CS 498 MCS Algorithms ✧ Fall 2026

Homework 0

Due Tuesday, September 1, 2026 at 9pm Central Time

- **This homework is required, but it will not contribute to your final grade.** This homework serves two purposes. First, we want to jog your memory about some prerequisite topics. But more importantly, we want to understand your background, so that we can properly calibrate this brand-new class. Honest good-faith effort is *far* more valuable to us than complete, perfectly correct answers.

  Please answer each problem to the best of your ability. If you get stuck, please explain as best you can *where* you are stuck. Even answers like "I have no idea how to start" or "I don't know what these words mean" are valuable information for us. On the other hand, if you find a problem too easy, feel free to tell us or give a high-level sketch of the solution.

  For this homework only, each student must work alone and submit individual solutions. Do not use other students, external sources, or generative AI to help you solve these problems; that defeats the purpose of this assignment. You are welcome to consult course materials and other sources to refresh your memory about definitions and underlying concepts, but not for help with the specific problems in this homework.

  Future homeworks will allow group work, collaboration, and unlimited use of external sources; they will also count toward your final grade.

- **Submit your solutions electronically on Gradescope as PDF files.**
  - Submit a separate PDF file for each numbered problem.
  - You can find a LaTeX solution template on the course web site, which we encourage you to use to typeset your homework.
  - If you plan to submit scanned handwritten solutions, please use dark ink (not pencil) on white unlined paper (not notebook or graph paper), and use a scanner or a scanning app to create a high-quality PDF for submission (not a raw photo). We reserve the right to reject submissions that are difficult to read.
  - If you plan to use a tablet and a note-taking app, please make sure your submitted PDF is broken into standard US-letter sized pages (not a long scroll).

See the course web site for more information.

If you have any questions about these policies,
please don't hesitate to ask in lecture, in office hours, or online.

---

1. Sort the following functions in asymptotic order. More precisely, if $f(n) = o(g(n))$, then $f(n)$ should come before $g(n)$ in the ordering. If $f(n) = \Theta(g(n))$, you can list those two functions in either order, but clearly indicate the tie.

   To simplify notation notation, write $f(n) \ll g(n)$ to indicate $f(n) = o(g(n))$, and write $f(n) \equiv g(n)$ to indicate $f(n) = \Theta(g(n))$. For example: $n \equiv 10n + \sqrt{n} \ll n^{10} \ll n^{10} \lg n$.

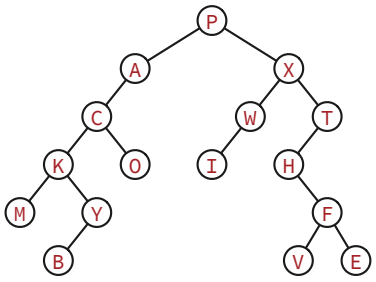
   No proofs are required or should be submitted, but you should work them out anyway to make sure your answers are correct.

$$n \qquad 3^n \qquad n^n \qquad \log(n!) \qquad (1+1/n)^n \qquad n \lg n$$

2. Solve the following recurrences. Give a tight asymptotic bound for each function of the form $f(n) = \Theta(g(n))$, where $f(n)$ is the function and $g(n)$ is composed of simple known functions. No proofs are required or should be submitted, but you should work them out anyway to make sure your answers are correct.

   - $A(n) = A(n/5) + n$ for $n > 5$ and $A(n) = 1$ for $n < 5$.
   - $B(n) = B(n/2) + 1$ for $n \geq 2$ and $B(n) = 1$ for $n = 1$.
   - $C(n) = 2C(n/2) + n$ for $n \geq 2$ and $C(n) = 1$ for $n = 0, 1$.

3. (a) Let T be the following binary tree. Each node in T is labeled with a unique letter from the English alphabet:



   List (the labels of) the nodes in T in each of the following traversal orders:

   - Preorder
   - Inorder
   - Postorder

   (Part (b) shows the format we are expecting for each traversal order.)

   (b) Professor George O'Jungle has a 27-node binary tree, in which every node is labeled with a unique letter of the English alphabet or the character `&`. Preorder and postorder traversals of the tree visit the nodes in the following order:

   - Preorder: `I Q J H L E M V O T S B R G Y Z K C A & F P N U D W X`
   - Postorder: `H E M L J V Q S G Y R Z B T C P U D N F W & X A K O I`

   Draw George's binary tree.

4. Recall that a binary search tree is a rooted binary tree T in which every node v has three associated values:

   - $v.left$: a pointer to the left child of v (or NULL)
   - $v.right$: a pointer to the left child of v (or NULL)
   - $v.key$: a number called the *search key* associated with v

   Let T_v denote the subtree of T rooted at v. The search keys in T are distinct and satisfy the following recursive property: For every node v, all keys in the left subtree of T_v are less than $v.key$, and all keys in the right subtree of T_v are greater than v.key.

   Now suppose we *augment* the tree by adding an additional value to each node:

   - $v.size$: the number of nodes in the subtree rooted at v (including v itself)

   Describe and analyze an algorithm to determine, given a binary search tree T (augmented with subtree sizes) and a number x, the number of search keys in T that are strictly greater than x.

   Your algorithm should run in $O(d)$ time, where d is the depth of T. In particular, if T is balanced, meaning its depth is $O(\log n)$, your algorithm should run in $O(\log n)$ time, where n is the number of nodes in T.

5. Submit a proof for ***one*** of the following problems.

   (a) Suppose we want to prove that $\sum_{i=1}^{n} \frac{1}{i^2} < 2$ by induction on n. This turns out to be not so easy directly. But it is quite a bit easier to prove the *stronger* statement $\sum_{i=1}^{n} \frac{1}{i^2} \le 2 - \frac{1}{n}$. Prove this stronger statement by induction on n.

   (b) Recall that the nth Fibonacci number F_n is defined recursively as follows:

   $$F_n = \begin{cases} F_0 & \text{if } n = 0 \\ F_1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

   Prove by induction on n that F_n is even if and only if n is divisible by 3.

6. Suppose you are given a directed graph $G = (V, E)$, where each vertex is colored either red, green, or blue. Recall that a vertex $v \in V$ is *reachable* from another vertex $u \in V$ if and only if G contains a directed path from u to v.

   Describe and analyze an algorithm for ***one*** of the following problems, whichever you find *more* difficult. (A solution to the part (b) immediately implies a solution to part (a).)

   (a) Given the graph G and a vertex $s \in V$, determine whether *any* red vertex in G is reachable from s.

   (b) Given the graph G and a vertex $s \in V$, return the number of red vertices in G that are reachable from s.

   Express the running time of your algorithm in terms of $m = |E|$ and $n = |V|$.