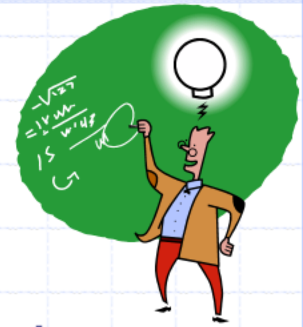
The general recurrence form for divide & conquer methods

$$T(n) = \begin{cases} \text{time for solving trivial problem} & n \leq d \\ \underbrace{\#\text{subproblems}}_{a} \times T\left(\dfrac{n}{\underbrace{\text{factor reduction}}_{b}}\right) \; \underbrace{\boxed{\begin{array}{l} + \text{ divide time} \\ + \text{ conquer time} \end{array}}}_{f(n)} & n > d \end{cases}$$

Need not be equal (see integer mult for eg)

Master Method (Appendix)



- Many divide-and-conquer recurrence equations have the form:

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

Includes divide & conquer for current stage

- The Master Theorem:

Merge sort

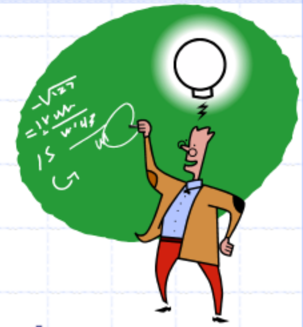
1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$

last example

2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$

3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

Master Method (Appendix)



- Many divide-and-conquer recurrence equations have the form:

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

Includes divide & conquer for current stage

Intuition: Iterative substitution

$$T(n) = aT(n/b) + f(n) = a^2 T(n/b^2) + a f(n/b) + f(n)$$

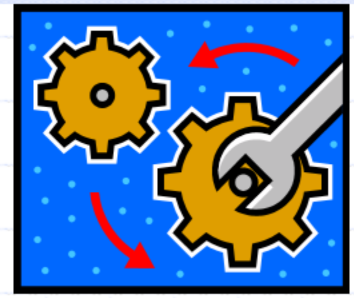
$$= a^3 T(n/b^3) + a^2 f(n/b^2) + a f(n/b) + f(n)$$

$$\cdots = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + a^{\log_b n} T(1)$$

$$n^{\log_b a}$$

Note: Let $a = b^k$, $n = b^\ell$, Then $a^{\log_b n} = a^\ell = (b^k)^\ell = (b^\ell)^k = n^k = n^{\log_b a}$

Iterative "Proof" of the Master Theorem

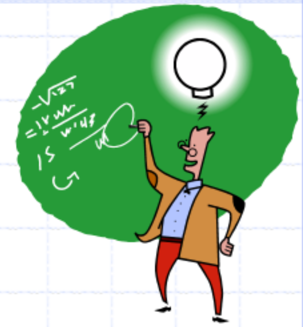


- Using iterative substitution, let us see if we can find a pattern:

$$
\begin{aligned}
T(n) &= aT(n/b) + f(n) \\
&= a(aT(n/b^2)) + f(n/b)) + bn \\
&= a^2T(n/b^2) + af(n/b) + f(n) \\
&= a^3T(n/b^3) + a^2 f(n/b^2) + af(n/b) + f(n) \\
&= \ldots \\
&= a^{\log_b n}T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i) \\
&= n^{\log_b a}T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i)
\end{aligned}
$$

- We then distinguish the three cases as
  - The first term is dominant
  - Each part of the summation is equally dominant
  - The summation is a geometric series → (i.e second term dominates)

Master Method, Example 1



- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$
- The Master Theorem: would like ϵ to be as large as possible

  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

  $\epsilon > 0$

- Example: $$T(n) = 4T(n/2) + n$$ Integer multiplication

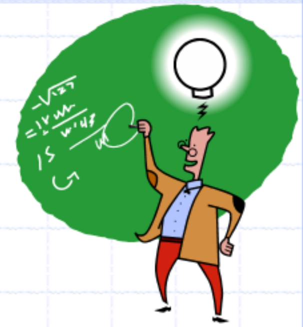
Solution: $\boxed{\log_b a = 2,}$ so case 1 says T(n) is $O(n^2)$.

Look at $\log_b a$ & $f(n)$

(2) $b=2$, $a=4$ Does $\exists\, k$ s.t $n = \Theta(n^2 \log^k n)$? No

(1) $b=2$, $a=4$, does $\exists\, \epsilon$ s.t $n = O(n^{2-\epsilon})$? Yes: $\epsilon = 1$

Master Method, Example 2



- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$$
- The Master Theorem:

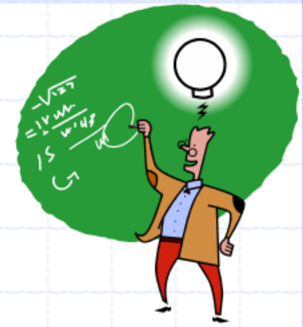
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
     provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.
- Example:

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

Solution: $\log_b a = 1$, so case 2 says T(n) is $O(n \log^2 n)$.

① nlogn can never be $O(n^{1-\varepsilon})$
② nlogn is $\Theta(n \log^k n)$ for $k=1$

Master Method, Example 3



- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$
- The Master Theorem:
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.
- Example:

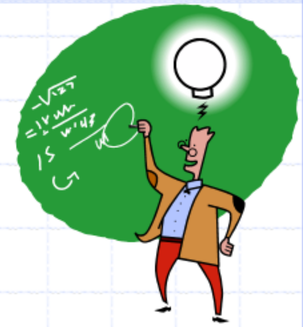
$$T(n) = T(n/3) + n \log n$$

Solution: $\log_b a = 0$, so case 3 says T(n) is O(n log n).

3 Is $n\log n = \Omega(n^{\epsilon})$ for some $\epsilon > 0$?
Ans: Yes $\epsilon = 1$

H/W

Master Method, Example 4

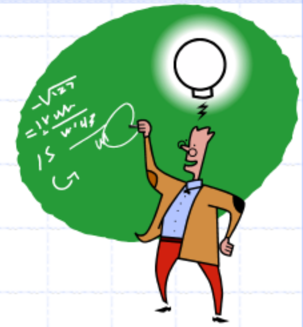


- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \ge d \end{cases}$$
- The Master Theorem:
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.
- Example:

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

Solution: $\log_b a = 3$, so case 1 says T(n) is $O(n^3)$.

Master Method, Example 5



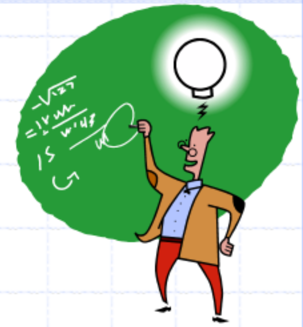
- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$
- The Master Theorem:
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,

     provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.
- Example:

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

Solution: $\log_b a=2$, so case 3 says T(n) is O(n^3).

Master Method, Example 6

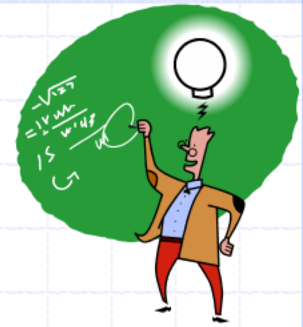


- The form: $$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$
- The Master Theorem:
  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
     provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.
- Example:

$$T(n) = T(n/2) + 1 \quad \text{(binary search)}$$

Solution: $\log_b a = 0$, so case 2 says $T(n)$ is $O(\log n)$.

Master Method, Example 7



- The form: $T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$
- The Master Theorem:

  1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$ $1 > \varepsilon > 0$
  2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
  3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
     provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- Example:

$$T(n) = 2T(n/2) + \log n$$ (heap construction)

bottom up

Solution: $\log_b a=1$, so case 1 says T(n) is O(n).

Integer Multiplication

- Algorithm: Multiply two n-bit integers I and J.
  - Divide step: Split I and J into high-order and low-order bits

$$I = I_h 2^{n/2} + I_l$$

$$J = J_h 2^{n/2} + J_l$$

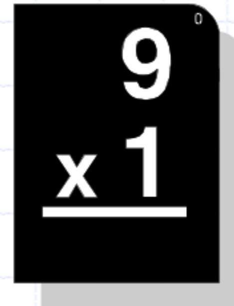
  - We can then define I*J by multiplying the parts and adding:

$$\begin{aligned} I * J &= (I_h 2^{n/2} + I_l) * (J_h 2^{n/2} + J_l) \\ &= I_h J_h 2^n + I_h J_l 2^{n/2} + I_l J_h 2^{n/2} + I_l J_l \end{aligned}$$

  - So, $T(n) = 4T(n/2) + n$, which implies $T(n)$ is $O(n^2)$.
  - But that is no better than the algorithm we learned in grade school.

$\log_b a = \log_2 4 = 2$, $f(n) = n$ is $O(n^{2-\epsilon})$ for $\epsilon = 1 \Rightarrow T(n) = \Theta(n^2)$

An Improved Integer Multiplication Algorithm



- Algorithm: Multiply two n-bit integers I and J.
  - Divide step: Split I and J into high-order and low-order bits

$$I = I_h 2^{n/2} + I_l$$

$$J = J_h 2^{n/2} + J_l$$

  - Observe that there is a different way to multiply parts:

$$I * J = I_h J_h 2^n + [(I_h - I_l)(J_l - J_h) + I_h J_h + I_l J_l]2^{n/2} + I_l J_l$$

$$= I_h J_h 2^n + [(I_h J_l - I_l J_l - I_h J_h + I_l J_h) + I_h J_h + I_l J_l]2^{n/2} + I_l J_l$$

$$= I_h J_h 2^n + (I_h J_l + I_l J_h)2^{n/2} + I_l J_l$$

  - So, $T(n) = 3T(n/2) + n$, which implies $T(n)$ is $O(n^{\log_2 3})$, by the Master Theorem.
  - Thus, $T(n)$ is $O(n^{1.585})$.

$\log_b a = \log_2 3$ & $f(n) = n = O(n^{\log_2 3 - \epsilon})$ for $\epsilon \leq \log_2 3 - 1 \ldots T(n) = \Theta(n^{\log_2 3})$