Recitation 3 - Recurrence Relations, Divide & Conquer

Sebastian Laudenschlager

sebastian.laudenschlager@colorado.edu

February 2, 2018

Basics of recurrence relations

- What are they?
- Why do we need them?
- How do we "solve" them?

Recurrence relations, cont.

- Recurrence relations specify the *n*-th value in a sequence based on previous value(s) in the sequence (hence the term recurrence / recursion)
  - Note that recurrence relations can be based on multiple previous values, e.g. $T(n) = T(n-1) + T(n-2)$
- Recurrence relations arise naturally when analyzing the complexity of recursive algorithms (like merge sort)

Expansion method

- One way to solve recurrence relations is via the so-called "expansion method"
- This method basically involves expanding the recurrence until a pattern is discernible, then applying the base case and solving

Recurrence relation examples

Consider the recurrence relation

$$
\begin{aligned}
T(n) &= T(n-1) + 3n \\
T(0) &= 1
\end{aligned}
$$

- Find the pattern by using the expansion method
- Apply the base case
- Then solve to get a $\mathcal{O}$ bound

Example, cont.

$$
\begin{aligned}
T(n) &= T(n-1) + 3n \\
&= T(n-2) + 3(n-1) + 3n \\
&= T(n-3) + 3(n-2) + 3(n-1) + 3n
\end{aligned}
$$

- What's the pattern?

Example, cont.

$$T(n) = T(n-k) + 3(n-k+1) + \ldots + 3n$$

- Base case?
- The base case is $T(0)$, which will occur when $n - k = 0$

$$\begin{aligned} T(n) &= T(0) + 3(1) + 3(2) + \ldots + 3n \\ &= T(0) + 3(1 + 2 + 3 + \ldots + n) \\ &= T(0) + 3\left(\frac{n(n+1)}{2}\right) \\ &= 1 + \frac{3}{2}n^2 + \frac{3}{2}n \\ &\in \mathcal{O}(n^2) \end{aligned}$$

Another example

Consider the recurrence relation

$$T(n) = T\left(\frac{n}{2}\right) + 1$$
$$T(1) = 1$$

- Here, $n > 1$ and is a power of 2.
- Solve it and prove it by induction.

Solution

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + 1 \\
&= T\left(\frac{n}{4}\right) + 2 \\
&= T\left(\frac{n}{8}\right) + 3 \\
&\vdots \\
&= T\left(\frac{n}{2^k}\right) + k
\end{aligned}
$$

Base case

- Eventually we should end up at our base case.
- Via our pattern, this would mean that $\frac{n}{2^k} = 1 \Rightarrow k = \log n$.

Plug it in

- Plugging this value in for k, we get:

$$\begin{aligned} T(n) &= T(1) + \log n \\ &= \log n + 1 \end{aligned}$$

- This is the solution to the recurrence relation.
- But really, we should prove that it is in fact correct.
- $\rightarrow$ Induction!

Proof of correctness

- So we want to show that the solution to our recurrence relation is $T(n) = \log n + 1$.
- Base case?
  - $T(1) = \log 1 + 1 = 1$

Induction hypothesis & step

- Assume that $T(k) = \log k + 1$.

$$\begin{aligned} T(2k) &= T\left(\frac{2k}{2}\right) + 1 \\ &= T(k) + 1 \\ &= \log k + 2 \\ &= \log k + \log 2 + 1 \\ &= \log(2k) + 1 \end{aligned}$$

- So we're done.

Recurrence Tree method

- Visualization of the recurrence relation.
- Draw a recursion tree and keep track of the cost at each recursive level.
- Sum up the cost of all recursion levels.
- Remember the general form of a recurrence relation:
  - $T(n) = aT(\frac{n}{b}) + f(n)$.
  - Here, a represents the number of subproblems to split into.
  - b represents the fractional size of each subproblem.
  - $f(n)$ represents the cost of the subproblem of size n.

Example

- Consider the recurrence relation $T(n) = 3T(\frac{n}{4}) + cn^2$.
- Let's start by translating this into a recursion tree.
- At each level of recursion, split into 3 subproblems of size $\frac{n}{4}$.
- How many levels are there?
  - Bottom level has subproblems of size 1, and subproblems at level i have size $\frac{n}{4^i}$.
  - Thus, we need $\frac{n}{4^i} = 1 \Rightarrow i = \log_4 n$.
  - This means our tree depth is $\log_4 n$.

Example, cont.

- Since each problem splits into 3 subproblems and our tree depth is $\log_4 n$, the bottom level has $3^{\log_4 n}$ nodes.
- $3^{\log_4 n} = n^{\log_4 3}$, and each node has a constant cost, this means that the cost of the bottom level is $\Theta(n^{\log_4 3})$.

Example, cont.

- So the total cost of this recursion tree is given by

$$
\begin{aligned}
T(n) &= cn^2 + \left(\frac{3}{16}\right) cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \ldots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 \\
&\quad + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1 - \frac{3}{16}} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= \mathcal{O}(n^2)
\end{aligned}
$$

Master method

- Generalization of solutions to recurrences.
- Learn the Master Theorem.
- Easy to use, but care must be taken to use the Master method properly.

Master Theorem

Master Theorem

For a recurrence relation of the form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1,\ b > 1,\ c > 0$.

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \textit{if} \quad f(n) = \mathcal{O}(n^{\log_b a - \epsilon}) \\ \Theta(n^{\log_b a} \log n) & \textit{if} \quad f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)) & \textit{if} \quad f(n) = \Omega(n^{\log_b a + \epsilon}) \end{cases}$$

Note that the third case also requires that $af(\frac{n}{b}) \leq cf(n)$ for some constant $c < 1$ and for all sufficiently large n. Also, $\epsilon > 0$ for case 1 and case 3.

Example

- Consider the following recurrence relation:

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

- What are the coefficients a, b in our general recurrence form?
- How about $f(n)$?
- Does the Master method apply to this recurrence? Which case?

Example, cont.

- $a = 4$, $b = 2$.
- Note that $f(n) = n = \mathcal{O}(n^{\log_2 4 - \epsilon}) = \mathcal{O}(n^{2-\epsilon})$, which is true for some $0 < \epsilon < 1$.
- So by case 1 of the Master Theorem, we have that $T(n) = \Theta(n^{\log_2 4}) = \Theta(n^2)$.

Another example

- Consider the following recurrence relation:

$$T(n) = \begin{cases} 3T(\frac{n}{2}) + \Theta(n) & \textit{if} \quad n > 1 \\ \Theta(1) & \textit{if} \quad n \leq 1 \end{cases}$$

- Apply the Master Theorem to show that the solution to this recurrence relation is $\mathcal{O}(n^{\log_2 3})$.

Solution

- $f(n) = \Theta(n) = \mathcal{O}(n^{\log_2 3 - \epsilon})$ for any $0 < \epsilon < \log_2 3 - 1$
- Case 2 of master theorem $\Rightarrow T(n) = \Theta(n^{\log_2 3}) \Rightarrow T(n) = \mathcal{O}(n^{\log_2 3})$.