CS 5800 Algorithms V. Pavlu, K. Lieberherr

Homework 2

Due Date: Jan. 20, 2014, midnight.

1 Submission Rules

- Where. https://nuonline.neu.edu
- No Handwriting. It is strongly suggested that you submit your homework in typeset form. Good tools to use are Latex and Word, converted into PDF. Some algorithm classes successfully used https://www.writelatex.com/. Do not take pictures of your handwritten solutions because they tend to be hard to read. An exception might be if you have drawn manually a nice picture of some data structure or graph.
- Naming. You should always submit one file with the name FirstName.LastName.HWx.pdf/.zip, where x is the homework number. If your solution consists of multiple files (e.g., code, pictures, pdf files, etc.) zip them into one file.
- Code submission. Write instructions to run your program. You may use the PL of your choice although we provide supporting software in Java. Test cases should be included as part of the program.
- Academic Honesty. List the student or outside collaborators you got information from to find your answers. List other works you used (except text books or class materials).
- Lateness. Every student can use one late pass (up to 7 days) for no penalty. After that we deduct 25% per business day.
- Proof submission. Write the claim you prove using the logic notation (predicate logic) we introduced in the first lecture and which we use to define algorithmic problems. Test cases should be included to show how the proof works. Those test cases are semantic games you have played within your group.

Problems

1. (20p)Two linked lists (simple link, not double link) heads are given: $headA$, and $headB$; it is also given that the two lists intersect, thus after the intersection they have the same elements to the end. Find the first common element, without modifying the lists elements or using additional datastructures.
   a) A linear algorithm is: count the lists first, then use the count difference as an offset in the longer list, before traversing the lists together. Write a formal pseudocode (the pseudocode in the lecture is vague), using "*next*" as a method/pointer to advance to the next element in a list.
   b) Write the actual code in a programming language (C/C++, Java, Python etc) of

your choice and run it on a made-up test pair of two lists. A good idea is to use pointers to represent the list linkage.

2. (10p) Exercise 3.1-1

3. (5p) Exercise 3.1-4

4. (15p) Rank the following functions in terms of asymptotic growth. In other words, find an arrangement of the functions $f_1, f_2, \ldots$ such that for all i, $f_i = \Omega(f_{i+1})$.

   $\sqrt{n}\ln n \qquad \ln\ln n^2 \qquad 2^{\ln^2 n} \qquad n! \qquad n^{0.001} \qquad 2^{2\ln n} \qquad (\ln n)!$

5. (40p) Provide a good asymptotic bound for HSRnk-min(n,2). See the HSR document:

   `http://www.ccs.neu.edu/home/lieber/courses/algorithms/cs5800/sp14/help-sessions/Jan14/HSR-problem-CS5800-2.doc`

6. (25p) **Task**: When multiplying n numbers $P = a_1 * a_2 * ... * a_n$, there are many choices of multiplication order, *or parenthesizing*, for example $(a_1 * a_2) * (a_3 * a_4)$, $((a_1 * a_2) * a_3) * a_4$, $(a_1 * (a_2 * (a_3 * a_4))$, etc. We want to know in how many different ways can we put the parentheses.

   Solution : divide and conquer; lets say T(n) is the number of ways to put parentheses on the product of n numbers. First we decide the last multiplication, say its between numbers a_k and a_{k+1}. Thus we decided $P = (a_1 * a_2 * ... * a_k) * (a_{k+1} * a_{k+2} * ... * a_n)$ There are $n-1$ possible such k, each leading to different parenthesis structure no matter how left and right sides get their parentheses.

   For each k, we have now two subproblems. We shall use the "earlier results" on subproblems: count the number of ways to parenthesize the k first half of numbers, that is T(k), and the number of ways to parenthesize the $n-k$ second half of numbers, that is T($n-k$). Then we have to combine these, and iterate across all k, to get back T(n).

   - What is the recurrence for T(n) embedded in this solution?