CS515: Algorithms and Data Structures, Fall 2017

Homework 1*

Due: Tue, 10/10/17

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 or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded.
2. 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.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
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 German mathematician developed a new variant of the Towers of Hanoi game, known in the US literature as the "Liberty Towers" game. In this variant, there is a row of k pegs, numbered from 1 to k. In a single turn, you are allowed to move the smallest disk on peg i to either peg $i-1$ or peg $i+1$, for any index i; as usual, you are not allowed to place a bigger disk on a smaller disk. Your mission is to move a stack of n disks from peg 1 to peg k.

(a) Describe a recursive algorithm for the case $k = 3$. Exactly how many moves does your algorithm make?

(b) Describe a recursive algorithm for the case $k = n + 1$ that requires at most $O(n^2)$ moves. [Hint: First design an recursive algorithm that requires at most $O(n^3)$ moves.]

(c) [*extra credit*] Describe a recursive algorithm for the case $k = \sqrt{n}$ that requires at most a polynomial number of moves. (What polynomial??)

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

Problem 2. **[25 pts]** An inversion in an array $A[1 \dots n]$ 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 runningtime.

Problem 3. **[25 pts]** Suppose we are given two sorted arrays $A[1 \dots n]$ and $B[1 \dots n]$ and an integer k. Describe an algorithm to find the kth smallest element in the union of A and B in $O(\log n)$ time. For example, if $k = 1$, your algorithm should return the smallest element of $A \cup B$; if $k = n$, your algorithm should return the median of $A \cup B$.) You can assume that the arrays contain no duplicate elements. [Hint: First solve the special case $k = n$.]

Problem 4. **[25 pts]** Suppose you are given a sequence of integers separated by + and − signs; for example:

$$1 + 3 - 2 - 5 + 1 - 6 + 7.$$

You can change the value of this expression by adding parentheses in different places. For example:

$$\begin{aligned} 1 + 3 - 2 - 5 + 1 - 6 + 7 &= -1 \\ (1 + 3 - (2 - 5)) + (1 - 6) + 7 &= 9 \\ (1 + (3 - 2)) - (5 + 1) - (6 + 7) &= -17 \end{aligned}$$

Describe and analyze an algorithm to compute, given a list of integers separated by + and − signs, the maximum possible value the expression can take by adding parentheses. You may only use parentheses to group additions and subtractions; in particular, you are not allowed to create implicit multiplication as in $1 + 3(-2)(-5) + 1 - 6 + 7 = 33$.

Here are a set of practice problems on asymptotic running time analysis, and recursion. Do *not* submit solutions for the following problems, they are just for practice.

Practice Problem A. Prove that $\log(n!) = \Theta(n \log n)$. (Logarithms are based 2)

Practice Problem B. For each of the following, indicate whether $f = O(g)$, $f = \Omega(g)$ or $f = \Theta(g)$.

(a) $f(n) = 2n - 5$, $g(n) = 1235813n + 2016$.

(b) $f(n) = n \log n$, $g(n) = 1235813n + 2016$.

(c) $f(n) = n^{2/3}$, $g(n) = 7n^{3/4}$.

(d) $f(n) = n^{1.00000001}$, $g(n) = n \log n$.

(e) $f(n) = n5^n$, $g(n) = 7^n$.

Practice Problem C. Suppose you are given a stack of n pancakes of different sizes. You want to sort the pancakes so that smaller pancakes are on top of larger pancakes. The only operation you can perform is a flip – insert a spatula under the top k pancakes, for some integer k between 1 and n, and flip them all over.

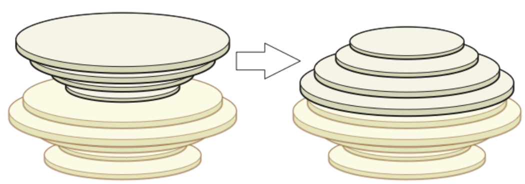

Figure 1: Flipping the top four pancakes.

(a) Describe an algorithm to sort an arbitrary stack of n pancakes using as few flips as possible. Exactly how many flips does your algorithm perform in the worst case?

(b) Now suppose one side of each pancake is burned. Describe an algorithm to sort an arbitrary stack of n pancakes, so that the burned side of every pancake is facing down, using as few flips as possible. Exactly how many flips does your algorithm perform in the worst case?