any data structure(s) you want to use.

The Timed Bit

Each test should consist of copying the data from the array into your temporary data structure and sorting the data using your sorting code.

Not timed

Finally, the sorted words from the final test run should be extracted from their temporary data structure and output with a space after each word except for every 10th word which is followed by a newline. A sample output file has been provided. Your sort routine is considered correct if your output file matches mine (if you run diff on your answer and my answer you should get no differences).

[1]If you can do a parallel sort on gpus, then you probably deserve the marks (but you won't get them—focus on the algorithms).

Submission instructions

Submit your assignment by typing the command:
```
submit -u <username> -c CSCI203 -a 2 <sourcefiles> makefile design.pdf
```
from the directory containing your files, where `<username>` is your SOLS username and `<sourcefiles>` are all the source files needed for your program.

1. Late submissions will be marked with a 25% deduction for each day.
2. Submissions more than three days late will not be marked, unless an extension has been granted.
3. If you need an extension apply through SOLS, if possible **before** the assignment deadline.
4. Plagiarism is treated seriously. If we suspect any work is copied, all students involved are likely to receive zero for the entire assignment.