

ECE 537 - Foundations of Computing
Prof. Sen
Homework #6
Solutions

1. Work problem 4.1-1 in the Cormen et. al text.

Solution:

We must show that

$$T(n) = c \lg n$$

for an appropriate choice of c , and for all n when n is sufficiently large. Substituting this into the recurrence we get

$$\begin{aligned} T(n) &\leq c \lg \left(\frac{n}{2}\right) + 1 \\ &= c \lg n + 1 - \lg 2 \\ &= c \lg n \end{aligned}$$

which verifies that $T(n) = O(\lg n)$.

2. Work problem 4.1-6 in the Cormen et. al text.

Solution:

Let $m = \lg n$, then $2^m = n$ and $\sqrt{n} = 2^{m/2}$. Thus, we can rewrite the recurrence as

$$T(2^m) = 2T(2^{m/2}) + 1.$$

Now, letting $S(m) = T(2^m)$, we can write

$$S(m) = 2S\left(\frac{m}{2}\right) + 1.$$

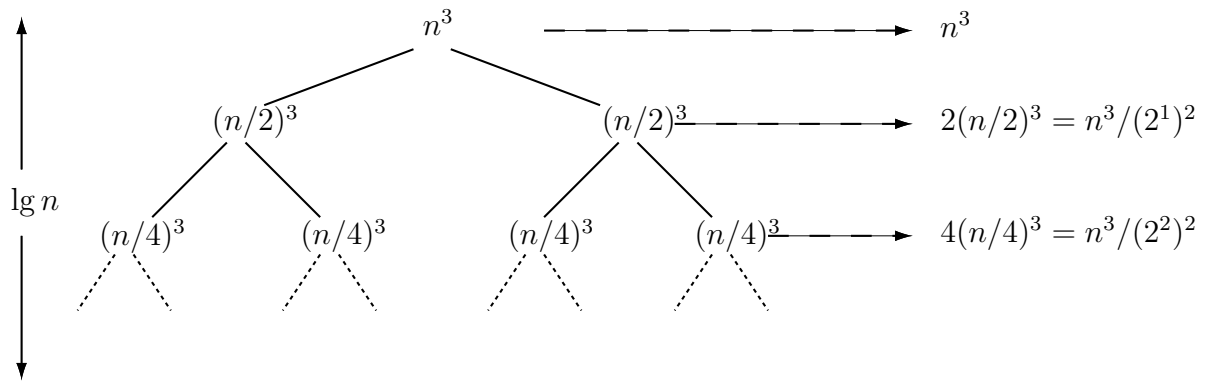
It is easy to verify that the solution to this recurrence is $S(m) = \Theta(\lg m)$ (and your solution to this problem should include this verification). Thus, $T(2^m) = \Theta(\lg 2^m) = \Theta(m)$, and $T(n) = \Theta(\lg n)$.

3. Work problem 4-1 (at the end of the chapter) in the Cormen et. al text.

Solution:

(a) $T(n) = 2T(n/2) + n^3$.

Solution: This is a divide-and-conquer recurrence with $a = 2$, $b = 2$ and $f(n) = n^3$. Let's solve it using the recursion tree method. Specifically, the following recursion tree captures the total amount of computation performed by the recurrence:



The root of this tree corresponds to the place where the first recursive steps are performed. That is, at the root, the problem size is n , and since the recurrence shows that for a problem of size n , an amount of computation corresponding to $f(n) = n^3$ needs to be performed, we write n^3 at the root. That is, ignoring the recursive calls (for the amount of computation contained in these will be shown in lower levels of the tree), n^3 work is performed at the root level. Now from the root level, there will be two (a) recursive calls, and that is why we show two branches from the root. In addition, since for each of these recursive call, the problem size is cut in half ($1/b$), the nodes at this level have a problem size of $n/b = n/2$, and an amount of computation corresponding to $f(n/b) = (n/2)^3$ must be performed for each recursive call. Thus, we write $(n/2)^3$ at each node on this level. Next, each of these nodes spawns two more recursive calls, for a total of four recursive calls on the next level, each having a computational load of $f(n/b^2) = (n/4)^3$. The problem size will continue to be reduced until the recurrence “bottoms out” when $n = 1$. That is, the recurrence will bottom out at the i th level when $n/2^i = 1$. That is, when $i = \lg n$. Thus, the recursion tree has $\lg n + 1$ levels, but the last one corresponds to the nodes where $n = 1$, so we don’t count them (i.e., we’re assuming $T(1) = \Theta(1)$). Thus, we show the height of the recursion tree as $\lg n$ on the LHS of the figure. Notice that since this recurrence only involves one previous value of $T(\cdot)$ on the RHS of the original recurrence, every path in the tree bottoms out on the same level.

Now to calculate the total amount of computation performed by the recurrence, we need to sum the work performed at each node. This is easiest to do by first summing across the levels, and then summing the levels. Thus, on the RHS of the above figure we show the results we obtain by summing across the levels. In general, we can write that on the i -th level, an amount of computation equal to:

$$\frac{n^3}{(2^i)^2} = \frac{n^3}{4^i}$$

is performed. Finally, summing the work performed in all levels, we obtain

$$\begin{aligned}
 T(n) &= \sum_{i=0}^{\lg n-1} \frac{n^3}{4^i} \\
 &= n^3 \sum_{i=0}^{\lg n-1} \frac{1}{4^i} \\
 &\leq n^3 \sum_{i=0}^{\infty} \frac{1}{4^i} \\
 &= n^3 \cdot \frac{1}{1-1/4} \\
 &= O(n^3).
 \end{aligned}$$

So we have an upper bound. By observing the original recurrence, it is easy to see that

$$T(n) = \Omega(n^3)$$

since that amount of work is required on the very first recursive call. Taken together, we therefore have shown that

$$T(n) = \Theta(n^3).$$

I will not provide this level of detail for the remaining problems.

(b) $T(n) = T(9n/10) + n.$

Solution: This recursion tree is linear, i.e., it has a single branch from the root to the leaf. At each successive level, the problem size is reduced by $9/10$, and the work is linear in the problem size. Thus, the amount of work performed on level i is $(9/10)^i n$, and the recurrence bottoms out when $(9/10)^i n = 1$, i.e., when $n = (10/9)^i$, or $i = \log_{10/9} n$. Thus,

$$\begin{aligned}
 T(n) &= \sum_{i=0}^{\log_{10/9} n-1} (9/10)^i n \\
 &= n \sum_{i=0}^{\log n-1} (9/10)^i \\
 &\leq n \sum_{i=0}^{\infty} (9/10)^i \\
 &= n \cdot \frac{1}{1-9/10} \\
 &= O(n).
 \end{aligned}$$

It is easy to see that $T(n) = \Omega(n)$, and therefore $T(n) = \Theta(n)$.

(c) $T(n) = 16T(n/4) + n^2.$

Solution: At each level of the recursion tree, the problem size is reduced by $1/4$, but the

number of subproblems increases by a factor of 16. Thus, the amount of work performed on level i is $16^i(n/4^i)^2 = n^2$, and the recurrence bottoms out when $i = \log_4 n$. Thus,

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_4 n - 1} n^2 \\ &= \log_4 n \cdot n^2 \\ &= \Theta(n^2 \log n). \end{aligned}$$

(d) $T(n) = 7T(n/3) + n^2$.

Solution: This is a divide-and-conquer recurrence with $a = 7$, $b = 3$, $f(n) = n^2$, and $n^{\log_b a} = n^{\log_3 7}$. Since $1 < \log_3 7 < 2$, we have that $n^2 = \Omega(n^{\log_3 7 + \epsilon})$ for some constant $\epsilon > 0$. We also have $a/b^k = 7/3^2 = 7/9 < 1$, so case 3 of the Master Theorem applies, and $T(n) = \Theta(n^2)$.

(e) $T(n) = 7T(n/2) + n^2$.

Solution: This is a divide-and-conquer recurrence with $a = 7$, $b = 2$, $f(n) = n^2$, and $n^{\log_b a} = n^{\log_2 7}$. Since $2 < \log_2 7 < 3$, we have that $n^2 = O(n^{\log_2 7 - \epsilon})$ for some constant $\epsilon > 0$. Thus, case 1 of the Master Theorem applies, and $T(n) = \Theta(n^{\log_2 7})$.

(f) $T(n) = 2T(n/4) + \sqrt{n}$.

Solution: This is a divide-and-conquer recurrence with $a = 2$, $b = 4$, $f(n) = \sqrt{n}$, and $n^{\log_b a} = n^{\log_4 2} = n^{\log_4 4^{1/2}} = n^{1/2} = \sqrt{n}$. Since $\sqrt{n} = \Theta(n^{\log_4 2})$, case 2 of the Master Theorem applies, and $T(n) = \Theta(\sqrt{n} \log n)$.

(g) $T(n) = T(n - 1) + n$.

Solution: This one is not a divide-and-conquer recurrence. We can solve it using a recursion tree. The tree is linear (one branch). At each level of the tree, the problem size is reduced by 1, and the amount of computation is the same as the problem size at each level. That is,

$$\begin{aligned} T(n) &= n + (n - 1) + (n - 2) + \cdots + 1 \\ &= \frac{n(n + 1)}{2} \\ &= \Theta(n^2). \end{aligned}$$

(h) $T(n) = T(\sqrt{n}) + 1$.

Solution: Let's solve this one using a change of variables. First let $m = \lg n$ and $S(m) = T(2^m)$. Then $T(2^m) = T(2^{m/2}) + 1$, and $S(m) = S(m/2) + 1$. It is easy to use the recursion tree method to see that $S(m) = \Theta(\log m)$. Thus, $T(n) = \Theta(\log \log n)$.

4. For each of the following recurrences, find the closed-form solution. Feel free to use the Master Method, but before applying it, you should check to make sure that the conditions of the Master Theorem are satisfied.

Discussion: The solutions we derived using the Characteristic Equation Method for divide-and-conquer recurrences of the form $T(n) = aT(n/b) + f(n)$, assumed that $f(n) = \Theta(n^k)$ for some positive integer k . This applies to many of the divide-and-conquer recurrence you will come

across in the study of algorithms. The Master Method is more powerful in that it only requires $f(n)$ to be an asymptotically positive function; however, there are some additional conditions that must be checked when using this method. Specifically, in the Master Method, which of the three cases you apply is determined by comparing $f(n)$ to $n^{\log_b a}$. If $n^{\log_b a}$ grows polynomially larger than $f(n)$, then Case 1 applies; if they grow at asymptotically the same rate (within a logarithmic factor), then Case 2 can be applied; and if $f(n)$ grows polynomially larger than $n^{\log_b a}$, then Case 3 applies, but only if an additional regularity condition is satisfied. Namely for this case we must also check to see that $f(n)$ satisfies the regularity condition that $af(n/b) \leq cf(n)$ for some constant $c < 1$. Notice that for Cases 1 and 3, growing asymptotically larger is not sufficient. Rather, we must show that $f(n)$ is either asymptotically larger or smaller than $n^{\log_b a}$ by a factor of n^ϵ . Thus, there can be cases where say $n^{\log_b a}$ grows asymptotically larger than $f(n)$, but not by a polynomial factor. This case would not be covered by the Master Method. In fact, it falls in between Cases 1 and 2. Similarly, there are recurrences that fall in between Cases 2 and 3. Thus, when applying the Master Method, be careful to check these conditions.

(a) $T(n) = 4T(\frac{n}{2}) + n$

Solution: This recurrence fits the form that we derived using the Characteristic Equation Method, and since $4 > 2^1$, $T(n) = \Theta(n^{\log_2 4}) = \Theta(n^2)$.

(b) $T(n) = 4T(\frac{n}{2}) + n^2$

Solution: This recurrence fits the form that we derived using the Characteristic Equation Method, and since $4 = 2^2$, $T(n) = \Theta(n^2 \log n)$.

(c) $T(n) = 4T(\frac{n}{2}) + n^2 \lg n$

Solution: We have $a = 4$, $b = 2$, $f(n) = n^2 \lg n$ and $n^{\log_2 4} = n^2$. Since $f(n) = \Theta(n^2 \lg^k n)$, with $k = 1$, Case 2 of the Master Method applies, and we can write $T(n) = \Theta(n^2 \log^{k+1} n) = \Theta(n^2 \log^2 n)$.

(d) $T(n) = 4T(\frac{n}{2}) + n^3$

Solution: This recurrence fits the form that we derived using the Characteristic Equation Method, and since $4 < 2^3$, $T(n) = \Theta(n^3)$.

(e) $T(n) = 5T(\frac{n}{2}) + n^2 \lg n$

Solution: We have $a = 5$, $b = 2$, $f(n) = n^2 \lg n$ and $n^{\lg 5} = n^{2.3219}$. Since $f(n) = O(n^{2.3219-\epsilon})$, for $\epsilon = 0.0219$, Case 1 of the Master Method applies, and we can write $T(n) = \Theta(n^{\lg 5}) = \Theta(n^{2.3219})$.

(f) $T(n) = 3T(\frac{n}{4}) + n \lg n$

Solution: We have $a = 3$, $b = 4$, $f(n) = n \lg n$ and $n^{\log_4 3} = n^{0.7925}$. Since $f(n) = \Omega(n^{0.7925+\epsilon})$ for $\epsilon = 0.2075$, Case 3 applies if the regularity condition holds. Since $3\frac{n}{4} \lg(\frac{n}{4}) = \frac{3}{4}n \lg n - \frac{3}{2}n \leq cn \lg n$ for $c = \frac{3}{4}$, the Master Method can be applied and we have $T(n) = \Theta(n \log n)$.

(g) $T(n) = 2T(\frac{n}{2}) + \frac{n}{\lg n}$

Solution: We have $a = 2$, $b = 2$, $f(n) = n/\lg n$ and $n^{\lg 2} = n$. Since $f(n) = O(n)$ it seems that Case 1 of the Master Method may apply; however, upon checking the exact form that must be satisfied, we find that $f(n) \neq O(n^{1-\epsilon})$ for any $\epsilon > 0$. (Note: To see this, recall that we must show that $n/\lg n \leq cn^{1-\epsilon} \forall n \geq n_0$, where c and n_0 are positive constants.

Rearranging, we must show that $n^\epsilon \leq c \lg n$, which is impossible.) Therefore, the Master Method does not apply, and this recurrence falls between Cases 1 and 2. Using a recursion tree we can show that $T(n) = O(n \log \log n)$, and using the substitution method we can also verify that $T(n) = \Omega(n \log \log n)$. Thus, $T(n) = \Theta(n \log \log n)$

(h) $T(n) = 4T(\frac{n}{2}) + \frac{n}{\lg \lg n}$

Solution: We have $a = 4, b = 2, f(n) = n^2 / \lg \lg n$ and $n^{\lg 4} = n^2$. Since $f(n) = O(n^{2-\epsilon})$, for $\epsilon = 1$, Case 1 of the Master Method applies, and we can write $T(n) = \Theta(n^2)$.

(i) $T(n) = 2T(\frac{n}{2}) + n \lg \lg n$

Solution: We have $a = 2, b = 2, f(n) = n \lg \lg n$ and $n^{\lg 2} = n$. Since $f(n) = \Omega(n)$ it seems that Case 3 of the Master Method may apply; however, upon checking the exact form that must be satisfied, we find that $f(n) \neq \Omega(n^{1+\epsilon})$ for any $\epsilon > 0$. (Note: To see this, recall that we must show that $n \lg \lg n \geq cn^{1+\epsilon} \forall n \geq n_0$, where c and n_0 are positive constants. Rearranging, we must show that $\lg \lg n \geq cn^\epsilon$, which is impossible.) Therefore, the Master Method does not apply.

(j) $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$

Solution: We have $a = 2, b = 2, f(n) = \sqrt{n}$ and $n^{\log_4 2} = \sqrt{n}$. Since $f(n) = \Theta(n^{1/2} \lg^k n)$, with $k = 0$, Case 2 of the Master Method applies, and we can write $T(n) = \Theta(\sqrt{n} \log^{k+1} n) = \Theta(\sqrt{n} \log n)$.

(k) $T(n) = 2T(\frac{n}{4}) + n^{0.51}$

Solution: We have $a = 2, b = 2, f(n) = n^{0.51}$ and $n^{\log_4 2} = \sqrt{n}$. Since $f(n) = \Omega(n^{0.5+\epsilon})$ when $\epsilon = 0.01$, and checking the regularity condition we find $2(\frac{n}{2})^{0.51} = 2(\frac{1}{2})^{0.51} n^{0.51} \leq cn^{0.51}$ for $c = (\frac{1}{2})^{0.51} = 0.7022$ when n is sufficiently large. Thus Case 3 of the Master Method applies, and $T(n) = \Theta(n^{0.51})$.

(l) $T(n) = 8T(\frac{n}{3}) + n!$

Solution: We have $a = 8, b = 3, f(n) = n!$ and $n^{\log_3 8} = n^{1.8928}$. Since $f(n) = \Omega(n^{1.89+\epsilon})$ when $\epsilon = 0.11$, and checking the regularity condition we find $8(\frac{n}{3})! \leq cn!$ for $c = 0.99$ when n is sufficiently large. Thus Case 3 of the Master Method applies, and $T(n) = \Theta(n!)$.

(m) $T(n) = T(\frac{n}{2}) + \lg n!$

Solution: We have $a = 1, b = 2, f(n) = \lg n!$ and $n^{\log_2 1} = 1$. Using the fact that $\lg n! = \Theta(n \lg n)$, we can see that $f(n) = \Omega(n^{\log_2 1 + \epsilon})$ when $\epsilon = 0.5$, and checking the regularity condition we find $\lg(\frac{n}{2})! \leq c \lg n!$ for $c = 0.99$ when n is sufficiently large. Thus Case 3 of the Master Method applies, and $T(n) = \Theta(\lg n!) = \Theta(n \log n)$.