CSCI 3100: Algorithms
Homework 0

A note about homework 0: A large goal of this particular assignment is to force you to review some of what was covered in discrete mathematics and in data structures, both of which are important prereqs for this course. You can't design and analyze algorithms without understanding fundamental data structures, proofs, and big-O notation! So expect to pull out your old textbooks if you're rusty on those, or look in the introduction of our textbook for some suggestions, or just email to ask me for some recommendations if you aren't sure where to look.

Academic integrity policy: In this class, while you're welcome to use any reference you'd like, you are responsible for both citing your sources AND re-writing the solution in your own words. So feel free to use the internet, but I'd recommend reading, thinking about it, adding a citation for what you read to your homework, and then putting all that away and writing it from memory, so you'll understand the answer properly. Please believe me when I say that ANY verbatim copying will get you a 0 on the homework, and any second offense will get you a 0 in the class! Please go check the syllabus for more details.

Required Problems

1. Sort the following functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice.

$$\begin{array}{ccccc} \log(2^{2n}) & 325143 & \sum_{i=1}^{n} i & \lg n & 5 \cdot 2^{\lg n} \\ \sin n + 12 & n \lg(2^n) & \sqrt{n} & 6n^2 - 4n + 81 & \frac{n!}{(n-3)!} \end{array}$$

   To simplify notation, write $f(n) \ll g(n)$ to mean $f(n) = o(g(n))$ and $f(n) \equiv g(n)$ to mean $f(n) = \Theta(g(n))$. For example, the functions n^2, n, $\binom{n}{2}$, n^3 could be sorted either as $n \ll n^2 \equiv \binom{n}{2} \ll n^3$ or as $n \ll \binom{n}{2} \equiv n^2 \ll n^3$.
   [Hint: Hopefully this will become obvious, but my goal here is to make you remember: big-O, logarithms, summations, and how to ignore constants of various sorts! Go back and check your discrete math book, as well as that chapter in your data structures book that talked about big-O, and hopefully these will start to look easier.]

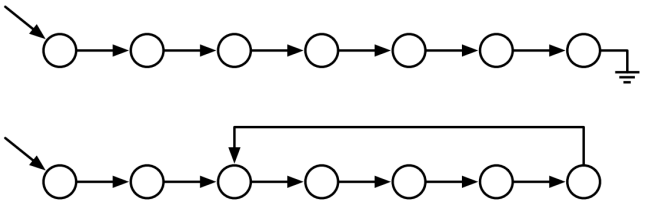
2. Dr. Chambers recently returned from Germany with a new favorite 24-node binary tree, in which every node is labeled with a unique letter from the German alphabet. (Note that this is pretty similar to English, but adds interesting characters like the umlaut and ß.) She gives you the following traversals:

   - Preorder: B K Ü E L Z I Ö R C P ß T S O A Ä D F M N U G
   - Inorder: Ü E K I Z R Ö C L P B S T O ß D Ä M F A U N G

   (a) List the nodes in a postorder traversal of the tree.

   (b) Draw the tree.

3. Suppose you are given a pointer to the head of singly linked list. Normally, each node in the list has a pointer to the next element, and the last node's pointer is NULL. Unfortunately, your list might have been corrupted (by a bug in somebody else's code, of course), so that some node's pointer leads back to an earlier node in the list.



Top: A standard singly-linked list. Bottom: A corrupted singly linked list.

Design and analyze an algorithm to detect if the list is corrupted or not. (So you need to give an algorithm, give its running time, and justify why it works.) Note that there are several ways to solve this. Any correct solution will be given *almost* full credit. However, to get full credit, I'm looking for an $O(n)$ time algorithm which uses only $O(1)$ extra space, which means you can't copy or mark the actual list, but can create only a few new variables.

4. A binomial tree of order k is defined recursively as follows:

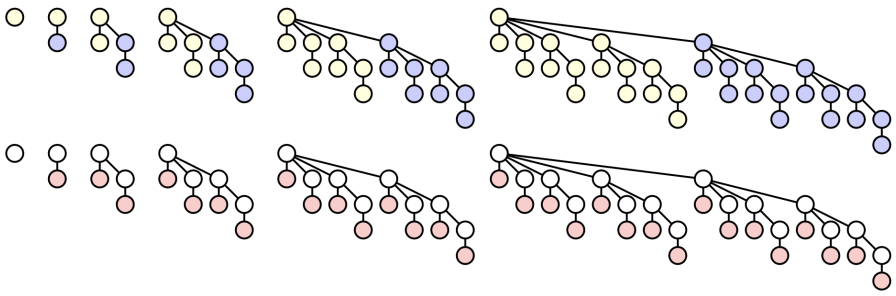
   - A binomial tree of order 0 is a single node.
   - For all $k > 0$, a binomial tree of order k consists of two binomial trees of order $k-1$, with the root of one tree connected as a new child of the root of the other. (See the figure below.)

   Prove the following claims:

   (a) For all non-negative integers k, a binomial tree of order k has exactly 2^k nodes.

   (b) For all positive integers k, attaching a leaf to every node in a binomial tree of order $k-1$ results in a binomial tree of order k.

   (c) For all non-negative integers k and d, a binomial tree of order k has exactly $\binom{k}{d}$ nodes with depth d



Binomial trees of order 0 through 5.
Top row: the recursive definition. Bottom row: the property claimed in part (b).