CS515: Algorithms and Data Structures, Fall 2015

Homework 1*

Due: Mon, Oct/12/15

Homework Policy:

1. Students should work on homework assignments in groups of preferably three people. Each group submits to TEACH one set of typeset solutions, and hands in a printed hard copy in class. The hard copy will be graded.
2. You are allowed to discuss the homework with other groups, however, you must mention their names in your submission. Also, you must cite any other source that you use.
3. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. Of course, you can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

Problem 1. [25 pts]

(a) Prove that the following algorithm actually sorts its input.

```
StoogeSort(A[0..n-1]):
  if n = 2 and A[0] > A[1]
      swap A[0] ↔ A[1]
  else if n > 2
      m = ⌈2n/3⌉
      StoogeSort(A[0..m-1])
      StoogeSort(A[n-m..n-1])
      StoogeSort(A[0..m-1])
```

(b) Would StoogeSort still sort correctly if we replaced $m = \lceil 2n/3 \rceil$ with $m = \lfloor 2n/3 \rfloor$? Justify your answer.

(c) State a recurrence (including the base case(s)) for the number of comparisons executed by StoogeSort.

*Problems are from Jeff Erickson's lecture notes. Looking into similar problems from his lecture notes on recursion and dynamic programming is recommended.

(d) Solve the recurrence, and prove that your solution is correct. [Hint: Ignore the ceiling.]

(e) Prove that the number of swaps executed by StoogeSort is at most $\binom{n}{2}$.

Problem 2. **[25 pts]** An inversion in an array $A[1..n]$ is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. The number of inversions in an n-element array is between 0 (if the array is sorted) and $\binom{n}{2}$ (if the array is sorted backward). Describe an algorithm to count the number of inversions in an n-element array in $O(n \log n)$ time. Prove your algorithm is correct and analyze its running time.

Problem 3. **[25 pts]**

A shuffle of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. For example, the string BANANAANANAS is a shuffle of the strings BANANA and ANANAS in several different ways.

$$\texttt{BANANA}_{\texttt{ANANAS}} \qquad \texttt{BAN}_{\texttt{ANA}}\texttt{ANA}_{\texttt{NAS}} \qquad \texttt{B}_{\texttt{AN}}\texttt{AN}_{\texttt{A}}\texttt{A}_{\texttt{NA}}\texttt{NA}_{\texttt{S}}$$

Similarly, the strings PRODGYRNAMAMMIINCG and DYPRONGARMAMMICING are both shuffles of DYNAMIC and PROGRAMMING:

$$\texttt{PRO}^{\texttt{D}}{}_{\texttt{G}}{}^{\texttt{Y}}{}_{\texttt{R}}{}^{\texttt{NAM}}{}_{\texttt{AMMI}}{}^{\texttt{I}}{}_{\texttt{N}}{}^{\texttt{C}}{}_{\texttt{G}} \qquad \texttt{DY}_{\texttt{PRO}}{}^{\texttt{N}}{}_{\texttt{G}}{}^{\texttt{A}}{}_{\texttt{R}}{}^{\texttt{M}}{}_{\texttt{AMM}}{}^{\texttt{IC}}{}_{\texttt{ING}}$$

Given three strings $A[1..m]$, $B[1..n]$, and $C[1..m + n]$, describe an algorithm to determine whether C is a shuffle of A and B. Prove your algorithm is correct and analyze its running time.

Problem 4. **[25 pts]** You are driving a bus along a highway, full of rowdy, hyper, thirsty students and a soda fountain machine. Each minute that a student is on your bus, that student drinks one ounce of soda. Your goal is to drop the students off quickly, so that the total amount of soda consumed by all students is as small as possible.

You know how many students will get off of the bus at each exit. Your bus begins somewhere along the highway (probably not at either end) and move s at a constant speed of 3.14 miles per hour. You must drive the bus along the highway; however, you may drive forward to one exit then backward to an exit in the opposite direction, switching as often as you like. (You can stop the bus, drop off students, and turn around instantaneously.)

Describe an efficient algorithm to drop the students off so that they drink as little soda as possible. Your input consists of the bus route (a list of the exits, together with the travel time between successive exits), the number of students you will drop off at each exit, and the current location of your bus (which you may assume is an exit). Prove your algorithm is correct and analyze its running time.