Data Structures and Algorithm Analysis (CSC317)

Divide and conquer (part 3)

Goals

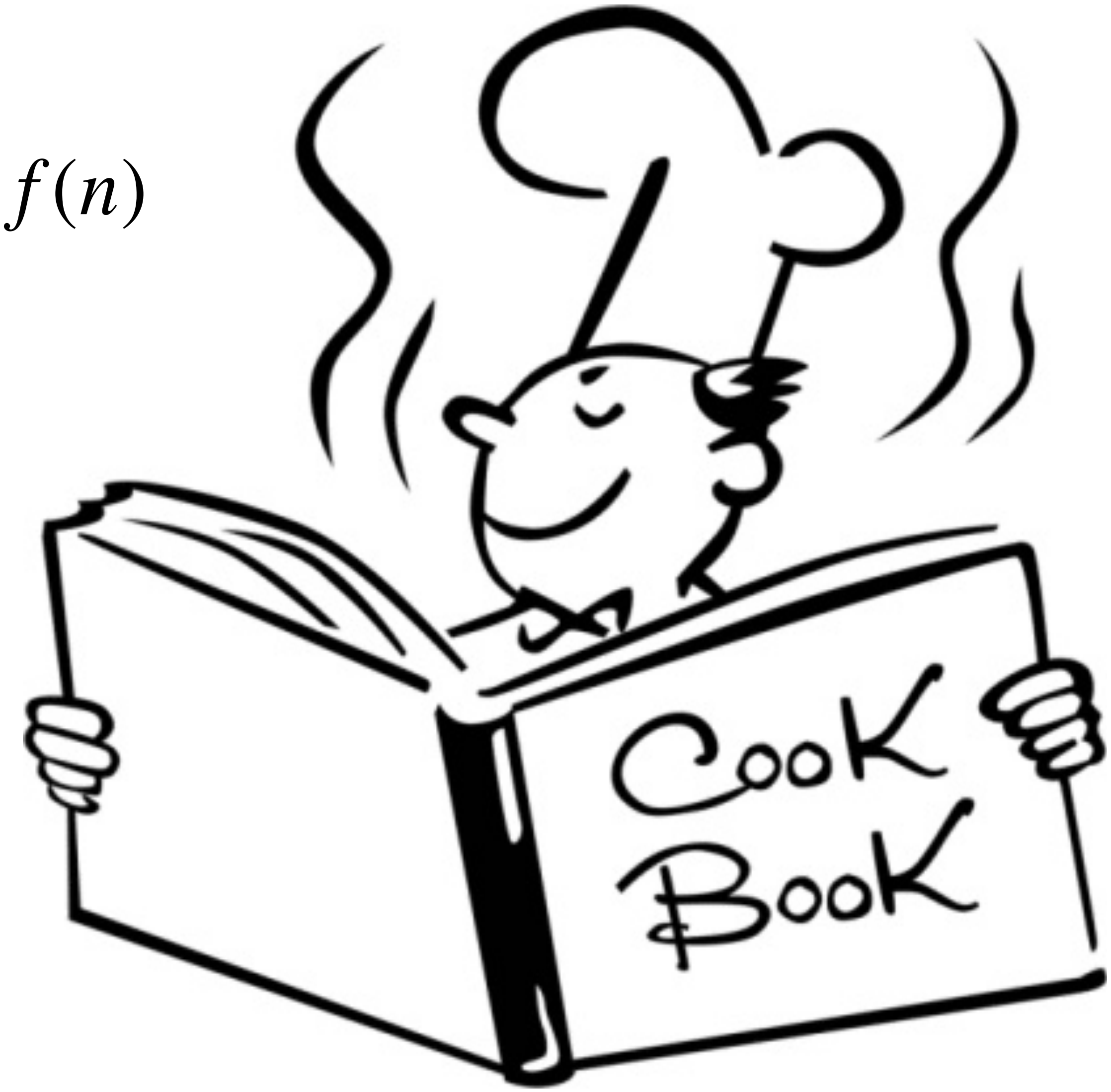
What kind of recurrences arise in algorithms
and how do we solve more generally (than what
we saw for merge sort)?

- More recurrence examples
- Revisit recursion trees more generally
- Master theorem as "recipe" for range of cases
- (Substitution method)

Master method

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

$$a \geq 1; b > 1$$



Master method

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

a subproblems
n/b size of each subproblem
f(n) cost of dividing problem and combining subproblem results

Master method

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

a subproblems
n/b size of each subproblem
f(n) cost of dividing problem and combining subproblem results

Competition between:

a number of recursive calls made – bad

Master method

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

a subproblems
n/b size of each subproblem
f(n) cost of dividing problem and combining subproblem results

Competition between:

a number of recursive calls made – bad
b how much problem size decreased each call – good

Master method

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

a subproblems
n/b size of each subproblem
f(n) cost of dividing problem and combining subproblem results

Competition between:

a number of recursive calls made – bad
b how much problem size decreased each call – good
f(n) determines work outside of recursive call we compare to

Master method

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

Competition between:

a number of recursive calls made – bad
b how much problem size decreased each call – good
f(n) determines work outside of recursive call we compare to

We'll be comparing:

$f(n)$ and $n^{\log_b a}$

Master method

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

Competition between:

a number of recursive calls made – bad
b how much problem size decreased each call – good
f(n) determines work outside of recursive call we compare to

We'll be comparing:

$$f(n) \text{ and } n^{\log_b a} = a^{\log_b n}$$

Intuitively, is there more work at the root or at the leaves? Like what we developed in recursion tree examples…

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let
T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$ then:

$$T(n) = \Theta(n^{\log_b a})$$

If $n^{\log_b a}$ polynomially larger than f(n) -> $n^{\log_b a}$ dominates

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$ then:

$$T(n) = \Theta(n^{\log_b a})$$

If $n^{\log_b a}$ polynomially larger than f(n) -> $n^{\log_b a}$ dominates

Like the leaves dominating in a recursion tree

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

2. If $f(n) = O(n^{\log_b a})$ then:

$$T(n) = \Theta(n^{\log_b a} \log n)$$

If $n^{\log_b a}$ equals f(n) -> $n^{\log_b a} \log n$

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

2. If $f(n) = O(n^{\log_b a})$ then:

$$T(n) = \Theta(n^{\log_b a} \log n)$$

If $n^{\log_b a}$ equals f(n) -> $n^{\log_b a} \log n$

Like merge sort - equal work each level

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ and regularity condition (ignore for now):

$$T(n) = \Theta(f(n))$$

If $n^{\log_b a}$ polynomially smaller than f(n) -> f(n) dominates

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

3. If $f(n) = \Omega(n^{\log_b a+\varepsilon})$ for some constant $\varepsilon > 0$ and regularity condition (ignore for now)

$$T(n) = \Theta(f(n))$$

If $n^{\log_b a}$ polynomially smaller than f(n) -> f(n) dominates

Like the root dominating in a recursion tree

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

3. … regularity condition:

$$af\left(\frac{n}{b}\right) \leq cf(n)$$

This regularity condition will hold in most examples we look at; it's intuitively saying the root will indeed dominate the work

Master theorem

Let a>=1 and b>1 be constants, f(n) a function, and let T(n) be defined on the nonnegative integers by the recurrence

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

Then T(n) has the following asymptotic bounds:

3. If $f(n) = \Omega(n^{\log_b a+\varepsilon})$ for some constant $\varepsilon > 0$ and $af\left(\frac{n}{b}\right) \leq cf(n)$, for some constant c<1, then:

$$T(n) = \Theta(f(n))$$

If $n^{\log_b a}$ polynomially smaller than f(n) -> f(n) dominates

Like the root dominating in a recursion tree

Master theorem summary – 3 cases

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

1 If $n^{\log_b a}$ > f(n) -> $n^{\log_b a}$ dominates

Like the leaves dominating in a recursion tree

2 If $n^{\log_b a}$ equals f(n) -> $n^{\log_b a} \log n$

Like merge sort - equal work each level

3 If $n^{\log_b a}$ < f(n) -> f(n) dominates

Like the root dominating in a recursion tree

Master theorem:

So in all cases we compare $n^{\log_b a}$ to f(n)
and look if they are equal or for polynomial differences

Intuition: Either the leaves and recursion process dominate the cost, or the root dominates the cost, or they are balanced

Proof: we won't show; but relies on recursion trees and geometric sums, similar to example cases we looked at

Master theorem: examples

On the board… we'll remember that:

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

1. If $n^{\log_b a}$ > f(n) -> $T(n) = \Theta(n^{\log_b a})$

2. If $n^{\log_b a}$ equals f(n) -> $T(n) = \Theta(n^{\log_b a} \log n)$

3. If $n^{\log_b a}$ < f(n) -> $T(n) = \Theta(f(n))$

Master theorem: example 1

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

Master theorem: example 1

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 8; b = 2; f(n) = \Theta(n^2)$$

Familiar??

Master theorem: example 1

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 8; b = 2; f(n) = \Theta(n^2)$$

Familiar??

Our first divide and conquer matrix multiplication

Master theorem: example 1

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 8; b = 2; f(n) = \Theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 8} = n^3$$

Which case?

Master theorem: example 1

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 8; b = 2; f(n) = \Theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 8} = n^3$$

Polynomially larger than $f(n) = n^2$ Case 1

Master theorem: example 1

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 8; b = 2; f(n) = \Theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 8} = n^3$$

Polynomially larger than $f(n) = n^2$ Case 1

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^3)$$

Master theorem: example 2

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

$$T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$$

Master theorem: example 2

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

$$T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 7; b = 2; f(n) = \Theta(n^2)$$

Familiar?? Strassen's method!

Master theorem: example 2

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 8; b = 2; f(n) = \Theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 7}$$

What case is this?

Master theorem: example 2

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

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

$$a = 8; b = 2; f(n) = \Theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 7}$$ Case 1

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.8})$$

Master theorem: example 3

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

$$T(n) = T(\frac{n}{3}) + 1$$

Master theorem: example 3

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

$$T(n) = T(\frac{n}{3}) + 1$$

$$a = 1; b = 3; f(n) = 1$$

$$n^{\log_b a} = n^{\log_3 1} = n^0 = 1$$

Master theorem: example 3

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

$$T(n) = T(\frac{n}{3}) + 1$$

$$a = 1; b = 3; f(n) = 1$$

$$n^{\log_b a} = n^{\log_3 1} = n^0 = 1$$

Equal to f(n)=1

Case 2

Master theorem: example 3

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

$$T(n) = T(\frac{n}{3}) + 1$$

$$a = 1; b = 3; f(n) = 1$$

$$n^{\log_b a} = n^{\log_3 1} = n^0 = 1$$

Equal to f(n)=1

Case 2

$$T(n) = \Theta(f(n)\log n) = \Theta(1\log n) = \Theta(\log n)$$

Master theorem: example 4

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

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

Master theorem: example 4

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

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

$$a = 3; b = 4; f(n) = n\log n$$

$$n^{\log_b a} = n^{\log_4 3} = \boxed{n^{0.793}}$$

Master theorem: example 4

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

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

$$a = 3; b = 4; f(n) = n\log n$$

$$n^{\log_b a} = n^{\log_4 3} = n^{0.793}$$

polynomially smaller than f(n) Case 3

$$T(n) = \Theta(f(n)) = \Theta(n\log n)$$

Master theorem: example 5

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

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

$$a = 3; b = 4; f(n) = n\log n$$

$$n^{\log_b a} = n^{\log_4 3} = n^{0.793}$$

polynomially smaller than f(n) Case 3

$$T(n) = \Theta(f(n)) = \Theta(n\log n)$$

Note: need to verify regularity condition holds

Master theorem: example 5

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

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

Master theorem: example 5

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

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

$$a = 2; b = 2; f(n) = n\log n$$

$$n^{\log_b a} = n^{\log_2 2} = n^1 = n$$

Master theorem: example 5

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

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

$$a = 2; b = 2; f(n) = n\log n$$

$$n^{\log_b a} = n^{\log_2 2} = n^1 = n$$

smaller than $f(n) = n\log n$ Case 3?

Master theorem: example 5

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

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

$$a = 2; b = 2; f(n) = n \log n$$

$$n^{\log_b a} = n^{\log_2 2} = n^1 = n$$

smaller than $f(n) = n \log n$ Case 3?

No, not polynomially smaller

$$\frac{n \log n}{n} = \log n < n^{\varepsilon}$$

One more recursion tree

Compare: (Like Merge Sort)

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

To: (Like an uneven split Merge Sort)

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

Better? Worse? Equal?

One more recursion tree

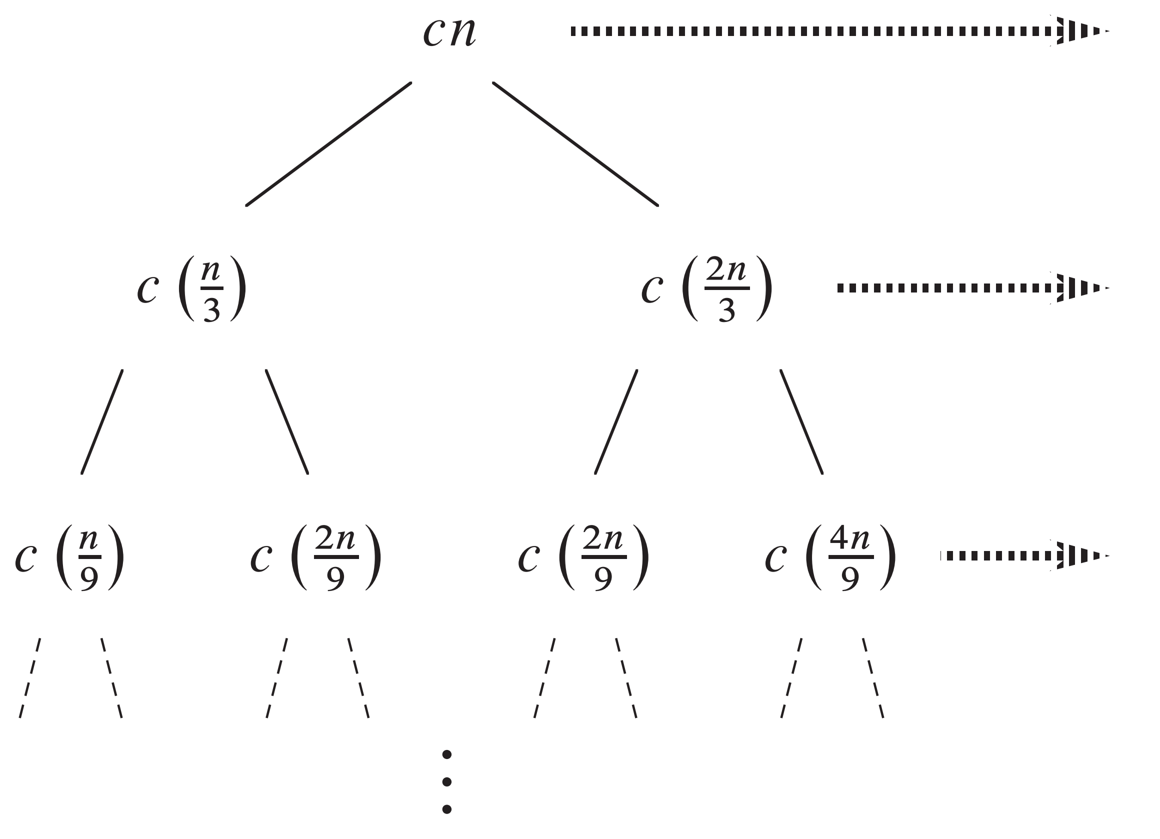
$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

$$cn$$

$$c\left(\frac{n}{3}\right) \qquad c\left(\frac{2n}{3}\right)$$

One more recursion tree

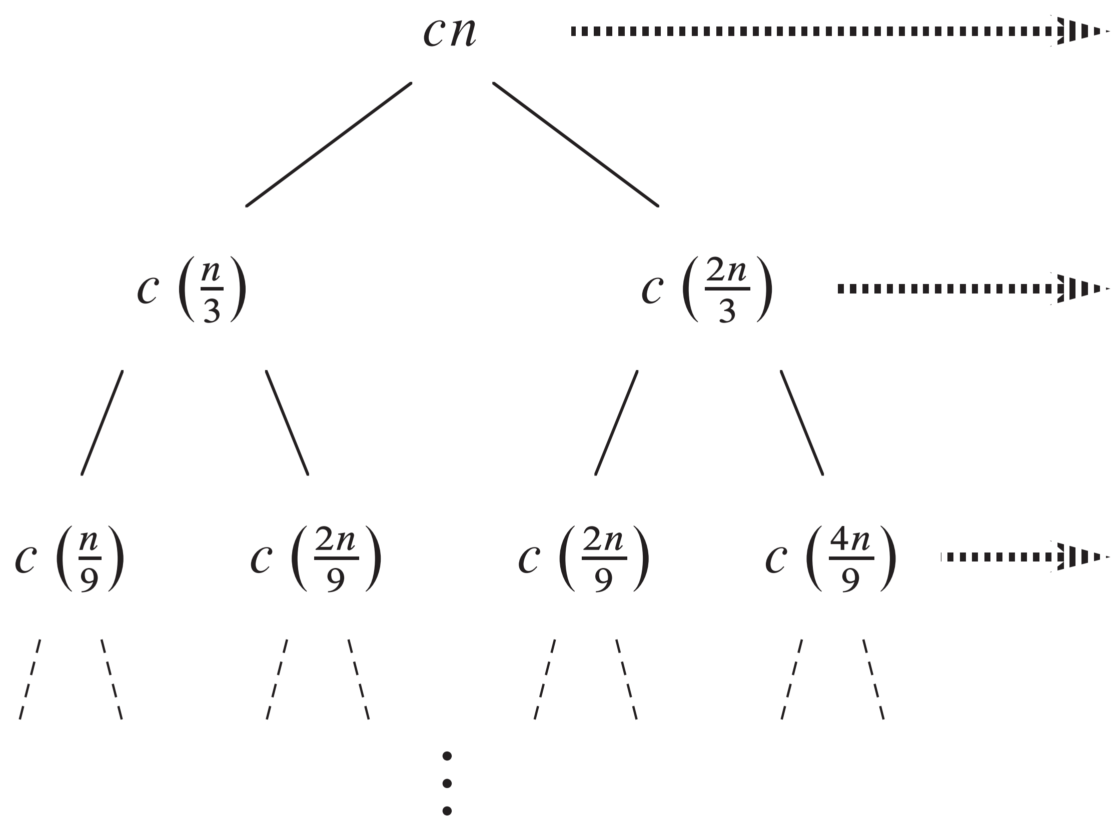
$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$



One more recursion tree

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

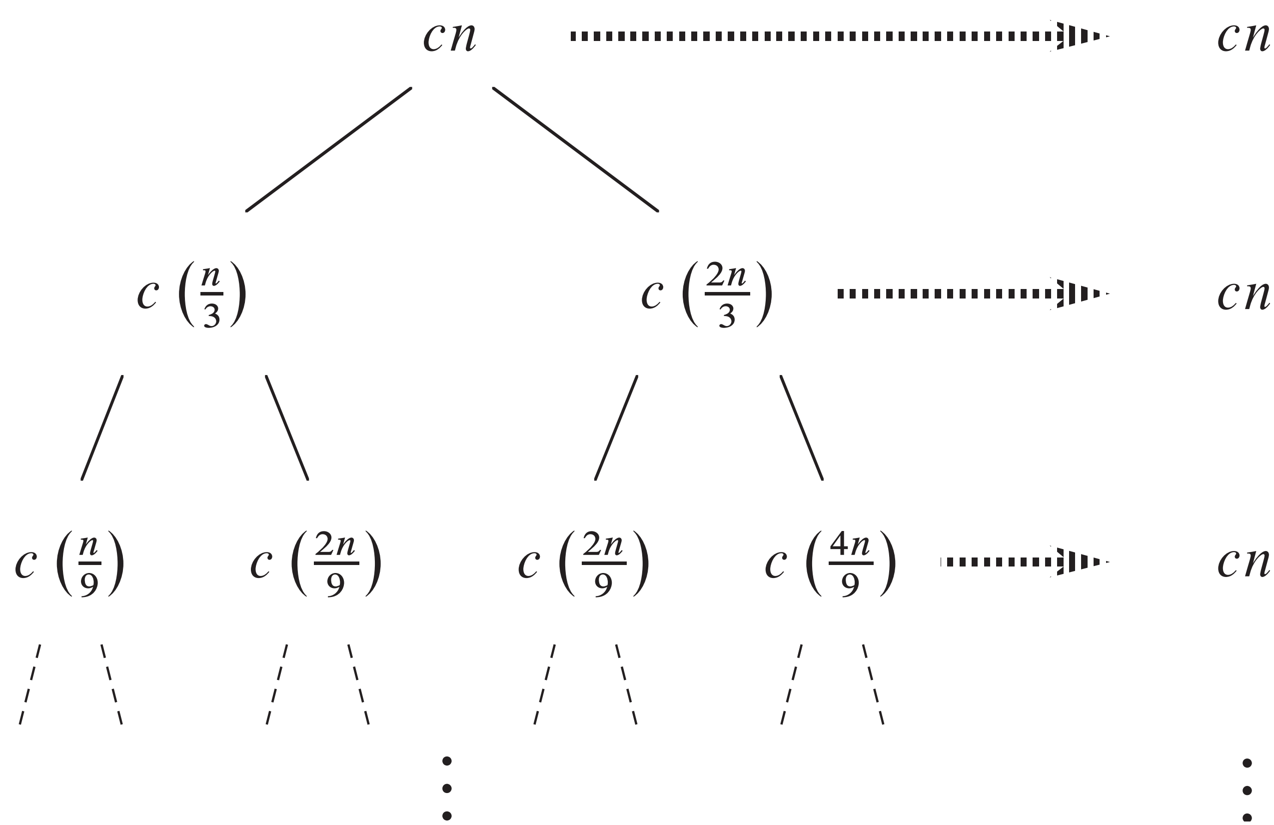
Work each level?



One more recursion tree

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

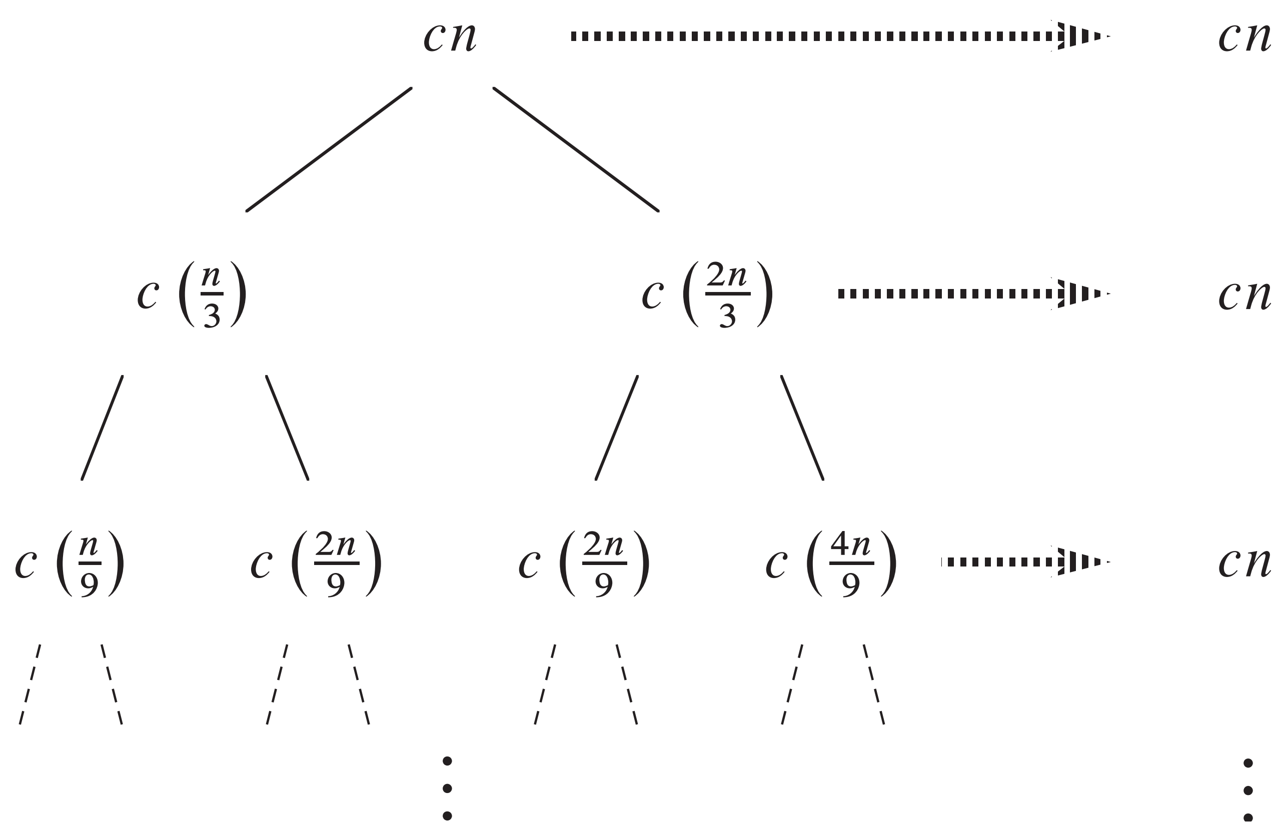
Work each level?



One more recursion tree

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

Height?



One more recursion tree

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

Height?

Longest path root to leaf:

$$n$$

$$\frac{2}{3}n$$

$$\left(\frac{2}{3}\right)^2 n$$

$$\dots$$

$$1$$

One more recursion tree

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

Height?

At the leaf:

$$\left(\frac{2}{3}\right)^k n = 1$$

One more recursion tree

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

Height?

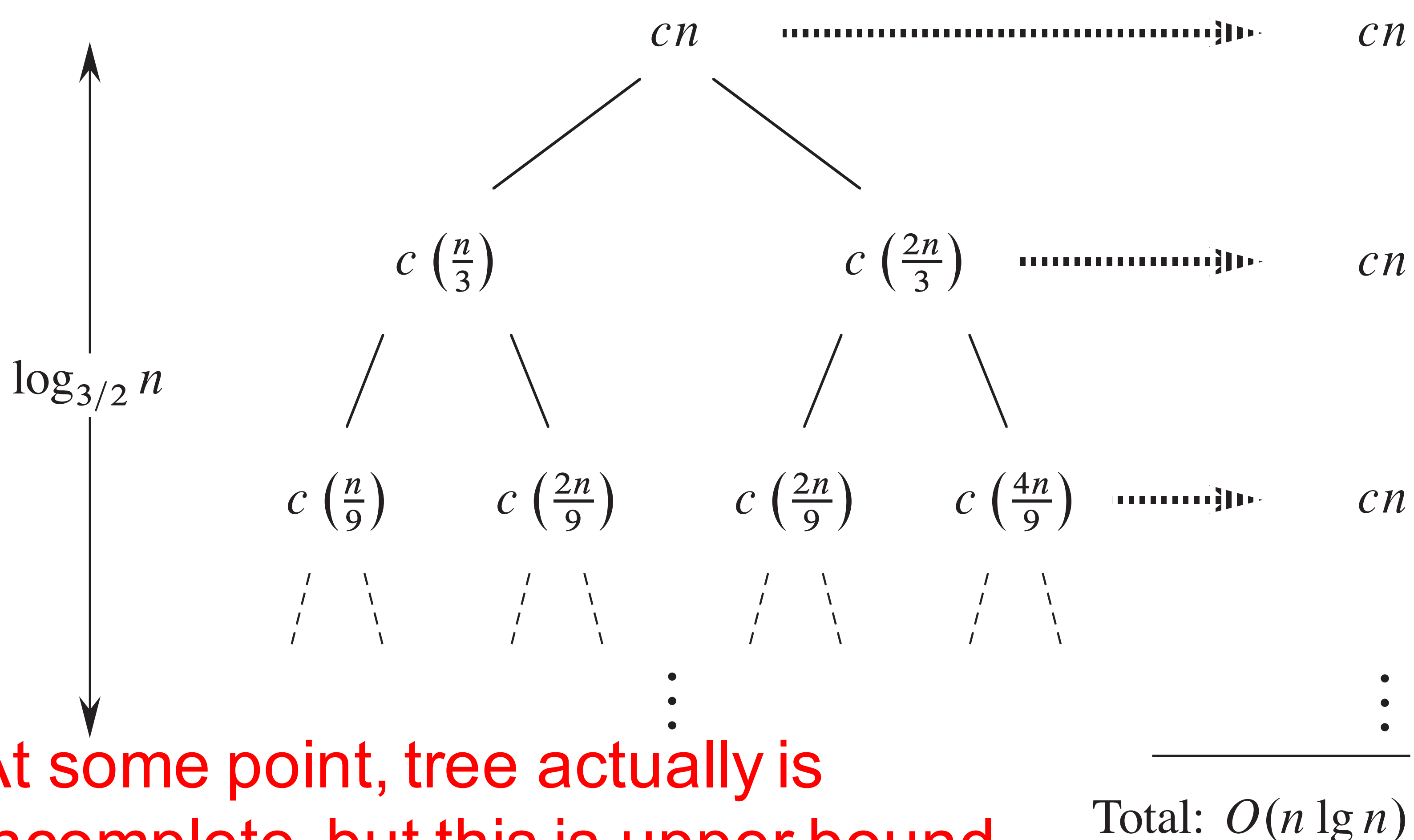
At the leaf: $\left(\frac{2}{3}\right)^k n = 1;$

$$n = \left(\frac{3}{2}\right)^k;$$

$$k = \log_{\frac{3}{2}} n$$

One more recursion tree

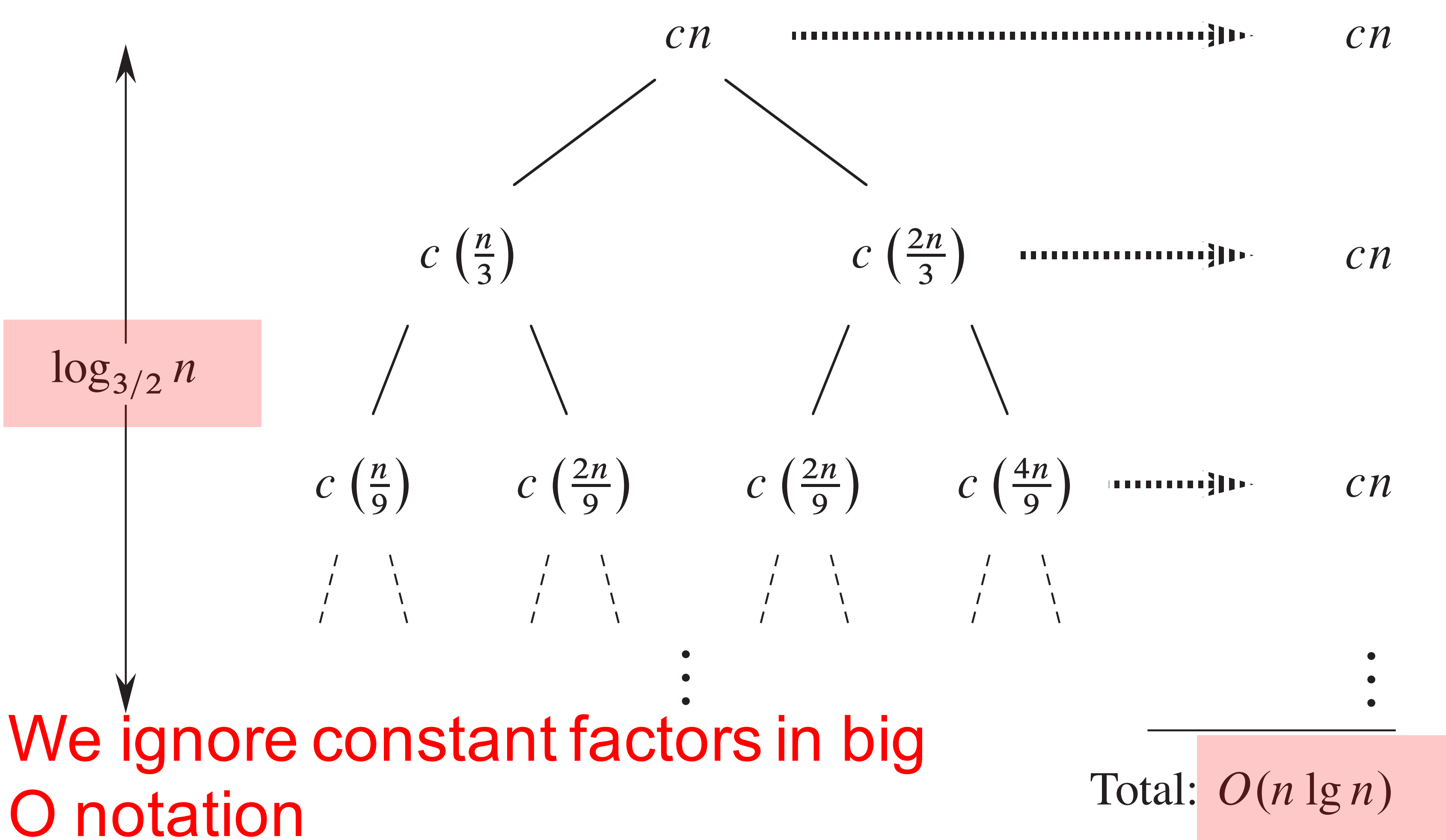
$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$



At some point, tree actually is
Incomplete, but this is upper bound

One more recursion tree

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$



We ignore constant factors in big O notation

One more recursion tree

Also note, we ignored base of algorithm, since constant factor:

$$\log_b a = \frac{\log_c a}{\log_c b}$$

One more recursion tree

Compare: (Like Merge Sort)

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

To: (Like an uneven split Merge Sort)

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$$

Better? Worse? Equal?

Asymptotically, similar

Goals

What kind of recurrences arise in algorithms and how do we solve more generally (than what we saw for merge sort)?

- More recurrence examples
- Revisit recursion trees more generally
- Master theorem as “recipe” for range of cases
- Substitution method

Substitution method

- Guess a bound

Substitution method

- Guess a bound (we need a guess!!)

Substitution method

- Guess a bound (we need a guess!!)
- Prove correct by induction
- Find constants in this process

Example

Prove that $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$ is

$O(n \log n)$

Example

Prove that $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$ is $O(n \log n)$

We need to prove that $T(n) \le cn \log n$

For appropriate choice of constant c>0

(can't use big Oh in substitution because of induction, need to write out definition with constants!)

Example

Induction step: assume

$$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \le c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le$$

Example

Induction step: assume

$$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \le c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

Example

Induction step: assume

$$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \le c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

…we want $T(n) \le cn\log(n)$

Example

Induction step: assume

$$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \le c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le$$

$$cn\log\left(\frac{n}{2}\right) + n =$$

…we want $T(n) \le cn\log(n)$

Example

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le$$

$$cn\log\left(\frac{n}{2}\right) + n =$$

$$cn\log n - cn\log 2 + n$$

…we want $T(n) \le cn\log(n)$

Example

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \leq \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \leq$$

$$cn\log\left(\frac{n}{2}\right) + n =$$

$$cn\log n - cn\log 2 + n$$

…we want $T(n) \leq cn\log(n)$

Example

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \leq \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \leq$$

$$cn\log\left(\frac{n}{2}\right) + n =$$

$$cn\log n - cn\log 2 + n =$$

$$\boxed{cn\log n} - cn + n$$

…we want $T(n) \leq cn\log(n)$

Example

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le$$

$$cn\log\left(\frac{n}{2}\right) + n =$$

$$cn\log n - cn\log 2 + n =$$

$$cn\log n - cn + n$$

$$\le cn\log n$$

When?

…we want $T(n) \le cn\log(n)$

Example

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \leq \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \leq$$

$$cn\log\left(\frac{n}{2}\right) + n =$$

$$cn\log n - cn\log 2 + n =$$

$$cn\log n - cn + n$$

$$\leq cn\log n$$

Holds for:

$$-cn + n \leq 0;$$

$$n \leq cn$$

$$c \geq 1$$

…we want $T(n) \leq cn\log(n)$

Example

Induction step: assume

$$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \le c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$

Then

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le \quad 2c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n =$$

$$cn\log n - cn\log 2 + n =$$

$$cn\log n - cn + n \le cn\log n$$

For c>=1

Example

Induction needs base condition. For n=1, assume:

$$T(1) = 1$$

Then:

$$T(1) \le c1\log 1 \qquad \text{(?)}$$

Example

Induction needs base condition. For n=1, assume:

$$T(1) = 1$$

Then:

$$T(1) \le c1\log 1 = c\log 1 \quad (?)$$

$$1 = T(1) \le c1\log 1 = 0 \quad \text{no}$$

Example

Induction needs base condition. For n=1, assume:

$$T(1) = 1$$

Then:

$$T(1) \leq c1\log 1 = c\log 1 \quad (?)$$

$$1 = T(1) \leq c1\log 1 = 0 \quad \text{no}$$

Asymptotic notation requires only for n>=no

$$T(n) \leq cn\log n$$

Example

Induction needs base condition.

$$T(n) \leq cn \log n$$

Asymptotic notation requires only for n>=no

Let's try n=3, so as not to depend directly on T(1):

$$T(1) = 1$$

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

$$T(3) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n = 2 + 3 = 5$$

Example

Induction needs base condition.

$$T(n) \le cn\log n$$

Asymptotic notation requires only for n>=no

Let's try n=3:

$$T(1) = 1$$

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

$$T(3) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n = 2 + 3 = 5$$

$$T(3) = 5 \le cn\log n = c3\log 3 = 3c(1.58)$$

Holds for c>=2

Example

We've shown for $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$

$$T(n) \le cn \log n$$

Asymptotic notation requires only for n>=3 c>=2

(both induction step and base case)

Change of variable

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

Change of variable

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

Define:

$$m = \log n \longrightarrow n = 2^m$$

Example 2

Change of variable

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

Define:

$$m = \log n \longrightarrow n = 2^m$$

$$\boxed{T(n) = T(2^m) = 2T(2^{\frac{m}{2}}) + m}$$

Familiar pattern?

Change of variable

$$T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

$$S(m) = 2S(\frac{m}{2}) + m$$

Change of variable

$$T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

$$S(m) = 2S(\frac{m}{2}) + m$$

Like what?

Change of variable

$$T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

$$S(m) = 2S(\frac{m}{2}) + m$$

Like what? Merge Sort

Change of variable

$$T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

$$S(m) = 2S(\frac{m}{2}) + m$$

Like what? Merge Sort

$$O(m \log m)$$

Change of variable

$$T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

$$S(m) = 2S(\frac{m}{2}) + m$$

Like what? Merge Sort

$$O(m \log m) = O(\log n \log(\log n))$$

Change variable back

Goals

Solving recurrences

- Revisit recursion trees more generally
- Master theorem as “recipe” for range of cases
- Substitution method

PROS / CONS?