Math Week 10
Master Theorem

Master Theorem → Is a shortcut which tells you a time complexity of divide and conquer algorithms.

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

Merge Sort → $T(n) = 2 \cdot T(n/2) + n$

Level 0: $f(n)$
Level 1: $a \times f(n/b)$
Level 2: $a^2 \times f(n/b^2)$
Level k: $a^k \times f(n/b^k)$

"Work per level"

Total cost = sum of work per level

Each case of the theorem tells you which levels matter the most:

[How f(n) compares is n^log_b(a) ?]

1. noticeably smaller → work at the last level is dominant → n^log_b(a)
2. about the same → work at every level is equal → n^log_b(a) * log(n)
3. noticeably larger → work at top level (level 0) is dominant → f(n)

$T(n) = 2 \times T(n/2) + n$
$a = 2$; $b = 2$; $f(n) = n$

f(n) vs n^log_b(a)
n vs n

ex. $T(n) = \Theta(n^2)$

ex: $T(n) = 4 \times T(n/2) + n$
$a = 4$; $b = 2$; $f(n) = n$

f(n) vs n^log_b(a)
n vs n^log_2(4)
n vs n^2
$T(n) = \Theta(n^2)$

[How f(n) compares against n^log_b(a)?]
[How *merge-time-at-top-level* compares against *number-of-merges-at-the-last-level*?

Example with cupcakes and sprinkles

1. Sprinkles are sparse

- You hardly use any sprinkles compared with the amount of decorating your friends do.

→ Most time is spent at the very bottom when each tiny cupcake finally gets its icing.

2. Sprinkles match decorating

- The sprinkle time each round is about equal to the decorating time.

→ Every round contributes equally, so total time is "one round times number of rounds" = log n rounds

3. Sprinkles are lavish

- You pour on so many sprinkles up front that the helpers' work is dwarfed.

→ Your initial sprinkle party dominates the total time.

Using ϵ, we are able to distinguish between:

1. lavish → "at least a little more sprinkle every time the cupcakes double in number"
2. sparse → "at least a little less".