Master Theorem

Viet Tung Hoang

The slides are loosely based on those of Dr Kevin Wayne, Princeton University
1. Overview
2. Master Theorem
3. When Master Theorem Doesn’t Apply
Recall: Divide & Conquer

- Divide the problem into multiple **independent** subproblems
- Solve each subproblem and then combine them to form solution for original problem
Master Method

**Goal:** Recipe for solving common divide-and-conquer recurrences

\[ T(n) = aT(n/b) + \Theta(n^k) \]

with \( T(0) = 0 \) and \( T(1) = \Theta(1) \) and \( a \geq 1, \ b \geq 2, \ k \geq 0 \)

- \( a \): # of sub-problems
- \( b \): factor by which subproblem size decreases
- \( n^k \): combining cost
**Master Method**

**Goal:** Recipe for solving common divide-and-conquer recurrences

\[ T(n) = aT(n/b) + \Theta(n^k) \]

with \( T(0) = 0 \) and \( T(1) = \Theta(1) \)

---

**Merge Sort:**
\[ a = 2, \ b = 2, \ k = 1 \]

**Karatsuba’s Integer Multiplication:**
\[ a = 3, \ b = 2, \ k = 1 \]

---

**Binary Search:**
\[ a = 1, \ b = 2, \ k = 0 \]

**Strassen’s Matrix Multiplication:**
\[ a = 7, \ b = 2, \ k = 2 \]
The Recursion Tree

\[ T(n) = aT(n/b) + \Theta(n^k) \quad \text{and} \quad T(1) = \Theta(1) \]

\[ a^{\log_b(n)} = n^{\log_b(a)} \]
Agenda

1. Overview

2. Master Theorem

3. When Master Theorem Doesn’t Apply
Master Theorem

\[ T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k) \]

**Case 1:**
\[ k < \log_b(a) \]

**Case 2:**
\[ k = \log_b(a) \]

**Case 3:**
\[ k > \log_b(a) \]
An Example for Case 1

Example: \[ T(n) = 2T(n/2) + 1 \text{ and } T(1) = 1 \]
A Closer Look at Cost

Root

1

2

$2^2$

\[ \ldots \]

Leaves

$2^{\log_2(n)} = n$

**Observation:** Cost concentrates at leaves

Total cost: $\Theta(n)$
A Simplified Abstraction

\[ T(n) = 2T(n/2) + 1 \text{ and } T(1) = 1 \]

\[ T(n) = \Theta(n) \]

Also the total cost
Case 1, Generic

\[ T(n) = aT(n/b) + \Theta(n^k) \text{ with } k < \log_b(a) \]

Cost at root is **strictly smaller** than that at leaves

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

Also the total cost
An Example for Case 2

Example: \( T(n) = 2T(n/2) + n \) and \( T(1) = 1 \)
A Closer Look at Cost

Root

\[ n \]

... 

Leaves

\[ n \]

Observation: Cost is evenly distributed from root to leaves

Total cost: \( \Theta(n \log(n)) \)
A Simplified Abstraction

\[ T(n) = 2T(n/2) + n \text{ and } T(1) = 1 \]

\[ T(n) = \Theta(n \log(n)) \]

Root

Leaves

Log(n)
Case 2, Generic

\[ T(n) = aT(n/b) + \Theta(n^k) \text{ and } k = \log_b(a) \]

Cost at root is the same as leaves

\[ T(n) = \Theta\left(n^k \log(n)\right) \]
An Example for Case 3

Example: \( T(n) = 2T(n/2) + n^2 \) and \( T(1) = 1 \)

\[
\begin{align*}
\log_2(n) + 1 & \quad n^2 \\
2 \cdot \left(\frac{n}{2}\right)^2 &= \frac{n^2}{2} \\
4 \cdot \left(\frac{n}{4}\right)^2 &= \frac{n^2}{4} \\
\frac{n^2}{2^i} &
\end{align*}
\]

\( 2^{\log_2(n)} = n \)
A Closer Look at Cost

Root

\[ n^2 \]

\[ \frac{n^2}{2} \]

\[ \frac{n^2}{4} \]

... 

Leaves

\[ n \]

Observation: Cost concentrates at root

Total cost: \( \Theta(n^2) \)
A Simplified Abstraction

\[ T(n) = 2T(n/2) + n^2 \text{ and } T(1) = 1 \]

Root

\[ n^2 \]

Also the total cost

\[ T(n) = \Theta(n^2) \]

Leaves
Case 3, Generic

\[ T(n) = aT(n/b) + \Theta(n^k) \text{ and } k > \log_b(a) \]

Cost at root is \textbf{strictly greater} than that at leaves

\[ n^k \]

Also the total cost

\[ T(n) = \Theta(n^k) \]

Root

Leaves

\[ n^{\log_b(a)} \]
Master Theorem

\[ T(n) = aT(n/b) + \Theta(n^k) \]

**Case 1:** \( k < \log_b(a) \)

\[ n^k \]

\[ n^{\log_b(a)} \]

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

**Case 2:** \( k = \log_b(a) \)

\[ \log(n) \]

\[ n^k \]

\[ T(n) = \Theta(n^k \log(n)) \]

**Case 3:** \( k > \log_b(a) \)

\[ n^k \]

\[ n^{\log_b(a)} \]

\[ T(n) = \Theta(n^k) \]
Exercise

Find the Big-Theta of the following recurrences:

**Textbook Integer Multiplication:**
\[ T(n) = 4T(n/2) + \Theta(n) \]

**Karatsuba’s Integer Multiplication:**
\[ T(n) = 3T(n/2) + \Theta(n) \]

**Textbook Matrix Multiplication:**
\[ T(n) = 8T(n/2) + \Theta(n^2) \]

**Strassen’s Matrix Multiplication:**
\[ T(n) = 7T(n/2) + \Theta(n^2) \]
Gaps In Master Theorem

- Division is **unbalanced**:

\[ T(n) = T(n/5) + T(7n/10) + n \]

Median-Finding algorithm

- Number of sub-problems is **not** a constant:

\[ T(n) = nT(n/2) + n^2 \]

- Combining cost is **problematic**:

\[ T(n) = 2T(n/2) + \frac{n}{\log_2(n)} \]