Problem 0. (-5 for leaving blank.) Write down all outside resources you consulted, and the names of any other students that you worked with in any way. By turning in this homework, you acknowledge that you have read and abide by all course policies on outside resources and collaboration as stated in the course syllabus. Use of Generative AI tools must be explicitly noted. *If you did not use outside resources, AI tools, or collaborators, state that explicitly.*

Problem 1. Let BinSearch(A, k) be a standard binary search algorithm that takes in a sorted array A of integers, and an integer k that is guaranteed to be contained somewhere in A, and returns an index j such that $A[j] = k$.

(a) Write pseudocode to implement this method. You may assume that $|A|$ is a power of 2. For your code, assume A is 1-indexed (i.e., $A[1]$ is the first element).

(b) Write down a recurrence relation for this algorithm.

Answer.

(a) Pseudocode is given below. Slight variations are also acceptable.

BinSearch(A, k)

$n = |A|$
if $|A| == 1$ **then**
return 1
if $A[n/2] \geq k$ **then**
return BinSearch$(A[1 : n/2], k)$
else
return BinSearch$(A[n/2 + 1 : n], k) + \frac{n}{2}$

(b) The recurrence relation is $T(n) = T(n/2) + O(1)$.

Problem 2. Consider the recurrence relations below. For each, there is some text that attempts to apply the Master Theorem (as shown at the end of the homework) to determine the runtime. In each case, state whether the analysis is correct, or whether there is an error

in the application of the Master Theorem. If there is an error, state what the error is and what the correct answer should be.

(a) Recurrence: $T(n) = 2T(n/4) + \sqrt{n}$.

Candidate explanation: In this case, $g(n) = n^{\log_b a} = n^{\log_4 2} = n^{\frac{1}{2}}$. Hence, $f(n) = \sqrt{n} = O(n^{\log_b a - \varepsilon})$ for a sufficiently small ε, and therefore Case 1 applies and we know $T(n) = \Theta(n^{1/2})$.

(b) Recurrence: $T(n) = 2T(n/4) + \sqrt{n}\log^2 n$.

Candidate explanation: In this case, $g(n) = n^{\log_b a} = n^{\log_4 2} = n^{\frac{1}{2}}$. Meanwhile, $f(n) = \sqrt{n}\log^2 n$. Hence, $f(n)$ is polynomially larger than $g(n)$ and we apply Case 3. The runtime is therefore $T(n) = \Theta(f(n)) = \Theta(\sqrt{n}\log^2 n)$.

(c) Recurrence: $T(n) = 17T(n/4) + O(n^2 \log n)$.

Candidate explanation: In this case, $g(n) = n^{\log_b a} = n^{\log_4 17} \approx n^{2.04}$. Meanwhile, $f(n) = O(n^2 \log n)$, which does not grow like a polynomial and is therefore neither polynomially larger nor polynomially smaller than $g(n)$. For this reason, the Master Theorem does not apply.

(d) $T(n) = 4T(n/4) + n \log n$.

Candidate explanation: In this case, $g(n) = n^{\log_b a} = n^{\log_4 4} = n$. Meanwhile, $f(n) = n \log n$. The comparison between $f(n)$ and $g(n)$ does not fit into any cases of the Master Theorem, so the theorem does not apply here.

Answer.

(a) This is incorrect. $f(n) = \sqrt{n} = \Theta(g(n))$, so Case 2 applies, and the runtime is $T(n) = \Theta(\sqrt{n})$.

(b) This is incorrect. Although $f(n)$ is asymptotically larger than $g(n)$, it is not polynomially larger than $g(n)$, so Case 3 does not apply. Cases 1 and 2 also don't apply, so the Master Theorem below simply does not apply. Note furthermore that the candidate explanation did not check the additional condition $af(n/b) \leq cf(n)$.

(c) This is incorrect. Whether or not $n^2 \log n$ is a polynomial, the function $n^{\log_4 17}$ is poly-

nomially larger. Therefore, Case 1 applies and the runtime is $O(n^{\log_4 17})$.

(d) This is correct. The recurrence does not fit into any of the cases.

Problem 3. Say you have discovered a way to multiply 7×7 matrices using $c > 50$ multiplications and 30 additions.[1]

(a) Write down the recurrence relation you could get for a new recursive matrix-matrix multiplication algorithm you could design by breaking up an $n \times n$ matrix into blocks of size $n/7 \times n/7$ and using your approach for multiplying 7×7 matrices. You may assume $n = 7^m$ where $m > 1$ is an integer.

(b) Write down an expression for the asymptotic runtime of this method in terms of c and n.

(c) What is the largest value for c such that the asymptotic runtime of your new method is better than the runtime you get from Strassen's algorithm.

Answer. Let $T(n)$ be the runtime of the algorithm for $n = 7^m$.

(a) The recurrence relation you get is

$$T(n) = c \cdot T(n/7) + O(n^2),$$

since $T(n/7)$ is the time for multiplying two $n/7 \times n/7$ blocks (which we do c times), and $O(n^2)$ is the time it would take to do 30 additions of $n/7 \times n/7$ matrices.

(b) Because $c > 50$, we know that $\log_7 c > 2$, which means that $O(n^2) = O(n^{\log_7 c - \varepsilon})$ for a small enough $\varepsilon > 0$. Applying the Master Theorem, we know that the runtime for the new method is

$$T(n) = \Theta(n^{\log_7 c})$$

(c) The largest value of c such that $\log_7 c < \log_2 7$ is $c = 235$, so this is the answer since the runtime for Strassen's method is $T(n) = \Theta(n^{\log_2 7})$.

Master Theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$

[1] Assume that this works without relying on the fact that $ab = ba$ for $a, b \in \mathbb{R}$.

be defined on the nonnegative integers by the relation:

$$T(n) = aT(n/b) + f(n).$$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.
