

Asymptotic

Asymptotic growth rate of functions

如何比较两个算法的优劣？

- 仅比较关键操作的数量
- 仅在输入规模很大的时候：规模趋于无穷大
- 仅考虑关键项
 - 仅考虑次数最高的项
 - 忽略常系数

引入**函数的渐近增长率** (Asymptotic growth rate of $f(n)$)，关注函数随输入规模增大，函数数值的增长速度

Asymptotic notation

Big Oh

$O(g)$: functions that grow no faster than g

定义 2.2 $f(n) = O(g(n))$

- Giving $g : \mathbb{N} \rightarrow \mathbb{R}^+$, then $O(g)$ is the set of $f : \mathbb{N} \rightarrow \mathbb{R}^+$, such that for some $c \in \mathbb{R}^+$ and some $n_0 \in \mathbb{N}$, $f(n) \leq cg(n)$ for all $n \geq n_0$
- $f(n) = O(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c < \infty$

即问题规模足够大时， $f(n)$ 的增长率不会超过 $g(n)$

Order of sum function

Big Ω

$\Omega(g)$: functions that grow at least as fast as g

定义 2.4 $f(n) = \Omega(g(n))$

- Giving $g : \mathbb{N} \rightarrow \mathbb{R}^+$, then $\Omega(g)$ is the set of $f : \mathbb{N} \rightarrow \mathbb{R}^+$, such that for some $c \in \mathbb{R}^+$ and some $n_0 \in \mathbb{N}$, $f(n) \geq cg(n)$ for all $n \geq n_0$
- $f(n) = \Omega(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c > 0$

即问题规模足够大时, $f(n)$ 的增长率不会低于 $g(n)$

The set Θ

$\Theta(g) = O(g) \cap \Omega(g)$: functions that grow at the same rate as g

定义 2.6 $f(n) = \Theta(g(n))$

- Giving $g : \mathbb{N} \rightarrow \mathbb{R}^+$, then $\Theta(g)$ is the set of $f : \mathbb{N} \rightarrow \mathbb{R}^+$, such that for some $c_1, c_2 \in \mathbb{R}^+$ and some $n_0 \in \mathbb{N}$
- $f(n) = \Theta(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c \quad (0 < c < \infty)$

即 $f(n)$ 和 $g(n)$ 的增长率在同一水平

注:

Little Oh

定义 2.3 $f(n) = o(g(n))$

- Giving $g : \mathbb{N} \rightarrow \mathbb{R}^+$, then $o(g)$ is the set of $f : \mathbb{N} \rightarrow \mathbb{R}^+$, such that for **any** $c \in \mathbb{R}^+$, there **exists some** $n_0 \in \mathbb{N}$
- $f(n) = o(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$

表示 $f(n)$ 和 $g(n)$ 的增长率间有 **实质性的差距**, 总可以通过增加问题的规模使得 $f(n)$ 和 $g(n)$ 之间有任何大的差距

Little ω

定义 2.5 $f(n) = \omega(g(n))$

- Giving $g : \mathbb{N} \rightarrow \mathbb{R}^+$, then $\omega(g)$ is the set of $f : \mathbb{N} \rightarrow \mathbb{R}^+$, such that for **any** $c \in \mathbb{R}^+$, there **exists some** $n_0 \in \mathbb{N}$
- $f(n) = \omega(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

表示 $f(n)$ 和 $g(n)$ 的增长率间有 **实质性的差距**，总可以通过增加问题的规模使得 $f(n)$ 和 $g(n)$ 之间有任何大的差距

Asymptotic Order

Logarithm

证明: [L'Hospital's rule](#)

Power

证明: [L'Hospital's rule](#)

Factorial

证明: [Stirling's formula](#)

Stirling formula

函数渐近增长率的性质

$O, \Omega, \Theta, o, \omega$ 均满足传递性

O, Ω, Θ 满足自反性

Θ 是一种等价关系: 渐近增长率的不同规模即是不同的等价类 (e.g. $\log n, n, n \log n, n^k, a^n$)

对偶关系:

Brute Force Enumeration

此部分简单介绍, Tutorial 1 会详细讲解

Iteration

Swapping array elements

一个有 n 个元素的数组, 如何将其前半部分和后半部分交换位置, 同时每部分内部元素次序不变

<time, space>

- $\langle O(n^2), O(1) \rangle$: 逐个向前交换
- $\langle O(n), O(n) \rangle$: 使用辅助数组
- $\langle O(n), O(1) \rangle$: 首先将数组反转, 然后将两个子部分分别反转

Max-sum Subsequence

一个有 n 个元素的数组，求其一个连续子序列，使子序列中各元素之和最大

- $O(n^3)$: 蛮力迭代，遍历数组
- $O(n^2)$: 避免冗余计算，仍是遍历数组，但是减少一层循环
- $O(n \log n)$: Divide & Conquer
- $O(n)$: 动态规划

Recursion

Job scheduling

Jobs: $J_i = [s_i, j_i)$

每个 Job 占据一定的时间段，求所有 Job 中互不冲突的 Job 的最大数量

- BF: 选择 a
 - 结果不包含 a : 递归求解 $J \setminus \{a\}$
 - 结果包含 a : 递归求解 $J \setminus \{a\} \setminus \{\text{Jobs overlapping with } a\}$
- Improvements:
 - DP
 - Greedy

Matrix Chain Multiplication

求解 $A_1 \times A_2 \times \cdots \times A_n$ 的最佳求解顺序

- 矩阵乘法具有结合性
- 计算顺序不同将带来操作数量的极大差别

BF & DP