CSE 40113: 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 - including any usage of ChatGPT or similar tools. 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, as well as being reported to the department. Any second offense will get you a 0 in the class, as well as be forwarded to the college for further disciplinary measures.

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 make sure you understand how you would prove these and what facts/identities are needed to simplify.

$$\begin{array}{ccccc} 1 & n & 3n^2 - 6n + 12 & \lg n & \lg \sqrt{n} \\ \cos n + 2 & n^{\lg n} & \lg 2^n & \sum_{i=1}^{n} \sum_{j=1}^{i} 1 & 2^{2 \lg n} \\ \sqrt{2^{\lg n}} & n! & n \lg n & \sum_{i=1}^{n} i^2 & \lg(\sqrt{2^n}) \end{array}$$

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: When considering two functions $f(\cdot)$ and $g(\cdot)$ it is sometime useful to consider the functions $\ln f(\cdot)$ and $\ln g(\cdot)$.]*

Solution: The final sorted list is as follows:

$$\cos n + 2 \equiv 1 << \log \sqrt{n} \equiv \lg n << \sqrt{2^{\lg n}} << n \equiv \log(\sqrt{2^n}) \equiv \log 2^n << n \lg n << 3n^2 - 6n + 12 \equiv \sum_{i=1}^{n} \sum_{j=1}^{i} 1 \equiv 2^{2 \lg n} << \sum_{i=1}^{n} i^2 << n^{\lg n} << n!$$

■

[Hint: Hopefully this will become obvious, but my goal here is to make you remember: big-O, logarithms, summations, and ignoring constants! Go back and check your discrete math book and that chapter in your data structures book that talked about big-O.]

2. (a) Your fellow student presents the following idea: in order to implement "find the maximum" in constant time, why not use a stack or a queue, but keep track of the maximum value inserted so far, then return that value for "find the maximum". Does this seem like a good idea, or do you see issues?

   (b) Suppose you have been asked to implement a stack, but the only data structure you have available is a queue (although you may use as many queues as you would like). Give pseudocode for implementing stack's push and pop function, which use only the queue(s) you will use to maintain the data. What is the asymptotic runtime of each of your functions, assuming the queue push and pop functions run in $O(1)$ time?

3. 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 H L Z I Ö R C ß T S O A Ä D F M N U G
   - Postorder: H I Ö Z R L E C Ü S O T A ß K D M U G N F Ä B

   (a) List the nodes in an in order traversal of the tree.

   (b) Draw the tree.

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

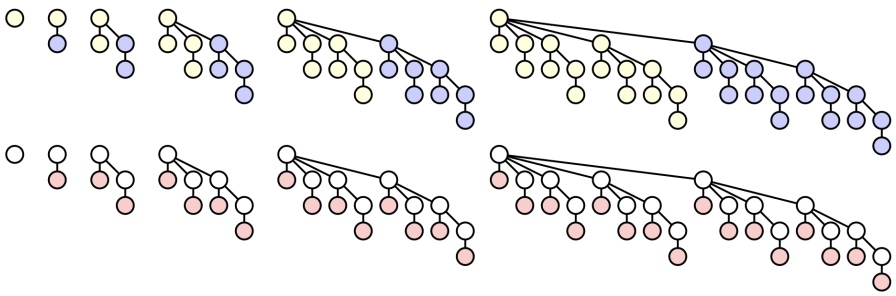
   - A binomial tree of order 0 is a single node.
   - For all $k > 0$, a binomial tree of order k consists of two binomial trees of order $k-1$, with the root of one tree connected as a new child of the root of the other. (See the figure below.)

   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) Prove that for all non-negative integers k and d, a binomial tree of order k has exactly $\binom{k}{d}$ nodes with depth d. (Hence the name!)



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