CS4311
Design and Analysis of Algorithms

Lecture 3: Recurrences

About this lecture

- Introduce some ways of solving recurrences
  - Substitution Method (If we know the answer)
  - Recursion Tree Method (Very useful !)
  - Master Theorem (Save our effort)

Substitution Method

(if we know the answer)

How to solve this?

$$T(n) = 2T(\lfloor n/2 \rfloor) + n, \quad \text{with } T(1) = 1$$

1. Make a guess

   E.g., $T(n) = O(n \log n)$

2. Show it by induction

   - E.g., to show upper bound, we find constants c and n_0 such that $T(n) \leq c\, f(n)$ for $n = n_0$, n_0+1, n_0+2, …

Substitution Method

(if we know the answer)

How to solve this?

$$T(n) = 2T(\lfloor n/2 \rfloor) + n, \quad \text{with } T(1) = 1$$

1. Make a guess ($T(n) = O(n \log n)$)
2. Show it by induction
   - Firstly, $T(2) = 4$, $T(3) = 5$.

     ➔ We want to have $T(n) \leq cn \log n$

     ➔ Let $c = 2$ ➔ $T(2)$ and $T(3)$ okay
   - Other Cases ?

Substitution Method

(if we know the answer)

- Induction Case:

Assume the guess is true for all n = 2,3,…,k

For n = k+1, we have:

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

$$\leq 2c\lfloor n/2 \rfloor \log \lfloor n/2 \rfloor + n$$

$$\leq cn \log (n/2) + n$$

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

Induction case is true

Substitution Method

(if we know the answer)

Q. How did we know the value of c and n_0 ?

A. If induction works, the induction case must be correct → $c \geq 1$

Then, we find that by setting $c = 2$, our guess is correct as soon as $n_0 = 2$

Alternatively, we can also use $c = 1.3$
Then, we just need a larger $n_0 = 4$

(What will be the new base cases? Why?)

Substitution Method

(New Challenge)

How to solve this?

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1, \quad T(1) = 1$$

1. Make a guess ($T(n) = O(n)$), and
2. Show $T(n) \leq cn$ by induction
   - What will happen in induction case?

Substitution Method
(New Challenge)

Induction Case:

(assume guess is true for some base cases)

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

$$\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1$$

$$= cn + 1$$

This term is not what we want …

Substitution Method
(New Challenge)

- The 1st attempt was not working because our guess for $T(n)$ was a bit "loose"

Recall: Induction may become easier if we prove a "stronger" statement

2nd Attempt: Refine our statement

Try to show $T(n) \leq cn - b$ instead

Substitution Method

(New Challenge)

Induction Case:

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

$$\leq c\lfloor n/2 \rfloor - b + c\lceil n/2 \rceil - b + 1$$

$$\leq cn - b$$

We get the desired term (when $b \geq 1$)

It remains to find c and n_0, and prove the base case(s), which is relatively easy

Substitution Method
(New Challenge 2)

How to solve this?

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

Hint: Change variable: Set $m = \log n$

Substitution Method
(New Challenge 2)

Set $m = \log n$, we get

$$T(2^m) = 2T(2^{m/2}) + m$$

Next, set $S(m) = T(2^m) = T(n)$

$$S(m) = 2S(m/2) + m$$

We solve $S(m) = O(m \log m)$

➔ $$T(n) = O(\log n \log \log n)$$

Recursion Tree Method

(Nothing Special… Very Useful !)

How to solve this?

$$T(n) = 2T(n/2) + n^2, \qquad \text{with } T(1) = 1$$

Recursion Tree Method

(Nothing Special… Very Useful !)

Expanding the terms, we get:

$$T(n) = n^2 + 2T(n/2)$$
$$= n^2 + 2n^2/4 + 4T(n/4)$$
$$= n^2 + 2n^2/4 + 4n^2/16 + 8T(n/8)$$
$$= \ldots$$
$$= \sum_{k=0 \text{ to } \log n-1} (1/2)^k n^2 + 2^{\log n}\, T(1)$$
$$= \Theta(n^2) + \Theta(n) = \Theta(n^2)$$

Recursion Tree Method

(Recursion Tree View)

We can express the previous recurrence by:

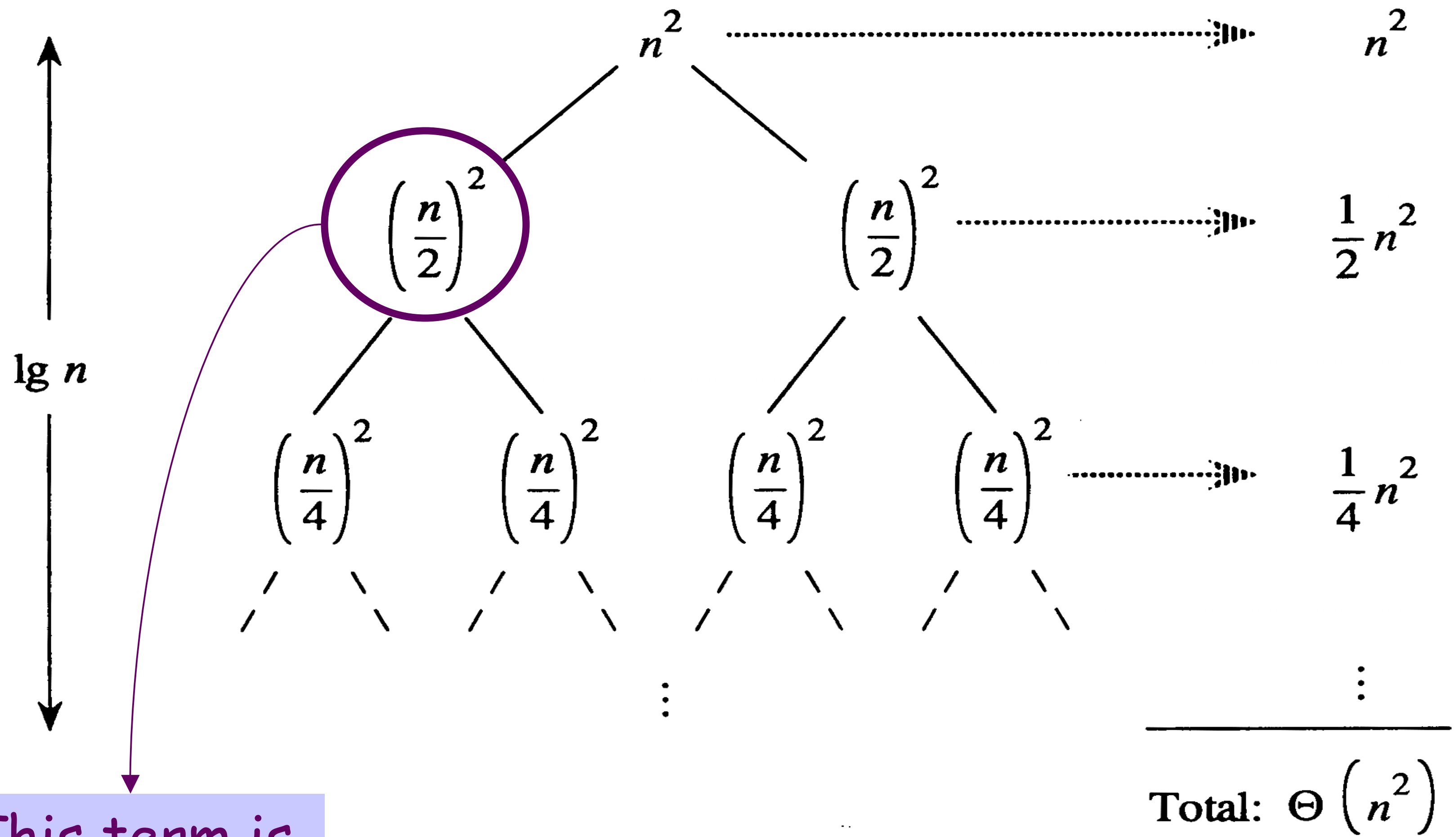
$$T(n)$$

(a)

$$\begin{array}{ccc} & n^2 & \\ \diagup & & \diagdown \\ T\left(\frac{n}{2}\right) & & T\left(\frac{n}{2}\right) \end{array}$$

(b)

Further expressing gives us:



(d)

This term is from T(n/2)

Recursion Tree Method

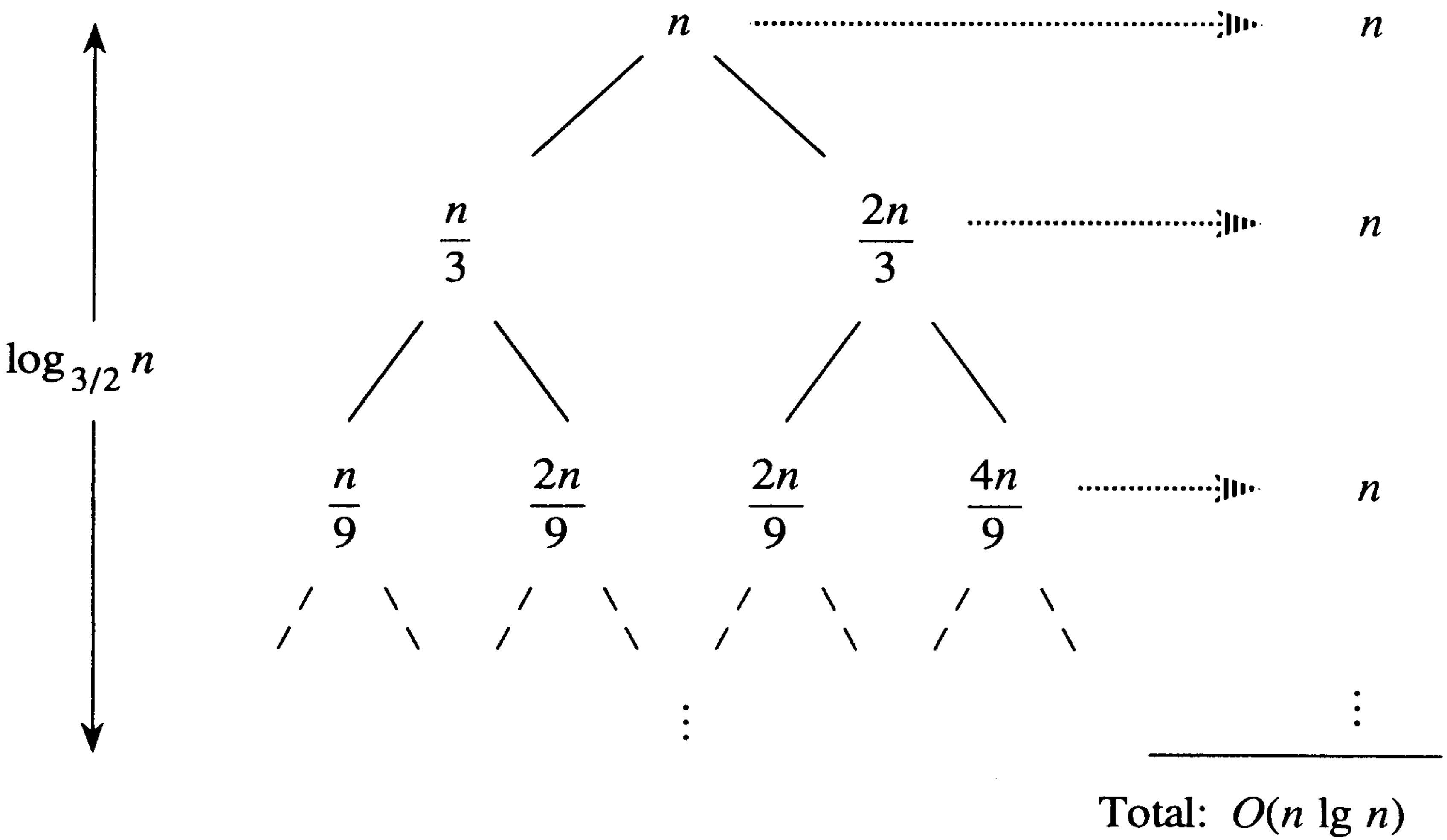
(New Challenge)

How to solve this?

$$T(n) = T(n/3) + T(2n/3) + n, \quad \text{with } T(1) = 1$$

What will be the recursion tree view?

The corresponding recursion tree view is:



Master Method

(Save our effort)

When the recurrence is in a special form, we can apply the Master Theorem to solve the recurrence immediately

The Master Theorem has 3 cases …

Master Theorem

Let $T(n) = aT(n/b) + f(n)$

with $a \geq 1$ and $b > 1$ are constants.

Theorem: (Case 1: Very Small $f(n)$)

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

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

Theorem: (Case 2: Moderate $f(n)$)

If $f(n) = \Theta(n^{\log_b a})$,

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

Theorem: (Case 3: Very large $f(n)$)

If (i) $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$

and (ii) $a\, f(n/b) \le c\, f(n)$ for some constant $c < 1$, all sufficiently large n

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

Master Theorem (Exercises)

1. Solve $T(n) = 9T(n/3) + n$

2. Solve $T(n) = 9T(n/3) + n^2$

3. Solve $T(n) = 9T(n/3) + n^3$

4. How about this?

   $T(n) = 9T(n/3) + n^2 \log n$?