Lecture #3:

When we study "Divide and Conquer" type algorithms, we often come across recurrence relations during the analysis phase. We need to solve these to know the complexity of these algorithms. We now take up some methods to solve such equations. This section is covered in Chapter 4 of your book.

Substitution Method (Chapter 4.1) **Step 1**: Guess the form of the solution. This is the most difficult part in this method. Very often we have seen a similar case before and gain intuition from it for this process.

Step 2: Using mathematical induction, show that the above guess works. If it does not, revise your guess and do it again!

Example 1 $t(n) = 2t(\lfloor \frac{n}{2} \rfloor) + n$

This is similar to the one we face in the analysis of merge sort.

Guess:$t(n) = O(n \lg n)$

So we need to show that there are constants $c_2 > 0$ and n_0 such that $t(n) \leq c_2 n \lg n$ for all $n \geq n_0$. We do this by using mathematical induction.

Induction Hypothesis: Assume that the above holds for smaller values of n and show that it holds for larger values. In particular, we know that $\lfloor \frac{n}{2} \rfloor < n$ for all $n \geq 3$. So we assume that the result holds for $\lfloor \frac{n}{2} \rfloor$ and show it holds for n.

$$\begin{aligned}
t(n) &= 2t(\left\lfloor \frac{n}{2} \right\rfloor) + n \\
&\leq 2c_2(\left\lfloor \frac{n}{2} \right\rfloor) \lg(\left\lfloor \frac{n}{2} \right\rfloor) + n \\
&\leq 2c_2(\frac{n}{2}) \lg(\frac{n}{2}) + n \\
&= c_2 n \lg n - c_2 n + n \\
&\leq c_2 n \lg n \text{ for } c_2 \geq 1
\end{aligned}$$

We need to also make sure that boundary conditions hold when choosing the value of c_2. We don't need to show that this holds for $n = 1$; since the definition only requires the result to hold for $n \geq n_0$, we can start this from a higher value of n. [See page 55 of your book for more on this topic].

Example 2 $t(n) = t(\lfloor \frac{n}{2} \rfloor) + t(\lceil \frac{n}{2} \rceil) + 1$

This looks like the previous example with the term n replaced by 1. The fact that we have used rounding up does not matter much as it turns out. So our guess will be a function that grows slower.

Guess: $t(n) = O(n)$

So we must show that there are constants $c_2 > 0$ and n_0 such that $t(n) \leq c_2 n$ for all $n \geq n_0$. We attempt this by induction again.

For $n \geq 3$, $\lfloor \frac{n}{2} \rfloor \leq \lceil \frac{n}{2} \rceil < n$. Therefore,

$$\begin{aligned} t(n) &= t(\left\lfloor \frac{n}{2} \right\rfloor) + t(\left\lceil \frac{n}{2} \right\rceil) + 1 \\ &\leq c_2(\left\lfloor \frac{n}{2} \right\rfloor) + c_2(\left\lceil \frac{n}{2} \right\rceil) + 1 \\ &= c_2 n + 1 \end{aligned}$$

But when $c_2 > 0$, $c_2 n + 1$ is not less than or equal to $c_2 n$ for any value of n. Our method did not work! We have two choices: Change our guess to a bigger function like $O(n^2)$ or "redo" the proof by strengthening our induction hypothesis!

New Guess:$t(n) \leq c_2 n - b$ for some positive b. Note that while this looks like a smaller value than before we may be choosing a larger value of c_2 this time around. We try again!

$$\begin{aligned} t(n) &= t(\left\lfloor \frac{n}{2} \right\rfloor) + t(\left\lceil \frac{n}{2} \right\rceil) + 1 \\ &\leq c_2(\left\lfloor \frac{n}{2} \right\rfloor) - b + c_2(\left\lceil \frac{n}{2} \right\rceil) - b + 1 \\ &= c_2 n - 2b + 1 \\ &\leq c_2 n - b \text{ for } b \geq 1 \end{aligned}$$

This time it worked! Moreover, $c_2 n - b < c_2 n$. Hence $t(n) = O(n)$ after all!

Again, we must choose these constants so that boundary conditions are satisfied.

0.0.1 Changing Variables:

Example 3 $t(n) = 2t(\sqrt{n}) + \lg n$

Let $m = \lg n$. The above recursion can now be written as

$$t(2^m) = 2t(2^{\frac{m}{2}}) + m$$

If we let $s(m) = t(2^m)$, this can be written as

$$s(m) = 2s(\frac{m}{2}) + m$$

This is much more manageable! In fact, we know the answer from previous examples. $s(m) = O(m \lg m)$

Therefore, we have:

$$t(n) = t(2^m) = s(m) = O(m \lg m) = O(\lg n \lg \lg n)$$

Please note that $\lg \lg n = \lg(\lg n)$.

Iteration Method (Chapter 4.2)

Example 4 $t(n) = 3t(\lfloor \frac{n}{4} \rfloor) + n$

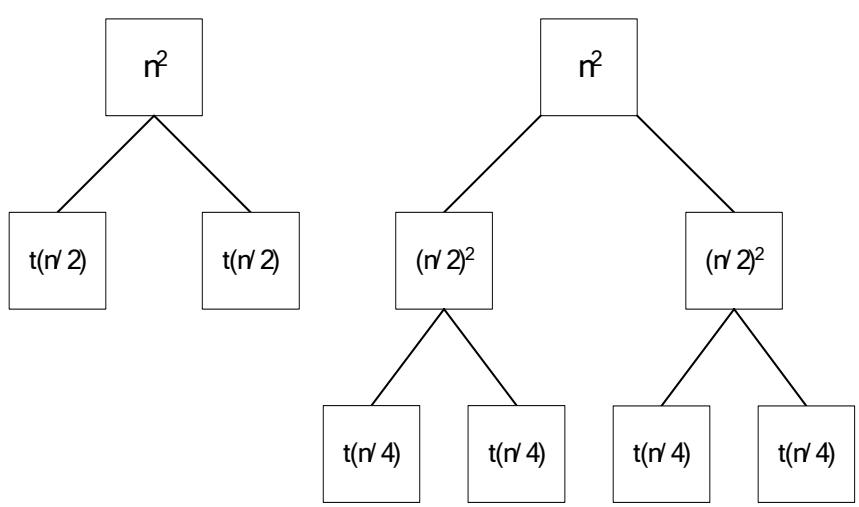
$$\begin{aligned}
t(n) &= n + 3t(\left\lfloor \frac{n}{4} \right\rfloor) \\
&= n + 3(\left\lfloor \frac{n}{4} \right\rfloor) + 9t(\frac{\lfloor \frac{n}{4} \rfloor}{4}) \\
&= n + 3(\left\lfloor \frac{n}{4} \right\rfloor) + 9t(\left\lfloor \frac{n}{16} \right\rfloor) \\
&= n + 3(\left\lfloor \frac{n}{4} \right\rfloor) + 9(\left\lfloor \frac{n}{16} \right\rfloor) + 27(\left\lfloor \frac{n}{64} \right\rfloor) + ...
\end{aligned}$$

i^{th} term in the above is $3^i \lfloor \frac{n}{4^i} \rfloor$. The series stops when $\lfloor \frac{n}{4^i} \rfloor = 1 \iff i \geq \log_4 n$. Since $\lfloor \frac{n}{4^i} \rfloor \leq \frac{n}{4^i}$,

$$\begin{aligned}
t(n) &\leq n + \frac{3}{4}n + \frac{9}{16}n + \frac{27}{64}n + ... + 3^{\log_4 n}\Theta(1) \\
&\leq \Theta(n^{\log_4 3}) + n\sum_{i=0}^{\infty}(\frac{3}{4})^i \\
&= \Theta(n^{\log_4 3}) + 4n \\
&= 4n + o(n) \\
&= O(n)
\end{aligned}$$

Recursion Trees

Example 5 $t(n) = 2t(\frac{n}{2}) + n^2$



The height of this tree is $\lg n$. Hence

$$\begin{aligned} t(n) &\leq n^2[1 + \frac{1}{2} + \frac{1}{4} + ... \\ &= 2n^2 \\ &= \Theta(n^2) \end{aligned}$$

0.0.2 Master Theorem:(page 62):

Theorem 6 (4.1) *: Let $a \geq 1, b > 1$. Let $t(n) = at(\frac{n}{b}) + f(n)$ where we interpret $\frac{n}{b}$ to include $\lfloor \frac{n}{b} \rfloor$ or $\lceil \frac{n}{b} \rceil$. $t(1) = \Theta(1)$. Then $t(n)$ can be bounded asymptotically as follows:*

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 $t(n) = \Theta(n^{\log_b a} \lg n)$.*
3. *If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(\frac{n}{b}) \leq cf(n)$ for some constant $c1$, and n sufficiently large, then $t(n) = \Theta(f(n))$.*

We will not go into the proof of this theorem (which is done using recursion trees), but mainly use it on several examples.

Example 7 *$t(n) = 2t(\frac{n}{2}) + \Theta(n)$ as in merge sort.*

Here $a = b = 2; \log_b a = 1; f(n) = \Theta(n)$. So we are in case 2 of the theorem. Hence $t(n) = \Theta(n \lg n)$

Example 8 *$t(n) = 3t(\frac{n}{2}) + cn; c > 0$.*

Here $a = 3; b = 2; \log_b a = \log_2 3 > 1; f(n) = \Theta(n)$. So we are in case 1 of the theorem. Hence $t(n) = \Theta(n^{\log_2 3})$

Example 9 *$t(n) = 7t(\frac{n}{2}) + cn^2; c > 0$.*

Here $a = 7; b = 2; \log_b a = \log_2 7 > 2; f(n) = \Theta(n^2)$. So we are in case 1 of the theorem. Hence $t(n) = \Theta(n^{\log_2 7})$

Example 10 *$t(n) = 9t(\frac{n}{3}) + n$.*

Here $a = 9; b = 3; \log_b a = \log_3 9 = 2; f(n) = \Theta(n)$. So we are in case 1 of the theorem. Hence $t(n) = \Theta(n^2)$

Example 11 *$t(n) = t(\frac{2}{3}n) + 1$.*

Here $a = 1; b = \frac{3}{2}; \log_b a = 0; f(n) = \Theta(1)$. So we are in case 2 of the theorem. Hence $t(n) = \Theta(\ln n)$

Example 12 *$t(n) = 3t(\frac{n}{4}) + n \lg n$*

Here $a = 3; b = 4; \log_b a = \log_4 3 < 1; f(n) = \Theta(n \lg n)$.

$af(\frac{n}{b}) = 3(\frac{n}{4}) \lg(\frac{n}{4}) \leq (\frac{3}{4}) n \lg n = cf(n)$ for $c = \frac{3}{4}$.
So we are in case 3 of the theorem. Hence $t(n) = \Theta(n \lg n)$

Example 13 $t(n) = 2t(\frac{n}{2}) + n \lg n$

Here $a = b = 2; \log_b a = 1; f(n) = \Theta(n \lg n) = \Omega(n)$. But $f(n) \neq \Omega(n^{1+\epsilon})$ for any $\epsilon > 0$. **So the master theorem does not apply.** A more general case is worked out below and it includes the above example.

Example 14 $t(n) = at(\frac{n}{b}) + n^{\log_b a} \cdot [\lg n]^k; b > 1; a \geq 1$

Assume that $n = b^j$;(the remaining cases are done exactly as in the book for master theorem – see Chapter 4.4.2.

$$
\begin{aligned}
t(n) &= t(b^j) \\
&= at(b^{j-1}) + a^j \cdot [j \lg b]^k \\
&= a^j \cdot [j \lg b]^k + a\{a^{j-1} \cdot [(j-1) \lg b]^k + at(b^{j-2})\} \\
&= a^j \cdot [j \lg b]^k + a^j \cdot [(j-1) \lg b]^k + a^2 t(b^{j-2}) \\
&= a^{\log_b n} [\lg b]^k \sum_{i=1}^{\log_b n} i^k + a^{\log_b n} \Theta(1) \\
&= n^{\log_b a} [\lg b]^k \sum_{i=1}^{\log_b n} i^k + n^{\log_b a} \Theta(1) \\
&= \Theta(n^{\log_b a} (\log_b n)^{k+1}) \\
&= \Theta(n^{\log_b a} (\lg n)^{k+1}) \\
&= \Theta(n^{\log_b a} \lg^{k+1} n)
\end{aligned}
$$