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

Assignments[1]

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

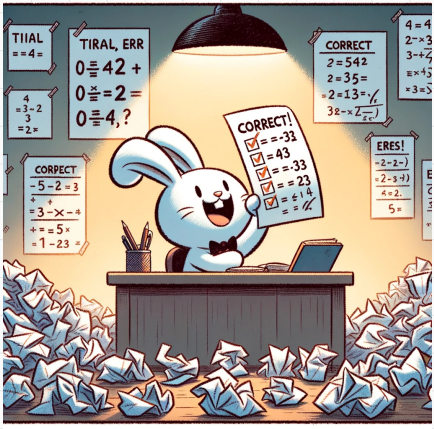

Figure 1: Math is 1% inspiration and 99% not eating the paper.

Let's Do Some Experiments

Consider the recursion —

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

Let's compute some values —

$$\langle T(1), T(2), \dots \rangle = \langle 1, 3, \_\_\_\_\_ \rangle$$

🍰 Can you make a guess?

$$T(n) = \_\_\_\_\_ \qquad \text{for all } n \geq 1$$

💡 You can also search the sequence on 👆 OEIS.

🤔 How can we prove the pattern is true?

😮 How Dominoes Fall

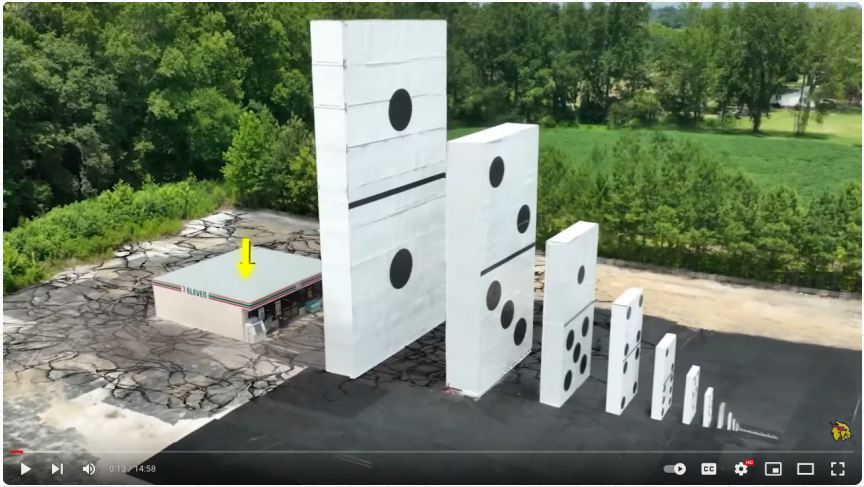

Figure 2: 👀 Watch 40 seconds of 👆 this YouTube video

✏️ 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* (👉中) — Show S_{n_0} is true.

✏️ 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* (👉🀄) — Show S_{n_0} is true.
- *Inductive Step* (🀄👉🀄) —
  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.

✏️ 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 $n \geq n_0$.

✏️ 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

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

📌 Example of Proof by Induction

We'll use induction to prove our guess

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

Base Case (👉🇭) — For $n = 1$, we have

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

📌 Example of Proof by Induction

We'll use induction to prove our guess

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

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.,

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

Then for $n = k + 1$,

$$T(n) = \_\_\_\_\_$$

✏️ 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 $S_{n_0}, S_{n_0+1}, \ldots, S_{n_1}$ 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* (👉🀄 ... 🀄) — Show $S_{n_0}, S_{n_0+1}, \ldots, S_{n_1}$ are true.
- *Inductive Step* (🀄🀄🀄...🀄👉🀄) —

✏️ 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* (👉🀄 ... 🀄) — Show $S_{n_0}, S_{n_0+1}, \ldots, S_{n_1}$ are true.
- *Inductive Step* (🀄🀄🀄...🀄👉🀄) —
  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* (👉🀄 ... 🀄) — Show $S_{n_0}, S_{n_0+1}, \ldots, S_{n_1}$ are true.
- *Inductive Step* (🀄🀄🀄...🀄👉🀄) —
  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* (👉🀄 ... 🀄) — Show $S_{n_0}, S_{n_0+1}, \ldots, S_{n_1}$ are true.
- *Inductive Step* (🀄🀄🀄...🀄👉🀄) —
  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* (👉🀄 ... 🀄) — Show $S_{n_0}, S_{n_0+1}, \ldots, S_{n_1}$ are true.
- *Inductive Step* (🀄🀄🀄...🀄👉🀄) —
  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 $n \geq n_0$.

💡 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.

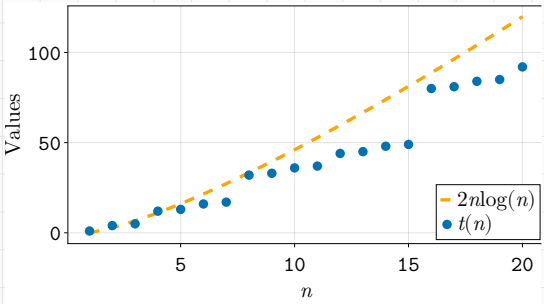

Figure 3: Guessing the upper bound of $T(n)$ by experiments.

Proving the Upper Bound

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

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

Proving the Upper Bound

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

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

Inductive Step — Assume that

$$T(n) \leq cn \log n \qquad \text{for all } n \in \{n_0, \ldots, k\} \tag{3}$$

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

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

Inductive Step — Assume that

$$T(n) \leq cn \log n \qquad \text{for all } n \in \{n_0, \ldots, k\} \tag{3}$$

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

Then for $n = k + 1$,

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

Proving the Upper Bound

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

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

Inductive Step — Assume that

$$T(n) \leq cn \log n \qquad \text{for all } n \in \{n_0, \ldots, k\} \tag{3}$$

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

Then for $n = k + 1$,

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

Thus

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

Proving the Upper Bound

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

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

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

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

Proving the Upper Bound

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

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

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

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

💡 Note that

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

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

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

Proving the Upper Bound

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

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

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

🍰 Is it possible to pick a c such that

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

💡 Note that

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

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

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

🙂 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.

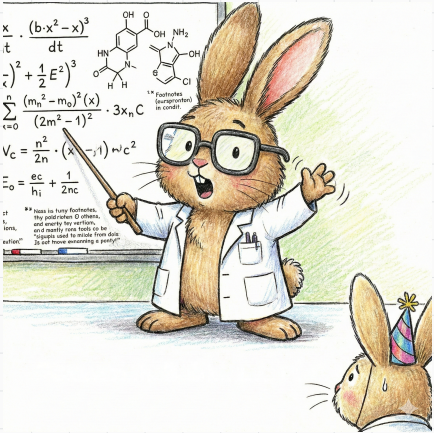

Figure 4: 😉 Giving too many details is a sign of 👎 scientist!

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

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

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

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

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

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

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

Then for $n = k + 1$,

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

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

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

Inductive Step — Thus

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

How to Make a Good Guess

Consider the recurrence relation

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

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

$$T(n) = O(\_\_\_\_\_).$$

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.

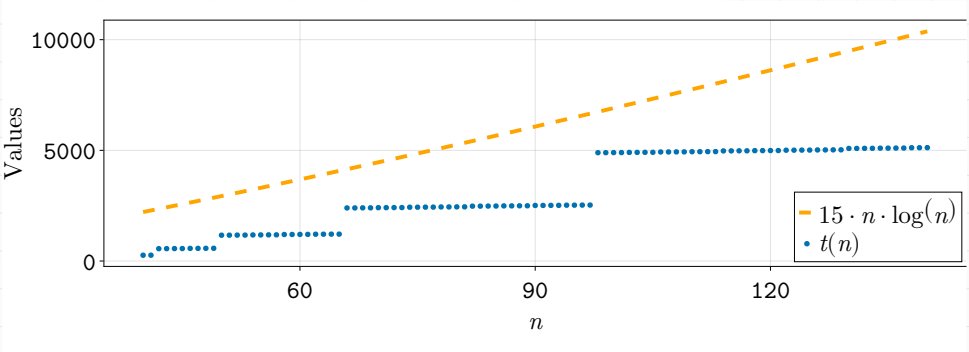

Figure 5: The upper bound of (13).

Start With Loose Bounds

Given

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

we can immediately see that

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

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

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

Inductive Step — Assume that

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

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

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

$$T(n) \leq 2(cn/2) + \Theta(1) \leq cn$$

A Trick—Subtracting Lower-Order Terms

Consider

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

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

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

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

Inductive Step — Assume that

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

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

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

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

😵 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

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

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

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

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

💡 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.

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

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

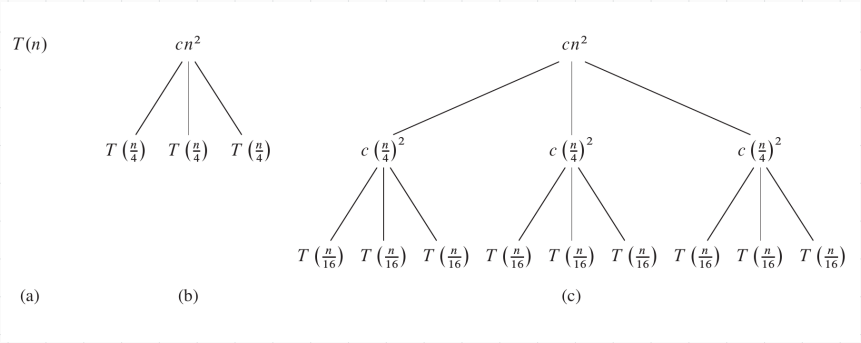

Figure 6: Recursion 🌲 for $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

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

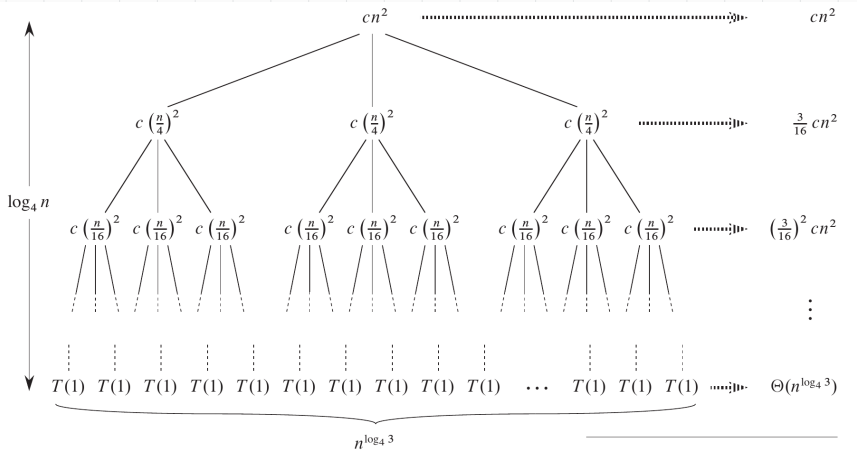

Figure 7: The complete recursion 🌲 for $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

Proving the Upper Bound

By the recursion tree, we have

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

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

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

Consider

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

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

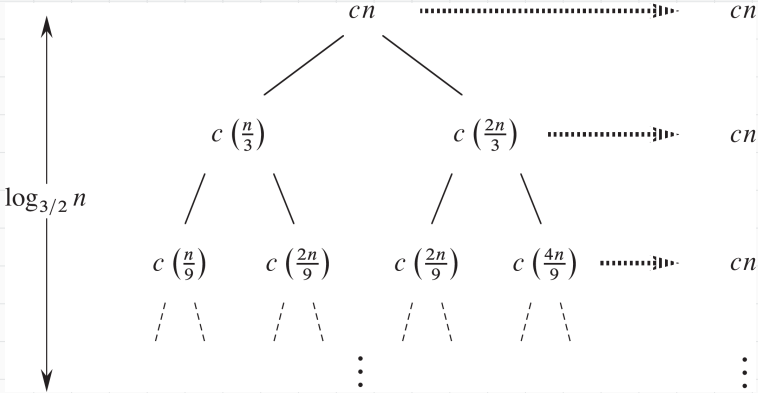

Figure 8: The recursion tree for $T(n)$ in (5)

ITA 4.5 Master's Theorem

Theorem 4.1 — Master's Theorem

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

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

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

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

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

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

📌 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

$$T(n) = 9T(n/3) + n$$

what should $a, b, f(n)$ and ϵ be?

📌 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?

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

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

we *cannot* apply Master's Theorem.

What are your main takeaways today? Any questions?