算法设计与分析

Lecture 2: Asymptotic Notation

卢杨

厦门大学信息学院计算机科学系

luyang@xmu.edu.cn

Asymptotic Notation

- Intuitively, just look at the dominant term.

$$T(n) = \cancel{0.1}n^3 + \cancel{10n^2 + 5n + 25}$$

  - Drop lower-order terms $10n^2 + 5\text{n} + 25$.
  - Ignore constant 0.1.
- But we can't say that $T(n)$ equals to n^3.
  - It grows like n^3. But it doesn't equal to n^3.
- We define asymptotic notations (渐进符号) like $T(n) = \Theta(n^3)$ to describe the asymptotic running time of an algorithm.
  - "Asymptotic" here means "as something tends to infinity", as we want to compare algorithms for very large n.

Logarithm Review

Definition

$\log_b a$ is the unique number c s.t. $b^c = a$.

- Notations:
  - $\lg n = \log_2 n$ (binary logarithm)
  - $\ln n = \log_e n$ (natural logarithm)
  - $\lg^k n = (\lg n)^k$ (exponentiation)
  - $\lg \lg n = \lg(\lg n)$ (composition)
- Derivative:
  - $\frac{d(\log_a x)}{dx} = \frac{1}{x \ln a}$

- Useful identities for all real $a > 0$, $b > 0$, $c > 0$, and n, and where logarithm bases are not 1:
  - $\log_c(ab) = \log_c a + \log_c b$
  - $\log_b a^n = n \log_b a$
  - $\log_b \left(\frac{1}{a}\right) = -\log_b a$
  - $\log_b a = (\log_a b)^{-1}$
  - $a^{\log_b c} = c^{\log_b a}$
  - $\log_b a = \frac{\log_c a}{\log_c b}$
  - $a = b^{\log_b a}$

Big O Notation

Definition 2.2

For a given complexity function $g(n)$, $O\big(g(n)\big)$ is the set of complexity functions $f(n)$ for which there exists some positive real constant c and some nonnegative integer n_0 such that for all $n \geq n_0$,

$$0 \leq f(n) \leq cg(n).$$

- $O\big(g(n)\big)$ is a set of functions in terms of $g(n)$ that satisfy the definition.
- If $f(n) = O\big(g(n)\big)$, it represents that $f(n)$ is an element in $O\big(g(n)\big)$. We say that $f(n)$ is "big O (大O)" of $g(n)$.
  - Strictly, we should use "∈" instead of "=". However, it is conventional to use "=" for asymptotic notations.

Big O Notation

- No matter how large $f(n)$ is, it will eventually be smaller than $cg(n)$ for some c and some n_0.
- Big O notation describes an upper bound. We use it to bound the worst-case running time of an algorithm on arbitrary inputs.

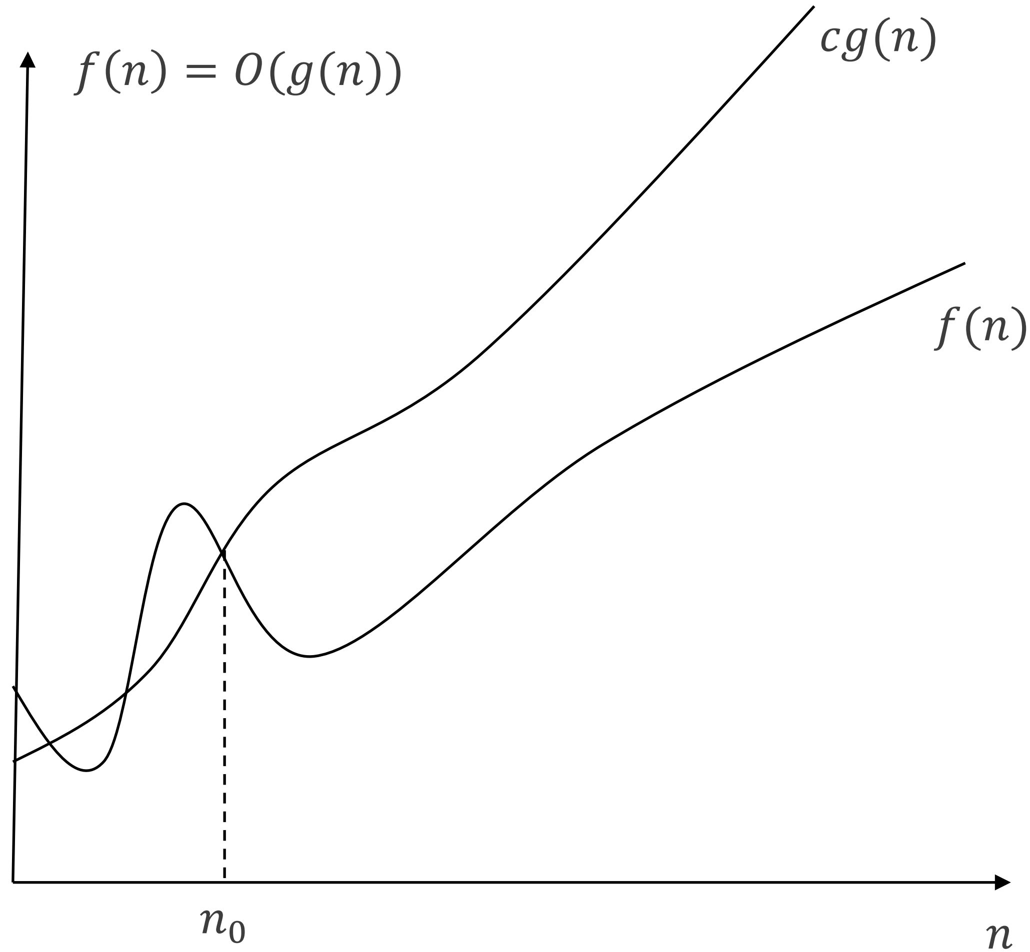


Image source: 图2.2, 张德富, 算法设计与分析, 国防工业出版社, 2009.

Display of Growth of Functions

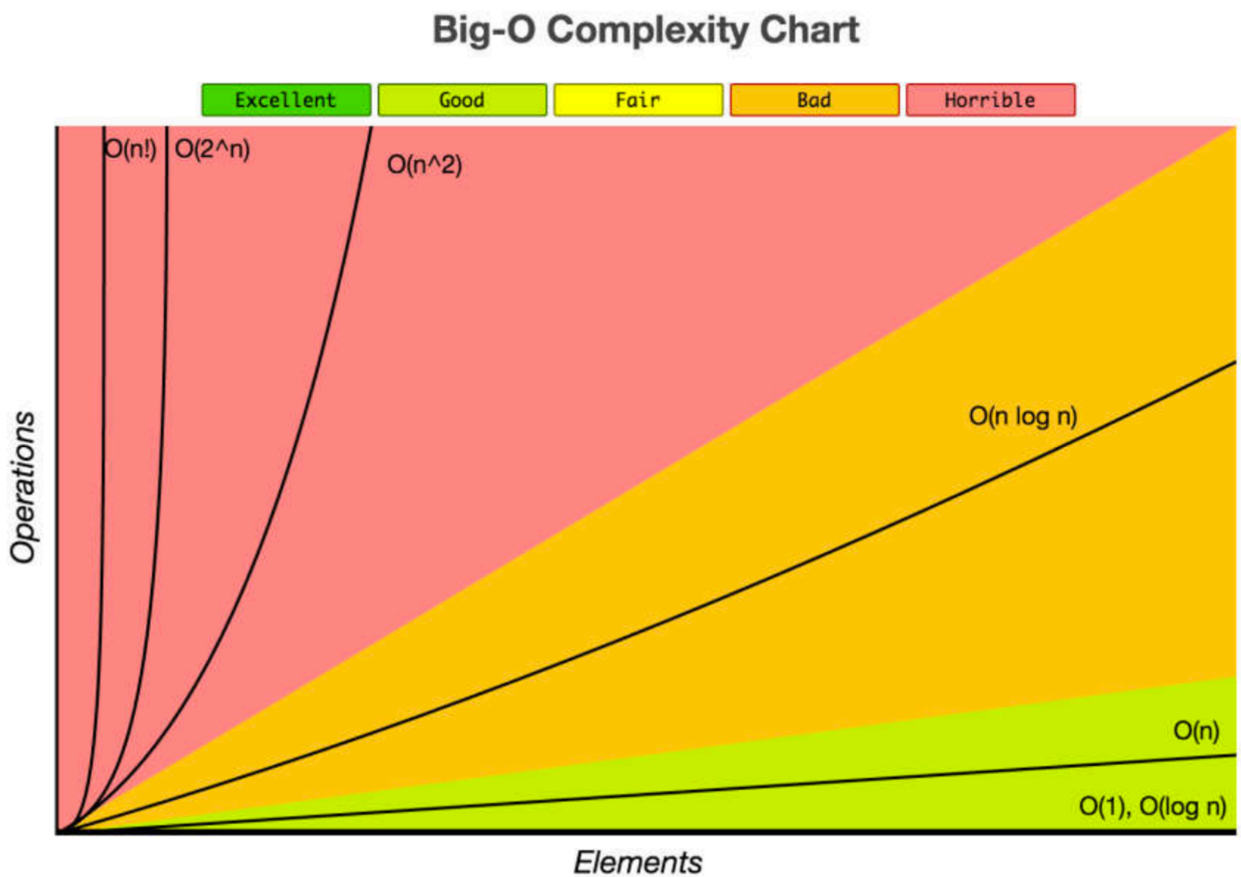


Image source: http://bigocheatsheet.com/img/big-o-complexity-chart.png

Big O Notation

Example 1

We show that $n^2 + 10n = O(n^2)$. Because, for $n \geq 1$,

$$n^2 + 10n \leq n^2 + 10n^2 = 11n^2,$$

we can take $c = 11$ and $n_0 = 1$ to obtain our result.

- To show a function is in big *O* of another function, the key is to find a specific value of c and n_0 that make the inequality hold.
- More examples of functions in $O(n^2)$:
  - n^2, $n^2 + n$, $n^2 + 1000n$, $1000n^2 + 1000n$, n, $n/1000$, $n^{1.99999}$, $n^2/\lg\lg\lg n$.

Classroom Exercise

Use the definition of Big *O* notation to show:

Is $2^{2n} = O(2^n)$?

Classroom Exercise

Proof:

We prove it by contradiction. Assume there exist constants $c > 0$ and $n_0 \geq 0$, such that

$$2^{2n} \leq c2^n,$$

for all $n \geq n_0$. Then

$$2^{2n} = 2^n 2^n \leq c2^n,$$
$$2^n \leq c.$$

But we can't find any constant c is greater than 2^n for all $n \geq n_0$. So the assumption leads to a contradiction. Then we can certify that $2^{2n} \neq O(2^n)$.

How about $2^{n+1} = O(2^n)$?

Big Ω Notation

Definition 2.3

For a given complexity function $g(n)$, $\Omega\big(g(n)\big)$ is the set of complexity functions $f(n)$ for which there exists some positive real constant c and some nonnegative integer n_0 such that for all $n \geq n_0$,

$$0 \leq cg(n) \leq f(n).$$

- $\Omega\big(g(n)\big)$ is the opposite of $O\big(g(n)\big)$.
- If $f(n) = \Omega\big(g(n)\big)$, it represents that $f(n)$ is an element in $\Omega\big(g(n)\big)$. We say that $f(n)$ is "big Ω (大Ω)" of $g(n)$.

Big Ω Notation

- No matter how small $f(n)$ is, it will eventually be larger than $cg(n)$ for some c and some n_0.
- Big Ω notation describes an lower bound. We use it to bound the best-case running time of an algorithm on arbitrary inputs.

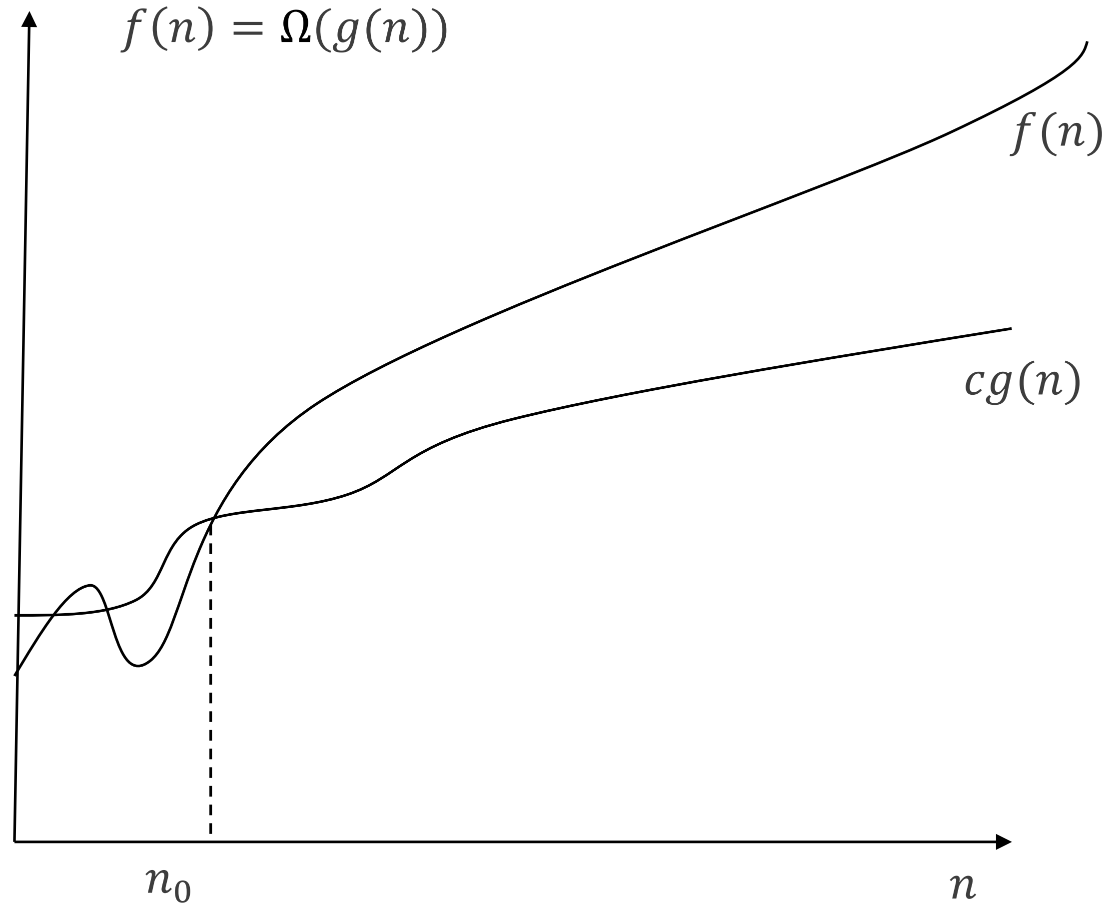


Image source: 图2.3, 张德富, 算法设计与分析, 国防工业出版社, 2009.

Big Θ Notation

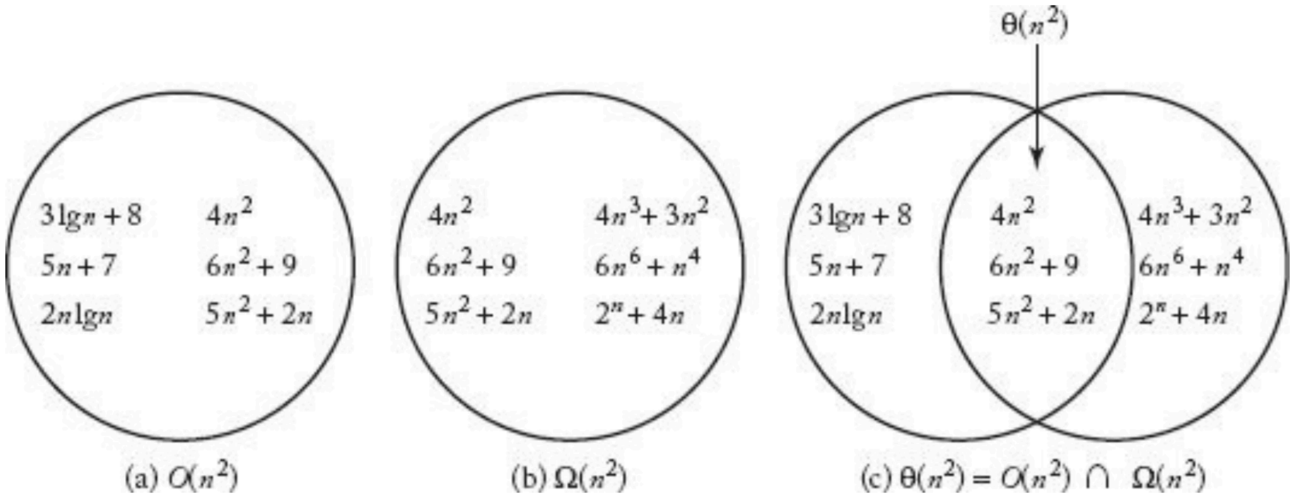
Definition 2.1

For a given complexity function $g(n)$, $\Theta(g(n))$ is the set of complexity functions $f(n)$ for which there exists some positive real constants c_1 and c_2 and some nonnegative integer n_0 such that, for all $n \geq n_0$,

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n).$$

- If $f(n) = \Theta(g(n))$, we say that $f(n)$ is "big Θ (大Θ)" or has the same order (数量级) of $g(n)$.
- $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$.

Relation between Big O, Big Ω and Big Θ



- Now we have O, Θ, and Ω. Intuitively, they just like "≤", "=", and "≥" for complexity functions.

Image source: Figure 1.6, Richard E. Neapolitan, Foundations of Algorithms (5th Edition), Jones & Bartlett Learning, 2014

Big Θ Notation

- $f(n) = \Theta\big(g(n)\big)$ implies $f(n) = O\big(g(n)\big)$ and $f(n) = \Omega\big(g(n)\big)$.
- Big Θ can also be used to bound the worst-case time complexity.
  - For insertion sort, the worst-case is both $\Theta(n^2)$ and $O(n^2)$.
- However, we usually use Big O notation because we don't care the best-case.

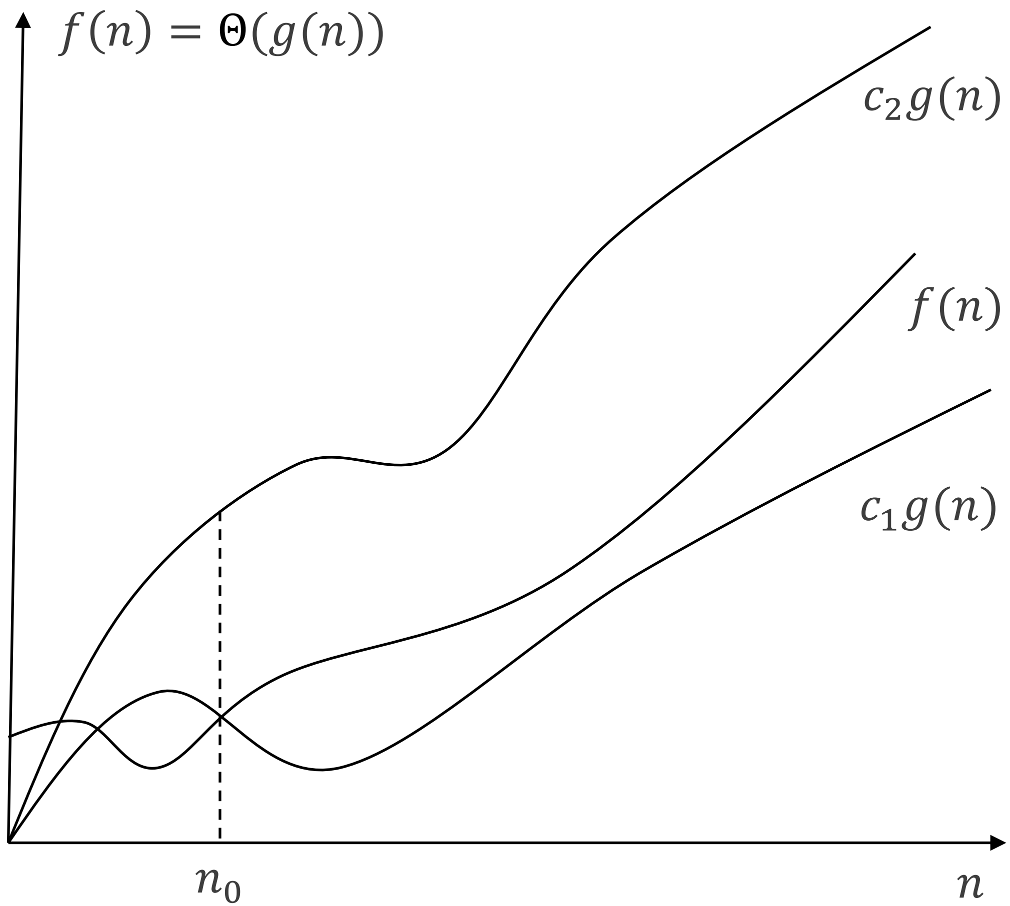


Image source: 图2.1, 张德富, 算法设计与分析, 国防工业出版社, 2009.

Properties of Asymptotic Notations

Theorem 2.1

For any two functions $f(n)$ and $g(n)$, $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

- $\Theta = \text{O}$ and Ω.

Theorem 2.2

For any two functions $f_1(n)$ and $f_2(n)$, if $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, we have $f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$.

- Pick the larger one.

Properties of Asymptotic Notations

- Transitivity (传递性)
  - If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$ then $f(n) = \Theta(h(n))$.
  - Same for O and Ω.
- Additivity (可加性)
  - If $f(n) = \Theta(h(n))$ and $g(n) = \Theta(h(n))$ then $f(n) + g(n) = \Theta(h(n))$.
  - Same for O and Ω.
- Reflexivity (自反性)
  - If $f(n) = \Theta(f(n))$.
  - Same for O and Ω.
- Symmetry (对称性)
  - $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$.
  - Not hold for O and Ω.

Properties of Asymptotic Notations

- Consider the following ordering of complexity categories:

  $\Theta(\lg n)\ \ \Theta(n)\ \ \Theta(n\lg n)\ \ \Theta(n^2)\ \ \Theta\left(n^j\right)\ \ \Theta\left(n^k\right)\ \ \Theta(a^n)\ \ \Theta(b^n)\ \ \Theta(n!)$

  where $k \geq j \geq 2$ and $b \geq a \geq 1$.
- If $f(n)$ is to the left of $g(n)$ in the above sequence, then

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

- Notice: Big Θ is a set of functions. We can't say $\Theta(\lg n) < \Theta(n)$.

Properties of Asymptotic Notations

Example 2

Given $f(n) = \frac{1}{2}n(n-1)$, prove that $f(n) = \Theta(n^2)$

Proof:

By the property, we first show that $f(n) = O(n^2)$:

$$\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2 \text{ (for } c = \frac{1}{2} \text{ and } n_0 = 0\text{).}$$

Then we show that $f(n) = \Omega(n^2)$:

$$\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \geq \frac{1}{2}n^2 - \frac{1}{2}n\frac{1}{2}n = \frac{1}{4}n^2 \text{ (for } c = \frac{1}{4} \text{ and } n_0 = 2\text{).}$$

Thus $f(n) = \Theta(n^2)$.

Using Limit to Determine Order

- In addition to proving by definition, we can also use limit to get asymptotic notations.

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}\begin{cases} = c & \text{implies } f(n)=\Theta\big(g(n)\big) \quad \text{if } 0<c<\infty \\ \neq \infty & \text{implies } f(n)=O\big(g(n)\big) \\ \neq 0 & \text{implies } f(n)=\Omega\big(g(n)\big) \end{cases}$$

Using a Limit to Determine Order

Example 3

Compare the orders of growth of $\frac{1}{2}n(n-1)$ and n^2.

$$\lim_{n\to\infty}\frac{\frac{1}{2}n(n-1)}{n^2}=\frac{1}{2}\lim_{n\to\infty}\frac{n^2-n}{n^2}=\frac{1}{2}\lim_{n\to\infty}(1-\frac{1}{n})=\frac{1}{2},$$

Thus, $\frac{1}{2}n(n-1)=\Theta(n^2)$.

Classroom Exercise

Compare the orders of growth of a^n and b^n, when $b > a > 0$

Classroom Exercise

Solution:

$$\lim_{n\to\infty}\frac{a^n}{b^n} = \lim_{n\to\infty}\left(\frac{a}{b}\right)^n = 0.$$

The limit is 0 because $0 < \frac{a}{b} < 1$. Thus, $a^n = O(b^n)$.

Using a Limit to Determine Order

- When calculating $\lim_{n \to \infty} \frac{f(n)}{g(n)}$, how to deal with the following cases?

$$\lim_{n \to \infty} f(n) = \lim_{n \to \infty} g(n) = 0 \text{ or } \pm\infty$$

Using a Limit to Determine Order

Image source: https://tieba.baidu.com/p/5933589166

Using a Limit to Determine Order

L'Hôpital's Rule (洛必达法则)

If $f(x)$ and $g(x)$ are both differentiable with derivatives $f'(x)$ and $g'(x)$, respectively, and if

$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} g(x) = 0 \text{ or } \pm\infty,$$

then

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{f'(x)}{g'(x)},$$

whenever the limit on the right exists.

Using a Limit to Determine Order

Example 4

$$\lg n = O(n)$$

$$\frac{d(\log_a x)}{dx} = \frac{1}{x \ln a}$$

because

$$\lim_{x\to\infty} \frac{\lg x}{x} = \lim_{x\to\infty} \frac{d(\lg x)/dx}{dx/dx} = \lim_{x\to\infty} \frac{1/(x \ln 2)}{1} = 0.$$

Exercises

Show the correctness of the following statements.

- $\lg n = O(n)$
- $n = O(n \lg n)$
- $n \lg n = O(n^2)$
- $2^n = \Omega(5^{\ln n})$
- $\lg^3 n = O(n^{0.5})$

Conclusion

After this lecture, you should know:

- Why do we need asymptotic notation?
- What are the meaning of these asymptotic notations big O, big Θ, or big Ω?
- How to prove a complexity function is big O, big Θ, or big Ω?
- How to compare the order of two complexity function?

厦门大学信息学院 SCHOOL OF INFORMATICS XIAMEN UNIVERSITY 厦门大学计算机科学系 Computer Science Department of Xiamen University

Homework

- Page 19

  2.1

  2.2

  2.3

  2.9

谢谢

有问题欢迎随时跟我讨论