

CSE 21: Math for Algorithm & System Analysis Summer II 2006

Lecture 10

Wenjing Rao
wrao@cs.ucsd.edu

Review

- Solving recursions
 - Master Method
 - Works for recurrences of the form
 - $T(n) = a T(n / b) + f(n)$
- 3 general cases: (in the long run):
 - Compare $f(n)$ with $n^{\log_b a}$
 - $f(n)$ grows faster
 - $f(n)$ grows slower
 - $f(n)$ grows proportional

Today's Topic

- Performance of an algorithm – growth of a function
- O , Ω , and Θ notation
 - Upper bound
 - Lower bound
 - Tight bound

Performance

- What's more important than performance?
- Why do we care about performance

How to Measure Performance

- Classic problem: sorting
 - Input: sequence of numbers:
 - $(a_1, a_2, a_3, \dots, a_n)$
 - Output: sorted permutation
 - $(a'_1, a'_2, \dots, a'_n)$, such that
 - $a'_1 \leq a'_2 \leq \dots \leq a'_n$
 - Example:
 - Input: 8 2 4 9 3 6
 - Output: 2 3 4 6 8 9

One algorithm to sort

- Insertion sort
- 8 2 4 9 3 6

Another algorithm to sort

- Merge sort

Which one is better in performance?

- Insertion vs Merge ?
- Performance depends on

Kinds of analyses

- **Worst-case** (usually)
- **Average-case** (sometimes)
- **Best-case** (bogus)

Machine-independent time

- What is insertion sort's worst-case time?
 - Depends on the speed of the computer
- Big idea:
 - Ignore the machine-dependent constants
 - Look at the growth of $T(n)$ as $n \rightarrow \infty$

“Asymptotic Analysis”

Determine the performance

- Typically a function on input size: $T(n)$
 - Rarely a simple answer

Θ -notation

- $f(n)$ is $\Theta(g(n))$
 - **Math:**
 - there exist positive constant A, B and n_0 such that $A|g(n)| \leq |f(n)| \leq B|g(n)|$ for all $n > n_0$
 - **Engineering**
 - Drop low-order terms
 - Ignore leading constants
 - **Example**

Θ -notation

- $f(n)$ is $\Theta(g(n))$
 - **Math:**
 - there exist positive constant A, B and n_0 such that $A|g(n)| \leq |f(n)| \leq B|g(n)|$ for all $n > n_0$
 - **Engineering**
 - Drop low-order terms & ignore leading constants
 - **Curve**

Asymptotic performance

- When n gets large enough, a $\Theta(n^2)$ algorithm always beats a $\Theta(n^3)$ algorithm

O-notation

- $f(n)$ is $O(g(n))$
 - **Math:**
 - there exist positive constant B and n_0 such that $|f(n)| \leq B|g(n)|$ for all $n > n_0$
 - Example

O-notation

- $f(n)$ is $O(g(n))$
 - **Math:**
 - there exist positive constant B and n_0 such that $|f(n)| \leq B|g(n)|$ for all $n > n_0$
 - Curve

O-notation

- **Set definition of O-notation:**

- $O(g(n)) = \{f(n) : \text{there exist positive constant } B \text{ and } n_0 \text{ such that } |f(n)| \leq B|g(n)| \text{ for all } n > n_0\}$

- **Convention:**

- $f(n) = n^3 + O(n^2)$ means?
- $n^2 + O(n) = O(n^2)$ means?

Properties of big-O

- **Transitivity:**

- if $f = O(g)$ and $g = O(h)$, then $f = O(h)$

- **Additivity:**

- $O(f) + O(g) = O(\max\{f, g\})$

- **Multiplication by a constant:**

- $O(k \times g(n)) = O(g(n)) \quad (k > 0)$

Example

- Prove that $3n^3 - 10n^2 + n - 10$ is $O(n^3)$

Example

- Prove that $3n^3 - 10n^2 + n - 10$ is not $O(n^2)$

On polynomial orders

- $a_n x_n + a_{n-1} x_{n-1} + \dots + a_1 x + a_0$ is ?

– $\Theta(?)$

– $O(?)$

Ω -notation

- $f(n)$ is $\Omega(g(n))$
 - **Math:**
 - there exist positive constant A and n_0 such that $|A g(n)| \leq |f(n)|$ for all $n > n_0$
 - **Set definition:**
 - $\Omega(g(n)) = \{f(n): \text{there exist ...}\}$
 - **Example**

On polynomial orders

- $a_n x_n + a_{n-1} x_{n-1} + \dots + a_1 x + a_0$ is ?
 - $\Theta(?)$
 - $O(?)$
 - $\Omega(?)$

Θ , O and Ω

- $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$
 - Example

Example

- Prove that $3n^3 - 10n^2 + n - 10$ is $\Omega(n^3)$

Asymptotic Order of Growth

- Upper bounds
 - Big-Oh: $f(n) = O(g(n))$
- Lower bounds
 - Big-Omega: $f(n) = \Omega(g(n))$
- Tight bounds
 - Big-Theta: $f(n) = \Theta(g(n))$

Example

- $f(n) = \sum_{i=0}^n i$

Example

- $f(n) = \log_{2.3} n^3$

Example

- $f(n) = \log n!$