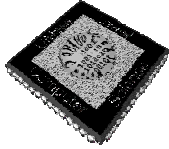
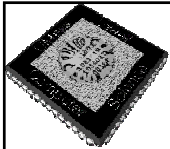


Asymptotic Notations



Text
Chapters 3



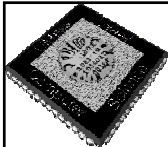
Asymptotic Notation

- What does “the order of” mean
- Big O, Ω , and Θ notations
- Properties of asymptotic notation
- Limit rule



A notation for “the order of”

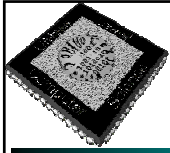
- We’d like to measure the efficiency of an algorithm
 - Determine mathematically the resources needed
- There is no such a computer which we can refer to as a standard to measure computing time
- We introduce “asymptotic” notation
 - An asymptotically superior algorithm is often preferable even on instances of moderate size



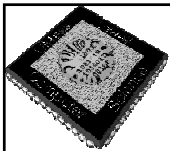
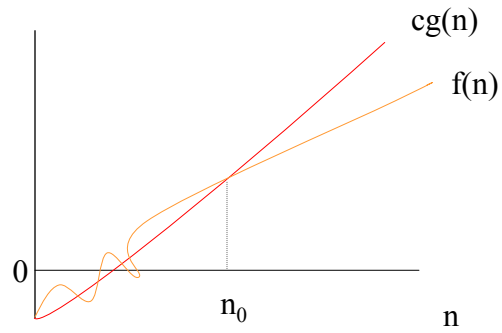
Definition of big O

$$O(g(n)) = \{f(n) \mid (\exists c \in R^+, n_0 \in N)(\forall n \geq n_0)[0 \leq f(n) \leq cg(n)]\}$$

- Typically used for *asymptotic upper bound*
- Attention
 - $O(f(n))$ is a **set** of functions
- Pitfall
 - Traditionally we say $n^2 = O(n^2)$ as used in our text book
 - It really means $n^2 \in O(n^2)$



A graphical view of asymptotic definition



Example

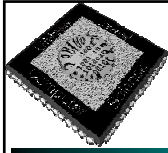
□ Prove that following statements

$$13n^2 + n + 5 \in O(n^2)$$

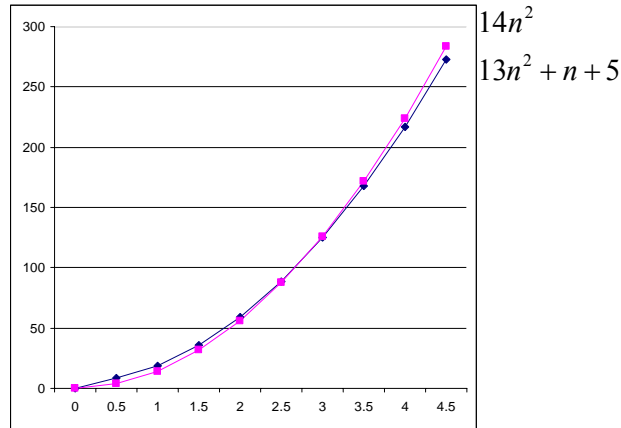
$$13n^2 + n + 5 \in O(n^2 \log n)$$

$$f(n) \in O(n) \rightarrow f^2(n) \in O(n^2)$$

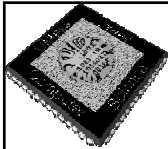
$$O(n) \subset O(n^2)$$



$$13n^2 + n + 5 \in O(n^2) \quad \forall n \geq 3, \quad 13n^2 + n + 5 \leq 14n^2$$



What are c and n_0 ?

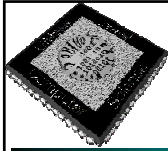


Several notations

- Logarithm time $O(\log n)$
- Linear time $O(n)$
- Quadratic time $O(n^2)$
- Cubic time $O(n^3)$
- Exponential time $O(c^n)$, $c > 1$

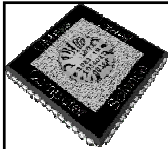
$$O(\lg n) \subset O(n^\epsilon) \subset O(n^\epsilon \lg n) \subset O(n^{c+\epsilon} \lg n) \subset O(d^n) \quad c, \epsilon > 0, d > 1$$

- Order of growth



The Maximum rule

- Let $f, g : N \rightarrow R^{\geq 0}$,
then $O(f(n) + g(n)) = O(\max(f(n), g(n)))$
- Proof
 - the key is $\max(f(n), g(n)) \leq f(n) + g(n) \leq 2 * \max(f(n), g(n))$
- Examples
 - $O(12n^3 - 5n + n \log n + 36)$
- The maximum rule let us ignore lower-order terms



Example

- True or false
 - ? $5 = O(\log n)$
 - ? $\log n = O(5)$
 - ? $n = O(n^{0.6} \log n)$
 - ? $n^{0.6} \log n = O(n)$
 - ? $n^8 = O((n^2 - 3n + 5)^4)$



Definition of Ω

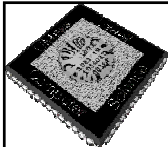
$$\Omega(g(n)) = \{f(n) \mid (\exists c \in R^+, n_0 \in N)(\forall n \geq n_0)[f(n) \geq cg(n) \geq 0]\}$$

□ Ω is typically used to describe *asymptotic lower bound*

- For example, insertion sort take time in $\Omega(n)$

□ Ω for algorithm complexity

- We use it to give the lower bounds on the intrinsic difficulty of solving problems
- Example, any comparison-based sorting algorithm takes time $\Omega(n \log n)$



Example of $\Omega(n^2)$

- n^2
- n^2+n
- n^2-n
- $n^{2.0001}$
- $n^2 \log n$
- 2^n



The Θ notation

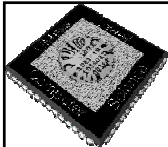
Definition:

$$\Theta(g(n)) = \{f(n) \mid (\exists c_1, c_2 \in R^+, n_0 \in N)(\forall n \geq n_0)[0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)]\}$$

Equivalent to:

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

- Used to describe *asymptotically tight bound*
- Example: selection sort take time in $\Theta(n^2)$



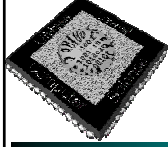
The Limit Rule

□ Let $f, g : N \rightarrow R^{\geq 0}$, then

1. If $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \in R^+$ then $f(n) \in \Theta(g(n))$

2. If $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$ then $f(n) \in O(g(n))$ but $f(n) \notin \Omega(g(n))$

3. If $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = +\infty$ then $f(n) \in \Omega(g(n))$ but $f(n) \notin O(g(n))$



Example

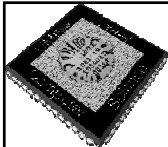
$$(n^c)' = cn^{c-1}$$

$$(\ln n)' = \frac{1}{n} \quad (\ln n \text{ means } \log_e n, \text{ the text use } \log)$$

When $c > 0$

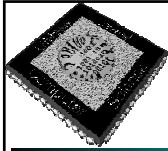
$$\lim_{n \rightarrow \infty} \frac{\ln n}{n^c} = \lim_{n \rightarrow \infty} \frac{(\ln n)'}{(n^c)'} = \lim_{n \rightarrow \infty} \frac{1/n}{cn^{c-1}} = \lim_{n \rightarrow \infty} \frac{1}{cn^c} = 0$$

$$\ln n \in O(n^c) \quad \text{for any } c > 0$$



Semantics of big-O and Ω

- When we say an algorithm takes worst-case time $t(n) = O(f(n))$, then there exist a real constant c such that $c * f(n)$ is an upper bound for any instances of size of sufficiently large n
- When we say an algorithm takes worst-case time $t(n) = \Omega(f(n))$, then there exist a real constant d such that there exists at least one instance of size n whose execution time $\geq d * f(n)$, for any sufficiently large n
- Example
 - Is it possible an algorithm takes worst-case time $O(n)$ and $\Omega(n \log n)$?



Practice Problems

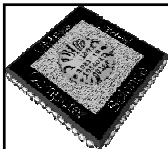
True or false

```

anAlgorithm( int n)
{
  // if (x) is an elementary
  // operation
  if (x) {
    some work done
    by n2 elementary
    operations;
  } else {
    some work done
    by n3 elementary
    operations;
  }
}

```

- The algorithm takes time in $O(n^2)$
- The algorithm takes time in $\Omega(n^2)$
- The algorithm takes time in $O(n^3)$
- The algorithm takes time in $\Omega(n^3)$
- The algorithm takes time in $\Theta(n^3)$
- The algorithm takes time in $\Theta(n^2)$
- The algorithm takes worst case time in $O(n^3)$
- The algorithm takes worst case time in $\Omega(n^3)$
- The algorithm takes worst case time in $\Theta(n^3)$
- The algorithm takes best case time in $\Omega(n^3)$



Definition of o and ω

Definition

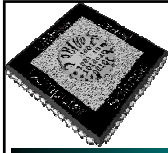
$$o(g(n)) = \{f(n) \mid (\forall c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0)[0 \leq f(n) < cg(n)]\}$$

$$\omega(g(n)) = \{f(n) \mid (\forall c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0)[f(n) > cg(n) \geq 0]\}$$

Denote upper/lower bounds that are not asymptotically tight

Example $1000n \in o(n^2)$; $1000n^2 \notin o(n^2)$
 $1000n^2 \in \omega(n)$; $1000n^2 \notin \omega(n^2)$

Properties $f(n) \in o(g(n)) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$
 $f(n) \in \omega(g(n)) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$



Relational Properties

- ❑ Transitivity: $O, o, \Omega, \omega, \Theta$
- ❑ Reflexivity: O, Ω, Θ
- ❑ Symmetry: $f(n) = \Theta(g(n)) \Leftrightarrow g(n) \in \Theta(f(n))$
- ❑ Transpose symmetry (Duality)
 - $f(n) = O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$
 - $f(n) = o(g(n)) \Leftrightarrow g(n) \in \omega(f(n))$
- ❑ Analogy
 - $f(n) \in O(g(n)) \approx a \leq b$
 - $f(n) \in \Omega(g(n)) \approx a \geq b$
 - $f(n) \in \Theta(g(n)) \approx a = b$
 - $f(n) \in o(g(n)) \approx a < b$
 - $f(n) \in \omega(g(n)) \approx a > b$