The master theorem

The master method depends on the following theorem.

Theorem 4.1 (Master theorem)
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ 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 $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Strassen's mm

$T(n) = 7\,T(n/2) + \Theta(n^2)$

Ex 1

$a = 7$ ←

$b = 2$ ← division

$\Theta(n^{2.8})$

LHS

RHS

$n \to O(n^{\log_2 7 - \varepsilon})$

$2.8 - 0.8$

ε

fraction

$T(n) = 8T(n/2) + \Theta(n^2)$

Ex:2

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

a 8 | b 2 | f(n) $\Theta(n^2)$

Apply Case 1

$f(n) = O(n^{\log_b a - \varepsilon})$, $\varepsilon > 0$

LHS $\Theta(n^2)$ = RHS $n^{\log_2 2^3 - \varepsilon}$

$$\theta(n^2) = n^{3-\varepsilon}$$

$$\varepsilon \Rightarrow 1$$

$$\theta(n^2) = n^2$$ ∴ Case 1 applies

LHS = RHS

Therefore

solution $T(n) = \theta(n^{\log_b a})$

$$\theta(n^{\log_2 8}) \Rightarrow \theta(n^3)$$

Ex.3

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

$a \quad 2 \quad b \quad 2 \quad 1 \quad f(n) \quad \Theta(n)$

(we test)

$f(n) = O(n^{\log_b a - \varepsilon})$

LHS $\Theta(n)$

$n^{\log_2 2 - \varepsilon}$ RHS

$n^{1-\varepsilon}$

n

L.H.S

n

R.H.S

only for $\varepsilon > 0$

But $\varepsilon = 0$ not allowed.

Case 2

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

LHS $= n$

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

$\Theta(n^1)$

RHS

$=$ Therefore, we decide that test case 2 is matching

Soln is

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

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

$$\Theta(n \log n)$$

Maximum Subarray Problem

With respect to stocks scenario described we want to find a sequence of days over which the net change from the first day to the last day is maximum.

Instead of looking at the daily prices, let us consider daily change in price.

Day	0	1	2	3	4
Price	10	11	7	10	6
Change		1	-4	3	-4

Buy after day 2 & sell after day 3 to earn max profit, $3

Code version 1
does not use accumulated sum
from previous subarray. Hence, one
more loop.
Code version 2
Uses accumulated summation
for next subarray. Hence, we show that only
for within for loop only.
know that thus algorithms can be
improvised.

Assignment:

Apply divide and conquer to finding maximum subarray using DAC

Solve the recurrence using master theorem.

---

note that case 3, regularity requirement as well as gap between cases, the limitation of master theorem yet to discuss.