Lecture Notes #4

Recurrence Relations

Used to analyze the running time of divide-and-conquer algorithms.

Substitution Method

Step 1: Guess the form of the solution.

Step 2: Use mathematical induction to prove the guessed solution.

The power of this method is limited because it can be hard to guess solutions in many ways.

Example 1

\[
T(1) = 1,
T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + n.
\]

We guess \( T(n) = O(n \log n) \). But we need to show there exists \( c > 0 \) and \( n_0 > 0 \) such that \( T(n) \leq cn \log n \) for \( n > n_0 \).

Induction Hypothesis: Suppose that \( T(n) \leq cn \log n \) for \( n = 1, 2, 3, \cdots m \).

Induction step: Consider \( n = m + 1 \). Then

\[
T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + n
\leq 2c \left\lfloor \frac{n}{2} \right\rfloor \log \left\lfloor \frac{n}{2} \right\rfloor + n \text{ by hypothesis}
\leq 2c \frac{n}{2} \log \frac{n}{2} + n
= cn \log \frac{n}{2} + n
= cn \log n - cn + n
\leq cn \log n \text{ for } c \geq 1
\]

But for \( T(2) = 2(2 \log 2) = 4, T(3) = 5 < 2(3 \log 3) \). Thus, \( T(n) \leq cn \log n \) for all \( n > n_0 = 1 \) if we choose \( c = 2 \).
Example 2

\[ T(1) = 1, \]
\[ T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + T\left(\left\lceil \frac{n}{2} \right\rceil \right) + 1. \]

We guess \( T(n) = O(n) \). Need to show there exists \( c > 0 \) and \( n_0 > 0 \) such that \( T(n) \leq cn \) for \( n > n_0 \).

By substitution, we have

\[ T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + T\left(\left\lceil \frac{n}{2} \right\rceil \right) + 1 \leq c \left\lfloor \frac{n}{2} \right\rfloor + c \left\lceil \frac{n}{2} \right\rceil + 1 = cn + 1 \]

This induction shows that the method does not work! (no such \( c \) exists.)

We have two choices: (1) guess a bigger function such as \( O(n \log n) \) or \( O(n^2) \). (2) Strengthen the induction hypothesis and try again. Consider (2).

We give a new guess: \( T(n) \leq cn - b \). Then,

\[ T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + T\left(\left\lceil \frac{n}{2} \right\rceil \right) + 1 \leq c \left\lfloor \frac{n}{2} \right\rfloor - b + c \left\lceil \frac{n}{2} \right\rceil - b + 1 = cn - 2b + 1 \leq cn - b \quad \text{for any } b \geq 1 \]

Now, we choose \( b = 1 \). Then \( c \) must be chosen to be large enough to handle boundary conditions (when \( n = 1, 2, 3, 4 \)).

\[ T(2) = T(1) + T(1) + 1 = 3 \leq 2 \times 2 - 1 \]
\[ T(3) = T(1) + T(2) + 1 = 1 + 3 + 1 = 5 \leq 2 \times 3 - 1 \]
\[ T(4) = T(2) + T(2) + 1 = 3 + 3 + 1 = 7 \leq 2 \times 4 - 1 \]

Choose \( c = 2 \). Then, \( T(n) \leq 2n - 1 = O(n) \).
Changing Variables
A method for transforming a seemingly difficult recurrence relation to an easier one.

Example 3

\[
\begin{align*}
T(1) &= 1, \\
T(n) &= 2T(\lfloor \sqrt{n} \rfloor) + \lg n.
\end{align*}
\]

Let \( m = \lg n \) (i.e. \( n = 2^m, \sqrt{n} = 2^{\frac{m}{2}} \)). Then, by ignoring rounding off and making \( \sqrt{n} \) to be integer, we have

\[
T(n) = T(\lfloor \sqrt{n} \rfloor) + \lg n \Leftrightarrow T(2^m) = 2T(2^{\frac{m}{2}}) + m.
\]

Renaming \( S(m) = T(2^m) \), we have

\[
S(m) = 2S\left(\frac{m}{2}\right) + m.
\]

By Example 1, we know that

\[
S(m) = O(m \lg m).
\]

Then, \( T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg(\lg n)) \).

Recursion Tree
Useful for getting a good guess.

Example 4

\[
\begin{align*}
T(1) &= 1, \\
T(n) &= 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2).
\end{align*}
\]

Consider the recursion tree obtained in Figure 1. We have the following observations:

- The subtree rooted at level \( i \) corresponds to \( T(\frac{n}{4^i}) \).
- The tree has \( \log_4 n + 1 \) levels (level 0, 1, ..., \( \log_4 n \)).
- \( 3^i \) nodes in level \( i \).
- Each node in level \( i \) contributes a value of \( 3^i c\left(\frac{n}{4^i}\right)^2 = (\frac{n}{4^i})^i cn^2 \).
- Last level, level \( \log_4 n \), has \( 3^{\log_4 n} n = n^{\log_4 3} \) nodes, each contributing a value \( T(1) \) for a total cost of \( n^{\log_4 3} T(1) = O(n^{\log_4 3}) \).
Figure 1: Recursion tree for $T(n) = 3T\left(\left\lfloor \frac{n}{4} \right\rfloor \right) + cn^2$. 

(a) $c n^2$  
\[ T(n/4) \quad T(n/4) \quad T(n/4) \]

(b) $c n^2$  
\[ (n/4)^2 \quad (n/4)^2 \quad (n/4)^2 \]
\[ T(n/16) \quad T(n/16) \quad T(n/16) \quad T(n/16) \quad T(n/16) \quad T(n/16) \quad T(n/16) \quad T(n/16) \]

(c) $c n^2$  
\[ (n/4)^2 \quad (n/4)^2 \quad (n/4)^2 \]
\[ (n/16)^2 \quad (n/16)^2 \quad (n/16)^2 \quad (n/16)^2 \quad (n/16)^2 \quad (n/16)^2 \]
\[ T(1) \quad T(1) \quad T(1) \quad T(1) \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad T(1) \quad \text{level } \log_4 n \]
Total value:

\[
T(n) = \sum_{i=0}^{\log_4 n-1} \left( \left( \frac{3}{16} \right)^i cn^2 \right) + \Theta(n^{\log_4 3})
\]
\[
< \sum_{i=0}^{\infty} \left( \left( \frac{3}{16} \right)^i cn^2 \right) + \Theta(n^{\log_4 3})
\]
\[
= \frac{1}{1 - \frac{3}{16}} cn^2 + \Theta(n^{\log_4 3}) \quad \text{by } \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}, \text{ if } x < 1 \ (\text{see } (A.6) \text{ on page 1060})
\]
\[
= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3})
\]

By this derivation, we guess that \(T(n) = dn^2\). We use substitution to verify this:

\[
T(n) \leq 3T \left( \left\lfloor \frac{n}{4} \right\rfloor \right) + cn^2
\]
\[
\leq 3d \left( \left\lfloor \frac{n}{4} \right\rfloor \right)^2 + cn^2
\]
\[
\leq 3d \left( \frac{n}{4} \right)^2 + cn^2
\]
\[
= \frac{3}{16} dn^2 + cn^2
\]
\[
\leq dn^2 \quad \text{as long as } d \geq \frac{16}{13} c
\]

Note: \( \frac{3}{16} dn^2 + cn^2 \leq dn^2 \Rightarrow \frac{3}{16} d + c \leq d \Rightarrow d - \frac{3}{16} d \geq c \Rightarrow d \geq \frac{16}{13} c.\)