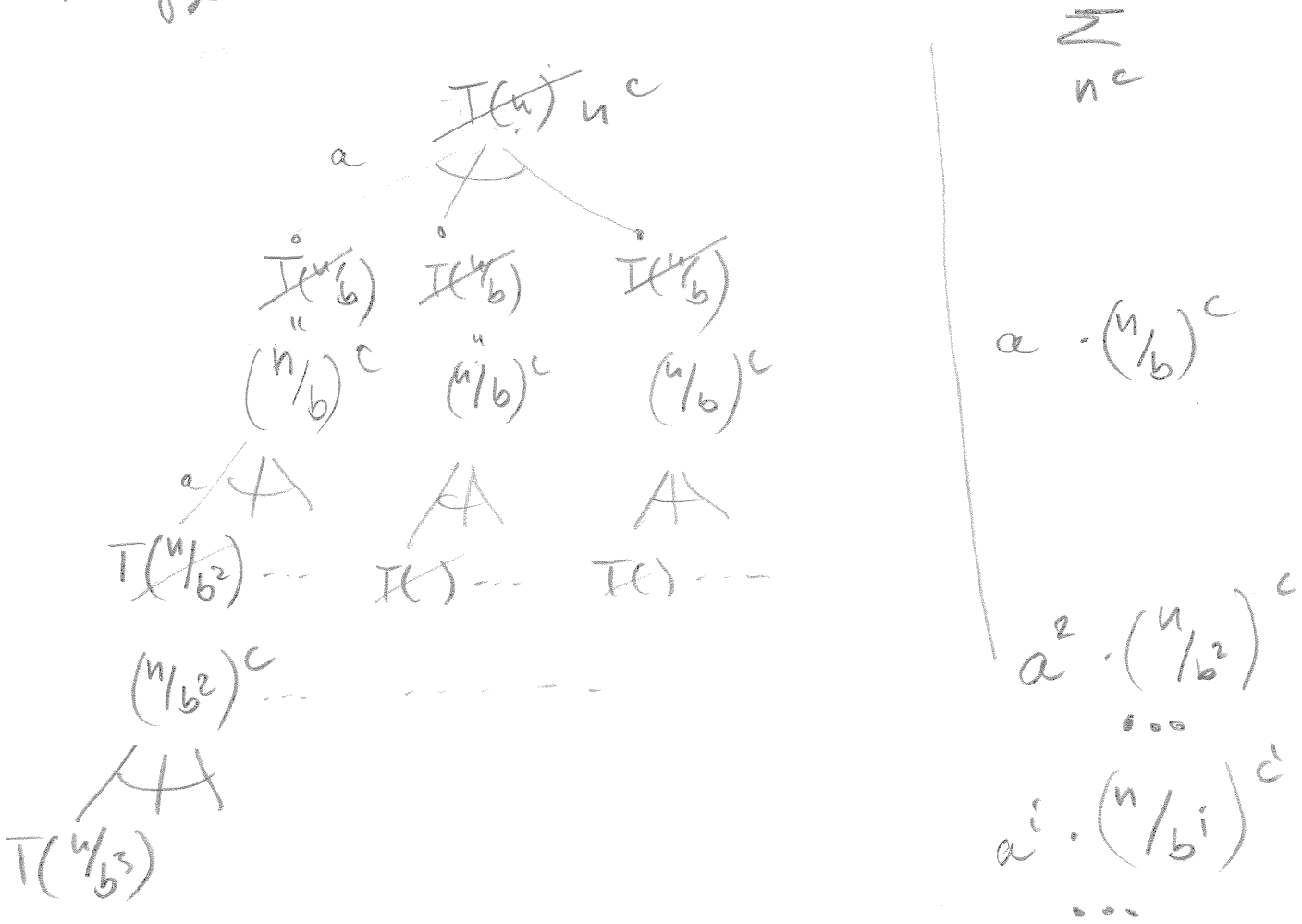
How can we solve recurrences

$$T(n) = a \cdot T(n/b) + n^c \quad ?$$

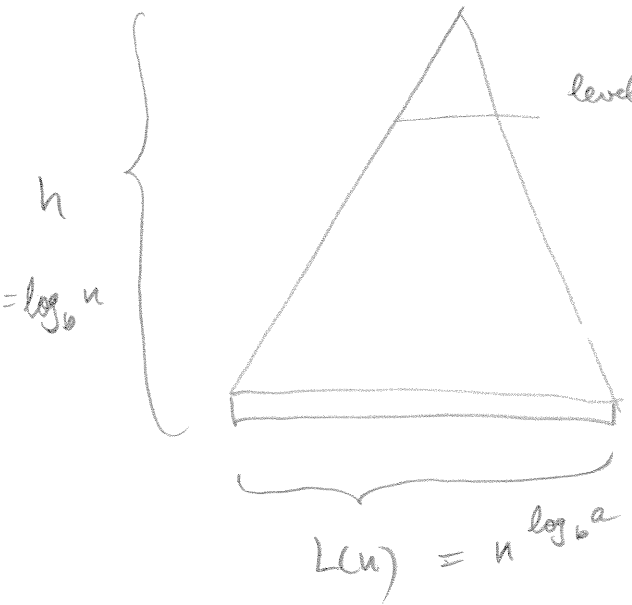
Answered by **Master Theorem**

Analyze Recursion tree $(n = b^k)$



h = # levels = $k + 1 = 1 + \log_b n$

$L(n)$ = # leaves = $a^k = a^{\log_b n} = n^{\log_b a}$



level i:
- #nodes $= a^i$
- cost per node $= \left(\frac{n}{b^i}\right)^c = n^c \left(\frac{1}{b^i}\right)^c$
- total cost $= n^c \cdot \frac{a^i}{(b^c)^i} = n^c \left(\frac{a}{b^c}\right)^i$

Total cost of inner nodes

$$S(n) = n^c \sum_{i=0}^{h-1} \left(\frac{a}{b^c}\right)^i$$

Case 1: $\frac{a}{b^c} = 1 \iff a = b^c \iff c = \log_b a$

$$S(n) = n^c \sum_{i=0}^{h-1} 1 = n^c \cdot h = n^c \cdot \log_b(n)$$

$$\Rightarrow T(n) = S(n) + \Theta(1) \cdot L(n)$$
$$= n^c \cdot \log_b n + n^c = \Theta(n^c \cdot \log_b n)$$

Merge Sort: $T(n) = 2\,T(n/2) + \Theta(n)$

$a = 2, b = 2, c = 1$ $\Rightarrow$ Case 1

$\log_2 2 = 1$

$$\Rightarrow T(n) = \Theta(n \cdot \log_2 n)$$

Observe: $\frac{a}{b^c} \neq 1 \iff a \neq b^c$

$$S(n) = n^c \sum_{i=0}^{h-1} \left(\frac{a}{b^c}\right)^i$$

$$= n^c \frac{1 - \left(\frac{a}{b^c}\right)^h}{1 - \left(\frac{a}{b^c}\right)}$$

$D := \left(1 - \left(\frac{a}{b^c}\right)\right)$ $\quad D > 0 \iff a < b^c \iff \log_b a < c$

$D < 0 \iff a > b^c \iff \log_b a > c$

$$\left(\frac{a}{b^c}\right)^h = \frac{a^h}{(b^c)^h} \overset{h = \log_b n}{=} \frac{a^{\log_b n}}{\left(b^{\log_b n}\right)^c}$$

$$= \frac{n^{\log_b n}}{n^c}$$

$$S(n) = \frac{1}{D} n^c \left(1 - \frac{n^{\log_b n}}{n^c}\right)$$

$$= \frac{1}{D} \underbrace{\left(n^c - n^{\log_b a}\right)}_{\text{who wins?}}$$

$=$ ...

$$= \begin{cases} \frac{b^c}{b^c - a}\left(n^c - n^{\log_b a}\right) & c > \log_b a \\ \frac{b^c}{a - b^c}\left(n^{\log_b a} - n^c\right) & \log_b a > c \end{cases}$$

Intuition

$c \triangleq$ growth rate of division and combination work

$\log_b a \triangleq$ growth rate of subproblems

$$\Rightarrow \quad S(n) = \begin{cases} \Theta(n^c) & c > \log_b a \\ \Theta(n^{\log_b a}) & c < \log_b a \end{cases}$$

Remember: $L(n) = n^{\log_b a}$

Summary ($T(n) = S(n) + L(n) \cdot \Theta(1)$):

$$T(n) = \begin{cases} \Theta(n^c) & \text{if } c > \log_b a \\ \Theta(n^c \cdot \lg n) & \text{if } c = \log_b a \\ \Theta(n^{\log_b a}) & \text{if } c < \log_b a \end{cases}$$

Master Theorem

Let $a \geq 1$, $b > 1$, $c \geq 0$

Let $T(n)$ be a function satisfying the recurrence

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ a \cdot T(n/b) + n^c & n > 1 \end{cases}$$

Then

- $T(n) = \Theta(n^c)$ if $c > \log_b a$
- $T(n) = \Theta(n^c \cdot \lg n)$ if $c = \log_b a$
- $T(n) = \Theta(n^{\log_b a})$ if $c < \log_b a$

Intuition

Case 1: work concentrated at top of tree

Case 2: work evenly distributed over tree

Case 3: work concentrated at bottom of tree

Application

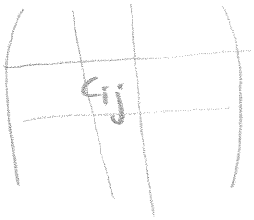
Algorithm	a	b	c	$\log_b a$	$T(n)$
Mergesort	2	2	1	1	$\Theta(n \cdot \lg n)$
Binary search, Powering	1	2	0	0	$\Theta(1 \cdot \lg n) = \Theta(\lg n)$
Smart Multiplication	3	2	1	1.58..	$\Theta(n^{1.58..})$
Tromino tiling	4	2	0	2	$\Theta(n^2)$

Matrix Multiplication

$n \times n$ matrices

$$A = (a_{ij})_{i,j=1}^{n} \qquad B = (b_{ij})_{i,j} \qquad C = (c_{ij})$$

$$C = A \cdot B \quad \text{iff} \quad c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$



scalar product of
- i-th row of A
- j-th column of B

Running time
- n^2 elements in C
- n multiplications, n additions per element

```
for i := 1 to n
    for j := 1 to n
    C[i][j] := 0
    for k := 1 to n
        C[i][j] := C[i][j] + A[i][k] * B[k][j]
```

$\Rightarrow T(n) = \Theta(n^3)$

Matrix Multiplication : Divide and Conquer

$n \times n$ matrix = 2×2 block matrix
of $n/2 \times n/2$ submatrices

$$C = \left(\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right) = \left(\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right) \cdot \left(\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right)$$

A B

$A \cdot B = C$ if

$$\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}$$

Idea: recursively multiply 8 submatrices of size $n/2$
4 additions of cost $\Theta(n^2)$
as matrices have $(n/2)^2$ elements

Analysis: $a = 8$, $b = 2$, $c = 2$

$\Rightarrow \log_b a = \log_2 8 = 3 > 2 = c$

$\Rightarrow T(n) = \Theta(n^3)$ ☹

Where could we gain? $b = 3 \Rightarrow a = 9 \cdot 3 = 3^3$

Can we reduce a?

Strassen (1969)

Compute 7 products

$$P_1 = a \cdot (f - h)$$
$$P_2 = (a + b) \cdot h$$
$$P_3 = (c + d) \cdot e$$
$$P_4 = d \cdot (g - e)$$
$$P_5 = (a + d) \cdot (e + h)$$
$$P_6 = (b - d) \cdot (g + h)$$
$$P_7 = (a - c) \cdot (e + f)$$

Submatrices

$$r = P_4 + P_5 + P_6 - P_2$$
$$s = P_1 + P_2$$
$$t = P_3 + P_4$$
$$u = P_1 + P_5 - P_3 - P_7$$

(4)

Analysis:

- 18 additions/subtractions
- 7 multiplications

$$T(n) = 7\,T(n/2) + \Theta(n^2)$$
$$= \Theta(n^{\log_2 7}) = \Theta(n^{2.81})$$

In practice, faster than naive multiplication for $n \geq 12$.

Best algorithm so far $\Theta(n^{2.3})$

9a

(*)

Algorithm

1) Divide A, B into $a, \ldots, h$,
   compute terms for 7 products

2) Conquer
   compute 7 products

3) Combine
   compute r, s, t, u from $p_1, \ldots, p_7$