EC330 HW3 Solution

Emre Ateş

October 5, 2016

1. The master method is given below, which is used for some of the solutions:

Let $T(n) = aT(\frac{n}{b}) + f(n)$ $\qquad (a \geq 1, b > 1, f(n) \geq 0, n \in \mathbb{N})$

(i) $f(n) = O(n^{\log_b a - \epsilon})\ \exists \epsilon > 0 \implies T(n) = \Theta(n^{\log_b a})$

(ii) $f(n) = \Theta(n^{\log_b a}) \implies T(n) = \Theta(n^{\log_b a} \log n)$

(iii) $f(n) = \Omega(n^{\log_b a + \epsilon})\ \exists \epsilon > 0,\ af(\frac{n}{b}) \leq cf(n)\ \exists c < 1\ \forall n > n' \implies T(n) = \Theta(f(n))$

(a) Using master method, a = 16, b = 4, $f(n) = 2n^2$. We are in case ii, since $f(n) = 2n^2 = \Theta(n^{\log_4 16}) = \Theta(n^2)$.
Therefore, $T(n) = \Theta(n^2 \log n)$

(b) Using master method, a = 7, b = 2, $f(n) = n^2$. We are in case i, since $f(n) = n^2 = O(n^{\log_2 7 - \epsilon})$ where $0 < \epsilon < 3$.
Therefore, $T(n) = \Theta(n^{\log_2 7})$

(c) We can solve by iteration:

$$\begin{aligned}
T(n) &= \cancel{T(n-1)} + n \\
\cancel{T(n-1)} &= \cancel{T(n-2)} + n - 1 \\
\cancel{T(n-2)} &= \cancel{T(n-3)} + n - 2 \\
&\dots \\
\cancel{T(2)} &= T(1) + 2 \qquad \text{(If we sum up all of the equations u to here:)} \\
T(n) &= T(1) + \sum_{k=2}^{n} k \\
&= O(n^2)
\end{aligned}$$

(d) Via the master method, a = 2, b = 2, $f(n) = n^4$. We are in case iii, since $f(n) = n^4 = \Omega(n^{1+\epsilon})$ where $3 > \epsilon > 0$, and $2(n/2)^4 \leq cn^4$, $c > 2^{-3}$.
Therefore, $T(n) = \Theta(n^4)$.

(e) Via the master method, a = 2, b = 4, $f(n) = 3\sqrt{n}$. We are in case ii, since $f(n) = 3\sqrt{n} = \Theta n^{\log_4 2} = \Theta\sqrt{n}$.
Therefore, $T(n) = \Theta(\sqrt{n} \log n)$

(f) We will evaluate using a recursion tree, given in the last page. The resulting expression is:

$$\begin{aligned}
T(n) &= \Theta\left(\sum_{i=0}^{k} 2^i \cdot \frac{\frac{n}{2^i}}{\lg \frac{n}{2^i}}\right) \\
&= \Theta\left(\sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - \lg 2^i}\right) \\
&= \Theta\left(\sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - i}\right) \\
&= \Theta\left(n \sum_{i=1}^{\lg n} \frac{1}{i}\right) \\
&= \Theta\left(n \ln \lg n\right) && \text{(Using the harmonic series formula)} \\
T(n) &= \Theta(n \log \log n)
\end{aligned}$$

(g) We can take the log of both sides:

$$\begin{aligned}
\log T(n) &= \log(nT(n/2)^2) \\
&= 2 \log T(n/2) + \log n && \text{(Let } S(n) = \log T(n).\text{)} \\
S(n) &= 2S(n/2) + \log n && \text{(Case (i) in master method)} \\
S(n) &= \Theta(n) \\
T(n) &= 10^{\Theta(n)}
\end{aligned}$$

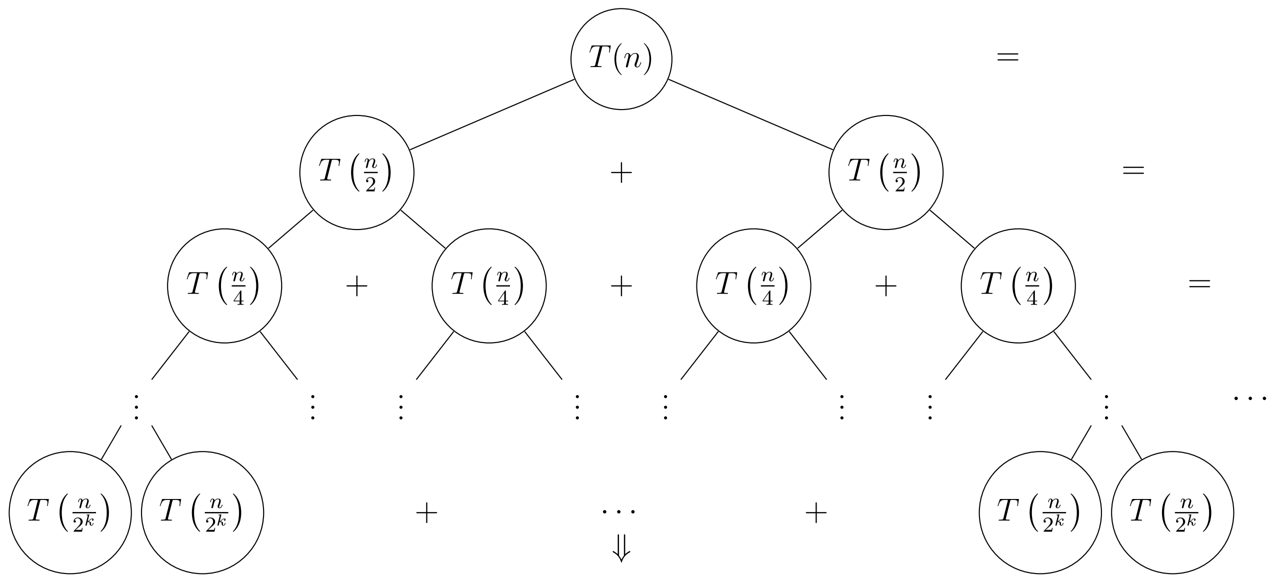
2. (a) The inner loop executes n^2 times, regardless of the outer loop. The outer loop executes n times. The total complexity is $T(n) = \Theta(n^3)$.

(b) This is a recursion that calls $f(n-1)$ 1 times per execution until completion. Therefore, $T(n) = T(n-1) + \Theta(1)$. We can solve this by iteration.

$$\begin{aligned}
T(n) &= \cancel{T(n-1)} + 1 \\
\cancel{T(n-1)} &= \cancel{T(n-2)} + 1 \\
\cancel{T(n-2)} &= \cancel{T(n-3)} + 1 \\
&\cdots \\
\cancel{T(2)} &= T(1) + 1 && \text{(If we sum up all of the equations u to here:)} \\
T(n) &= T(1) + \sum_{k=2}^{n} 1 \\
&= O(n)
\end{aligned}$$

(c) The program calls itself with the parameter n/2 2 times, until completion, and the other calculations are constant time. Therefore, the recursion relation is $T(n) = 2T(n/2) + \Theta(1)$. This is case i in the master method, since $\Theta(1) = O(n^{1-\epsilon})$. Therefore, $T(n) = \Theta(n)$.

(d) The function has an if statement, which results in different complexities for $n > 1000$ and otherwise. Since we are looking at complexity for $n > n'$, we only need to look at the $n > 1000$ case. For that case, $T(n) = \Theta(n)$, since there is only one for loop.



$$\Theta\left(\frac{n}{\lg n}\right) + 2\Theta\left(\frac{\frac{n}{2}}{\lg \frac{n}{2}}\right) + 4\Theta\left(\frac{\frac{n}{4}}{\lg \frac{n}{4}}\right) + \cdots + 2^k\Theta\left(\frac{\frac{n}{2^k}}{\lg \frac{n}{2^k}}\right)$$

$$\Downarrow$$

$$\Theta\left(\sum_{i=0}^{k} 2^i \cdot \frac{\frac{n}{2^i}}{\lg \frac{n}{2^i}}\right) = \Theta\left(\sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - \lg 2^i}\right)$$