

# Bounding Recurrences

CS 420, Fall '11, McConnell

## 1 Iteratively expanding a recurrence to get a summation.

**Example 1:**  $T(n) = 2T(n/2) + n^2$

Such an expression arises when we divide the analysis of the work of a recursive algorithm into two parts: the work that is performed by the recursive calls and the work that is performed outside of the recursive calls.

In this case, for an instance of a problem of size  $n$ , the algorithm performs two recursive calls on problems of size  $n/2$ , and spends  $O(n^2)$  time putting together the data for the recursive calls and then computing the returned value from the values that the recursive calls returned. Since we are doing big-O analysis, we can assume that there are exactly  $n^2$  operations performed outside of the recursive calls without affecting the outcome of the analysis.

For simplicity, assume that  $n$  is a power of two greater than one. If  $n \geq 4$ , then let  $m = n/2$ . Since  $T(m) = 2T(m/2) + m^2$ , it follows that  $T(n/2) = 2T(n/4) + (n/2)^2$ , hence  $T(n/2) = 2T(n/4) + n^2/4$ .

In any algebraic expression, you can replace any subexpression with something of equal value. We can replace the  $T(n/2)$  in  $T(n) = 2T(n/2) + n^2$  with  $2T(n/4) + n^2/4$ . Therefore  $T(n) = 2[2T(n/4) + n^2/4] + n^2$ . Simplifying, we get:

**2.**  $T(n) = 4T(n/4) + n^2/2 + n^2$ .

What is this expression saying? This divides the analysis into work that is performed by recursive calls that are two levels deep in the recursion tree (in the grandchildren of the main call), and work that is performed outside of those calls. There are four recursive calls generated at level two, there are  $n^2/2$  operations performed at level 1, and  $n^2$  operations performed at level 0.

If  $n \geq 8$ , let  $m = n/4$ . Since  $T(m) = 2T(m/2) + m^2$ ,  $T(n/4) = 2T(n/8) + n^2/16$ . Substituting this for  $T(n/4)$  in the expression  $T(n) = 4T(n/4) + n^2/2 + n^2$ , we get  $T(n) = 4[2T(n/8) + n^2/16] + n^2/2 + n^2$ . Simplifying, we get

**3.**  $T(n) = 8T(n/8) + n^2/4 + n^2/2 + n^2$ .

This divides the analysis into work that's performed inside the great-grandchildren calls, and work that is performed outside of them. There are  $n^2/4$  operations at level 2,  $n^2/2$  at level 1, and  $n^2$  at level 0.

A pattern is starting to emerge: each time we iterate, we pop out a new term to the right that's half as big as the previous one, the coefficient in front of the  $T$  doubles, and the parameter to  $T()$  halves. To see this, you may find it helpful to rewrite Expression 3 in a way that makes the pattern more obvious:

4.  $T(n) = 2^3T(n/2^3) + n^2/2^2 + n^2/2^1 + n^2/2^0.$

It's easy to show by induction on the number of iterations that the pattern will continue until we hit the base case of  $T(1)$ . What will it look like then?

Imagine that someone set up a tripod and took a movie lasting a few seconds of some cars traveling at constant speeds on a road. They show you the first few frames of the film, and challenge you to draw a picture of what the last frame will look like.

Your first question: *How many frames does the film have?* If they answer this, you can measure how far each car moves from one frame to the next, extrapolate, and draw a picture of what the last frame will look like without even seeing it. This assumes that nothing unexpected will happen such as cars hitting the brakes or having an accident.

Think of the initial recurrence and expressions 2 and 3 as the initial frames of a movie. The last frame is where you hit the base case of  $T(1)$ . You have to show what the last frame looks like. We can show by induction that nothing unexpected will happen to alter the pattern of change from frame to frame. This is just like the problem about the cars.

*How many frames does this movie have?* The initial parameter to  $T()$  is  $n$ , and in each frame, the parameter is half as big as it was in the previous frame. The number of frames is therefore the number of times in a row that  $n$  can be divided by two before it reaches 1. In other words, there are  $\log_2 n$  frames. Extrapolating, we can see what the last frame will look like:

5.  $T(n) = 2^{\log_2 n}T(n/2^{\log_2 n}) + n^2/2^{\log_2 n-1} + n^2/2^{\log_2 n-2} + \dots + n^2/2^2 + n^2/2^1 + n^2/2^0.$

Simplifying, we get  $T(n) = nT(1) + \sum_{i=0}^{(\log_2 n)-1} n^2/2^i$ , hence:

6.  $T(n) = n + \sum_{i=0}^{\log_2 n-1} n^2/2^i.$

The summation is a geometric series, since each term differs from the previous by a constant factor (1/2). We have previously given a nice trick for getting an asymptotic bound on a geometric series, which is to put a big- $\Theta$  around the largest term and throw away the rest. Therefore:

7.  $T(n) = n + \Theta(n^2) = \Theta(n^2).$

Let us now relax the requirement that  $n$  is a power of two. Then  $n/2$  is not necessarily an integer, so we need to add floors and ceilings to make the parameters to the recursive calls into integers. The book explains that, although this complicates the algebra, it doesn't change the outcome of the big-O analysis. This allows to get away with assuming that  $n$  is a convenient power of an integer for the analysis, and then claiming that the bound still holds when we relax this assumption.

**Example 2:**  $T(n) = 4T(n/2) + n$

Iteratively expanding this, we get:

- $T(n) = 4[4T(n/4) + n/2] + n = 4^2T(n/2^2) + 2n + n$
- $T(n) = 4^2T[4T(n/2^3) + n/4] + 2n + n = 4^3T(n/2^3) + 2^2n + 2^1n + 2^0n$ , etc.

Extrapolating to the end of the movie, which has  $\log_2 n$  frames, we see that the last frame will look like this:

- $T(n) = 4^{\log_2 n} T(1) + \sum_{i=0}^{\log_2 n - 1} 2^i n.$

Since  $a^{\log_b c} = c^{\log_b a}$  (pp. 55-56 of our text) and  $T(1) = 1$ ,  $4^{\log_2 n} T(1) = n^{\log_2 4} = n^2$ . The summation is a geometric series whose largest term is  $2^{(\log_2 n) - 1} n = \Theta(2^{\log_2 n} n) = \Theta(n^2)$ . Summarizing:

- $T(n) = \Theta(n^2).$

**Example 3:**  $T(n) = 2T(n/4) + n^{1/2}$

- $T(n) = 2[2T(n/4^2) + (n/4)^{1/2}] + n^{1/2} = 2^2 T(n/4^2) + n^{1/2} + n^{1/2}.$

- $T(n) = 4[2T(n/4^3) + (n/4^2)^{1/2}] + n^{1/2} + n^{1/2} = 2^3 T(n/4^3) + n^{1/2} + n^{1/2} + n^{1/2}, \text{ etc.}$

This time, there are  $\log_4 n$  frames of the movie instead of  $\log_2 n$ , since the parameter to  $T()$  is reduced by a factor of four each time. Extrapolating to the last frame, we get:

- $T(n) = 2^{\log_4 n} T(1) + \sum_{i=0}^{\log_4 n - 1} n^{1/2} = n^{\log_4 2} + \sum_{i=0}^{\log_4 n - 1} n^{1/2} = n^{1/2} + \sum_{i=0}^{\log_4 n - 1} n^{1/2}$

This time, the summation is not a geometric series, but it is easy to bound because all of the terms are the same:  $\sum_{i=0}^{\log_4 n - 1} n^{1/2} = n^{1/2} \log_4 n$ . Summarizing:

- $T(n) = n^{1/2} + \Theta(n^{1/2} \log n) = \Theta(n^{1/2} \log n).$

## 2 A special case of the master theorem

Notice that the three cases above illustrate three things that can happen in the analysis:

- You get a geometric series where the first term that we produce, is the largest. Since we can bound a geometric series by discarding all but the largest term and putting a big- $\Theta$  around it, we do this with the first term we produce.
- You get a geometric series where the *last* term we produce is the largest. We need to extrapolate forward to see what that will be. It's still a geometric series, so we put a big- $\Theta$  around that term.
- All terms are the same. We just multiply that term by the number of terms. We extrapolate forward to see how many terms that will be.

Now look at the master theorem: this has three cases. I want to show you that these three cases correspond to the three scenarios above. To achieve full generality, the book has to deal with a lot of tedious details. However, I can easily show you the strategy of the proof, which is a trick that is general to many problems, provided we stick with a special case of the master theorem, for recurrences of the form  $T(n) = aT(n/b) + n^x$ , with  $x > 0, b > 1$ , and  $a$  an integer greater than or equal to 1.

All of the above cases observe this form. However, in all of the above derivations, we assigned values to  $a$ ,  $b$ , and  $x$  *before* we did the iterative expansion. We committed ourselves prematurely to one instance of a recurrence.

Let's wriggle out of being pinned down by leaving  $a$ ,  $b$ , and  $x$  as variables when we do the algebra. Our final summation will be one where we can plug in values for  $a$ ,  $b$ , and  $x$  *after* we have done all that work. We don't need to repeat it all each time one of these variables changes! We will then see that the master theorem describes what we get as the big- $\Theta$  bound when we do.

- $T(n) = a[aT(n/b^2) = (n/b)^x] + n^x = a^2T(n/b^2) + (a/b^x)n^x + n^x$
- $T(n) = a^2[aT(n/b^3) + (n/b^2)^x] + (a/b^x)n^x + n^x = a^3T(n/b^3) + (a/b^x)^2n^x + (a/b^x)n^x + n^x$

Extrapolating, we get  $T(n) = a^{\log_b n}T(n/b^{\log_b n}) + \sum_{i=0}^{\log_b n - 1} (a/b^x)^i n^x = n^{\log_b a}T(1) + \Theta(\sum_{i=0}^{\log_b n - 1} (a/b^x)^i n^x)$   
 Note that  $\log_b a = \log_c a / \log_c b$  (p. 53), so  $n^{\log_b a} = n^{(\log_b a)/(\log_b b)} = n^{(\log_b a)/(1/x)} = n^{x \log_b a}$ .

- $T(n) = n^{x \log_b a} + \Theta(\sum_{i=0}^{\log_b n} (a/b^x)^i n^x)$ .

We can consider three cases:

- If  $(a/b^x) < 1$ , so is  $\log_b a$ , so  $n^{x \log_b a} = o(n^x)$ . The summation is a geometric series with largest term  $n^x$ , so  $T(n) = \Theta(n^x)$ .
- If  $(a/b^x) > 1$ , the summation is a geometric series with largest term  $\Theta((a/b^x)^{\log_b n} n^x)$ .  $T(n) = n^{\log_b a} + \Theta((a/b^x)^{\log_b n} n^x) = n^{\log_b a} + \Theta(a^{\log_b n} / (b^{\log_b n})^x n^x) = n^{\log_b a} + \Theta((a^{\log_b n} / n^x) n^x) = n^{\log_b a} + \Theta(a^{\log_b n}) = n^{\log_b a} + \Theta(n^{\log_b a}) = \Theta(n^{\log_b a})$ .
- If  $(a/b^x) = 1$ , then  $\log_b a = 1$ .  $T(n) = n^{x \log_b a} + \sum_{i=0}^{\log_b n - 1} n^x = n^{x*1} + \Theta(n^x \log n) = \Theta(n^x \log n)$ .

Summarizing, the extreme values of the summation are  $n^x$  and  $\Theta(n^{\log_b a})$ . If they differ, we have a geometric series and we select the larger of the two as the bound. If they're tied, we have  $O(\log n)$  terms, each of size  $n^x$ , so we get a  $\Theta(n^x \log n)$  bound.

The three cases are exhaustive, so we don't need to add any disclaimers about  $n^\epsilon$  or the "regularity condition." (The regularity condition is met by  $f(n) = n^x$  and if  $f(n) = n^x$  for  $x > 0$ , then  $n^{\log_b a}$  is either tied with  $n^x$  or differs by the required ratio  $n^\epsilon$ .)

### 3 The full master theorem

The master theorem is proved with a generalization of this proof that allows  $f(n)$  to be arbitrary, except for the disclaimer that it must meet the "regularity condition."

Why does it also have the disclaimer in cases 1 and 3 that  $n^{\log_b a}/f(n)$  or  $f(n)/n^{\log_b a}$  is an increasing function that is faster growing than  $n^\epsilon$  for some  $\epsilon > 0$ ?

Consider  $T(n) = 2T(n/2) + n \log_2 n$ . This is not covered by the special case of the master theorem above, since  $f(n)$  is not of the form  $n^x$ . Note that  $f(n) = n \log_2 n$  and  $n^{\log_b a} = n^{\log_2 2} = n$ . The ratio is  $O(\log n)$ , which is slower growing than  $n^\epsilon$  for any  $\epsilon > 0$ .

- $T(n) = 2[2T(n/4) + n/2 \log_2(n/2)] + n \log_2 n = 4T(n/4) + n \log_2(n/2) + n \log_2 n$

- $T(n) = 8T(n/8) = n \log_2 n/4 + n \log_2 n/2 + n \log_2 n$
- $T(n) = 2^{\log_2 n} T(1) + n(\log_2 2 + \log_2 4 + \dots + \log_2 n) = n + n(1 + 2 + 3 + 4 + \dots + \log_2 n)$ .

At this point of the analysis of our special case above, we showed that the summation was a geometric series or a sum of constant terms. Here, however, the summation is an arithmetic series. Our proof just fell apart! The caveats about  $n^\epsilon$  are disclaimers about such cases.

Note that we can still bound the recurrence easily, however. Letting  $m = \log_2 n$ , we see that  $T(n) = n + n(1 + 2 + 3 + \dots + m) = n + n\Theta(m^2) = \Theta(nm^2) = \Theta(n \log^2 n)$ .

In general, iteratively expanding a recurrence is a good backup plan when the master theorem doesn't apply.