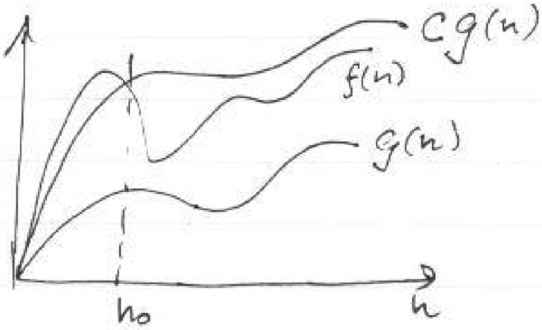
O-notation

* 

$f(n) = O(g(n))$

* $n = O(n)$ $\quad n \le 1 \cdot n \quad$ for all $n \geq 1$

$n = O(n^2)$ $\quad n \le 1 \cdot n^2 \quad n \geq 1$

$n + \sqrt{n} = O(n)$ $\quad n + \sqrt{n} \le 2n \quad n \geq 1$

* $\log_3 n = O(\log_2 n)$ $\quad \log_3 n \le \log_2 n$

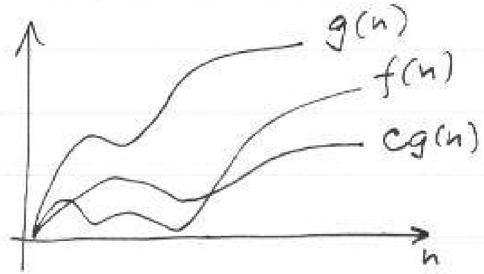
$\log_2 n = O(\log_3 n)$ $\quad \log_2 n \le 2 \log_3 n$ $\quad \left\{ \begin{array}{l} x = 2^{\log_2 x} \\ \log_3 x = \log_2 x \end{array} \right.$

$O(\log_a n) = O(\log_b n)$ $\Rightarrow$ never write basis always only $O(\log n)$

$\log n = O(n^\alpha)$ $\quad$ for all $\alpha > 0$

$$\lim_{x \to \infty} \frac{\log x}{x^\alpha} = \lim \frac{1/x}{x^{\alpha-1}} = 0$$

Ω- notation



* $n^2 = \Omega(n)$
* $n = \Omega(n)$
* $n^2 + n = \Omega(n^2)$

Θ - notation $\quad O(g(n)) \cap \Omega(g(n))$

* $n^2 + n + 1 = \Theta(n^2)$

Properties

1) transitivity (O, Ω, Θ) $\quad f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

2) symmetry $\quad f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$

- looks like ordering, but not total

e.g. $f(n) = n \quad g(n) = n^{1+\sin n} \quad f(n) \neq O(g(n))$
$g(n) \neq O(f(n))$

Abuses of Notation

$$* \quad n^2 + O(n) = O(n^2)$$

$$\forall f(n) \in \Theta(n), \quad n^2 + f(n) \in O(n^2)$$

$$* \quad \sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$$

$$\forall f(i) \in \Theta(i), \quad \sum_{i=1}^{n} f(i) = \Theta(n^2)$$

$$c_1 i \le f(i) \le c_2 i \qquad i \ge i_0$$

$$\sum c_1 i \le \sum f(i) \le \sum c_2 i$$

$$c_1 \frac{n(n+1)}{2} \le \sum f(i) \le c_2 \frac{n(n+1)}{2}$$

Algoritm Analysis

$T(n)$ is the runtime of an algorithm applied to input of size n.

$$T(n) = a \cdot T(n/b) + f(n)$$ divide and conquer

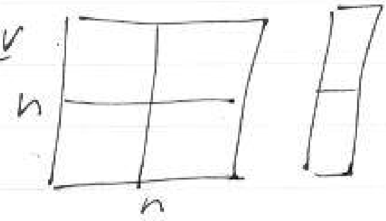
$$= \begin{cases} \Theta(n^{\log_b a}) & f(n) = O(n^{\log_b a - \varepsilon}), \varepsilon > 0 \\ \Theta(n^{\log_b a} \log n) & f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)) & f(n) = \Omega(n^{\log_b a + \varepsilon}) \end{cases}$$

(when recursed all the way)

- it is correct for $\lfloor \frac{n}{b} \rfloor$ or $\lceil \frac{n}{b} \rceil$

Examples:

1. M.V



* generic $\Theta(n^2)$

* divide & conquer

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) \qquad a=4 \quad b=2$$

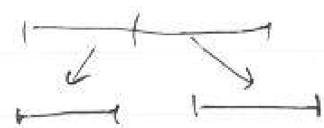
$$\log_2 4 = 2$$

$\Rightarrow$ first case

$\Rightarrow T(n) = \Theta(n^2)$ didn't win anything

2. Mergesort:

- sort positive integers



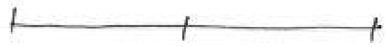
$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

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

$a = b = 2 \quad \log_2 2 = 1$
(second case)

complexity of sorting is $\Theta(n \log n)$

3. Find an element in a sorted list



- cut in the middle and compare
- if this is it $\Omega(1)$
  best case

- worst case

$$T(n) = T\left(\frac{n}{2}\right) + O(1) \quad (\cancel{\text{first}}\ \text{second case}) \quad a=1,\ b=2 \quad \log_2 1 = 0$$

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

4. Multiplying two polynomials

$$p(x) = a_n x^n + \dots + a_0$$
$$q(x) = b_n x^n + \dots + b_0$$

$$p(x)q(x) = O(n^2)$$

$4\mu, 1\alpha$

Karatsuba algorithm

e.g. $$(a+bx)(c+dx) = ac + (ad+bc)x + bd\,x^2$$
$$= ac + ((a+b)(c+d) - ac - bd)x + bd\,x^2$$

$3\mu, 4\alpha$

---

n even: $$p(x) = p_0(x^2) + x\,p_1(x^2)$$
$$q(x) = q_0(x^2) + x\,q_1(x^2)$$

$$\Rightarrow T(n) = 3 \cdot T\left(\frac{n}{2}\right) + \Theta(n) \rightarrow$$ additions of 4 polynomials
$$= \Theta(n^{\log_2 3})$$

we got rid of 1 expensive poly multiplication even though we paid by 3 extra poly additions