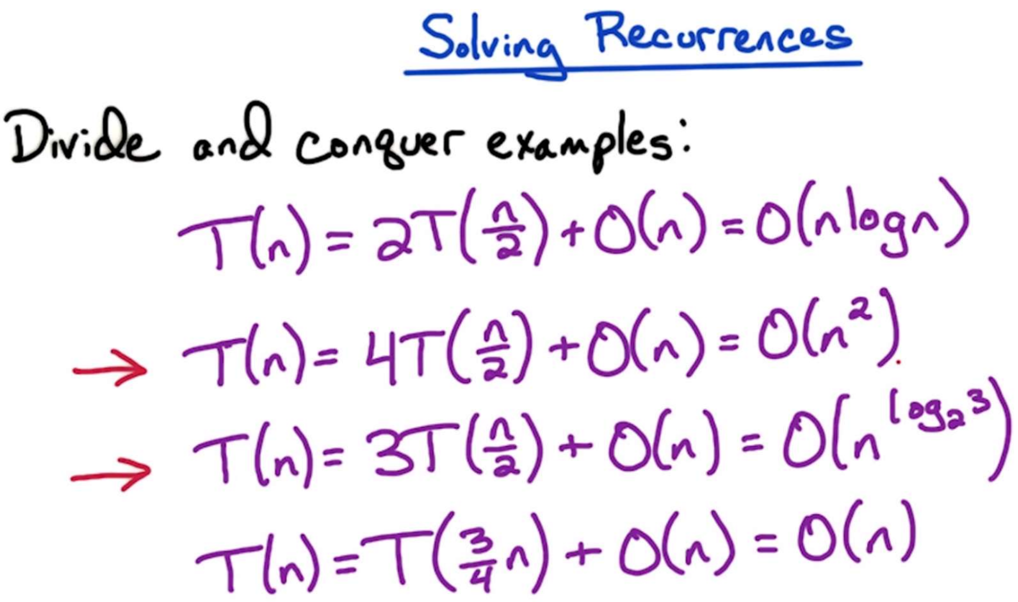
Lesson: DC3: Solving Recurrences
(Slide 1) Solving Recurrences
***



Solving Recurrences

Divide and conquer examples:

$T(n) = 2T(\frac{n}{2}) + O(n) = O(n \log n)$

→ $T(n) = 4T(\frac{n}{2}) + O(n) = O(n^2)$

→ $T(n) = 3T(\frac{n}{2}) + O(n) = O(n^{\log_2 3})$

$T(n) = T(\frac{3}{4}n) + O(n) = O(n)$

This is a short refresher lecture on Solving Recurrences. We're going to focus on the type of recurrences that arise in Divide and Conquer algorithms.

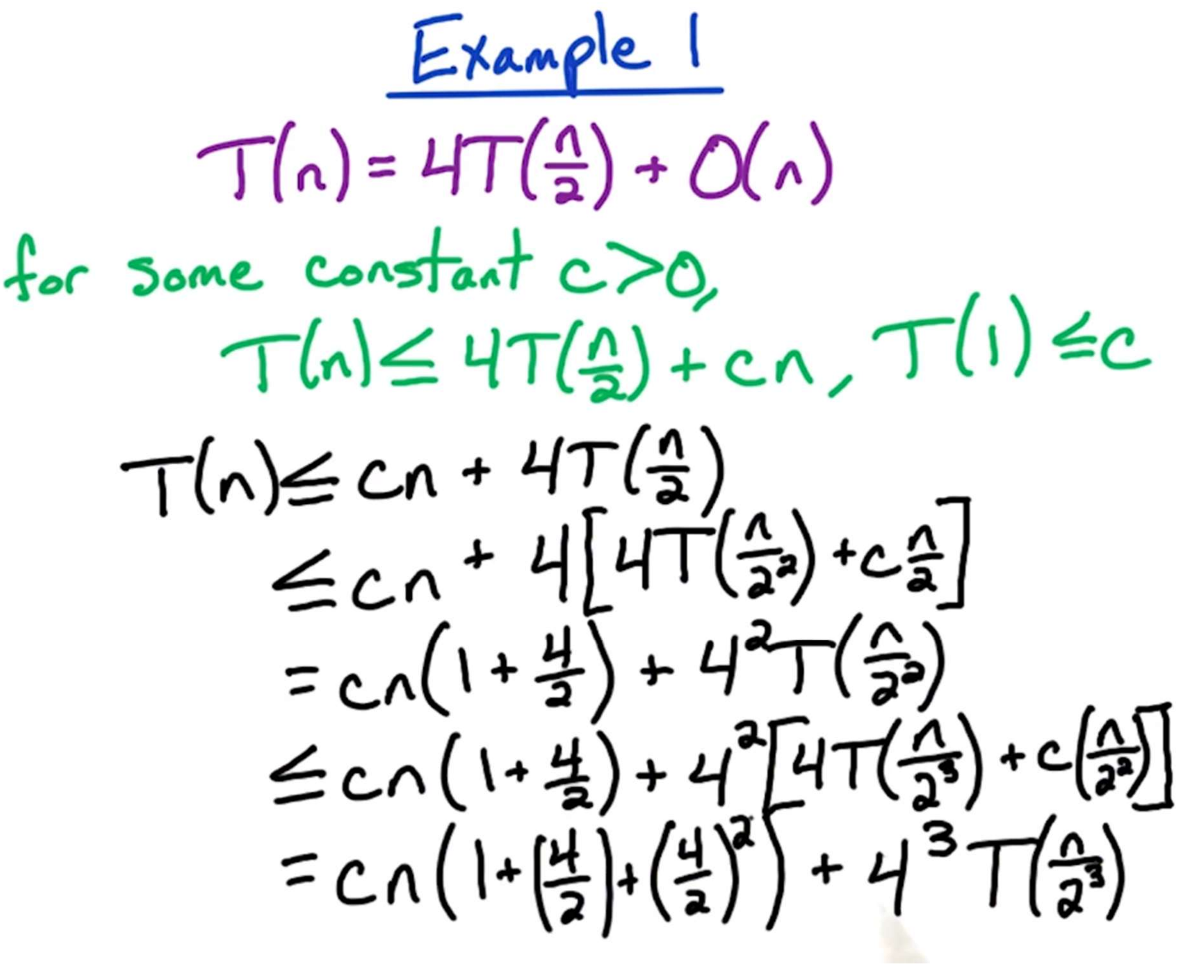
A classical recurrence is that for mergesort T(n) = 2 T(n/2) + O(n) We partition the list into the first n/2 numbers and the last n/2 numbers. We recursively sort those two sub lists, that takes time 2 T(n/2) and then it takes O(n) to merge these two sorted list together. As you probably know, this recurrence solves to O(n log n).

Let's look at some other recurrences that we've seen so far in divide and conquer examples.

- The naive algorithm for integer multiplication had the following recurrence: T(n) = 4 T(n/2) + O(n). And we claim that this solves to O(n^2). We'll see why momentarily.
- Our improved algorithm for integer multiplication had three subproblems instead of four subproblems. So we had the following occurrence: T(n) = 3 T(n/2) + O(n). We claim that this solves to O(n^log$_2$3). And recall this exponent is roughly one point six.
- Now another recurrence which we saw in the median finding algorithm was T(n) = T(3/4 n) + O(n). And we claim that this recurrence solves to O(n).

Now let's go ahead and solve these recurrences. We'll start with this recurrence. The naive algorithm for integer multiplication then we'll look at this slightly more complicated example. And finally we'll look at the general solution for these recurrences of this form.

(Slide 2) Example 1
***



Example 1

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

for some constant $c>0$,

$$T(n) \le 4T\left(\frac{n}{2}\right) + cn, \quad T(1) \le c$$

$$\begin{aligned} T(n) &\le cn + 4T\left(\frac{n}{2}\right) \\ &\le cn + 4\left[4T\left(\frac{n}{2^2}\right) + c\frac{n}{2}\right] \\ &= cn\left(1 + \frac{4}{2}\right) + 4^2 T\left(\frac{n}{2^2}\right) \\ &\le cn\left(1 + \frac{4}{2}\right) + 4^2\left[4T\left(\frac{n}{2^3}\right) + c\left(\frac{n}{2^2}\right)\right] \\ &= cn\left(1 + \left(\frac{4}{2}\right) + \left(\frac{4}{2}\right)^2\right) + 4^3 T\left(\frac{n}{2^3}\right) \end{aligned}$$

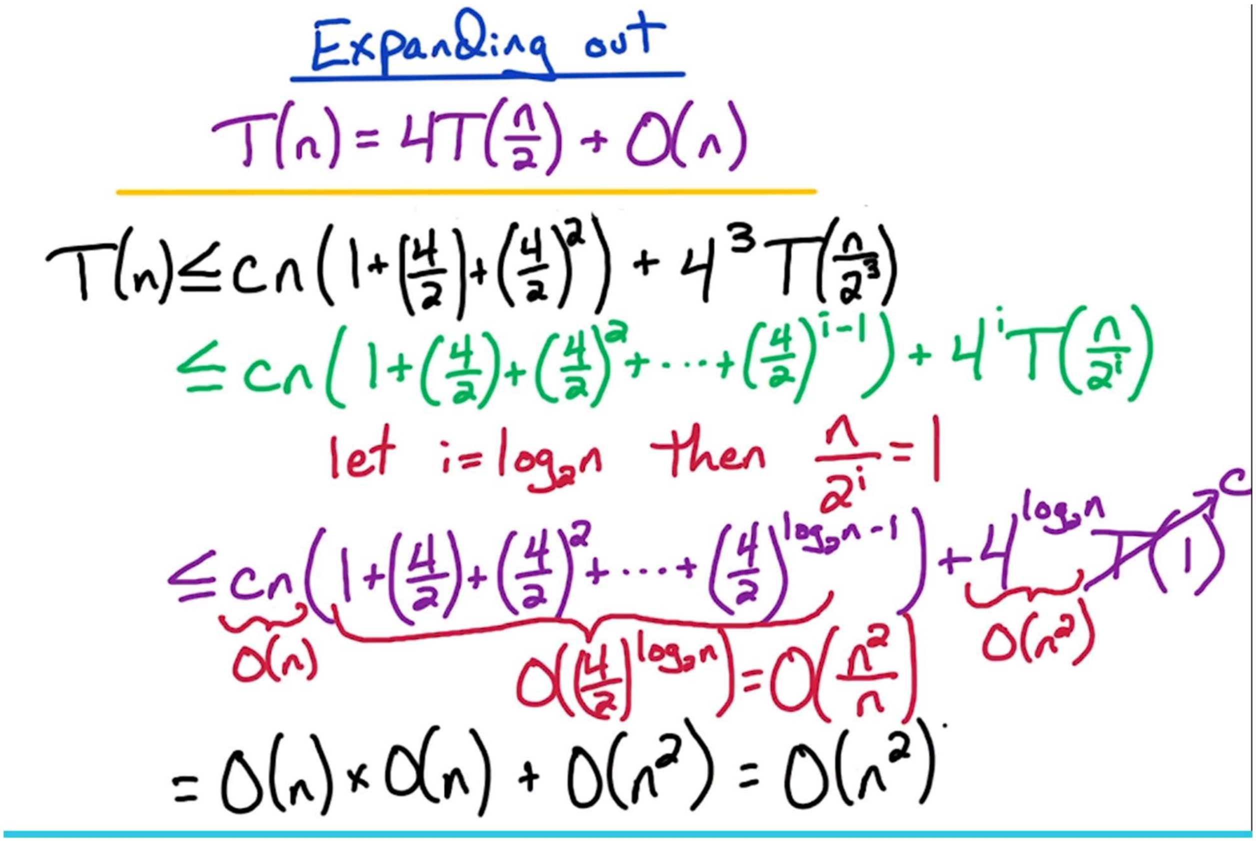
We're looking first at this recurrence, T(n) = 4T (n/2) +O (n).

- Let's start by getting rid of the big O notation. Order n means that there's some constant c so that T(n) <= 4 T(n/2) + cn. So we replaced O(n) by at most cn.
- We also have the base case, let's say the base cases of size 1. Since the input is of size 1 it must solve in O(1) runtime. So we can place that O(1) by a constant c and notice we can use the same constant c for both of these by taking the max of these two constants and calling that c.
- Let's solve for T(n). So I simply restated the recurrence, though, for convenience, I rearranged the terms so T(n) <= cn + 4T(n/2).
- Now let's replace T(n/2) by this recurrence. So replace n by n/2 so we get this term c n again plus we get 4 times … and then we replace T(n/2) with this expression. So we have 4 T(n) and we're replacing n by n/2 so we get $n/2^2$ and then for the last term we're replacing n by n/2 so instead of cn, we get c n/2. Now let's rearrange this and collect similar terms So we want to collect cn with 4 c n/2.

- So these have a common factor of cn … this term is 1 … this term is 4/2 cn and then finally, the last term is 4^2 $T(n/2^2)$.
- Now let's repeat this process and replace $T(n/4)$ by this expression. Now we're replacing n by $n/2^2$. So the first term becomes $4T(n/2^3)$, and the last term becomes $c(n/2^2)$
- Rearranging once again … so bringing this term over here.
  So we have $T(n) <= cn\ (1 + (4/2) + (4/2)^2) + 4^3\ T(n/2^3)$.
  Notice we're starting to get a geometric series here.

(Slide 3) Expanding out

***



Expanding out

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

$$T(n) \leq cn\left(1+\left(\frac{4}{2}\right)+\left(\frac{4}{2}\right)^2\right) + 4^3 T\left(\frac{n}{2^3}\right)$$

$$\leq cn\left(1+\left(\frac{4}{2}\right)+\left(\frac{4}{2}\right)^2+\dots+\left(\frac{4}{2}\right)^{i-1}\right) + 4^i T\left(\frac{n}{2^i}\right)$$

let $i = \log_2 n$ then $\frac{n}{2^i} = 1$

$$\leq \underbrace{cn}_{O(n)}\underbrace{\left(1+\left(\frac{4}{2}\right)+\left(\frac{4}{2}\right)^2+\dots+\left(\frac{4}{2}\right)^{\log_2 n-1}\right)}_{O\left(\left(\frac{4}{2}\right)^{\log_2 n}\right)=O\left(\frac{n^2}{n}\right)} + \underbrace{4^{\log_2 n}}_{O(n^2)}\overset{c}{\cancel{T(1)}}$$

$$= O(n) \times O(n) + O(n^2) = O(n^2)$$

What we've seen so far is that for this recurrence T(n) = 4 T(n/2) + O(n), by substituting in two times, we get to T(n) <= cn (1 + 4/2 + $(4/2)^2$) + 4^3 T(n/2^3). Now notice we're leaving this 4/2 and $(4/2)^2$, instead of 2 and 2^2 so that we see the geometric series that's going to arise.

And this illustrates how the geometric series that arises is related to the original recurrence. The four comes from here and the two comes from this term. At this point, we see the pattern.

So now, suppose instead of substituting in two times we did i times. Well, to be precise, let's suppose we're substituting i-1 times. So, this geometric series becomes (1 + (4/2) + $(4/2)^2$ + … + $(4/2)^{i-1}$) + 4 T(n/2^i) And with the last term, it becomes (1 + (4/2) + $(4/2)^2$ + … + $(4/2)^{i-1}$) + 4 T(n/2^i).

Now, when do we stop? What i do we stop at? We start when we get to the base case which is when this quantity is one and whether, we stop at one or a hundred, it doesn't matter. So, let i = log2n then n/2^i = 1. So, substituting in i=log2n, we get the following expression.

Now, let's try to simplify the terms here. First off, what is T(1)? T of one is the most c. So, we can upperbound this by c. What $4^{\log 2n}$? Well, this is O(n^2) cn is O(n).

Now, what about this geometric series? Now, the key thing is that this geometric series is increasing. 4/2 > 1. It's 2. So, the last term dominates. So, this whole expression is dominated by order of the last term. So, this geometric series is $O((4/2)^{log2n})$.

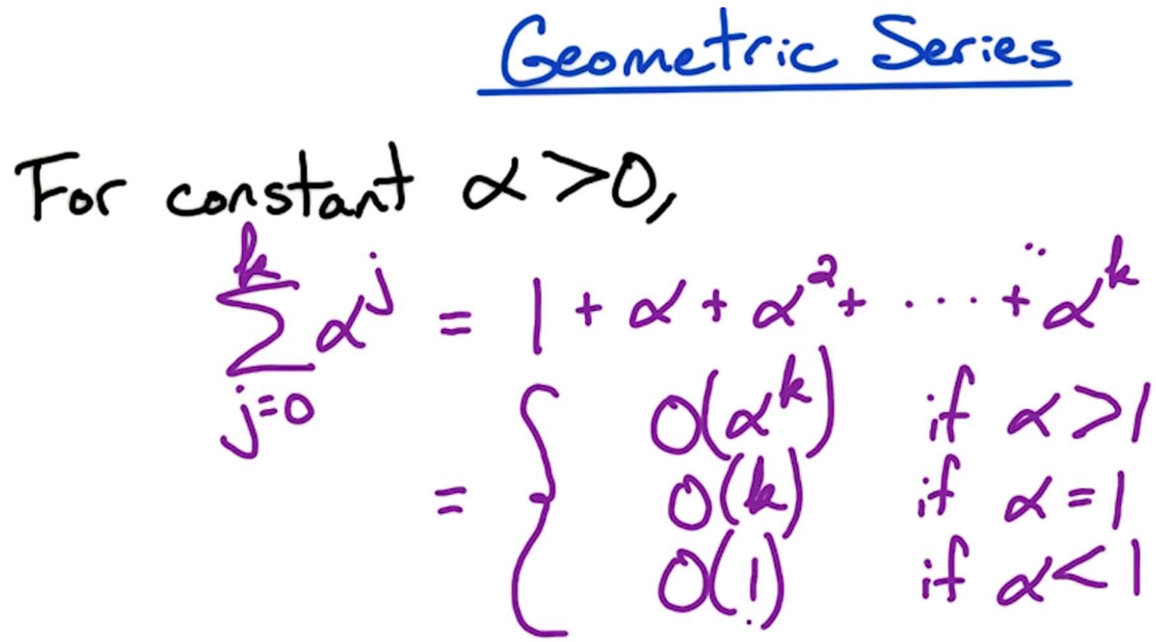
We can forget about the -1 because that's hidden in the Big O notation because that's just an extra factor of a half.

Now, let's simplify this quantity. The numerator is four to the log base 2 n, which we said before is O(n^2). The denominator is 2^log2n which is, of course, n. So, this quantity is O(n).

Plugging it in, we have O(n) x O(n) + O(n^2). So, this is O(n^2) + O(n^2) which is O(n^2). So this recurrence solves to O(n^2). The key element of these recurrences is this geometric series that arises.

So, let's take a look at geometric series in general.

(Slide 4) Geometric Series
***



Geometric Series

For constant $\alpha > 0$,

$$\sum_{j=0}^{k} \alpha^j = 1 + \alpha + \alpha^2 + \cdots + \alpha^k$$

$$= \begin{cases} O(\alpha^k) & \text{if } \alpha > 1 \\ O(k) & \text{if } \alpha = 1 \\ O(1) & \text{if } \alpha < 1 \end{cases}$$

Consider some positive constant $\alpha > 0$. In our previous example α was 4/2 and we're looking at this geometric series. (see slide for the series) Now since we are solving our recurrences within big O notation, we don't need to solve this expression exactly. We simply need to solve it within constant factors, so that makes life much easier.

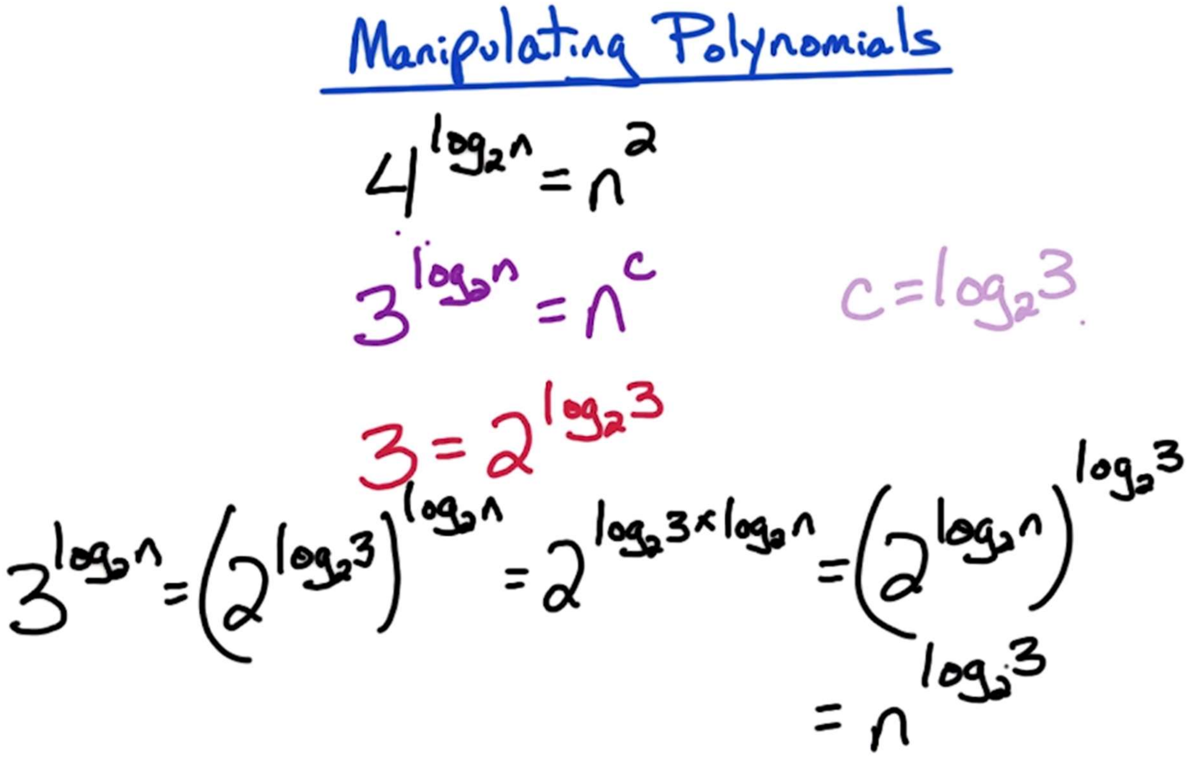
Now the key for a geometric series is to figure out which term dominates either the first term, the last term, or all terms are equal. The three cases are whether alpha > 1, alpha = 1, or alpha < 1.

- Now if alpha > 1, then each term is growing, so the last term is going to dominate. So the whole expression is going to be on the order of the last term.
- Conversely, if alpha < 1 then the terms are decreasing and the first term dominates, so the whole expression is on the order of the first term, so it's order 1.
- Finally, the middle case is if alpha = 1 then every term is exactly 1 and then the number of terms is k + 1, so the whole thing is O(k).

This first case where alpha is bigger than 1 is what arises in the recurrence we just analyzed. In that case alpha= 2 and what we saw there was that the last term dominates. When we look at mergesort, then we'll have alpha equals one, all the terms will be the same and we'll get an order again from that.

The final case is what arises in the example of the median finding algorithm, there we'll have alpha=3/4 so the geometric series which arises will be on the order of 1.

(Slide 5) Manipulating Polynomials
***



Manipulating Polynomials

$$4^{\log_2 n} = n^2$$

$$3^{\log_2 n} = n^c \qquad c = \log_2 3$$

$$3 = 2^{\log_2 3}$$

$$3^{\log_2 n} = \left(2^{\log_2 3}\right)^{\log_2 n} = 2^{\log_2 3 \times \log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 3}$$

$$= n^{\log_2 3}$$

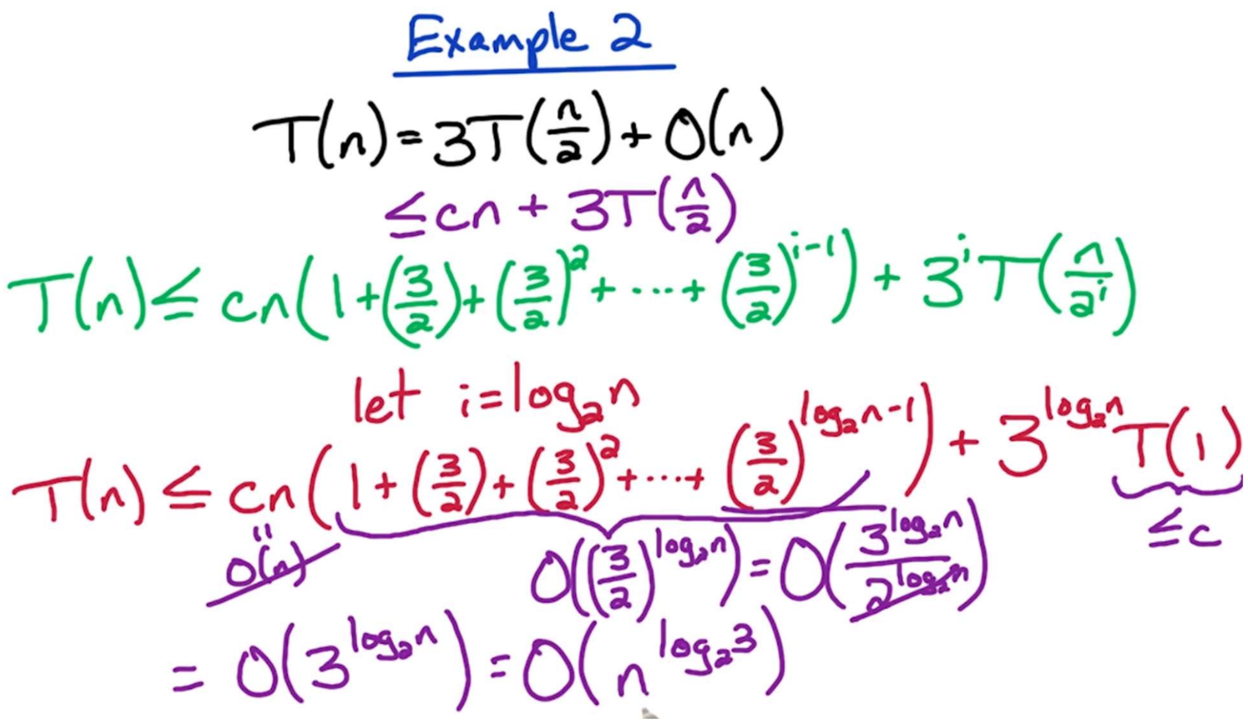
The final tool that we'll need is manipulating polynomials. What we saw in the previous example is that 4^log2(n) is the same as n^2. Now that example is quite straightforward but we're going to have other examples which arise such as 3^log2(n). And we want to convert this into a polynomial. So we want to convert this into n^c for some constant c and we want to figure out what is this constant c.

So let's see the basic recipe for converting this into a polynomial. The key is that the base of this log is 2. So I want to change this base from 3 to 2. Well, 3 is the same as 2^log2(3). That's by the definition of log.

So let's substitute this in, to this expression. So we have 3^log2(n) and then we replace 3 by this expression. So we have (2^log2(3))^log2(n). Notice these exponents multiply. So this is the same as 2^(log2(3) x log2(n)). And we can rewrite this as (2^log base2(n))^log2(3).

What we've done is just swapped these exponents. Now look at this inner quantity. That's just n, so we have n^log2(3). So we see that this exponent c is log2(3).

(Slide 6) Example 2
***



Example 2

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$$\leq cn + 3T\left(\frac{n}{2}\right)$$

$$T(n) \leq cn\left(1 + \left(\frac{3}{2}\right) + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^{i-1}\right) + 3^i T\left(\frac{n}{2^i}\right)$$

let $i = \log_2 n$

$$T(n) \leq \underset{\cancel{O(n)}}{\overset{\shortparallel}{cn}}\Big(\underbrace{1 + \left(\frac{3}{2}\right) + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^{\log_2 n - 1}}_{O\left(\left(\frac{3}{2}\right)^{\log_2 n}\right) = O\left(\frac{3^{\log_2 n}}{\cancel{2^{\log_2 n}}}\right)}\Big) + 3^{\log_2 n} \underbrace{T(1)}_{\leq c}$$

$$= O\left(3^{\log_2 n}\right) = O\left(n^{\log_2 3}\right)$$

Now let's take a look at the following recurrence, T(n) = 3T(n/2) + O(n). This is the recurrence which arised in the more sophisticated integer multiplication algorithm.

Just as before, rewriting this to eliminate the big O notation, we have:

T(n) <= cn + 3 T(n/2)

Now substituting in as we did before and repeating it i-1 times we get the following recurrence:

T(n) <= cn (1 + (3/2) + (3/2)^2 + … + (3/2)^i-1) + 3^i T(n/2^i).

Notice that, before, we had a geometric series with base (4/2) and now we have a geometric series with base (3/2). And instead of 4^i we get 3^i, this is because we replaced four with three here. Now once again we're going to stop when we get to the base cases which is when this is one. So we're going to stop when i = log2(n).

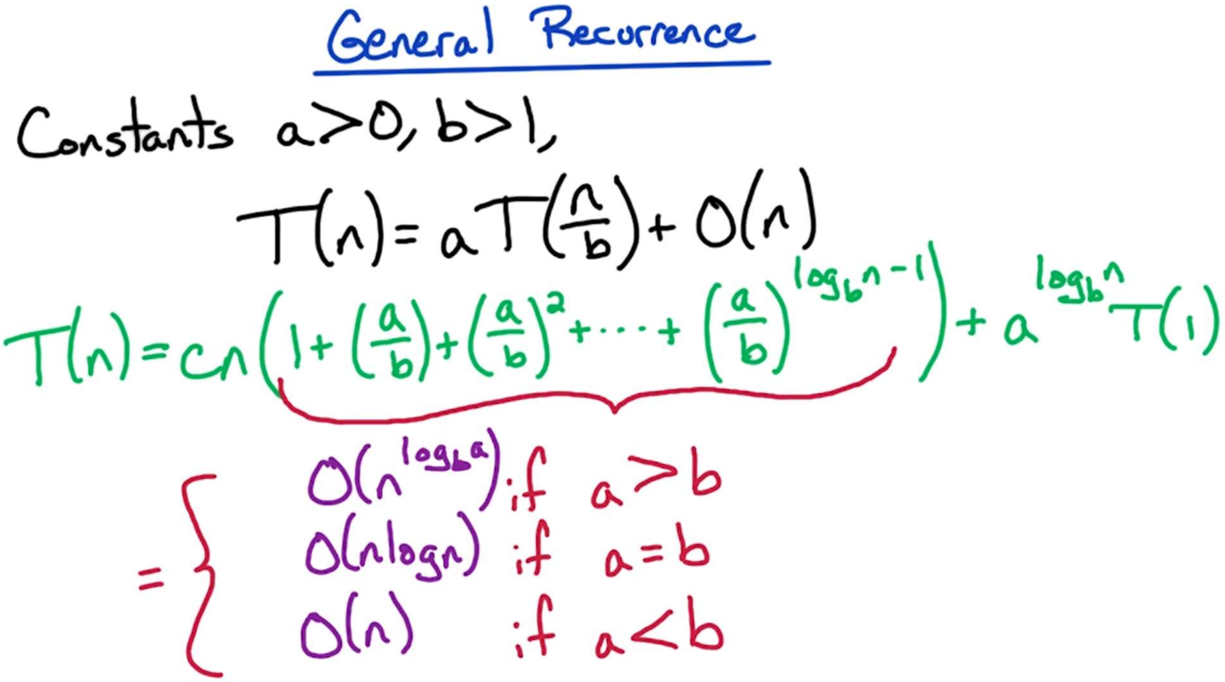
Why is it the base two? Because we have 2^i. So, if we would have had subproblems of size n/3 then we would stop when we have log3 n. Now plugging this in we get the following expression:

T(n) <= cn (1 + (3/2) + (3/2)^2 + … + (3/2)^log2(n)-1) + 3^log2(n) T(1).

T(1) <= c; cn = O(n). What about this geometric series? Well, the base is (3/2). So it's an increasing geometric series, the last term dominates. We can forget about the -1, that's just an extra factor of 2/3rd. So this is 0((3/2)^log2(n)), the numerator is 3^log2(n), the denominator is 2^log2(n). 2^log2(n) is n so that cancels with this n, this O(n) cancels with this one over order n. So we get O(1) x (O(3^log2n) and this term is also O(3^log2 n). That's not a coincidence. This should be the last term of this geometric series, so they should be on the same order of magnitude. Finally we want to manipulate this into a polynomial and we just saw that this is on the O(n^log2(3)).

(Slide 7) General Recurrence

***



General Recurrence

Constants $a>0, b>1$,

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

$$T(n) = cn\Bigg(\underbrace{1 + \left(\frac{a}{b}\right) + \left(\frac{a}{b}\right)^2 + \cdots + \left(\frac{a}{b}\right)^{\log_b n - 1}}\Bigg) + a^{\log_b n}T(1)$$

$$= \begin{cases} O(n^{\log_b a}) & \text{if } a > b \\ O(n\log n) & \text{if } a = b \\ O(n) & \text{if } a < b \end{cases}$$

Now, let's look at the more general form of recurrences. So for positive constants a and b greater than one, Let's look at the recurrence T(n) = a T(n/b) + O(n).

- Our previous two examples had b=2, and we did a case where a=4, and a=3.
- Now, the recurrence for the median finding algorithm has a = 1 and has b = 4/3.
- And mergesort has a = 2 and b = 2.

Now, when you expand this out you get c * n from this O(n) times the following quantity,
T(n) = cn (1 + (a/b) + (a/b)^2 +…+ (a/b)^logb(n)-1) + a^logb(n) T(1)
The series is going to stop when we get the logb(n). The last term is going to be a^logb(n) T(1), which is O(1).

Now, the key is what happens for this geometric series? It depends on whether a/b > 1, a/b = 1, or a/b < 1.

- If a > b, then we get an increasing geometric series and the last term dominates.
- If a = b, then we get a geometric series where i term is one.
- If a < b, then we get a decreasing geometric series and the first term dominates.

Let's do the easy case first, a < b, so the first term dominates. So this geometric series is on the O(1). So we get O(1) for this geometric series, get O(n) here, so the whole thing is going to be O(n), so we get O(n), in this case.

What happens when a = b? Then each of these terms is one. So we get log n terms. So the geometric series is O(log n), so we have O(n) times O(logn). So that's going to be O(n log n). So in the case, a = b, we get O(n log n) - that's for merge sort.

Finally, what happens in the case when a > b? Now, the last term dominates. Notice the denominator here is b ^logb(n) which is n, so this term, cancels this term. So the whole thing is on the order of a^logb(n). Converting that into a polynomial, we get that this recurrence in the case a > b is on the order of n ^logb(a).

And of course, one can look at more general forms of recurrences where here, we have instead of O(n), we have O(n^d), where d is a constant. This is commonly referred to as the master theorem and you can look at your textbook for the form of the master theorem and the solution to the master theorem.