

绪论

渐进复杂度：层级划分

e1 - c4

邓俊辉

deng@tsinghua.edu.cn

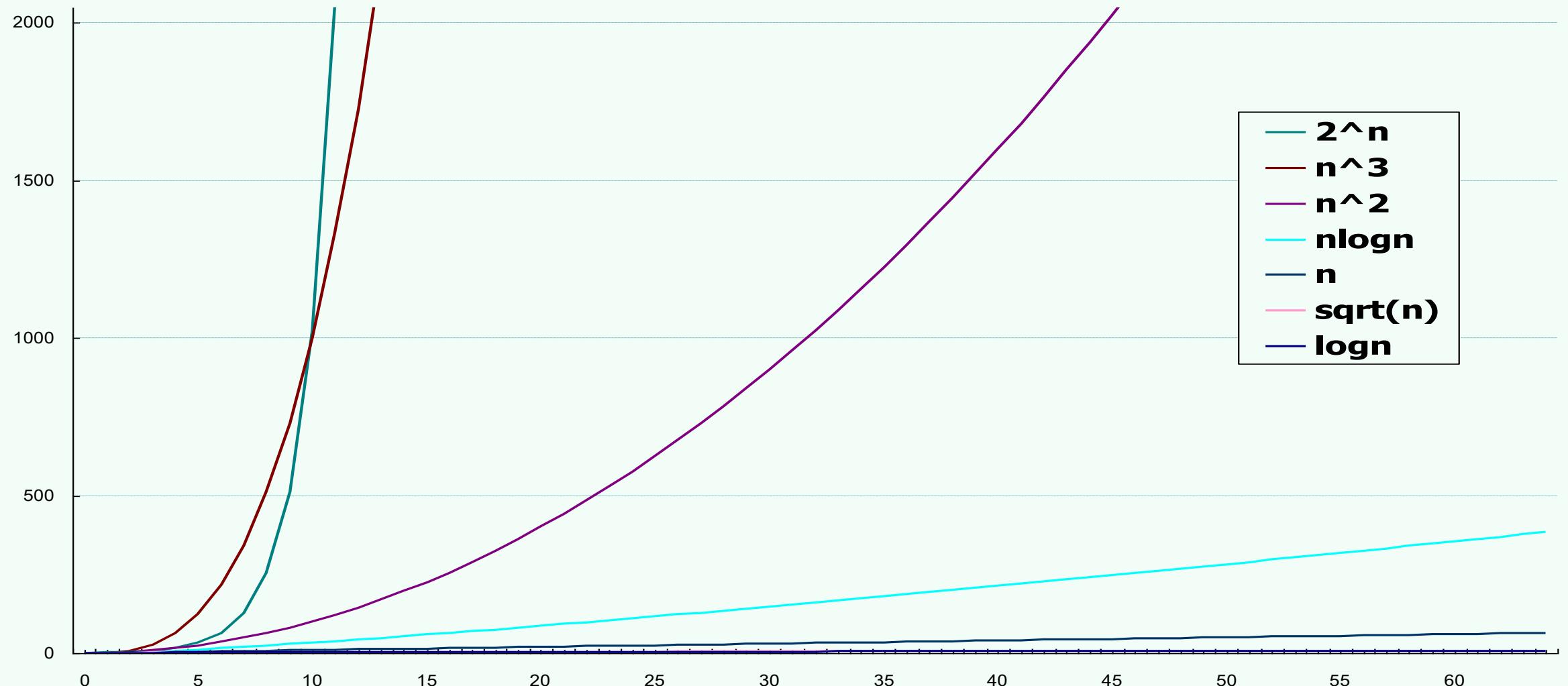
好读书，不求甚解；每有会意，便欣然忘食

主啊，我向你承认，我依旧不明了时间是什么

增长速度：先胖不算胖



增长速度：路遥知马力



层次级别

常数	$\mathcal{O}(1)$	再好不过，但难得如此幸运	对数据结构的基本操作
	$\mathcal{O}(\log^* n)$	在这个宇宙中，几乎就是常数	逆Ackermann函数
对数	$\mathcal{O}(\log n)$	与常数无限接近，且不难遇到	有序向量的二分查找 堆、词典的查询、插入与删除
线性	$\mathcal{O}(n)$	努力目标，经常遇到	树、图的遍历
	$\mathcal{O}(n \log^* n)$	几乎几乎几乎...就是线性	某些MST算法
	$\mathcal{O}(n \log \log n)$	非常非常非常...接近线性	某些三角剖分算法
	$\mathcal{O}(n \log n)$	最常出现，但不见得最优	排序、EU、Huffman编码
平方	$\mathcal{O}(n^2)$	所有输入对象两两组合	Dijkstra算法
立方	$\mathcal{O}(n^3)$	不常见	矩阵乘法
多项式	$\mathcal{O}(n^c)$	P问题 = 存在多项式算法的问题	
指数	$\mathcal{O}(2^n)$	很多问题的平凡算法，再尽可能优化	
...		绝大多数问题，并不存在算法	

❖ 证明、证否

- Fibonacci数 $\text{fib}(n) = \Theta(2^n)$
- $12n + 5 = \Theta(n \log n)$
- $\log^2(n^{1024} - 2*n^6 + 101) = \Theta(?)$
- $\log^d n = \Theta(n^c), \forall c > 0, d > 1$
- $\log^{1.001} n = \Theta(\log(n^{1001}))$
- $(n^2 + 1) / (2n + 3) = \Theta(n)$
- $n^{2013} = \Theta(n!)$
- $n! = \Theta(n^{2019})$
- $2^n = \Theta(n!)$

❖ k-Subset :

任给整数集 S , 判定 S

可否划分为 k 个不交子集, 使其和均为 $(\sum S)/k$

❖ 证明或证否 :

$(k+1)$ -Subset 的难度, 不低于 k -Subset

❖ 自学: small-o notation