CSE331 Introduction to Algorithm
Lecture 8: Solving Recurrences II

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

March 29, 2021

Introduction

- Assignment 1 is due today.
- This is the second part of the lecture on solving recurrences.
- Reference: Sections 4.3, 4.4 and 4.5 of the textbook Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein.

The Recursion Tree Method: Example 2

Problem

Find a good upper bound for $T(n) = T(n/3) + T(2n/3) + O(n)$.

(See textbook p. 91)

The Recursion Tree Method: Example 3

- Guess an upper bound on the running time of Karatsuba's algorithm, which satisfies:

$$T(n) = 3T(n/2) + \Theta(n)$$

  (Done in class.)

The Master Method

Theorem (Master Theorem)

Let $a \geqslant 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n) > 0$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. *If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, 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 $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \leqslant cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.*

The Master Method

- I will not prove the master theorem in CSE331.
- But here is some intuition.
- **Case 2:** $f(n) = \Theta(n^{\log_b a})$.
  - A simple calculation shows that for each level of the recursion tree, the cost is $\Theta(n^{\log_b a})$.
  - As $b > 1$, the height of the recursion tree is $\Theta(\log n)$.
  - So the running time is $T(n) = \Theta(n^{\log_b a} \log n)$.
- **Case 1:** $f(n)$ is much smaller than $n^{\log_b a}$.
  - As there are $n^{\log_b a}$ leaves, the cost of the leaves is $\Theta(n^{\log_b a})$.
  - So the leaves dominate the running time, and $T(n) = \Theta(n^{\log_b a})$.
- **Case 3:** $f(n)$ is much larger than $n^{\log_b a}$.
  - In this case the cost is dominated by the root node of the tree.
  - So $T(n) = \Theta(f(n))$.

The Master Method: Example 1

- The running time of BINARY SEARCH is given by:

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

- So $a = 1$, $b = 2$ and $f(n) = 1$.
- $n^{\log_b a} = n^0 = 1$.
- As $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$, we are in Case 2, and thus

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

The Master Method: Example 2

- The running time of MERGE SORT is given by:

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

- The floor and ceiling function can be discarded when we apply the master theorem, so we rewrite it:

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

- Thus $a = 2$, $b = 2$ and $f(n) = \Theta(n)$.
- $n^{\log_b a} = n$.
- We are in Case 2, and thus

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

The Master Method: Example 3

- The running time of Karatsuba's algorithm satisfies:

$$T(n) = 3T(n/2) + \Theta(n)$$

- So $a = 3$, $b = 2$ and $f(n) = \Theta(n)$.
- $n^{\log_b a} = n^{\log_2 3} \approx n^{1.59}$.
- As $f(n) = O(n^{\log_2(3)-1/2})$, we are in Case 1, and thus

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3}).$$

The Master Method: Example 4

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

- We have $a = 9$, $b = 3$ and $f(n) = n$.
- So $n^{\log_b a} = n^{\log_3 9} = n^2$.
- Therefore $f(n) = n = O(n^{\log_b a - \varepsilon})$ for $\varepsilon = 1$.
- We are in Case 1 of the master theorem, and thus $f(n) = \Theta(n^2)$.

The Master Method: Example 5

- Consider the recurrence relation $T(n) = T(2n/3) + 1$
- $a = 1$, $b = 3/2$ and $f(n) = 1$.
- $n^{\log_b a} = n^0 = 1$.
- We are in Case 2, and

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

- What is the connection with binary search?

The Master Method: Example 6

- Consider the recurrence relation $T(n) = 3T(n/4) + n \log n$
- $a = 3$, $b = 4$ and $f(n) = n \log n$.
- $n^{\log_b a} \simeq n^{0.8}$.
- $f(n) = \Omega(n^{\log_b a})$, so we are in Case 3, and

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

The Master Method: Example 7

- Consider the recurrence relation $T(n) = 2T(n/2) + n \log n$
- $a = 2$, $b = 2$ and $f(n) = n \log n$.
- $n^{\log_b a} = n$.
- None of the three cases applies, because we don't have $n \log n = \Omega(n^{1+\varepsilon})$ for any $\varepsilon > 0$.
  - Reason: $\log n$ grows more slowly than n^{ε} for every $\varepsilon > 0$.

The Master Method

- The master theorem provides a general method for solving recurrences.
- The three cases of the master theorem do not cover all possibilities.
  - (See previous slide.)
- You do not need to memorize the master theorem statement. I will give it in the exam.