CSCE 411, Fall 2025: Homework 1

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.*

Used ChatGPT for Problems 2(c) and 3(c)

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).

Algorithm 1 Binary search algorithm

```
procedure BinSearch($A, k$)
    $left \leftarrow 1$
    $right \leftarrow length(A)$
    while $left \leq right$ do
        $mid \leftarrow \lfloor (left + right)/2 \rfloor$
        if $k < A[mid]$ then
            $right \leftarrow mid - 1$
        else if $k > A[mid]$ then
            $left \leftarrow mid + 1$
        else
            return $mid$
```

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

$n = length(A)$

$T(n) = T(n/2) + O(1)$

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 $T(n) = \Theta(n^{1/2})$.

There is an error. The candidate is incorrect in claiming that $f(n) = O(n^{\log_b a - \varepsilon})$

My Answer: $a = 2$, $b = 4$, $f(n) = \sqrt{n}$, $g(n) = n^{\log_b a} = n^{\log_4 2} = n^{1/2}$. Hence, $f(n) = \sqrt{n} = \Theta(n^{\log_b a})$. Case 2 applies, and we know $T(n) = \Theta(\sqrt{n} \log n)$

(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 $T(n) = \Theta(f(n)) = \Theta(\sqrt{n} \log^2 n)$.

There is an error. The candidate is incorrect in claiming that $f(n)$ is polynomially larger than $g(n)$.

My Answer: $a = 2$, $b = 4$, $f(n) = \sqrt{n} \log^2 n$, $g(n) = n^{\log_b a} = n^{\log_4 2} = n^{1/2}$. So, $f(n) = g(n) \log^2 n$. This matches Case 2 when $f(n) = g(n) \log^k n$ and Case 2 says $T(n) = \Theta(g(n) \log^{k+1} n)$ in this situation. Since $k = 2$, $T(n) = \Theta(\sqrt{n} \log^3 n)$

(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.

There is an error. The candidate claims that the Master Theorem does not apply

My Answer: $a = 17$, $b = 4$, $f(n) = O(n^2 \log n)$, $g(n) = n^{\log_b a} = n^{\log_4 17} \approx n^{2.04}$. Thus, $f(n)$ is asymptotically smaller than $g(n)$ meaning Case 1 applies. $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_4 17})$

(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.

There is an error. The candidate claims that the Master Theorem does not apply

My Answer: $a = 4$, $b = 4$, $f(n) = n \log n$, $g(n) = n^{\log_b a} = n^{\log_4 4} = n$. So, $f(n) = g(n) \log n$. This matches Case 2 when $f(n) = g(n) \log^k n$ and Case 2 says $T(n) = \Theta(g(n) \log^{k+1} n)$ in this situation. Since $k = 1$, $T(n) = \Theta(n \log^2 n)$

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.

$T(n) = cT(n/7) + O(n^2)$

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

$a = c$, $b = 7$, $f(n) = O(n^2)$, $g(n) = n^{log_b a} = n^{log_7 c}$ Since $n^{log_7 c} > n^2$ when $c > 50$, then $f(n) = O(n^{log_7 c - \epsilon})$ for some $\epsilon > 0$ so Case 1 applies. Therefore, $T(n) = \Theta(n^{log_b a}) = \Theta(n^{log_7 c})$

(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.

Recurrence realtion of the Strassen algorithm: $7T(n/2) + O(n^2)$

$g(n) = n^{log_b a} = n^{log_2 7} \approx n^{2.8}$

We want to find c where $log_7 c < log_2 7$

$c < 7^{1/log_7 2} \approx 235.6$

So, $c = 235$

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 $a, b \in \mathbb{R}$. This must be assumed because matrix multiplication is not commutative, so if the strategy for 7×7 matrix multiplication relied somehow on commutativity, it would not apply when multiplying *blocks* of matrices.

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 $T(n) = \Theta(f(n))$.
