

1.3 绪论-大O记号

备注	
学习日	@2020/03/07
视频起始	https://www.bilibili.com/video/av75509584?p=8
讲义	01.Introduction.C.Big_o.pdf

▼ 01-C-1 大O记号

随着计算规模增大，计算成本如何增加

渐进分析：在问题规模足够大后，计算成本如何增长？

Asymptotic analysis: 当 $n \gg 2$ 后，对于规模为 n 输入，算法

- 所需执行基本操作次数 $T(n) = ?$
- 需占用的储存单元数 $S(n) = ?$

主要看长远地变化趋势！

- 大O记号

$T(n) = O(f(n))$ ，存在一个 c ，当 $n \gg 2$ 后，有 $T(n) < c f(n)$

通过不断简化，可以得出反应整体算法的增长趋势：

渐进分析：大O记号

❖ 大O记号 (big-O notation) //Paul Bachmann, 1894

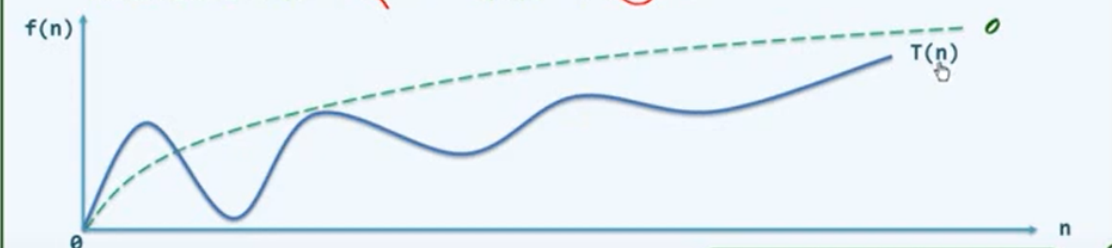
$T(n) = O(f(n))$ iff $\exists c > 0$, 当 $n \gg 2$ 后, 有 $T(n) < c \cdot f(n)$

✓ $\sqrt{5n \cdot [3n \cdot (n + 4) + 6]} < \sqrt{5n \cdot [6n^2 + 4] + 6} < \sqrt{35n^3 + 6} < 6 \cdot n^{1.5} = O(n^{1.5})$

❖ 与 $T(n)$ 相比， $f(n)$ 更为简洁，但依然反映前者的增长趋势

常系数可忽略： $O(f(n)) = O(c \cdot f(n))$

低次项可忽略： $O(n^a + n^b) = O(n^a)$, $a > b > 0$



Data Structures & Algorithms (Fall 2013), Tsinghua University

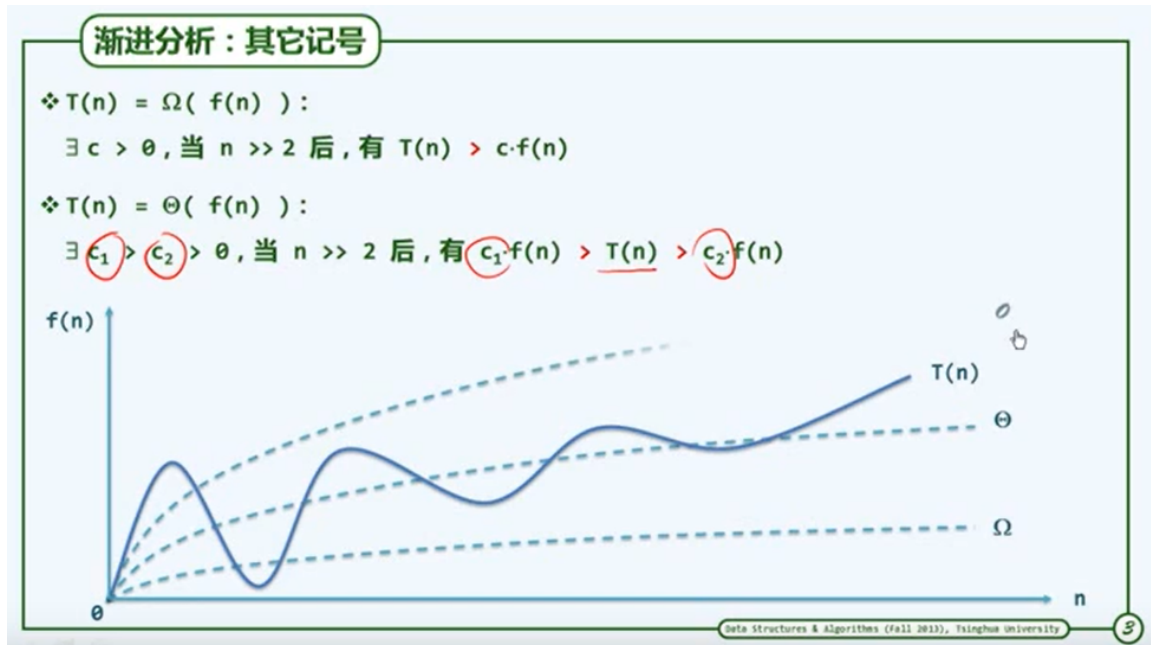
▼ 01-C-2 大Omega/Theta记号

- 大 Omega 记号

$T(n) = \Omega(f(n))$, 存在一个 c , 当 $n \gg 2$ 后, 有 $T(n) > c f(n)$

- Theta记号 (上下界定)

$T(n) = \Theta(f(n))$, 存在一个 $c_1 > c_2 > 0$, 当 $n \gg 2$ 后, 有 $c_1 f(n) > T(n) > c_2 f(n)$



- 常数复杂度 $O(1)$

❖ 常数 (constant function)

$2 = 2013 = 2013 \times 2013 = O(1)$, 甚至 $2013^{2013} = O(1)$ //含RAM各基本操作

不包含显式的循环, 只包含判断语句, 则顺序执行一般为常数复杂度

- 对数复杂度 (往往不标明具体的底数, 不影响复杂度)
 - 常底数无所谓

$$\forall a, b > 0, \log_a n = \log_a b \cdot \log_b n = \Theta(\log_b n)$$

不同的底数都可以通过常数乘法进行转化

- 常数次幂无所谓

$$\forall c > 0, \log n^c = c \cdot \log n = \Theta(\log n)$$

- 对数多项式 (poly-log func) 则只需要高次幂

$$123 \log^{123} n + \log^{10}(n^2 - n + 1) = \Theta(\log^{123} n)$$

- 这类算法非常高效。复杂度无限接近于常数

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

算法的复杂度无限接近于常数

- 多项式复杂度，一般直接取最高次项

$$a_k n^k + a_{k-1} n^{k-1} + \dots + c = \mathcal{O}(n^k)$$

- 线性函数，则幂次c=1
- 从O(n)到O(n²)则为变成习题主要覆盖的范围

直接看最高复杂度的计算即可

▼ 01-C-3 复杂度总结

- 指数 (exponential) 函数复杂度

$$T(n) = a^n$$

他永远构成了多项式函数上界

$$n^{1000} = \mathcal{O}(1.000001^n) = \mathcal{O}(2^n)$$

$$1.000001^n = \Omega(n^{1000})$$

计算成本增长极快!

- 从O(n^c)到O(2ⁿ)是从有效到无效的分水岭
O(2ⁿ)往往显而易见而O(n^c)极其不易
- 案例分析，能否把和为2m的n个正整数集合S，能否平均分为两部分，和为m

2-Subset

❖ 【问题描述】

S 包含 n 个正整数, $\sum S = 2m$

S 是否有子集 T , 满足 $\sum T = m$? $= \sum S \setminus T$

❖ 【选举人制】

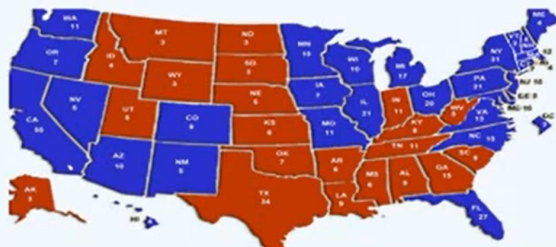
各州议会选出的选举人团投票
而不是由选民直接投票

50个州加1个特区, 共538票
获270张选举人票, 即可当选

❖ 但是...

❖ 若共有两位候选人
是否可能恰好各得269票?

55	California	11	Indiana	7	Connecticut	4	Idaho
34	Texas	11	Missouri	7	Iowa	4	Maine
31	New York	11	Tennessee	7	Oklahoma	4	New Hampshire
27	Florida	11	Washington	7	Oregon	4	Rhode Island
21	Illinois	10	Arizona	6	Arkansas	3	Alaska
21	Pennsylvania	10	Maryland	6	Kansas	3	Delaware
20	Ohio	10	Minnesota	6	Mississippi	3	D. C.
17	Michigan	10	Wisconsin	5	Nebraska	3	Montana
15	Georgia	9	Alabama	5	Nevada	3	North Dakota
15	New Jersey	9	Colorado	5	New Mexico	3	South Dakota
15	North Carolina	8	Louisiana	5	Utah	3	Vermont
13	Virginia	8	Kentucky	5	West Virginia	3	Wyoming
12	Massachusetts	8	South Carolina	4	Hawaii		
					538 = \sum		



Data Structures & Algorithms (Fall 2013), Tsinghua University

直觉算法：逐一枚举 S 的子集

$$|2^S| = 2^{|S|} = 2^n$$

这个2-Subse问题是t是NP完备问题！难以求解

- 增长速度

指数函数可能一开始增长速度不快，但是长远看来，算术复杂度增长最快！

