CSE548/AMS542 Fall 2019 Analysis of Algorithms

Jie Gao*

September 2, 2019

Due **September 17th** 9pm. No late homework accepted.

Each problem, unless specified otherwise, has a maximum of 10 points. Avoid too many details. A succinct and clean proof is the best. You may use the algorithms we covered in class without referring to the details. In both homework and exams, a question left empty received 15% of the total points, a wrong answer receives 0, and nearly correct answers with minor flaws receive partial points. Homework submission must be in the format of pdf, formatted by Latex, submitted through blackboard. Handwritten homework will be returned.

We may select a random subset of the problems to grade.

Homework 1

1. **Big-O notation (10pts)** Prove or disprove (i.e., give counter examples) for the following claims. $f(n), g(n)$ are non-negative functions.
   (a) $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.
   (b) $o(f(n)) \cap \omega(f(n)) = \emptyset$.
   (c) $(n + a)^b = \Theta(n^b)$, a, b are positive integers.
   (d) $f(n) = O(f(n)^2)$.
   (e) $f(n) = O(g(n))$ implies that $2^{f(n)} = O(2^{g(n)})$.

2. Sort the following functions from asymtotically smallest to asymptotically largest. That is, the function $f(n)$ and the next function $g(n)$ must always follow that $f(n) \in O(g(n))$. If the two functions have asymptotic the same order, i.e., $f(n) = \Theta(g(n))$, then also indicate that. No need to write down proofs. Remember $\lg n = \log_2 n$.
   n, $\lg n$, $\sqrt{n}$, $\sqrt{\lg n}$, $\lg \sqrt{n}$, 2^n, $2^{\sqrt{n}}$, $\sqrt{2^n}$, $2^{\lg n}$, $\lg(2^n)$, $2^{\lg \sqrt{n}}$, $2^{\sqrt{\lg n}}$, $\sqrt{2^{\lg n}}$, $\lg(\sqrt{2^n})$, $\sqrt{\lg(2^n)}$, 3^n, $3^{\sqrt{n}}$, $\sqrt{3^n}$, $3^{\lg n}$, $\lg(3^n)$, $3^{\lg \sqrt{n}}$, $3^{\sqrt{\lg n}}$, $\sqrt{3^{\lg n}}$, $\lg(\sqrt{3^n})$, $\sqrt{\lg(3^n)}$.

3. Textbook [Kleinberg & Tardos] Chapter 2, page 67, problem #6.

4. Recall that in a heap we keep the elements such that the parent is smaller than children. Thus the root (stored as the first element in the array) is the smallest item in the array. Insertion and of a new element can be done in $O(\lg n)$ time. Deletion of an element can be done in $O(\lg n)$ time. Thus one start from an empty heap and insert elements one by one to build a heap of n elements. Further, we can use it to sort n elements. Once the heap is built, we remove the root (the smallest element), which is the smallest of the n elements. Iterate and we will get all n elements output in the increasing sorted order. This algorithm is called HeapSort.

*Department of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA, jgao@cs.stonybrook.edu.

(a) What is the running time for HeapSort? Assume elements come in an arbitrary order and represent the running time in big-O notation. (2pts)

(b) What is the running time of HeapSort on n elements in increasing order? Represent the running time in big-Θ notation. (4pts)

(c) What is the running time of HeapSort on n elements in decreasing order? Represent the running time in big-Θ notation. (4pts)

5. **Young Tableaus** An $m \times n$ Young tableau is an $m \times n$ matrix such that the entries of each row are in increasing order from left to right and the entries of each column are in increasing order from top to bottom. Some of the entries of each column may be ∞, which are treated as nonexistent elements. A Young tableau with some elements as ∞ is not full.

   Give an algorithm that extracts MIN (i.e., delete the minimum element and restore the matrix to be a Young tableau) on a nonempty $m \times n$ Young tableau that runs in $O(m + n)$ time.