CS 461, Lecture 2

Jared Saia
University of New Mexico

Today's Outline

- Recurrence Relation Review
- Recursion Tree Method
- Master Method

Slide Bug

- Note that in last lecture, there's a bug in "Rule of Thumb Slide".
- Inequalities are wrong for O and Ω notation
- This is now fixed on slides on web page

Inequalities

- Often easier to prove that a recurrence is no more than some quantity than to prove that it equals something
- Consider: $f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$
- "Guess" that $f(n) \leq 2^n$

Inequalities (II)

Goal: Prove by induction that for $f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$, $f(n) \leq 2^n$

- Base case: $f(1) = 1 \leq 2^1$, $f(2) = 1 \leq 2^2$
- Inductive hypothesis: for all $j < n$, $f(j) \leq 2^j$
- Inductive step:

$$
\begin{aligned}
f(n) &= f(n-1) + f(n-2) && (1)\\
&\leq 2^{n-1} + 2^{n-2} && (2)\\
&< 2 * 2^{n-1} && (3)\\
&= 2^n && (4)
\end{aligned}
$$

Recursion-tree method

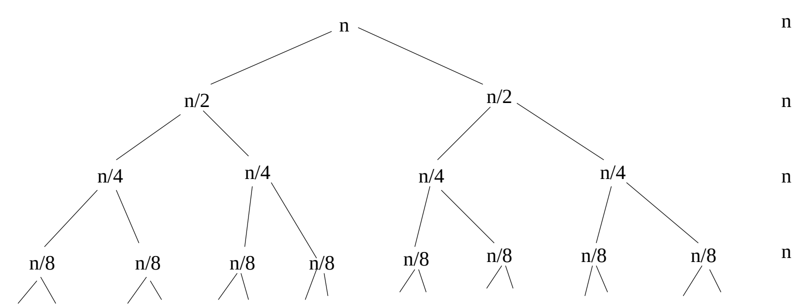
- Each node represents the cost of a single subproblem in a recursive call
- First, we sum the costs of the nodes in each level of the tree
- Then, we sum the costs of all of the levels

Recursion-tree method

- Used to get a good guess which is then refined and verified using substitution method
- Best method (usually) for recurrences where a term like $T(n/c)$ appears on the right hand side of the equality

Example 1

- Consider the recurrence for the running time of Mergesort: $T(n) = 2T(n/2) + n$, $T(1) = O(1)$

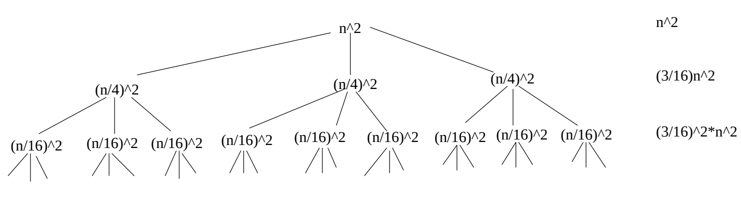


Example 1

- We can see that each level of the tree sums to n
- Further the depth of the tree is $\log n$ ($n/2^d = 1$ implies that $d = \log n$]).
- Thus there are $\log n + 1$ levels each of which sums to n
- Hence $T(n) = \Theta(n \log n)$

Example 2

- Let's solve the recurrence $T(n) = 3T(n/4) + n^2$
- Note: For simplicity, from now on, we'll assume that $T(i) = \Theta(1)$ for all small constants i. This will save us from writing the base cases each time.



Example 2

- We can see that the i-th level of the tree sums to $(3/16)^i n^2$.
- Further the depth of the tree is $\log_4 n$ ($n/4^d = 1$ implies that $d = \log_4 n$)
- So we can see that $T(n) = \sum_{i=0}^{\log_4 n} (3/16)^i n^2$

Solution

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n} (3/16)^i n^2 && (5)\\
&< n^2 \sum_{i=0}^{\infty} (3/16)^i && (6)\\
&= \frac{1}{1-(3/16)} n^2 && (7)\\
&= O(n^2) && (8)
\end{aligned}
$$

Master Theorem

- Divide and conquer algorithms often give us running-time recurrences of the form

$$T(n) = a\,T(n/b) + f(n) \tag{9}$$

- Where a and b are constants and $f(n)$ is some other function.
- The so-called "Master Method" gives us a general method for solving such recurrences when $f(n)$ is a simple polynomial.

Master Theorem

- Unfortunately, the Master Theorem doesn't work for all functions $f(n)$
- Further many useful recurrences don't look like $T(n)$
- However, the theorem allows for very fast solution of recurrences when it applies

Master Theorem

- Master Theorem is just a special case of the use of recursion trees
- Consider equation $T(n) = a\,T(n/b) + f(n)$
- We start by drawing a recursion tree

The Recursion Tree

- The root contains the value $f(n)$
- It has a children, each of which contains the value $f(n/b)$
- Each of these nodes has a children, containing the value $f(n/b^2)$
- In general, level i contains a^i nodes with values $f(n/b^i)$
- Hence the sum of the nodes at the i-th level is $a^i f(n/b^i)$

Details

- The tree stops when we get to the base case for the recurrence
- We'll assume $T(1) = f(1) = \Theta(1)$ is the base case
- Thus the depth of the tree is $\log_b n$ and there are $\log_b n + 1$ levels

Recursion Tree

- Let $T(n)$ be the sum of all values stored in all levels of the tree:

$$T(n) = f(n) + a\,f(n/b) + a^2\,f(n/b^2) + \cdots + a^i\,f(n/b^i) + \cdots + a^L\,f(n/b^L)$$

- Where $L = \log_b n$ is the depth of the tree
- Since $f(1) = \Theta(1)$, the last term of this summation is $\Theta(a^L) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$

A "Log Fact" Aside

- It's not hard to see that $a^{\log_b n} = n^{\log_b a}$

$$a^{\log_b n} = n^{\log_b a} \qquad (10)$$
$$a^{\log_b n} = a^{\log_a n * \log_b a} \qquad (11)$$
$$\log_b n = \log_a n * \log_b a \qquad (12)$$

- We get to the last eqn by taking $\log_a$ of both sides
- The last eqn is true by our third basic log fact

Master Theorem

- We can now state the Master Theorem
- We will state it in a way slightly different from the book
- Note: The Master Method is just a "short cut" for the recursion tree method. It is less powerful than recursion trees.

Master Method

The recurrence $T(n) = aT(n/b) + f(n)$ can be solved as follows:

- If $a\,f(n/b) \le f(n)/K$ for some constant $K > 1$, then $T(n) = \Theta(f(n))$.
- If $a\,f(n/b) \ge K\,f(n)$ for some constant $K > 1$, then $T(n) = \Theta(n^{\log_b a})$.
- If $a\,f(n/b) = f(n)$, then $T(n) = \Theta(f(n) \log_b n)$.

Proof

- If $f(n)$ is a *constant factor larger* than $a\,f(n/b)$, then the sum is a descending geometric series. The sum of any geometric series is a constant times its largest term. In this case, the largest term is the first term $f(n)$.
- If $f(n)$ is a *constant factor smaller* than $a\,f(n/b)$, then the sum is an ascending geometric series. The sum of any geometric series is a constant times its largest term. In this case, this is the last term, which by our earlier argument is $\Theta(n^{\log_b a})$.
- Finally, if $a\,f(n/b) = f(n)$, then each of the $L + 1$ terms in the summation is equal to $f(n)$.

Example

- $T(n) = T(3n/4) + n$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 1$,$b = 4/3$,$f(n) = n$
- Here $a\,f(n/b) = 3n/4$ is smaller than $f(n) = n$ by a factor of $4/3$, so $T(n) = \Theta(n)$

Example

- **Karatsuba's multiplication algorithm: $T(n) = 3T(n/2) + n$**
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 3$,$b = 2$,$f(n) = n$
- Here $a\,f(n/b) = 3n/2$ is bigger than $f(n) = n$ by a factor of $3/2$, so $T(n) = \Theta(n^{\log_2 3})$

Example

- **Mergesort:** $\boldsymbol{T(n) = 2T(n/2) + n}$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 2, b = 2, f(n) = n$
- Here $a\,f(n/b) = f(n)$, so $T(n) = \Theta(n \log n)$

Example

- $T(n) = T(n/2) + n \log n$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 1, b = 2, f(n) = n \log n$
- Here $a\,f(n/b) = n/2 \log n/2$ is smaller than $f(n) = n \log n$ by a constant factor, so $T(n) = \Theta(n \log n)$

In-Class Exercise

- Consider the recurrence: $\boldsymbol{T(n) = 4T(n/2) + n \lg n}$
- Q: What is $f(n)$ and $a\,f(n/b)$?
- Q: Which of the three cases does the recurrence fall under (when n is large)?
- Q: What is the solution to this recurrence?

In-Class Exercise

- Consider the recurrence: $\boldsymbol{T(n) = 2T(n/4) + n \lg n}$
- Q: What is $f(n)$ and $a\,f(n/b)$?
- Q: Which of the three cases does the recurrence fall under (when n is large)?
- Q: What is the solution to this recurrence?

Take Away

- Recursion tree and Master method are good tools for solving many recurrences
- However these methods are limited (they can't help us get guesses for recurrences like $f(n) = f(n-1) + f(n-2)$)
- For info on how to solve these other more difficult recurrences, review the notes on annihilators on the class web page.

Todo

- Read Chapter 3 and 4 in the text
- Work on Homework 1