

LECTURE 06 — DIVIDE AND CONQUER (PART 2)

COMPSCI 308 — DESIGN AND ANALYSIS OF ALGORITHMS

Xing Shi Cai <https://newptcai.gitlab.io>

Duke Kunshan University, Kunshan, China

Jan-Mar 2026

SUMMARY

ITA 4.3 The Substitution Method

ITA 4.4 The Recursion-Tree Method

ITA 4.5 Master's Theorem

A method to guess
what is $T(n) = O(?)$

A theorem to cover
most common situation

ASSIGNMENTS¹



Practice makes perfect!

📘 Required Readings:

- Section 4.3
- Section 4.4
- Section 4.5

✏️ Required Exercises:

- Exercises 4.3: 1–3,
- Exercises 4.4: 1–3,
- Exercises 4.5: 1–4.

¹ ⚪ Assignments will not be collected; however, quiz problems will be selected from them. (This includes both Readings and Exercises.)

ITA 4.3 THE SUBSTITUTION METHOD

EXPERIMENTAL MATHEMATICS

Experimental Mathematics is a branch of mathematics which employs computations and simulations to test conjectures and identify patterns.

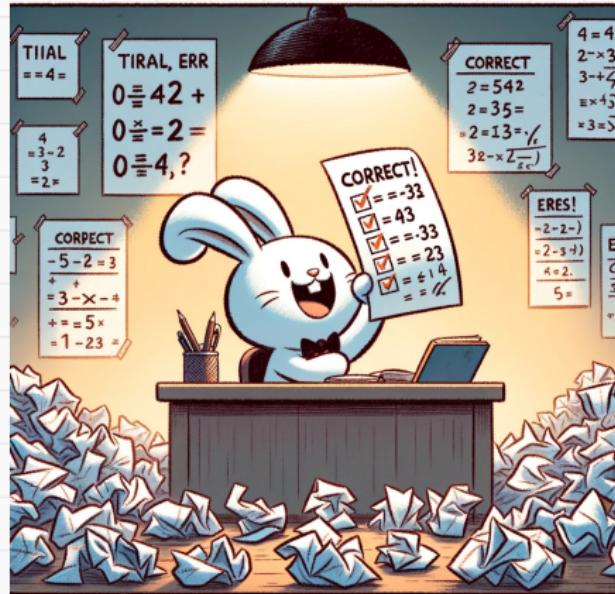


Figure 1: Math is 1% inspiration and 99% not eating the paper.

LET'S DO SOME EXPERIMENTS

Consider the recursion —

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n-1) + 1 & \text{otherwise} \end{cases} \quad n \geq 2.$$

Let's compute some values —

$$\langle T(1), T(2), \dots \rangle = \langle 1, 3, \underline{7}, \underline{(5, 31)}, \underline{63, 127}, \dots \rangle$$
$$\begin{aligned} T(3) &= 2T(2) + 1 \\ &= 2(2T(1) + 1) + 1 \\ &= 2(2 \cdot 1 + 1) + 1 \\ &= 2(3) + 1 \\ &= 7 \end{aligned}$$
$$\begin{aligned} T(4) &= 2T(3) + 1 \\ &= 2(7) + 1 \\ &= 15 \end{aligned}$$

🎂 Can you make a guess?

$$T(n) = \underline{2^n - 1} \quad \text{for all } n \geq 1$$

💡 You can also search the sequence on  OEIS.

🤔 How can we prove the pattern is true?

↳ Online Encyclopedia
of Integer sequences.

😱 How DOMINOES FALL

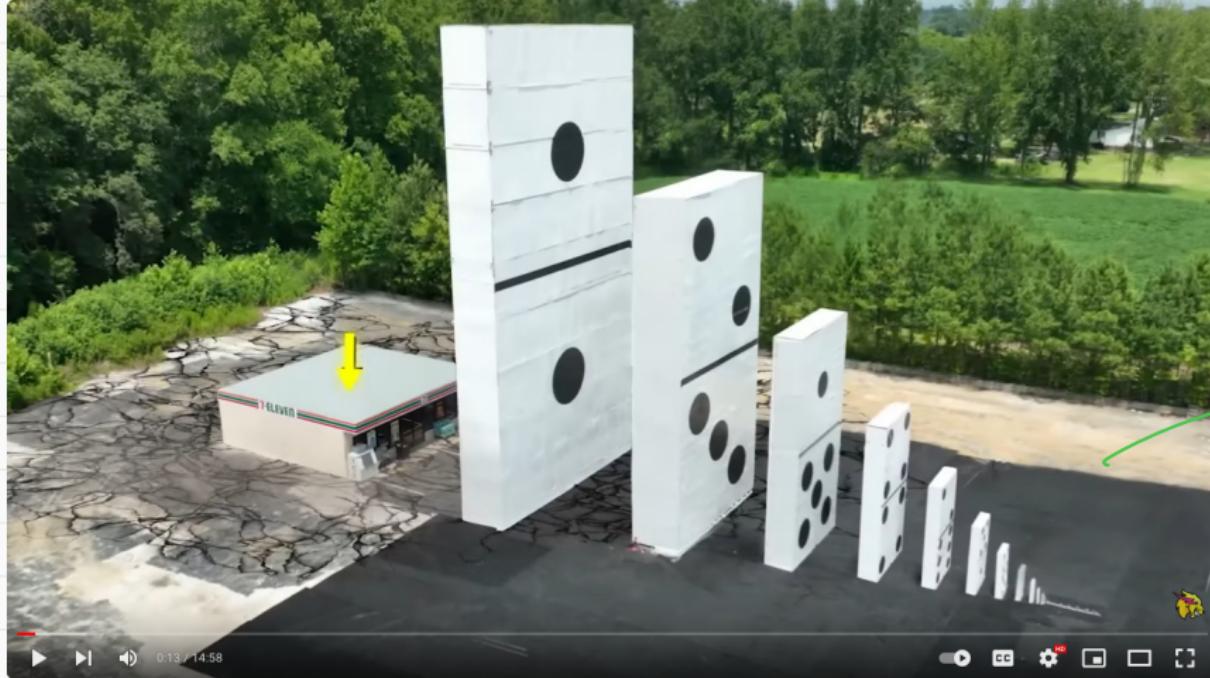


Figure 2: •• Watch 40 seconds of this YouTube video

PROOF BY INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use induction —

- *Base Case* () — Show S_{n_0} is true.



麻-将 牌



PROOF BY INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **induction** —

- *Base Case* (👉 中) — Show S_{n_0} is true.
- *Inductive Step* (中👉中) —



PROOF BY INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **induction** —

- *Base Case* (👉 中) — Show S_{n_0} is true.
- *Inductive Step* (中👉中) —
 1. Assume that the statement S_k is true for some integer $k \geq n_0$.

PROOF BY INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **induction** –

- **Base Case** () – Show S_{n_0} is true.
- **Inductive Step** () –

1. Assume that the statement S_k is true for some integer $k \geq n_0$.
2. Show that S_{k+1} is true.

Induction hypothesis.



one domino pushes down the next.



PROOF BY INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **induction** —

- *Base Case* (👉➡️) — Show S_{n_0} is true.
- *Inductive Step* (➡️👉➡️) —
 1. Assume that the statement S_k is true for some integer $k \geq n_0$.
 2. Show that S_{k+1} is true.
- Conclude that S_n is true for all $n \geq n_0$.

Try to prove
} $S_k \rightarrow S_{k+1}$



PROOF BY INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **induction** —

- *Base Case* (👉 中) — Show S_{n_0} is true.
- *Inductive Step* (中👉中) —
 1. Assume that the statement S_k is true for some integer $k \geq n_0$.
 2. Show that S_{k+1} is true.
- Conclude that S_n is true for all $n \geq n_0$.

💡 Note that the exact value of n_0 usually does not matter for algorithm analysis as we are allowed to choose n_0 arbitrarily.

 EXAMPLE OF PROOF BY INDUCTION

We'll use induction to prove our guess

$$T(n) = 2^n - 1, \quad \text{for all } n \geq 1 \quad (1)$$

EXAMPLE OF PROOF BY INDUCTION

We'll use induction to prove our guess

$$T(n) = 2^n - 1,$$

for all $n \geq 1$

✓

$$\begin{cases} T(n) = 2 \cdot T(n-1) + 1, \\ \text{if } n \geq 2, \\ T(1) = 1. \end{cases} \quad (1)$$

Base Case (👉 中) — For $n = 1$, we have

n_0

$$T(1) = 2^1 - 1 = 1$$

EXAMPLE OF PROOF BY INDUCTION

We'll use induction to prove our guess

$$T(n) = 2^n - 1, \quad \text{for all } n \geq 1 \quad (1)$$

Base Case (👉 中) — For $n = 1$, we have

$$T(1) = 2^1 - 1 = 1$$

Inductive Step (中👉中) — Assume that (1) holds for $n = k \geq 1$, i.e.,

$$T(k) = 2^k - 1, \quad (\text{Induction Hypothesis})$$

Then for $n = k + 1$,

$$\begin{aligned} T(n) &= T(k+1) = 2 \cdot T(k) + 1 \stackrel{?}{=} 2 \cdot (2^k - 1) + 1 \\ &= 2^{k+1} - 1 = 2^n - 1. \end{aligned}$$

STRONG INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **strong induction** —

- *Base Case* (  ... S_{n_0}, S_{n_0+1}, \dots, S_{n_1} are true.

STRONG INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **strong induction** —

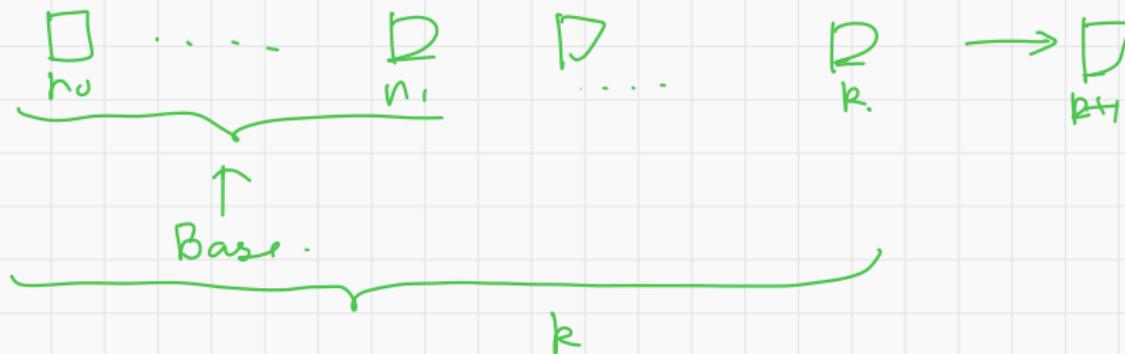
- *Base Case* (👉 中 ... 中) — Show $S_{n_0}, S_{n_0+1}, \dots, S_{n_1}$ are true.
- *Inductive Step* (中 中 中 ... 中👉 中) —

STRONG INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **strong induction** —

- *Base Case* (👉 中 ... 中) — Show $S_{n_0}, S_{n_0+1}, \dots, S_{n_1}$ are true.
- *Inductive Step* (中 中 中 ... 中👉 中) —
 1. Assume that the statement S_n is true for all n with $n_0 \leq n \leq k$ for some integer $k \geq n_1$.



STRONG INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **strong induction** —

- *Base Case* ( 中 ... 中) — Show $S_{n_0}, S_{n_0+1}, \dots, S_{n_1}$ are true.
- *Inductive Step* (中 中 中 ... 中  中) —
 1. Assume that the statement S_n is true *for all* n with $n_0 \leq n \leq k$ for some integer $k \geq n_1$.
 2. Show that S_{k+1} is true.

STRONG INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **strong induction** —

- *Base Case* (👉 中 ... 中) — Show $S_{n_0}, S_{n_0+1}, \dots, S_{n_1}$ are true.
- *Inductive Step* (中 中 中 ... 中👉 中) —
 1. Assume that the statement S_n is true *for all* n with $n_0 \leq n \leq k$ for some integer $k \geq n_1$.
 2. Show that S_{k+1} is true.
- Conclude that S_n is true for all $n \geq n_0$.

STRONG INDUCTION

Let S_n be a statement.

To prove S_n is true for all $n \geq n_0$, we can use **strong induction** —

- *Base Case* ( ... S_{n_0}, S_{n_0+1}, \dots, S_{n_1} are true.
 - *Inductive Step* (   ...   
 - Conclude that S_n is true for all $n \geq n_0$.
-  The choices of n_0 and n_1 depend on how many  need to fall in order for every  to fall.

GUESSING AN UPPER BOUND

Consider

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1, \\ 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & \text{otherwise.} \end{cases}$$

It is not difficult to guess that $T(n) = O(n \log n)$. What remains is to prove it.

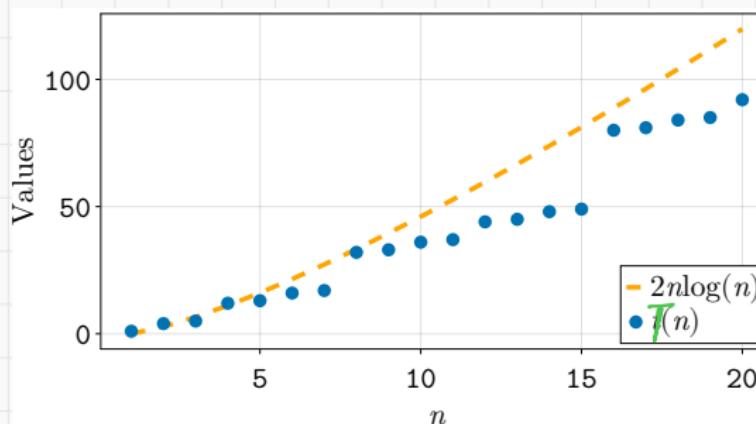


Figure 3: Guessing the upper bound of $T(n)$ by experiments.

PROVING THE UPPER BOUND

To prove $T(n) = O(n \log n)$, we show that there exists c and n_0 such that

Definition 

$$T(n) \leq cn \log n \quad \text{for all } n \geq n_0. \quad (2)$$

PROVING THE UPPER BOUND

To prove $T(n) = O(n \log n)$, we show that there exists c and n_0 such that

$$T(n) \leq cn \log n \quad \text{for all } n \geq n_0. \quad (2)$$

Inductive Step — Assume that

$$T(n) \leq cn \log n \quad \text{for all } n \in \{n_0, \dots, k\} \quad (3)$$

for some integer $k \geq n_1 \geq 2n_0 \geq 1$.

 The exact values of c , n_0 and n_1 can be decided in the base case.

PROVING THE UPPER BOUND

To prove $T(n) = O(n \log n)$, we show that there exists c and n_0 such that

$$T(n) \leq cn \log n \quad \text{for all } n \geq n_0. \quad (2)$$

Inductive Step — Assume that

$$T(n) \leq cn \log n \quad \text{for all } n \in \{n_0, \dots, k\} \quad (3)$$

for some integer $k \geq n_1 \geq 2n_0 \geq 1$.

Then for $n = k + 1$,

$$\lfloor n/2 \rfloor \in \{n_0, \dots, k\}.$$

$$n = k+1 > 2n_0 \rightarrow \left\lfloor \frac{n}{2} \right\rfloor \geq n_0$$

$$n = k+1 \rightarrow \left\lfloor \frac{n}{2} \right\rfloor \leq \frac{n}{2} = \frac{k+1}{2} \leq k.$$

Because
 $k \geq 1$

PROVING THE UPPER BOUND

To prove $T(n) = O(n \log n)$, we show that there exists c and n_0 such that

$$T(n) \leq cn \log n \quad \text{for all } n \geq n_0. \quad (2)$$

Inductive Step — Assume that

$$T(n) \leq cn \log n \quad \text{for all } n \in \{n_0, \dots, k\}$$

for some integer $k \geq n_1 \geq 2n_0 \geq 1$.

Then for $n = k + 1$,

$$\lfloor n/2 \rfloor \in \{n_0, \dots, k\}$$

Thus

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

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

$$= c \cdot n \log n - c \cdot n \log 2 + n. \quad \approx c \cdot n \cdot \log n.$$

We can pick
c large
enough. (3)

PROVING THE UPPER BOUND

To prove $T(n) = O(n \log n)$, we show that there exists c and n_0 such that

$$T(n) \leq cn \log n \quad \text{for all } n \geq n_0. \quad (2)$$

Base Case — We have to choose some appropriate n_0 , n_1 and c such that

$$n_1 \geq 2n_0 \geq 1$$

and that

$$T(n) \leq cn \log n \quad \text{for all } n \in \{n_0, \dots, n_1\}. \quad (3)$$

PROVING THE UPPER BOUND

To prove $T(n) = O(n \log n)$, we show that there exists c and n_0 such that

$$T(n) \leq cn \log n \quad \text{for all } n \geq n_0. \quad (2)$$

Base Case — Let's try $n_0 = 1$ and $n_1 = 2$ since $n_1 \geq 2n_0 \geq 1$.

🤔 Is it possible to pick a c such that

$$T(n) \leq cn \log n \quad \text{for all } n \in \{n_0, \dots, n_1\}. \quad (3)$$

{ 1, 2 }

💡 Note that

$$\langle T(1), T(2) \rangle = \langle 1, 4 \rangle$$

and for $f(n) = n \log n$

$$\langle f(1), f(2) \rangle \approx \langle 0, 2.77 \rangle$$

$$\begin{array}{cc} \uparrow & \downarrow \\ 1 \cdot \log 1 & 2 \cdot \log 2 \end{array}$$

PROVING THE UPPER BOUND

To prove $T(n) = O(n \log n)$, we show that there exists c and n_0 such that

$$T(n) \leq cn \log n \quad \text{for all } n \geq n_0. \quad (2)$$

Base Case — 🎉 What about $n_0 = 2$ and $n_1 = 4$?

🎂 Is it possible to pick a c such that

$$T(n) \leq cn \log n \quad \text{for all } n \in \{n_0, \dots, n_1\}. \quad (3)$$

💡 Note that

$$\tilde{\{T(2), T(3), T(4)\}} = \{2, 3, 4\}.$$

$$\langle T(2), T(3), T(4) \rangle = \langle 4, 5, 12 \rangle$$

and for $f(n) = n \log n$

$$\langle f(2), f(3), f(4) \rangle \approx \langle 2.8, 6.6, 11.1 \rangle$$

So we can simply take $c = \underline{10^6}$

😊 WHY CAN WE IGNORE THE BASE CASE?

Because we can pick n_0 and n_1 as large as we want, the base case is usually not a problem.

💡 Thus, in algorithm analysis we often simply skip checking the base case and do not bother with choosing n_0 and n_1 .

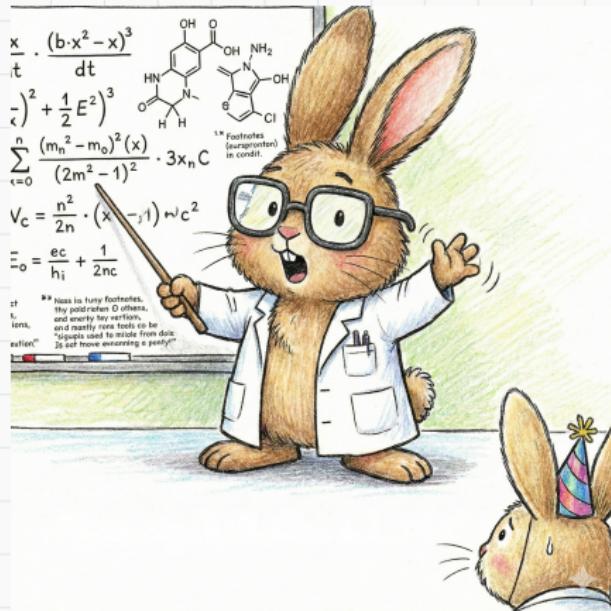


Figure 4: 😊 Giving too many details is a sign of 📈 scientist!

FROM EXACT RECURSION TO ASYMPTOTICS

Consider

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

We will prove that $T(n) = O(n \log n)$ by showing that there exists n_0 and c such that

$$T(n) \leq cn \lg n, \quad \text{for all } n \geq n_0.$$

FROM EXACT RECURSION TO ASYMPTOTICS

Consider

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

We will prove that $T(n) = O(n \log n)$ by showing that there exists n_0 and c such that

$$T(n) \leq cn \lg n, \quad \text{for all } n \geq n_0.$$

Inductive Step — Assume that there exists n_0 and c such that

$$T(n) \leq cn \log n \quad \text{for all } n \in \{n_0, \dots, k\} \tag{4}$$

for some integer $k \geq n_1 \geq 2n_0 \geq 1$. ← Hot really need.

Then for $n = k + 1$,



$$\left\lfloor \frac{n}{2} \right\rfloor \in \{n_0, \dots, k\}.$$

Assume that the induction hypothesis is true for $\left\lfloor \frac{n}{2} \right\rfloor$

FROM EXACT RECURSION TO ASYMPTOTICS

Consider

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

We will prove that $T(n) = O(n \log n)$ by showing that there exists n_0 and c such that

$$T(n) \leq cn \lg n, \quad \text{for all } n \geq n_0.$$

Inductive Step — Thus

$$\begin{aligned} &= 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n). \\ T(n) &\leq 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + \Theta(n) \\ &\leq 2(c(n/2) \lg(n/2)) + \Theta(n) \\ &= cn \lg(n/2) + \Theta(n) \\ &= cn \lg n - cn \lg 2 + \Theta(n) \\ &= cn \lg n - cn + \Theta(n) \quad \leq c_1 n. \\ &\leq cn \lg n. \end{aligned}$$

↑ We pick $c > c_1$

Induction.

How to MAKE A GOOD GUESS

Consider the recurrence relation

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor + 17\right) + \Theta(n)$$

We guess:

$$T(n) = O\left(\underbrace{n \cdot \log n}_{\text{---}}\right)$$

How to MAKE A GOOD GUESS

Consider the recurrence relation

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor + 17\right) + \Theta(n)$$

🎂 Since $\left\lfloor \frac{n}{2} \right\rfloor + 17 = \Theta(\frac{n}{2})$ when n is large, a 🤝 guess would be

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

How to MAKE A GOOD GUESS

Consider the recurrence relation

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor + 17\right) + \Theta(n)$$

Experiments can further confirm our guess.

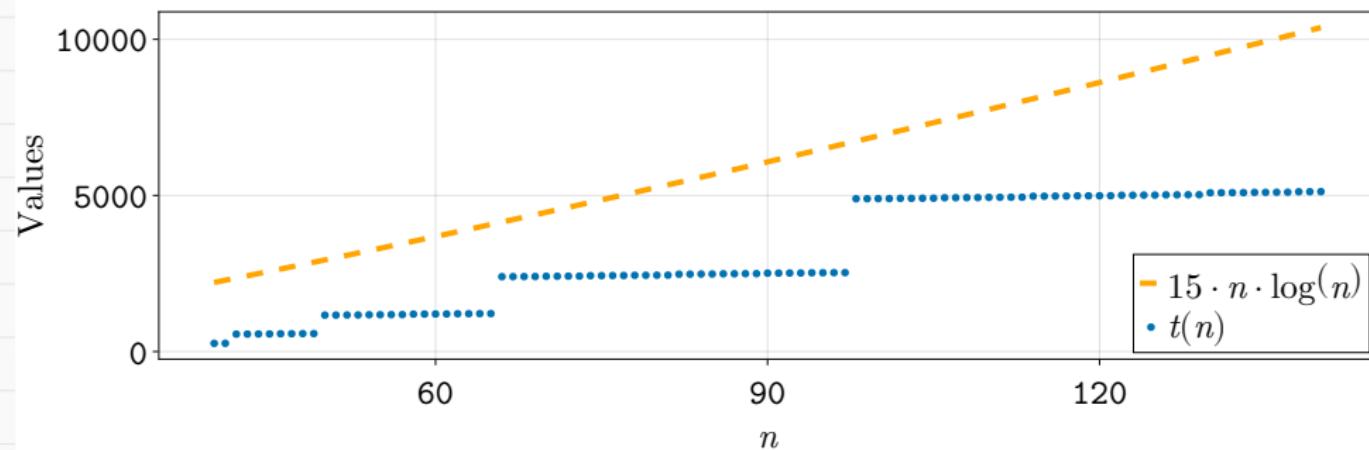


Figure 5: The upper bound of (13).

START WITH LOOSE BOUNDS

Given

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

we can immediately see that

$$T(n) = \Omega(n). \quad (\text{Why?})$$

It is also quite easy to show that

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

Using these as **stepping stones**, we can increase the lower and upper bounds until they match, which is

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

This trick is sometimes called **bootstrapping**.

A TRICK—SUBTRACTING LOWER-ORDER TERMS

Consider

$$T(n) = T(n/2) + \Theta(1).$$

Let us show that $T(n) = O(n)$.

A TRICK—SUBTRACTING LOWER-ORDER TERMS

Consider

$$T(n) = T(n/2) + \Theta(1).$$

Let us show that $T(n) = O(n)$.

We can try to prove that there exist c, n_0 such that

$$T(n) \leq cn \quad \text{for all } n \geq n_0.$$

Inductive Step — Assume that

$$T(n) \leq cn \quad \text{for all } n \in \{n_0, \dots, k\}, \tag{4}$$

for some integer $k \geq n_1 \geq 2n_0 \geq 1$.

🤔 Can we use this to prove that for $n = k + 1$,

$$\begin{aligned} T(n) &\leq 2(cn/2) + \Theta(1) \leq cn \\ &= 2T(n/2) + \Theta(1). \end{aligned}$$

Incorrect.

A TRICK—SUBTRACTING LOWER-ORDER TERMS

Consider

$$T(n) = T(n/2) + \Theta(1). \quad \Leftrightarrow \quad \exists c_1, c_2 \text{ s.t. } c_1 = c_1 \cdot | \leq T(n) \leq c_2 \cdot | \leq c_2$$

Let us show that $T(n) = O(n)$.

选拨 Now let's try something stronger — prove that there exist c, d, n_0 such that

$$T(n) \leq cn - d \quad \text{for all } n \geq n_0$$

Inductive Step — Assume that

$$T(n) \leq cn - d \quad \text{for all } n \in \{n_0, \dots, k\} \tag{4}$$

for some integer $k \geq n_1 \geq 2n_0 \geq 1$.

选拨 Can we use this to prove that for $n = k + 1$,

$$T(n) \leq cn - d.$$

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + \Theta(1) \\ &\leq 2\left[c\left(\frac{n}{2}\right) - d\right] + \Theta(1) \\ &\leq cn - d \quad [-d + \Theta(1)] \\ &\leq cn - d. \end{aligned}$$



WHY IS THIS WRONG?

Consider again

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

Can we prove $T(n) = O(n)$ by

$$\begin{aligned} T(n) &\leq 2O\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) \\ &\leq 2O(n) + \Theta(n) \\ &= O(n). \end{aligned}$$



WHY IS THIS WRONG?

Consider again

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

🤔 Can we prove $T(n) = O(n)$ by

$$\begin{aligned} T(n) &\leq 2O\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) \\ &\leq 2O(n) + \Theta(n) \\ &= O(n). \end{aligned}$$

🤔 What about

$$\begin{aligned} T(n) &\leq 2\left(c\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) \\ &\leq cn + \Theta(n) \\ &= O(n). \end{aligned}$$

ITA 4.4 THE RECURSION-TREE METHOD



THE RECURSION TREE METHOD

Consider

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

We will use a recursion tree to guess an upper bound of $T(n)$.

🍰 Do you see why

$$T(n) = \Omega(n^2)$$

 THE RECURSION TREE METHOD

Consider

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

Note that the above implies that there exist constants c and n_0 such that

$$T(n) \leq 3T(\lfloor n/4 \rfloor) + cn^2 \quad \text{for all } n \geq n_0.$$

 Often we omit mentioning n_0 by saying —

There exists a constant c such that

$$T(n) \leq 3T(\lfloor n/4 \rfloor) + cn^2$$

n is large enough.

THE RECURSION TREE METHOD

😊 We also drop the $\lfloor \cdot \rfloor$ and use only

$$T(n) \leq 3T(n/4) + cn^2.$$

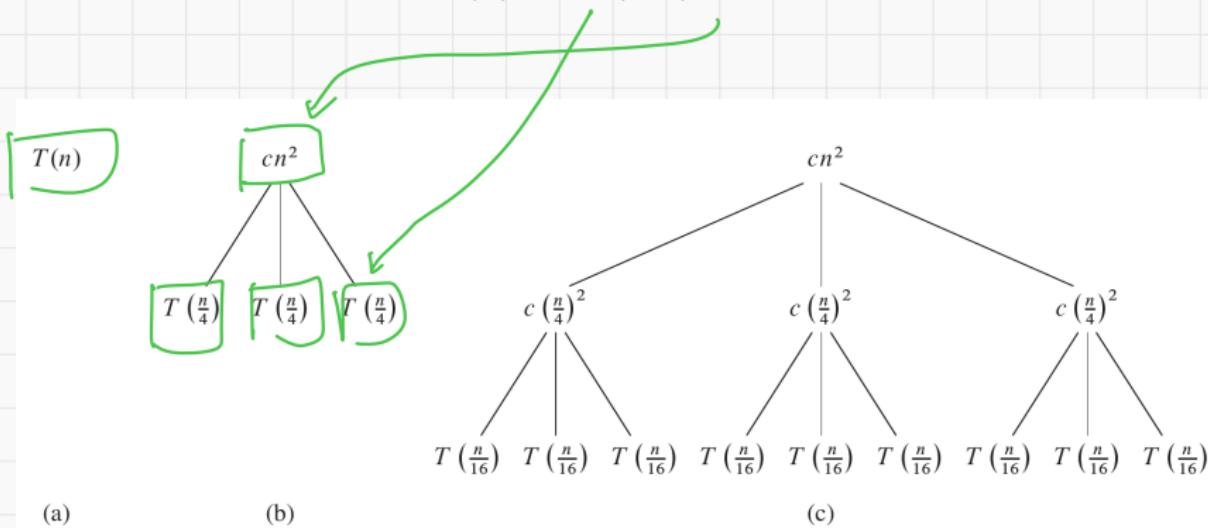


Figure 6: Recursion for $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

THE COMPLETE RECURSION TREE

Expanding the tree recursively, we eventually get the following tree –

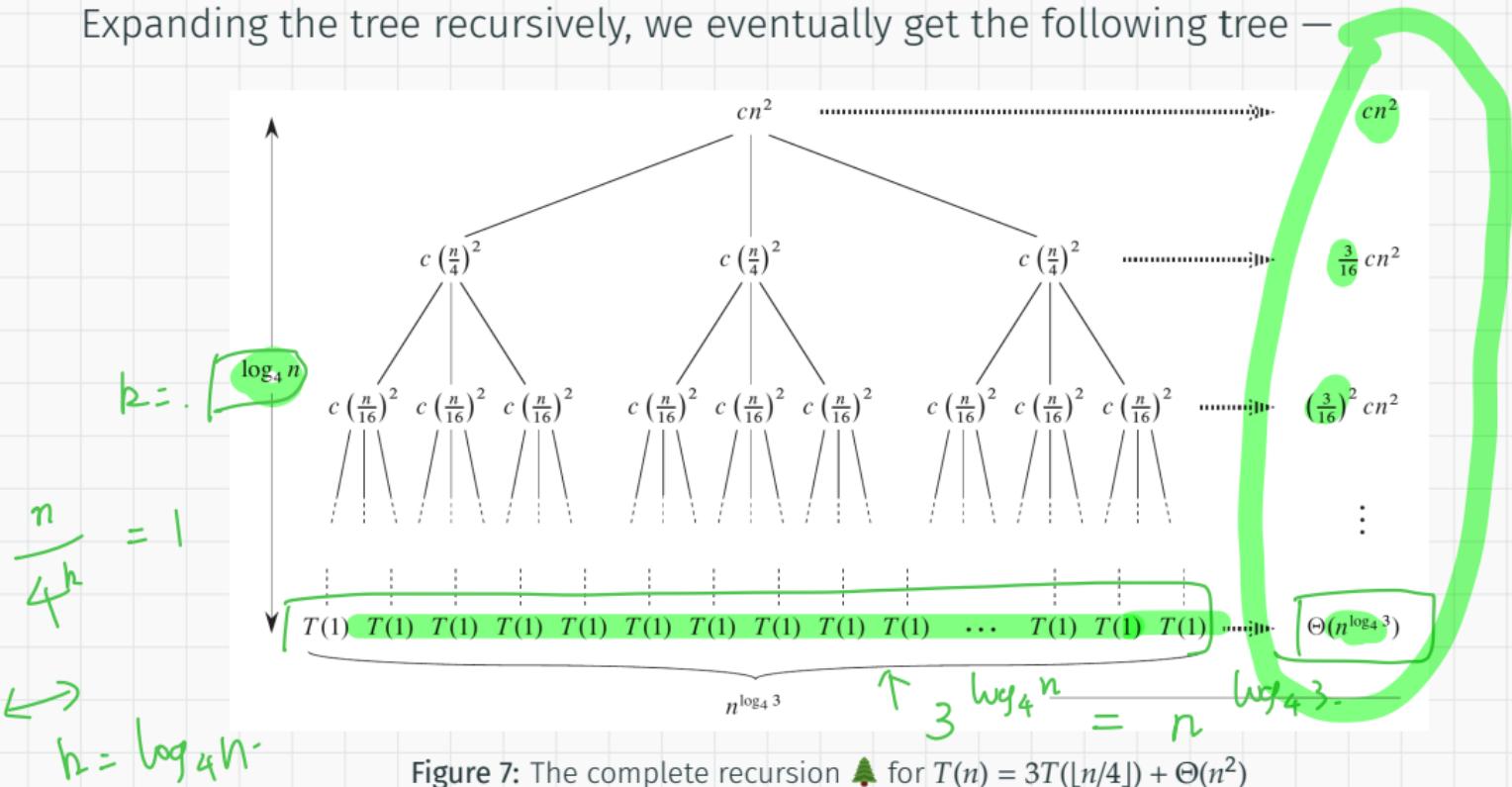


Figure 7: The complete recursion tree for $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

PROVING THE UPPER BOUND

By the recursion tree, we have

$$\begin{aligned} T(n) &\leq \sum_{i=0}^{\log_4(n)-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &\leq \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &= \frac{1}{1 - \frac{3}{16}} cn^2 + \Theta(n^{\log_4 3}) = O(n^2) \end{aligned}$$

Leave. 

Geometric progression.  $\log_4 3 < 1.$

So we can guess $T(n) = \Theta(n^2)$.

We can use induction (substitution method) to prove this.

PROVING THE UPPER BOUND

By the recursion tree, we have

$$\begin{aligned} T(n) &\leq \sum_{i=0}^{\log_4(n)-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &\leq \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\ &= \frac{1}{1 - \frac{3}{16}} cn^2 + \Theta(n^{\log_4 3}) = O(n^2) \end{aligned}$$

where we use the formula

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}, \quad \text{for all } |x| < 1.$$

AN IRREGULAR EXAMPLE

Consider

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

🎂 Can you get an upper bound of $T(n)$ from its recursion tree?

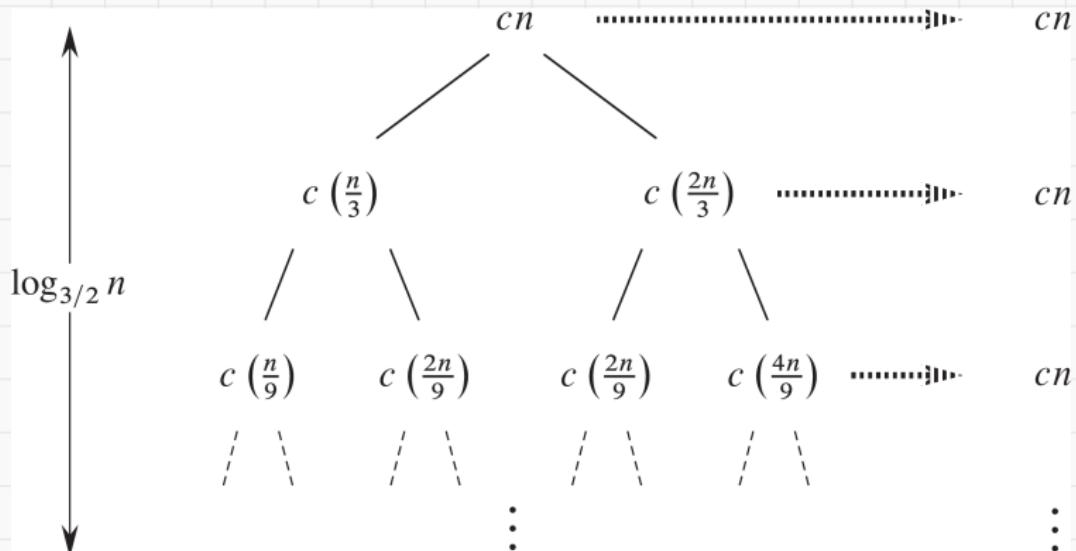


Figure 8: The recursion 🎄 for $T(n)$ in (5)

ITA 4.5 MASTER'S THEOREM

THEOREM 4.1 – MASTER'S THEOREM

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = \boxed{aT\left(\frac{n}{b}\right)} + f(n), \text{ for all } n \in \mathbb{N},$$

where $aT\left(\frac{n}{b}\right)$ actually means

$$a'T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + a''T\left(\left\lceil \frac{n}{b} \right\rceil\right)$$

and $a = a' + a''$.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.

$$\rightarrow f(n) = o(n^{\log_b a}).$$

e.g. if $a = 2$

$$b = 2.$$

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

$$+ T\left(\left\lceil \frac{n}{2} \right\rceil\right)$$

$$+ + (n).$$

THEOREM 4.1 – MASTER'S THEOREM

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ for all } n \in \mathbb{N},$$

where $aT\left(\frac{n}{b}\right)$ actually means

$$a'T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + a''T\left(\left\lceil \frac{n}{b} \right\rceil\right)$$

and $a = a' + a''$.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.

THEOREM 4.1 – MASTER'S THEOREM

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ for all } n \in \mathbb{N},$$

where $aT\left(\frac{n}{b}\right)$ actually means

$$a'T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + a''T\left(\left\lceil \frac{n}{b} \right\rceil\right)$$

and $a = a' + a''$.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$,

THEOREM 4.1 – MASTER'S THEOREM

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ for all } n \in \mathbb{N},$$

where $aT\left(\frac{n}{b}\right)$ actually means

$$a'T\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + a''T\left(\left\lceil \frac{n}{b} \right\rceil\right)$$

and $a = a' + a''$.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n , then $T(n) = \Theta(f(n))$.

😊 The proof is *not* required and you don't have to remember it.

EXAMPLE OF CASE 2

Master's Theorem

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ for all } n \in \mathbb{N}.$$

2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.

🍰 To apply the theorem to

what should $a, b, f(n)$ be?

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

$T(n) = 2T(n/2) + \Theta(n)$

\uparrow \uparrow
 $a=2$ $b=2$

← Same.

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

EXAMPLE OF CASE 1

Master's Theorem

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ for all } n \in \mathbb{N}.$$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.

🍰 To apply the theorem to

what should $a, b, f(n)$ and ϵ be?

$$\begin{aligned} T(n) &= 9T(n/3) + n \\ &\quad \uparrow \quad \curvearrowleft \quad \curvearrowleft \\ a &= 9 \quad b = 3 \quad n^{\log_3 9} \\ &= O(n^{2.0}) \\ &= n^{\log_3 9} \\ &= [n^2] \end{aligned}$$

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2).$$

EXAMPLE OF CASE 3

Master's Theorem

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ for all } n \in \mathbb{N}.$$

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n , then $T(n) = \Theta(f(n))$.

 To apply the theorem to

$$T(n) = 3T(n/4) + n \log n$$

what should $a, b, f(n)$ and ϵ be?

THE GAP BETWEEN CASE 1 AND 2

Master's Theorem

Let $a \geq 1$ and $b > 1$ be constants. Let

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ for all } n \in \mathbb{N}.$$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then

Consider

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

Since in this case, we have $n^{\log_b a} = n$ yet

$$f(n) = \frac{n}{\log(n)} = o(n), \quad \text{and} \quad f(n) = \frac{n}{\log(n)} = \omega(n^{1-\epsilon}) \text{ for all } \epsilon > 0,$$

we *cannot* apply Master's Theorem.

WHAT ARE YOUR MAIN TAKEAWAYS TODAY? ANY QUESTIONS?

