Introduction to Algorithms

Mong-Jen Kao (高孟駿)

Tuesday 10:10 – 12:00

Thursday 15:30 – 16:20

Solving Recurrence Formulas

Recurrence Formula

- A ***recurrence*** is an equation or inequality that describes a function in terms of its value on smaller inputs.

- We have already seen some of such examples.
  - For example, the time complexity of the Merge Sort algorithm can be described by the following recurrence.

$$T(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 2T\left(\frac{n}{2}\right) + \Theta(n), & \text{if } n > 1. \end{cases}$$

Recurrence Formula

- A ***recurrence*** is an equation or inequality that describes a function in terms of its value on smaller inputs.
- In this lecture, we will go through *three different methods* for ***obtaining asymptotic bounds*** on the solution.
  - The substitution method
  - The recursion-tree method
  - The master theorem

The Substitution Method

Make an *educated guess* and verify it.

The Substitution Method

- The substitution method for solving recurrences entails two steps.

  1. ***Guess*** the form of the solution.

  2. Use ***mathematical induction*** to ***find the constant*** and ***show*** that the solution works.

The Substitution Method

- As an example,
  consider the following recurrence

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

- We ***guess*** the solution is $T(n) = O(n \log n)$.
  - Then, we need to prove that for all $n \geq n_0$

$$T(n) \leq c \cdot n \log n$$

  for an *appropriate choice* of the constant $c > 0$ and n_0.

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

- We ***guess*** the solution is $T(n) = O(n \log n)$.
  - Then, we need to prove that for all $n \geq n_0$,

$$T(n) \leq c \cdot n \log n$$

  for an *appropriate choice* of the constant $c > 0$ and n_0.

- Assume as *in the inductive step* that the bound holds for $\lfloor n/2 \rfloor$.
  - That is,

$$T(\lfloor n/2 \rfloor) \leq c \cdot \lfloor n/2 \rfloor \cdot \log \lfloor n/2 \rfloor \ .$$

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

$$T(n) \leq c \cdot n \log n$$

for appropriate $c > 0$, $n_0 > 0$, and all $n \geq n_0$.

- Plug in the assumption into the recurrence and we obtain

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

$$\leq c \cdot n \cdot \log(n/2) + n$$

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

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

holds as long as $c \geq 1$.

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

$T(n) \le c \cdot n \log n$

for appropriate $c > 0$, $n_0 > 0$,
and all $n \ge n_0$.

- Plug in the assumption into the recurrence and we obtain

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

  - ***Hence, the inductive step holds***.

holds as long as $c \ge 1$.

- Next, we need to use ***boundary (initial) conditions*** to ***determine*** an *appropriate constant* c.

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

$T(n) \le c \cdot n \log n$

for appropriate $c > 0$, $n_0 > 0$,
and all $n \ge n_0$.

holds as long as $c \ge 1$.

- ***Hence, the inductive step holds***.

- Next, we need to use ***boundary (initial) conditions*** to ***determine*** an *appropriate constant* c.
  - We have $T(1) \le c \cdot 1 \log 1 = 0$ but $T(1) = 1$.
  - Hence, $n = 1$ is not consistent with our guess.
  - We need to see if a larger n_0 can be used.

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

$T(n) \le c \cdot n \log n$

for appropriate $c > 0$, $n_0 > 0$, and all $n \ge n_0$.

holds as long as $c \ge 1$.

- ***Hence, the inductive step holds***.

- Next, we need to use ***boundary (initial) conditions*** to ***determine*** an *appropriate constant* c.
  - For $n_0 = 2$ and $c = 2$, we have

$$4 = T(2) \le c \cdot 2 \log 2 \quad \text{and}$$
$$5 = T(3) \le c \cdot 3 \log 3 .$$

  - Hence, $n_0 = 2$ and $c = 2$ completes our guess for $T(n) = O(n \log n)$.

Making a Good Guess

- Unfortunately, there is no general way to guess the correct solutions for recurrences.

- Fortunately, there are some heuristic ways to do so.
  - Use the recursion-tree method (described next) to come up with a good *asymptotic guess*.

  - Make a good guess from similar recurrences.

  - Try & Refine the guess.

Observe from Similar Recurrences

- It is reasonable to make a similar guess from a similar recurrence you have seen before.
  - For example,

    $$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$$

    looks more difficult because of the additive term of 17.
  - However, the constant term 17 is negligible compared to $n/2$ when n is large enough.
  - Hence, $T(n) = O(n \log n)$ is still a good guess and will work.

Try & Refine the Guess

- One typical way is first to prove a loose bound and then refine the range of uncertainty.
  - For example, we can start with $T(n) = \Omega(n)$, $T(n) = O(n^2)$, and eventually obtain $T(n) = \Theta(n \log n)$.
  - When the asymptotic behavior you guess is wrong, it will lead to a contradiction in the inductive step.

Some Subtleties

- Sometimes you make a correct guess,
  but the inductive step ***fails due to smaller-order terms***.
  - For example, consider the recurrence

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

  - We guess $T(n) = O(n)$ and try to prove that $T(n) \le cn$.
    Then

$$T(n) \le c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 = cn + 1 > cn.$$

But it fails because of an $O(1) = o(n)$ term.

The asymptotic growth rate is correct.

Some Subtleties

- Sometimes you make a correct guess,
  but the inductive step ***fails due to smaller-order terms***.

  – One solution for this is to make a *slightly stronger* guess,
  so that ***less smaller-order error*** *is accumulated* during the inductive step.

  – To be precise, we guess $T(n) \le cn - b$.
  Then

$$T(n) \ \le \ c\lfloor n/2 \rfloor + c\lceil n/2 \rceil - 2b + 1 \ \le \ cn - b.$$

holds for any $b \ge 1$.

Avoiding Pitfalls

- When proving the inductive step, it is easy to err in the usage of asymptotic notations.
  - For example, we guess $\boldsymbol{T(n) \leq cn}$ and write

$$T(n) \leq 2c\lfloor n/2 \rfloor + n \leq cn + n = \boldsymbol{O(n)}.$$

$\nleq cn.$

  - Note that, this does not finish the proof for the inductive step!

Changing Variables

- Sometimes, a little algebraic manipulation can make an unknown recurrence similar to one you have seen before.
  - For example, consider the recurrence

$$T(n) = 2T\left(\lfloor\sqrt{n}\rfloor\right) + \log n .$$

  - By setting $m = \log n$, we obtain

$$T(2^m) = 2T\left(2^{m/2}\right) + m .$$

  - With $S(m) \coloneqq T(2^m)$,

    we get $S(m) = 2S(m/2) + m$, which we know how to solve.

$S(m) = O(m \log m)$.

$T(n) = T(2^m)$

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

The Recursion-Tree Method

Expand the recursion explicitly to see the result.

Recursion-Tree Method

- Sometimes it is difficult to come up with a good guess for the recurrence we are facing.

- Drawing out the recursion explicitly is a straightforward way to devise a good guess for the recurrence.

- In the recursion-tree method, we
  - expand the recurrence explicitly from the root, and
  - sum up the cost incurred at each level.

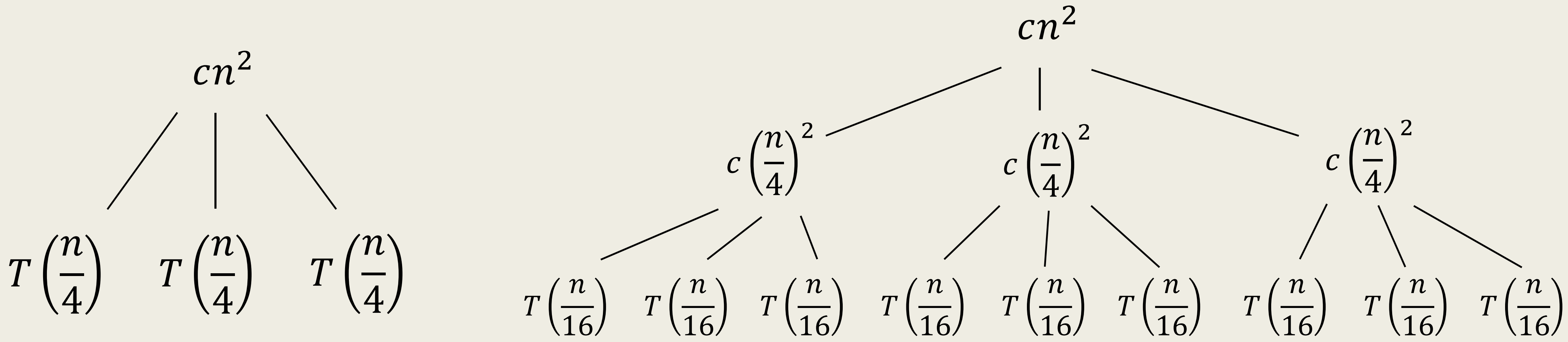
Recursion-Tree Method

- Let us consider the recurrence

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2),$$

which can be rewritten as

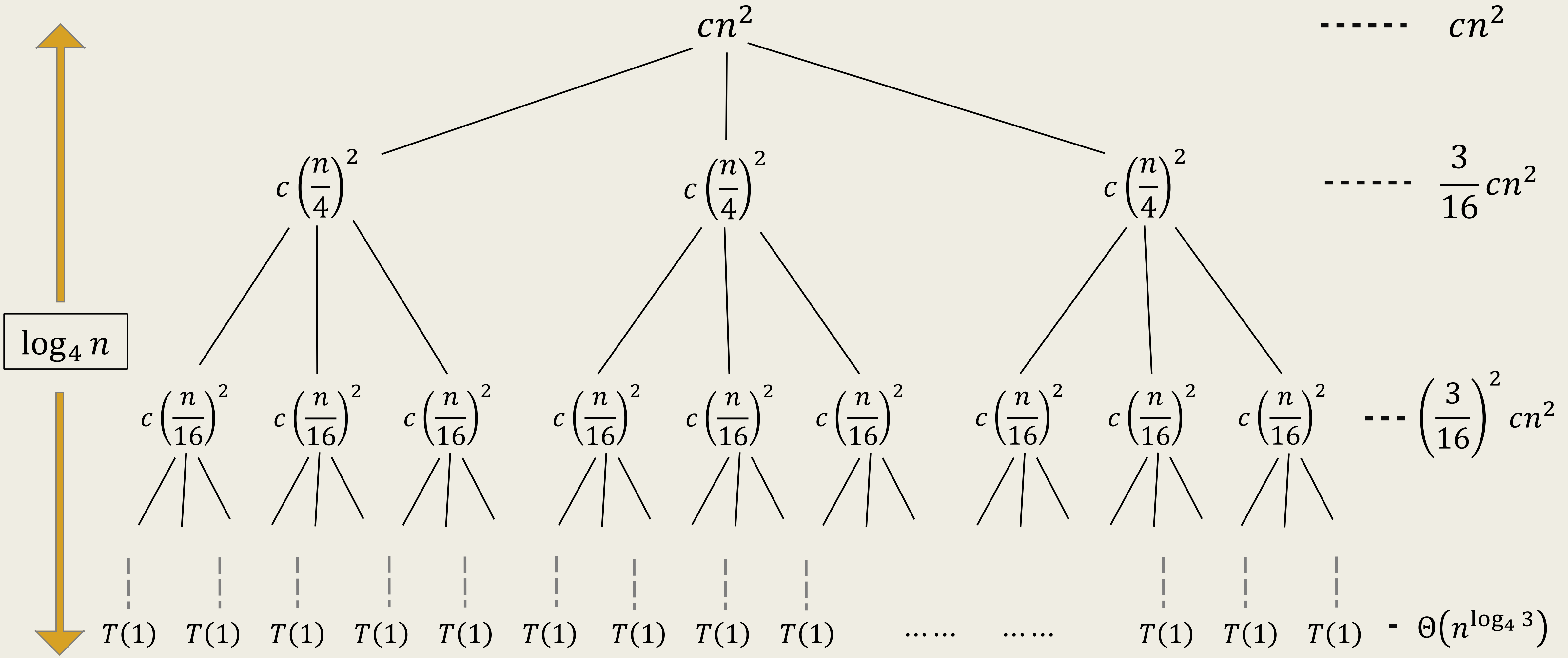
$$T(n) = 3T(\lfloor n/4 \rfloor) + c \cdot n^2 .$$



$$T(n) = 3T(\lfloor n/4 \rfloor) + c \cdot n^2 .$$

Guess $T(n) = O(n^2)$.

Total $O(n^2)$



The Master Theorem

A general theorem for $T(n) = aT(n/b) + f(n)$.

- **Theorem. (Master Theorem).**

  Let $a \geq 1$ and $b > 1$ be constants and $f(n)$ be a function.

  Let $T(n)$ be defined on non-negative integers by the recurrence

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

where n/b can either be $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.

Then $T(n)$ ***can be bounded asymptotically*** as follows.

- If $f(n) = O\left(n^{(\log_b a)-\epsilon}\right)$ for some $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.
- If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.
- If $f(n) = \Omega\left(n^{(\log_b a)+\epsilon}\right)$ for some $\epsilon > 0$ and
  if $a \cdot f(n/b) \leq c \cdot f(n)$ for some $c < 1$ and sufficiently large n,
  then $T(n) = \Theta\big(f(n)\big)$.