Master Method

The Precise Statement

Design and Analysis of Algorithms I

The Master Method

Cool Feature : a “black box” for solving recurrences.

Assumption : all subproblems have equal size.

Recurrence Format

1. Base Case : T(n) <= a constant for all sufficiently small n
2. For all larger n :

$$T(n) \leq aT(n/b) + O(n^d)$$

where

a = number of recursive calls (>= 1)

b = input size shrinkage factor (> 1)

d = exponent in running time of "combine step" (>=0)

[a,b,d independent of n]

The Master Method

-

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \quad \text{(Case 1)} \\ O(n^d) & \text{if } a < b^d \quad \text{(Case 2)} \\ O(n^{\log_b a}) & \text{if } a > b^d \quad \text{(Case 3)} \end{cases}$$

Base doesn't matter (only changes leading constants)

Base matters

Master Method

Examples

Design and Analysis of Algorithms I

The Master Method

If $T(n) \leq aT\left(\frac{n}{b}\right) + O(n^d)$

then

$$T(n) = \begin{cases} O\left(n^d \log n\right) & \text{if } a = b^d \quad \text{(Case 1)} \\ O(n^d) & \text{if } a < b^d \quad \text{(Case 2)} \\ O(n^{\log_b a}) & \text{if } a > b^d \quad \text{(Case 3)} \end{cases}$$

Example #1

Merge Sort

a = 2
b = 2
d = 1

$$b^d = a \Rightarrow Case\ 1$$

$$T(n) = O(n^d \log n) = O(n \log n)$$

Where are the respective values of a, b, d for a binary search of a sorted array, and which case of the Master Method does this correspond to?

- ○ 1, 2, 0 [Case 1] $a = b^d => T(n) = O(n^d \log n) = O(\log n)$
- ○ 1, 2, 1 [Case 2]
- ○ 2, 2, 0 [Case 3]
- ○ 2, 2, 1 [Case 1]

Example #3

Integer Multiplication Algorithm # 1

a = 4
b = 2
d = 1

$$b^d = 2 < a \ \ (Case \ \ 3)$$

$$=> T(n) = O(n^{\log_b a}) = O(n^{\log_2 4})$$

$$= O(n^2)$$

Same as grade-school algorithm

Where are the respective values of a, b, d for Gauss's recursive integer multiplication algorithm, and which case of the Master Method does this correspond to?

- ◯ 2, 2, 1 [Case 1]
- ◯ 3, 2, 1 [Case 1]
- ◯ 3, 2, 1 [Case 2]
- ◯ 3, 2, 1 [Case 3]

Better than the grade-school algorithm!!!

$$a = 3,\ b^d = 2\ a > b^d\ \ (Case\ 3)$$

$$=> T(n) = O(n^{\log_2 3}) = O(n^{1.59})$$

Example #5

Strassen's Matrix Multiplication Algorithm

a = 7

b = 2

d = 2

$$b^d = 4 < a \quad (Case\ 3)$$

$$=> T(n) = O(n^{\log_2 7}) = O(n^{2.81})$$

=> beats the naïve iterative algorithm !

Example #6

Fictitious Recurrence

$$T(n) \leq 2T(n/2) + O(n^2)$$

$\Rightarrow$ a = 2

$\Rightarrow$ b = 2

$\Rightarrow$ d = 2

$$b^d = 4 > a \quad (Case\ 2)$$

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