Fundamentals of Mathematics
Lecture 7: Asymptotics

Guan-Shieng Huang

National Chi Nan University, Taiwan

Spring, 2008

The Definition of Big-O Notation

Definition (Big O)

$f(n) = \mathrm{O}(g(n))$ iff there exist constants c and n_0 such that

$$|f(n)| \le c|g(n)| \quad \text{for all } n \ge n_0 \ .$$

In this definition, observe the following implications:

1. We only care about the behaviors of f and g when n is very large. (n_0)
2. A constant coefficient is ignored. (c)
3. g is an upper bound.

The Concept of an 'Upper Bound'

Definition (N. G. de Bruijn's L-notation)

$L(n)$ stands for a number whose absolute value $\leq n$.

$1 + L(5) = L(6)$, $L(2)L(3) = L(6)$, $L(2) + L(3) = L(5)$, $e^{L(5)} = L(e^5)$.
But $L(5) - L(3) = L(8)$.
Let $a = L(5)$ and $b = L(6)$. We cannot conclude that $a < b$, $|a| < |b|$, or even $L(a) = L(b)$.

$$1000 + 2000n = \mathrm{O}(n^2) \ .$$

Hence for small n, $f(n) = \mathrm{O}(g(n))$ may not imply $|f(n)| \leq |g(n)|$.

The Constant Coefficient c

$2000n = \mathrm{O}(n)$. However, $2000n \not\leq n$ for all n.

One-Way Equality

$f(n) = \mathrm{O}(g(n))$ cannot be written as $\mathrm{O}(g(n)) = f(n)$.

$$n = \mathrm{O}(n^2), \mathrm{O}(n^2) = n^2, \quad \text{but } n \neq n^2 \quad .$$

The meaning of '$=$' in Big O

- $\mathrm{O}(g(n))$ stands for the set of all functions $f(n)$ such that $|f(n)| \leq c|g(n)|$ for all $n \geq n_0$ for some c and n_0.
- $f(n) = \mathrm{O}(g(n))$ means $f(n) \in \mathrm{O}(g(n))$.
  $\mathrm{O}(f(n)) = \mathrm{O}(g(n))$ means $\mathrm{O}(f(n)) \subseteq \mathrm{O}(g(n))$.
- Let S and T be two sets of functions of n.

$$S + T := \{f(n) + g(n) |\ f(n) \in S \text{ and } g(n) \in T\}$$

  $S - T$, ST, S/T, $\sqrt{S}$, e^S, $\ln S$ are defined similarly.
  $\implies \mathrm{O}(f(n)) + \mathrm{O}(g(n))$ is defined accordingly.

Example

$\frac{n^2}{3} + \mathrm{O}(n^2) = \mathrm{O}(n^3)$ means

$$S_1 = \{\frac{n^2}{3} + f_1(n) |\ f_1(n) \in \mathrm{O}(n^2)\}$$

$$S_2 = \{f_2(n) |\ f_2(n) \in \mathrm{O}(n^3)\}$$

and $S_1 \subseteq S_2$.

Common Errors I

1. $f(n) = O(n)$ and $g(n) = O(n^2) \implies f(n) \leq g(n)$.
2. $1 + 2 + 3 + \cdots + n = O(n) + O(n) + 3 + \cdots + n$
   $= O(n) + 3 + 4 + \cdots + n = \cdots = O(n)$.
   Or, prove $1 + 2 + 3 + \cdots + n = O(n)$ by induction:
   - Basis: $n = 1$. $1 = O(1)$ holds.
   - Induction: Assume the assertion holds when $n = k$.

$$1 + 2 + \cdots + k + (k+1) = O(k) + (k+1) = O(k+1).$$

Common Errors II

3. For any two functions $f(n)$ and $g(n)$, either $f(n) = \mathrm{O}(g(n))$ or $g(n) = \mathrm{O}(f(n))$.

$$\text{Let } f(n) = \begin{cases} 0 & \text{when } n \text{ is odd} \\ 1 & \text{when } n \text{ is even} \end{cases}$$

$$g(n) = \begin{cases} 1 & \text{when } n \text{ is odd} \\ 0 & \text{when } n \text{ is even} \end{cases}$$

   Or, let $f(n) = \sin(n)$ and $g(n) = \cos(n)$.

4. $f(n) = \mathrm{O}(g(n)) \implies e^{f(n)} = \mathrm{O}(e^{g(n)})$.
   Let $f(n) = \ln n$, $g(n) = \frac{1}{2}\ln n$. Then $e^{f(n)} = n$, $e^{g(n)} = \sqrt{n}$, but $n \neq \mathrm{O}(\sqrt{n})$.

Common Errors III

5. $f(n) = \mathrm{O}(g(n)) \implies \lg f(n) = \mathrm{O}(\lg g(n))$.
   Let $f(n) = 2^{1+\frac{1}{n}}$, $g(n) = 2^{\frac{1}{n}}$. Then $\lg f(n) = 1 + \frac{1}{n}$, $\lg g(n) = \frac{1}{n}$, but $1 + \frac{1}{n} \neq \mathrm{O}(\frac{1}{n})$.
6. $f(n) = \mathrm{O}(1) \implies f(n)$ is a constant function.
   $\cos(n) = \mathrm{O}(1)$.

Other Asymptotic Notations

1. Ω: lower bound (omega)
   $f(n) = \Omega(g(n))$ iff $g(n) = \mathrm{O}(f(n))$.
2. Θ: at the same growth rate (theta)
   $f(n) = \Theta(g(n))$ iff $f(n) = \mathrm{O}(g(n))$ and $f(n) = \Omega(g(n))$.
3. o: (little oh)
   $f(n) = \mathrm{o}(g(n))$ iff $|f(n) \leq \epsilon|g(n)|$ for all $n \geq n_\epsilon$, for all constants $\epsilon > 0$.
   Or, we write $f(n) \prec g(n)$.
4. ω: (little omega)
   $f(n) = \omega(g(n))$ iff $g(n) = \mathrm{o}(f(n))$.
5. $\sim$: asymptotic to
   $f(n) \sim g(n)$ iff $f(n) = g(n) + \mathrm{o}(g(n))$.

Remark

$f(n) = \widetilde{\mathrm{O}}(g(n))$ *means* $f(n) = \mathrm{O}(g(n) \lg^k g(n))$ *for some* $k \in \mathbb{N}$.

How to Determine the Asymptotic Relationship Between Functions

1. $f(n) = \mathrm{O}(g(n)$ if $\lim_{n\to\infty} |\frac{f(n)}{g(n)}| \leq c$ for some constant c.
2. $f(n) = \Theta(g(n)$ if $\lim_{n\to\infty} |\frac{f(n)}{g(n)}| \leq c$ and $\lim_{n\to\infty} |\frac{g(n)}{f(n)}| \leq c$.
3. $f(n) = \mathrm{o}(g(n))$ iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
4. $f(n) \sim g(n)$ iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$.
5. $f(n) = \mathrm{O}(g(n)$ iff $\limsup_{n\to\infty} |\frac{f(n)}{g(n)}| \leq c$ for some constant c.

Useful Properties

1. $f(n) = \mathrm{o}(g(n))$ (or, $f(n) \prec g(n)$) $\implies f(n) = \mathrm{O}(g(n))$.
2. $f(n) \sim g(n) \implies f(n) = \Theta(g(n))$.

Useful Patterns

1. $n^\alpha \prec n^\beta$ iff $\alpha < \beta$
   $n^\alpha = \mathrm{O}(n^\beta)$ iff $\alpha \le \beta$
2. $\lg^k n \prec n^\epsilon$ for any constant $k > 0$ and $\epsilon > 0$.
3. $n^k \prec c^n$ for any constants k and $c > 1$.
4. $f_1(n) \prec g_1(n)$ and $f_2(n) \prec g_2(n) \implies f_1(n)f_2(n) \prec g_1(n)g_2(n)$.

A hierarchy:

$$1 \prec \lg\lg n \prec \lg n \prec n^\epsilon \prec n^c \prec n^{\lg n} \prec c^n \prec n! \prec n^n \prec c^{c^n}$$

where $0 < \epsilon < 1 < c$.

Example

What is the growth rate of $e^{\sqrt{\lg n}}$?
$e^{f(n)} \prec e^{g(n)}$ iff $\lim_{n\to\infty} (f(n) - g(n)) = -\infty$.
$1 \prec f(n) \prec g(n) \implies e^{|f(n)|} \prec e^{|g(n)|}$.

$$\because 1 \prec \lg\lg n \prec \sqrt{\lg n} \prec \epsilon \lg n$$

$$\therefore \lg n \prec e^{\sqrt{\lg n}} \prec n^{\epsilon} \quad .$$

Big-O Manipulation I

1. $n^m = \mathrm{O}(n^{m'})$ when $m \le m'$.
   $\mathrm{O}(f(n)) + \mathrm{O}(g(n)) = \mathrm{O}(|f(n)| + |g(n)|)$.
   Hence $\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} = \mathrm{O}(n^3) + \mathrm{O}(n^3) + \mathrm{O}(n^3) = \mathrm{O}(n^3)$.
2. $f(n) = \mathrm{O}(f(n))$;
   $c \cdot \mathrm{O}(f(n)) = \mathrm{O}(f(n))$ if c is a constant; $\mathrm{O}(\mathrm{O}(f(n))) = \mathrm{O}(f(n))$;
   $\mathrm{O}(f(n))\mathrm{O}(g(n)) = \mathrm{O}(f(n)g(n))$;
   $\mathrm{O}(f(n)g(n)) = f(n)\mathrm{O}(g(n)) = \mathrm{O}(f(n))\mathrm{O}(g(n))$.
3. $\mathrm{O}(f(n)^2) = \mathrm{O}(f(n))^2$
   Hence we can write $\mathrm{O}(\lg n)^2$ instead of $\mathrm{O}((\lg n)^2)$, but not $\mathrm{O}(\lg n)^{-1}$ instead of $\mathrm{O}((\lg n)^{-1})$.

Big-O Manipulation II

4. $\ln(1+\mathrm{O}(f(n))) = \mathrm{O}(f(n))$ if $f(n) \prec 1$.

$$\begin{aligned} |\ln(1+x)| &= \left|x\left(1-\frac{x}{2}+\frac{x^2}{3}-\cdots\right)\right| \\ &\le |x(1+\frac{c}{2}+\frac{c^2}{3}+\cdots)| = \mathrm{O}(x) \end{aligned}$$

when $|x| \le c < 1$ for some constant c.

5. $\exp(\mathrm{O}(f(n))) = 1+\mathrm{O}(f(n))$ when $f(n) = \mathrm{O}(1)$.

$$\begin{aligned} \exp(x) &= 1+x\left(\frac{x}{2!}+\frac{x^2}{3!}+\cdots\right) \\ &= 1+x\cdot\mathrm{O}(1) \quad \text{when } x=\mathrm{O}(1) \\ &= 1+\mathrm{O}(x) \end{aligned}$$

Big-O Manipulation III

6. $(1+\mathrm{O}(f(n)))^{\mathrm{O}(g(n))} = 1 + \mathrm{O}(f(n)g(n))$ if $f(n) \prec 1$ and $f(n)g(n) = \mathrm{O}(1)$.

$$\begin{aligned}
(1+\mathrm{O}(f(n)))^{\mathrm{O}(g(n))} &= \exp\left(\ln(1+\mathrm{O}(f(n)))^{\mathrm{O}(g(n))}\right) \\
&= \exp\left(\mathrm{O}(g(n))\ln(1+\mathrm{O}(f(n)))\right) \\
&= \exp\left(\mathrm{O}(g(n))\mathrm{O}(f(n))\right) \\
&= 1+\mathrm{O}(f(n)g(n))
\end{aligned}$$

The Analysis of Algorithms

- the time complexity of an algorithm
- the time complexity of a problem
- the analysis of the time complexity of an algorithm

- P: a problem
- A: an algorithm for solving P
- x : an instance of P

Complexity of Algorithms I

- the time complexity of A is $\mathrm{O}(f(n))$:
  for any x with $|x| = n$, the execution time of $A(x)$ is $\mathrm{O}(f(n))$.
  This is also called the worst-case time complexity of A.
- the time complexity of A is $\Omega(f(n))$:
  for any x with $|x| = n$, the execution time of $A(x)$ is $\Omega(f(n))$.
- the worst-case time complexity of A is $\Theta(f(n))$:
  the time complexity of A is $\mathrm{O}(f(n))$ and for any n, there exists x with $|x| = n$ such that the execution time of $A(x)$ is $\Omega(f(n))$.

Remark

People often use $\mathrm{O}(f(n))$ *instead of* $\Theta(f(n))$ *when refer to the worst-case time complexity of an algorithm.*

Complexity of Problems

- the time complexity of P is $\mathrm{O}(f(n))$:
  there exists an algorithm whose time complexity is $\mathrm{O}(f(n))$
- the time complexity of P is $\Omega(f(n))$:
  any algorithm that solves P must have worst-case time complexity $\Omega(f(n))$
- the time complexity of P is $\Theta(f(n))$: the lower bound and upper bound match

Worst-Case Time Complexity

- Only care about the hardest instances in a problem

Average-Case Time Complexity

- Care about the average-behavior of an algorithm

Discussion

- Is big O a good choice in the analysis of algorithm?
- Why do we usually analyze an algorithm by worst-case analysis?
- The time complexity of a problem depends on the model of computation.
  - Random-Access Machine
  - Turing Machine
- What is an algorithm?

References

- R. L. Graham, D. E. Knuth, O. Patashnik, Concrete Mathematics, 2nd Edition, Addison-Wesley, 1994.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, MIT Press, 2003.