

CS 535 Homework 1

Due: September 16 (W), in class.

Undergrads, please answer questions 1, 2, and 3; question 4 is a bonus question. Grad students, please answer all questions.

Reminders: You are allowed to collaborate with each other and consult resources other than your book BUT you must (a) indicate in your submission who you worked with, (b) cite the sources you used, and (c) you must write solutions in your own words. Homeworks are due at the beginning of the class. No late homeworks will be accepted.

- For this problem, order the functions based on their growth rate. That is, list $f(n)$ ahead of $g(n)$ if $f(n)$ is $O(g(n))$. Assume that the base of the log function is 2 so $2^{\log n} = n$.

$6n \log n$	2^{100}	$\log \log n$	$\log^2 n$	$2^{\log n}$
2^{2^n}	$\lceil \sqrt{n} \rceil$	$n^{0.01}$	$1/n$	$4n^{3/2}$
$3n^{0.5}$	$5n$	$\lfloor 2n \log^2 n \rfloor$	2^n	$n \log n$
4^n	n^3	$n^2 \log n$	$4^{\log n}$	$\sqrt{\log n}$

Hints:

- First, coarsely divide the functions into several groups – those that are decreasing (there is one!), those that are $O(1)$, those that are $O(\log^k n)$ for some constant k , those that are $O(n^k)$ for some $k > 0$, and those that are $O(a^n)$ for some $a > 1$. Then arrange the functions within each group.
 - When in doubt about two functions $f(n)$ and $g(n)$, consider $\log f(n)$ and $\log g(n)$ or $2^{f(n)}$ and $2^{g(n)}$.
- Verify the correctness of the following statements from Theorem 1.7 in your book. Assume $f(n), g(n), d(n), e(n)$ are functions mapping non-negative integers to non-negative real numbers. Suppose $f(n)$ is $O(g(n))$ and $d(n)$ is $O(e(n))$. Show that
 - $f(n) + d(n)$ is $O(g(n) + e(n))$.
 - $f(n)d(n)$ is $O(g(n)e(n))$.

Make use of the definition of the Big-Oh notation in your arguments.

- Suppose a $1 \times n$ array $A[0 \dots n - 1]$ contains numbers in non-decreasing order. That is, $A[i] \leq A[i + 1]$. Assume A is already stored in memory. We would like to answer queries of the form “Given x , does A contain the number x ?” efficiently. The brute force method would be to scan A , which will take $O(n)$ time in the worst case. A better method would be to first consider $A[\lfloor n/2 \rfloor]$ and compare it with x . If they are equal, we are done. Otherwise, if $A[\lfloor n/2 \rfloor] < x$, we resume our search on $A[0 \dots \lfloor n/2 \rfloor - 1]$; else, we resume our search on

$A[\lfloor n/2 \rfloor + 1 \dots n - 1]$. We keep doing this until we determine if x is in A or not.

- a. Computer scientists would say that we are doing *binary search* on A . Translate this solution into pseudocode.
 - b. What is the worst-case runtime of algorithm? Why?
4. Using pseudocode, write an algorithm for multiplying an $n \times m$ matrix A and an $m \times p$ matrix B . (Recall that the product $C = AB$ is an $n \times p$ matrix where $C[i][j] = \sum_{k=1}^m A[i][k]B[k][j]$.) Describe the runtime of your algorithm in terms of n, m and p .