θ1-C4

1.绪论

渐进分析

复杂度层级

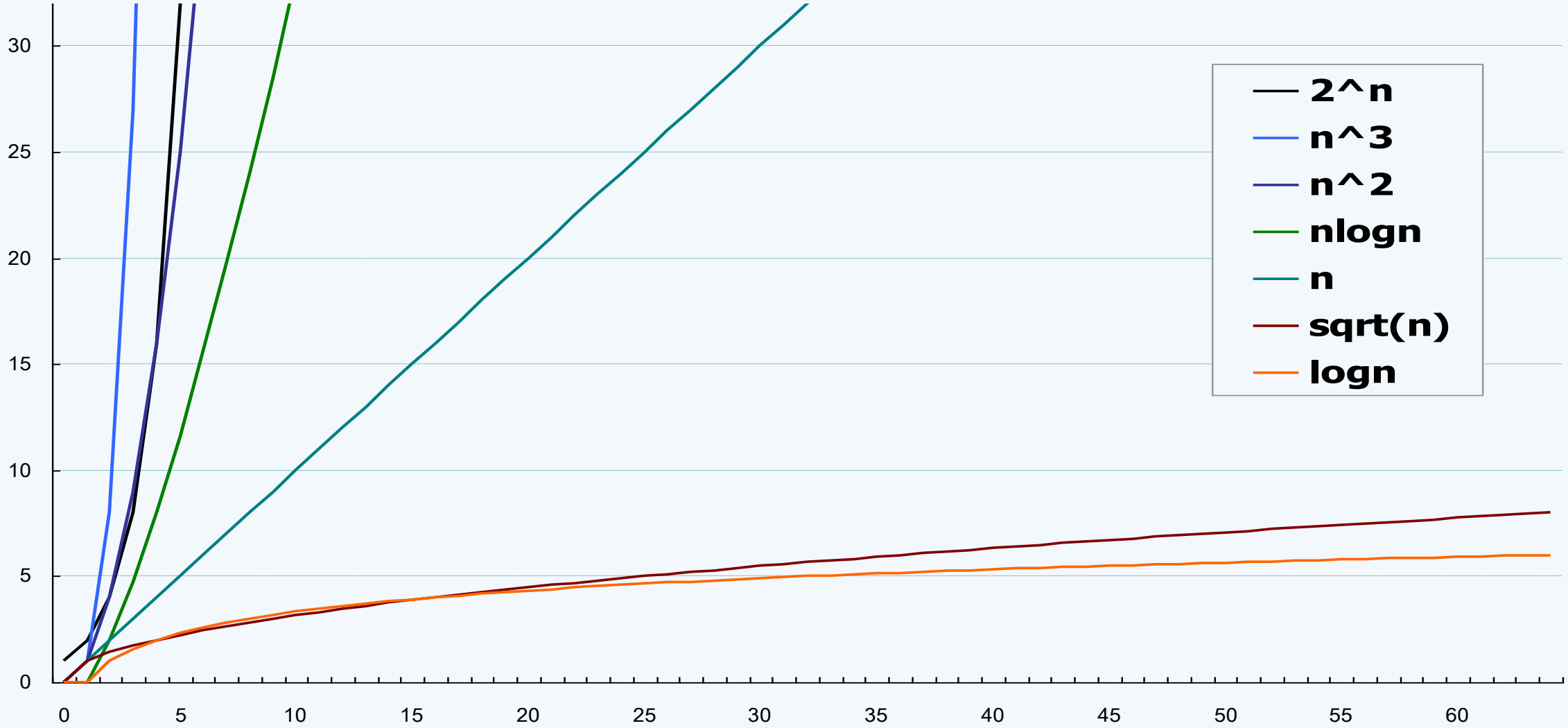
好读书，不求甚解

每有会意，便欣然忘食

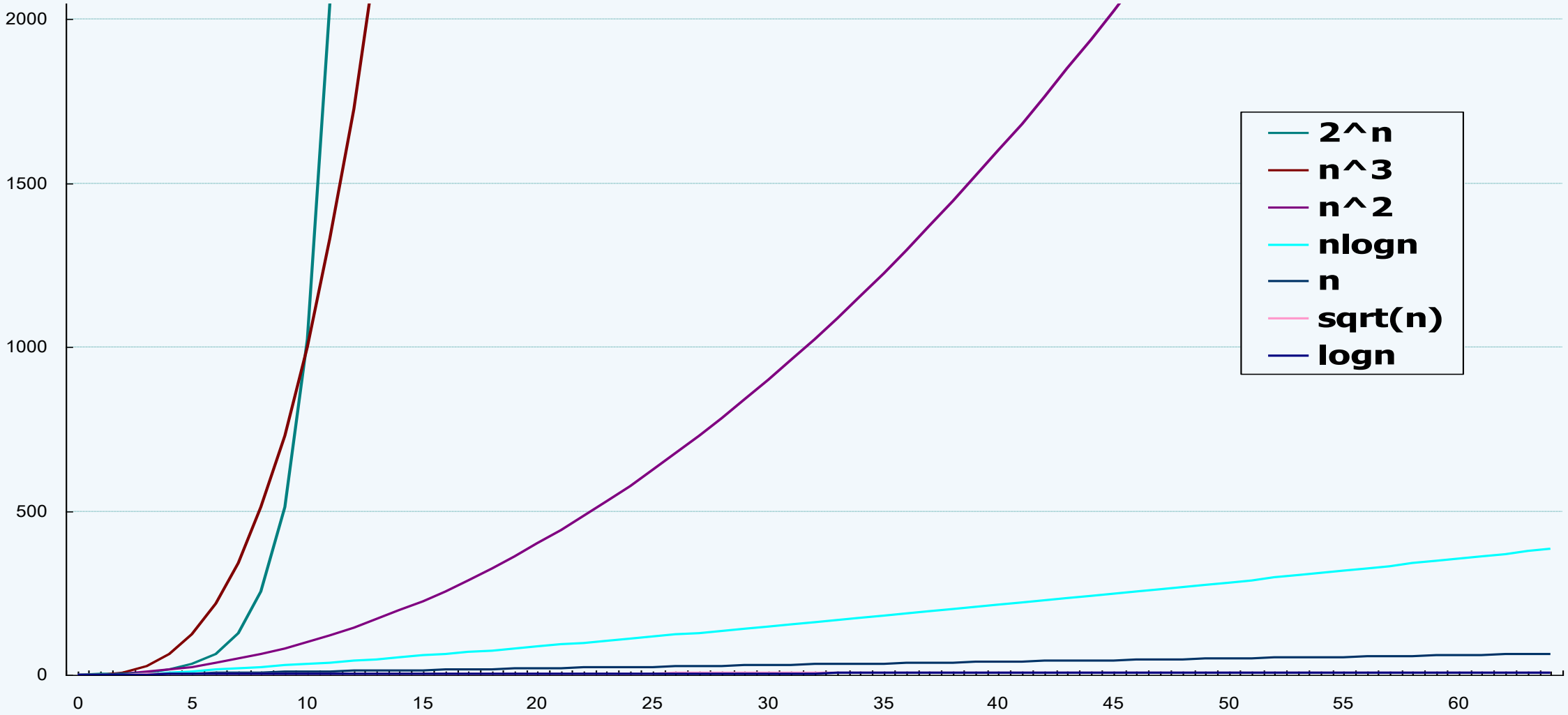
邓俊辉

deng@tsinghua.edu.cn

增长速度


30
25
20
15
10
5
0
0
5
10
15
20
25
30
35
40
45
50
55
60
2^n
n^3
n^2
nlogn
n
sqrt(n)
logn

增长速度


2000
1500
1000
500
0
0
5
10
15
20
25
30
35
40
45
50
55
60
2^n
n^3
n^2
nlogn
n
sqrt(n)
logn

层次级别

$O(1)$	常数复杂度	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	
$O(\log n)$	对数复杂度	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
$O(n)$	线性复杂度	努力目标，经常遇到	树、图的遍历
$O(n\log^* n)$		几乎几乎几乎接近线性	某些MST算法
$O(n\log\log n)$		几乎接近线性	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
$O(n^2)$	平方复杂度	所有输入对象两两组合	Dijkstra算法
$O(n^3)$	立方复杂度	不常见	矩阵乘法
$O(n^c)$，c常数	多项式复杂度	P问题 = 存在多项式算法的问题	
$O(2^n)$	指数复杂度	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、证否或计算： Fibonacci数 fib(n) = $\mathcal{O}(2^n)$

$12n + 5 = \mathcal{O}(n\log n)$

$\log^2(n^{1024} - 2*n^6 + 101) = \mathcal{O}(?)$

$\log^d n = \mathcal{O}(n^c),\ \forall c > 0,\ d > 1$

$\log^{1.001} n = \mathcal{O}(\log(n^{1001}))$

$(n^2 + 1) / (2n + 3) = \mathcal{O}(n)$

$n^{2013} = \mathcal{O}(n!)$

$n! = \mathcal{O}(n^{2013})$

$2^n = \mathcal{O}(n!)$

❖ k-Subset：任给整数集S，判定S可否划分为k个不交子集，其和均为(∑S)/k

证明或证否：(k+1)-Subset的难度，不低于k-Subset

❖ Google: small-o notation