Part2-Asymptotic Analysis

Completed on	@2020/03/16
Key videos	https://www.coursera.org/learn/algorithms-divide-conquer/lecture/yI6kU/additional-examples-review-optional https://www.coursera.org/learn/algorithms-divide-conquer/lecture/SxSch/big-omega-and-theta
Note	Introduction to some widely used notations for algorithms analysis.

Note

▼ The Gist

- 动机：掌握算法设计和分析的相关术语
- 注重最高次项，忽略常数项和低次项

High-level idea: Suppress constant factors and lower-order terms

too system-dependent

irrelevant for large inputs

Example: Equate $6n \log_2 n + 6$ with just $n \log n$.

Terminology: Running time is $O(n \log n)$

["big-Oh" of $n \log n$]

where n = input size (e.g. length of input array).

- 循环：单层/多层并列循环O(n)，双层（Two Nested Loop）循环O(n2)

▼ Big-Oh: Definition

- 定义：存在常数c与n0，使算法复杂度永远小于某一个**上界**

<u>Formal Definition</u> : T(n) = O(f(n)) if and only if there exist constants $c, n_0 > 0$ such that

$$T(n) \leq c \cdot f(n)$$

For all $n \geq n_0$

<u>Warning</u> : c, n_0 cannot depend on n

▼ Basic Examples

- 案例1

  可以把所有的低阶项视为高阶项的退化，则最终还是会小于最高次项的常数倍复杂度！

  <u>Claim</u> : if $T(n) = a_k n^k + ... + a_1 n + a_0$ then

  $$T(n) = O(n^k)$$

  <u>Proof</u> : Choose $n_0 = 1$ and $c = |a_k| + |a_{k-1}| + .. + |a_1| + |a_0|$
  Need to show that $\forall n \geq 1, T(n) \leq c \cdot n^k$
  We have, for every $n \geq 1$,

  $$\begin{aligned} T(n) &\leq |a_k|n^k + ... + |a_1|n + |a_0| \\ &\leq |a_k|n^k + ... + |a_1|n^k + |a_0|n^k \\ &= c \cdot n^k \end{aligned}$$

- 案例2

  证明k方的复杂度不可能≤(k-1)方

Claim : for every $k \geq 1$, n^k is not $O(n^{k-1})$

Proof : by contradiction. Suppose $n^k = O(n^{k-1})$
Then there exist constants c, n_0 such that

$$n^k \leq c \cdot n^{k-1} \quad \forall n \geq n_0$$

But then [cancelling n^{k-1} from both sides]:

$$n \leq c \quad \forall n \geq n_0$$

Which is clearly False [contradiction].

- Big Omega and Theta
  - Big Omega：存在下界

Definition : $T(n) = \Omega(f(n))$
If and only if there exist constants c, n_0 such that

$$T(n) \geq c \cdot f(n) \quad \forall n \geq n_0$$

  - Big Theta：存在一个上下界的区间

Definition : $T(n) = \theta(f(n))$ if and only if
$T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$

Equivalent : there exist constants c_1, c_2, n_0 such that

$$c_1 f(n) \leq T(n) \leq c_2 f(n)$$

$$\forall n \geq n_0$$

- 特别的案例！

Let $T(n) = \frac{1}{2}n^2 + 3n$. Which of the following statements are true ? (Check all that apply.)

- ☐ $T(n) = O(n)$.
- ☐ $T(n) = \Omega(n)$. $[n_0 = 1, c = 1/2]$
- ☐ $T(n) = \Theta(n^2)$. $[n_0 = 1, c_1 = 1/2, c_2 = 4]$
- ☐ $T(n) = O(n^3)$. $[n_0 = 1, c = 4]$

- Little O notation

  约束条件更强！对于所有的大于0的常数 vs 大O记号，存在一个常数即可

Little-Oh Notation

<u>Definition</u> : $T(n) = o(f(n))$ if and only if for all constants c>0, there exists a constant n_0 such that

$$T(n) \leq c \cdot f(n) \quad \forall n \geq n_0$$

<u>Exercise</u> : $\forall k \geq 1, n^{k-1} = o(n^k)$

▼ Addition Examples

- 案例1：做乘法即可选择到存在一个c

Example #1

Claim : $2^{n+10} = O(2^n)$

Proof : need to pick constants c, n_0 such that

$$(*) \quad 2^{n+10} \leq c \cdot 2^n \quad n \geq n_0$$

Note : $2^{n+10} = 2^{10} \times 2^n = (1024) \times 2^n$

So if we choose $c = 1024, n_0 = 1$ then (*) holds.

Q.E.D

- 案例2：不等式两边同时除

Example #2

Claim : $2^{10n} \neq O(2^n)$

Proof : by contradiction. If $2^{10n} = O(2^n)$ then there exist constants $c, n_0 > 0$ such that

$$2^{10n} \leq c \cdot 2^n \quad n \geq n_0$$

But then [cancelling 2^n]

$$2^{9n} \leq c \quad \forall n \geq n_0$$

Which is certainly false. Q.E.D

- 案例3，存在上下界，利用max()函数的性质！

Example #3 (continued)

<u>Proof</u> : $max\{f, g\} = \theta(f(n) + g(n))$

For every n, we have

$$max\{f(n), g(n)\} \leq f(n) + g(n)$$

And

$$2 * max\{f(n), g(n)\} \geq f(n) + g(n)$$

Thus $\frac{1}{2} * (f(n) + g(n)) \leq max\{f(n), g(n)\} \leq f(n) + g(n) \quad \forall n \geq 1$

$=> max\{f, g\} = \theta(f(n) + g(n))$ $[where\ n_0 = 1, c_1 = 1/2, c_2 = 1]$

Q.E.D

References

-