Recurrence Relations: Ch 4, Lect #3, HW #2.

These are required in the analysis of algorithms—particularly Divide-and-Conquer algorithms. First we discuss how they arise via a few examples. Then various methods of solving them.

Consider the example of "Binary Search".

Here we have

**Input**: A sorted array \( A[1, 2, \ldots, n] \) of numbers [totally ordered set] and a query element \( Z \) (also a number).

**Output**: An index \( j \) such that

\[ A[j] \leq Z < A[j+1] \]

(For sake of simplicity let us assume that no two elements in \( A \) are equal)

**Operations used**: Comparisons of the form


(Three-way Comparisons)
Now we describe a "particular" algorithm that is valid for this problem.

Algorithm: [An example of a divide-and-conquer alg]

Step 1: Compare $A\left[\left\lceil \frac{n}{2}\right\rceil\right] = z$

Step 2: if $A\left[\left\lceil \frac{n}{2}\right\rceil\right] = z$ Stop

& output $j = \left\lceil \frac{n}{2}\right\rceil$

Step 3 else if $A\left[\left\lceil \frac{n}{2}\right\rceil\right] < z$ do

BINARY-SEARCH $\left(A\left[\left\lceil \frac{n}{2}\right\rceil + 1, \ldots, n\right], z\right)$

Step 4 else (i.e. $A\left[\left\lceil \frac{n}{2}\right\rceil\right] > z$)

do BINARY-SEARCH $\left(A\left[1, \ldots, \left\lfloor \frac{n}{2}\right\rfloor\right], z\right)$

Above description is called a "pseudo-code" and will be used throughout this course.

Analysis: (a) time complexity

$t(n) = \# \ of \ steps \ (comparisons) \ taken \ by \ the$

above algorithm (in the worst-case) for an input of size $n$

$t(1) = 1 \ : \ BASIS \ CASE \ (Boundary \ Condition)$

$t(n) = t\left(\left\lceil \frac{n}{2}\right\rceil + 1\right) + 1 \ \ n \geq 2$

Called Recurrence Relation for this algorithm.
Example 2: (From children's toy) NAME: Knex Tower??

There are \( n \) rings stacked on a rod in
decreasing order of ring size (larger one
below smaller one). There are 2 "Spare"
rods. (Rods numbered 1, 2, 3)

\[ \text{Input:} \quad \text{Rod 1} \quad \text{Rod 2} \quad \text{Rod 3} \]

Want to move the "set" of rings to Rod 2.

Operation: Move the ring on top from one
rod to another; but with a
restriction: At no time should a
large ring sit on top of a
smaller one on ANY rod.

Want to minimize the number of ring moves.

Algorithm is stated in a "recursive" manner.

[Most divide-and-conquer algorithms are
done this way: "top-down" manner.]
Algorithm:

Recursive step: More "top" (n-1) rings from Rod 1 to Rod 3 using Rod 2 as a spare rod. (with all restrictions of original obeyed)

Step 2: Move the "last" (bottom) ring from Rod 1 to Rod 2 in one step. (Note: At this point Rod 2 is "empty")

Step 3: Now move the (n-1) rings on Rod 3 to Rod 2 using Rod 1 as a spare rod.

Recurrence relation: \[ t(1) = 1 \]
\[ t(n) = t(n-1) + 1 + t(n-1) = 2t(n-1)+1 \]
Solution Methods:
1. Substitution Method
2. Change of Variables
3. Recursion Trees - Geometric
4. Iteration Method - Algebraic
5. Master Theorem [as in Master Plan]
   [Not Master's Theorem]

1. Substitution Method:

Example 1: \( t(n) = 2t\left(\frac{n}{2}\right) + n \); \( t(1) = O(1) \) (constant)
   (You may take \( t(n) = 1 \) as well.

Step 1: Guess an answer (most difficult at this stage) (What if guess is not correct? → We will come to this later). How is the "guess" done?
   (We will talk more about this later)

Say guess is \( t(n) = O(n \log n) \) (X)

Step 2 (Attempt to) Prove that the guess is correct. Showing (X) is equivalent to

Showing \( \exists c < \infty , n_0 \) such that \( t(n) \leq c \cdot n \log n \) \( \forall n \geq n_0 \)
Proof: By induction on \( n \). There are two slightly different forms of proof by induction; we use "Strange" version.

Basis case is done by choosing \( c \), appropriately.

I.H.: Assuming that the result is true for value \( < n \) and show for \( n \).

Since \( \left\lfloor \frac{n}{2} \right\rfloor < n \quad \forall \; n \geq 1 \),

By I.H. we have \( t(\left\lfloor \frac{n}{2} \right\rfloor) \leq c \cdot \left\lfloor \frac{n}{2} \right\rfloor \log \left( \frac{n}{2} \right) \)

Hence \( t(n) \leq 2 \cdot c \cdot \left\lfloor \frac{n}{2} \right\rfloor \log \left( \frac{n}{2} \right) + n \)

\[ \leq 2 \cdot c \cdot \frac{n}{2} \cdot \log n \cdot ( \log n - 1 ) + n \]

\[ \leq c \cdot n \cdot \log n - (c - 1) \cdot n \]

If we choose \( c > 1 \), (we have the choice) then

\( t(n) \leq c n \cdot \log n \).

This completes proof and hence the guess is correct.
Example 2

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

When we compare this with our previous example, the "main" difference is we have 1 instead of \( n \). So we expect result for this example to be "smaller" than that of previous example when the solution was \( O(n \log n) \). So we make our guess also "smaller" than \( n \log n \). If we go too far in reduction, we may land up in trouble. So we guess for this example that \( t(n) = O(n) \).

To prove our guess is correct we need to show that \( \exists c > 0, \forall n \geq n_0 \) such that

\[ t(n) \leq cn \quad \forall n \geq n_0 \]  \( \star \star \)

Now we attempt this proof also by induction.

I. If \( \star \star \) holds for values \( < n \).

\[ \left\lfloor \frac{n}{2} \right\rfloor \leq \left\lceil \frac{n}{2} \right\rceil < n \quad \forall n \geq 2. \]
Hence

\[ t(\left\lfloor \frac{n}{2} \right\rfloor) \leq c \left\lfloor \frac{n}{2} \right\rfloor \]
\[ t(\left\lceil \frac{n}{2} \right\rceil) \leq c \left\lceil \frac{n}{2} \right\rceil \]

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

Note we "needed" to prove that \( t(n) \leq cn \)

And we failed. Now what? One idea is to "increase" our guess. on the basis that "guess" was incorrect. The other possibility (very counter-intuitive) is that our "proof method" was incorrect due to our "weak" I. H. and we should "strengthen" I. H. How? \( c < \alpha \)

New I. H.: \( t(n) \leq cn - b \), \( b > 0 \)

\[ \begin{align*}
\therefore t(\left\lfloor \frac{n}{2} \right\rfloor) & \leq c \left\lfloor \frac{n}{2} \right\rfloor - b \\
\quad t(\left\lceil \frac{n}{2} \right\rceil) & \leq c \left\lceil \frac{n}{2} \right\rceil - b \\
\therefore t(n) & \leq c \left\lfloor \frac{n}{2} \right\rfloor - b + c \left\lceil \frac{n}{2} \right\rceil - b + 1 \\
& = cn - b - (b - 1) \leq cn - b
\end{align*} \]

Proof works

\[ \therefore t(n) \in \mathcal{O}(n). \]
2. Change of Variables:

Example: $t(n) = 2t(\sqrt{n}) + \log n$.

The idea is to make a difficult case into an easier one by changing variables. What we don't like in the above is $\sqrt{n}$ on the right and we want to get "rid" of it.

So let $\log n = m$.

\[ n = 2^m \]

\[ \sqrt{n} = 2^{m/2} \]

Hence above equation becomes

\[ t(2^m) = 2t(2^{m/2}) + m \]

Now let $t(2^x) = s(x)$; the equation changes to

\[ s(m) = 2s(\frac{m}{2}) + m \]

(something we have seen before)

\[ = \Theta(m \log m) \]

\[ \therefore \quad t(n) = \Theta(\log n \log \log n) \]
3. Recursion Trees

Example: \( t(n) = 2t\left(\frac{n}{2}\right) + n^2 \)

\[ = 2t\left(\frac{n}{2}\right) + n^2 \]

Pictorial Form

\[ \begin{array}{c}
\text{n}^2 \\
\downarrow \\
t\left(\frac{n}{2}\right) \\
\text{t}(n) \\
\end{array} \]

Combine

\[ \begin{array}{c}
\text{n}^2 \\
\downarrow \\
\frac{\text{n}}{2} \\
\text{t}(\frac{n}{2}) \\
\end{array} \]

Keep expanding till we come to boundary state
\( t\left(\frac{n}{2^j}\right) = t(1) \). These will be leaf nodes.
Height of this tree is $h = \log n$.

$$2^j = n; \quad j = \log n$$

All leaf nodes: value $t(i) = \Theta(1) = \frac{1}{\text{say}}$

$$t(n) = n^2 \left(1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{j-1}}\right) + 2^j \Theta(1)$$

$$\leq n^2 \cdot \frac{1}{1-\frac{1}{2}} + \Theta(n)$$

$$= \Theta(n^2)$$

4. Iteration Method

This is the algebraic version of Recursion Tree; Comes in handy if coefficients might not be integers.

Example:

$$t(n) = 3 \cdot t \left(\left\lfloor \frac{n}{4} \right\rfloor \right) + n$$

(1)

$$t \left(\left\lfloor \frac{n}{4} \right\rfloor \right) = 3 \cdot t \left(\left\lfloor \frac{n}{4} \right\rfloor \right) + \frac{n}{4}$$

(2)

Using (2) in (1) we get

$$t(n) = n + 3 \cdot \frac{n}{4} + 3^2 \cdot t \left(\left\lfloor \frac{n}{4^2} \right\rfloor \right)$$

Expanding this further in a similar manner we get

$$t(n) = n + 3 \cdot \frac{n}{4} + 3^2 \cdot \frac{n}{4^2} + 3^3 \cdot \frac{n}{4^3} + \ldots$$

$i^{th}$ term: $3^i \cdot \frac{n}{4^i}$; Series stops: $\frac{n}{4^3} = 1$
Steps when $i = \log_4 n$. This is the height.

\[
\lfloor \frac{n}{4^i} \rfloor \leq \frac{n}{4^i}
\]

\[ t(n) = \frac{3}{4} n + \left(\frac{2}{4}\right)^2 n + \ldots + 3 \log_4 n \theta(n) \]

\[ \leq n \cdot \frac{1}{1 - \frac{2}{4}} + \frac{\log_3 3}{n} \sum_{i=0}^{\log_4 n} \log_4 3 \]

\[ = \Theta(n) \]

---

**Master Theorem**

Instead of repeatedly deriving similar results, we use a Master theorem that applies to several (but not all) instances at once. While there are different Master theorems, we use the one in your book.

**Theorem**: Let $a \geq 1$, $b > 1$ be constants independent of $n$. Let the recurrence relation be of the form:

\[ t(n) = a \cdot t\left(\frac{n}{b}\right) + f(n) \]

[For the purpose of this theorem, there is no distinction between $\left\lfloor \frac{n}{b} \right\rfloor$, $\left\lceil \frac{n}{b} \right\rceil$, and $\frac{n}{b}$. This is a great advantage from the use of this theorem.]
Then,

1. \( t(n) = \Theta(n \log^a n) \) if \( f(n) = O(n \log^a n - \varepsilon) \) for some (one) choice of \( \varepsilon > 0 \).

2. \( t(n) = \Omega(n \log^a \log n) \) if \( f(n) = \Theta(n \log^a n) \).

3. \( t(n) = \Omega(f(n)) \) if \( f(n) = \Omega(n \log^a n + \varepsilon) \) for some \( \varepsilon > 0 \) and \( \frac{n}{f(n)} \leq c \cdot f(n) \) for some \( c < 1 \).

The above cases are **mutually exclusive** (no two can happen simultaneously) but not collectively exhaustive (do not cover all equations of the type mentioned in the theorem). For these, we must go back to iteration method.

We will **ONLY** use Master theorem; proof is complicated as in the book [you are not responsible for the proof].