

绪论

渐进复杂度：层级划分

01-C4

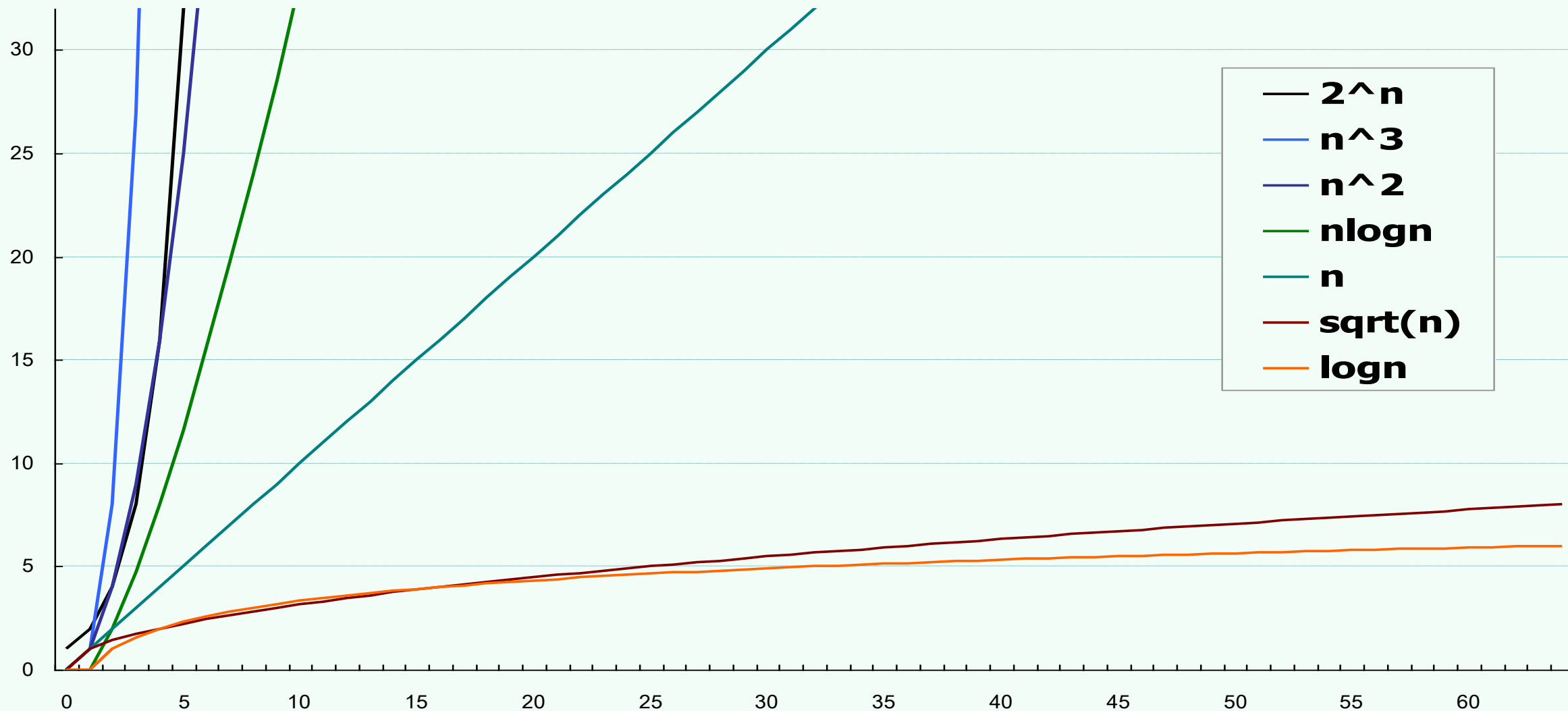
好读书，不求甚解；每有会意，便欣然忘食

主啊，我向你承认，我依旧不明白时间是什么

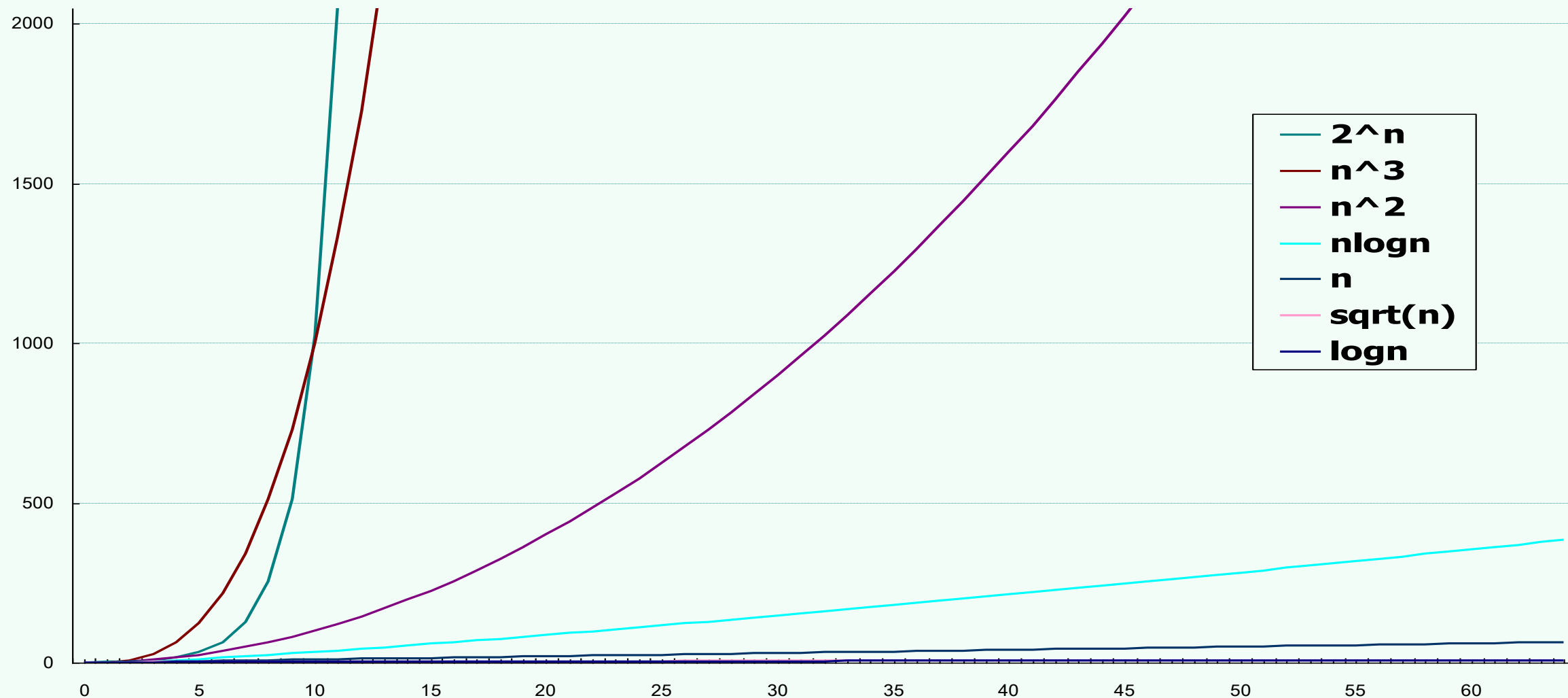
邓俊辉

deng@tsinghua.edu.cn

## 增长速度：先胖不算胖



# 增长速度：路遥知马力



# 层次级别

常数	$O(1)$	再好不过，但难得如此幸运	对数据结构的基本操作
	$O(\log^*n)$	在这个宇宙中，几乎就是常数	逆Ackermann函数
对数	$O(\log n)$	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	$O(n)$	努力目标，经常遇到	树、图的遍历
	$O(n \log^*n)$	几乎几乎几乎...就是线性	某些MST算法
	$O(n \log \log n)$	非常非常非常...接近线性	某些三角剖分算法
	$O(n \log n)$	最常出现，但不见得最优	排序、EU、Huffman编码
平方	$O(n^2)$	所有输入对象两两组合	Dijkstra算法
立方	$O(n^3)$	不常见	矩阵乘法
多项式	$O(n^c)$	P问题 = 存在多项式算法的问题	
指数	$O(2^n)$	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	

# 课后

## ❖ 证明、证否

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

## ❖ 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