

1. 绪论

渐进分析

大 θ 记号

Mathematics is more in need
of good notations than
of new theorems.

- A. Turing

邓俊辉

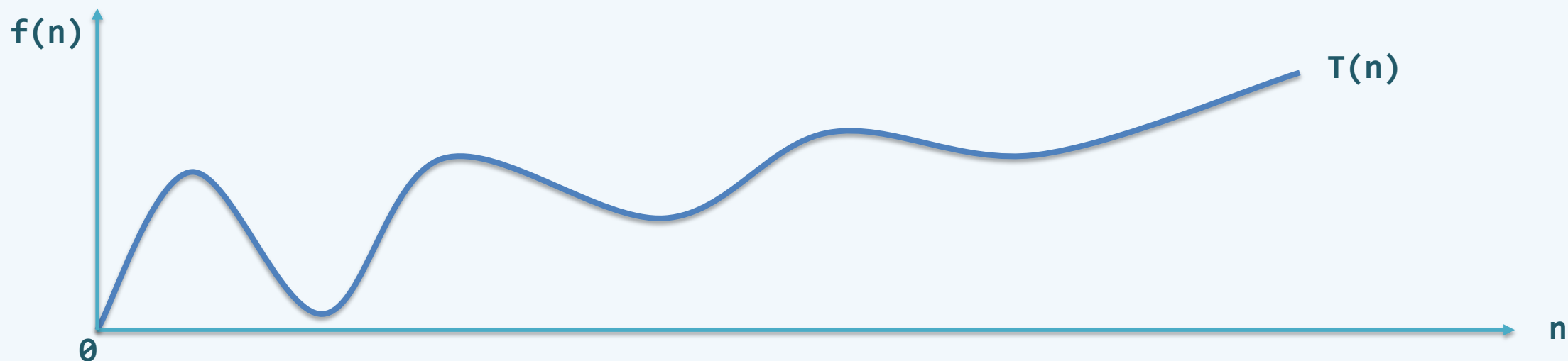
deng@tsinghua.edu.cn

渐进分析

❖ 回到原先的问题：随着问题规模的增长，计算成本如何增长？

注意：这里更关心足够大的问题，注重考察成本的增长趋势

❖ 在问题规模足够大后，计算成本如何增长？



渐进分析

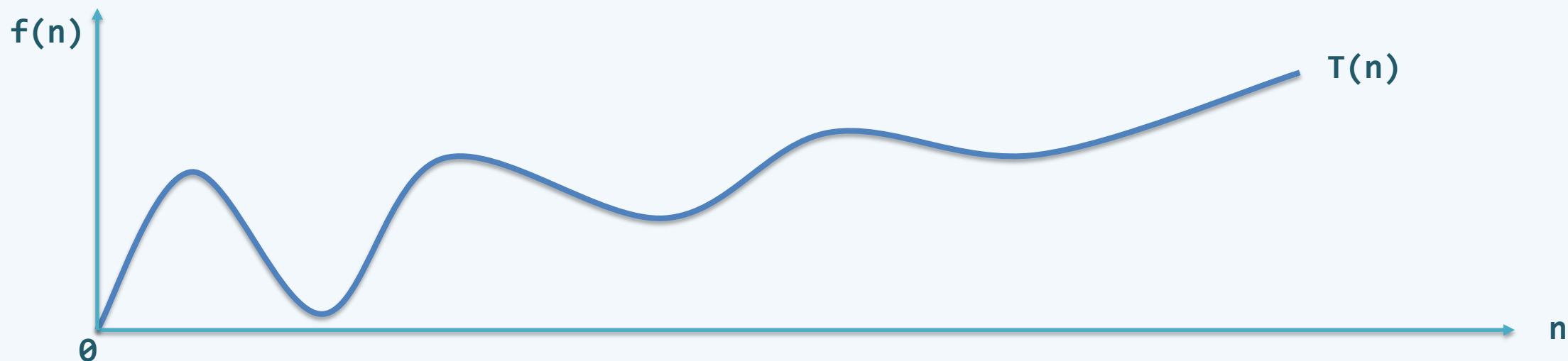
❖ 【Asymptotic analysis】

当 $n \gg 2$ 后，对于规模为 n 输入，算法

需执行的基本操作次数： $T(n) = ?$

需占用的存储单元数： $S(n) = ?$

//通常可不考虑，为什么？



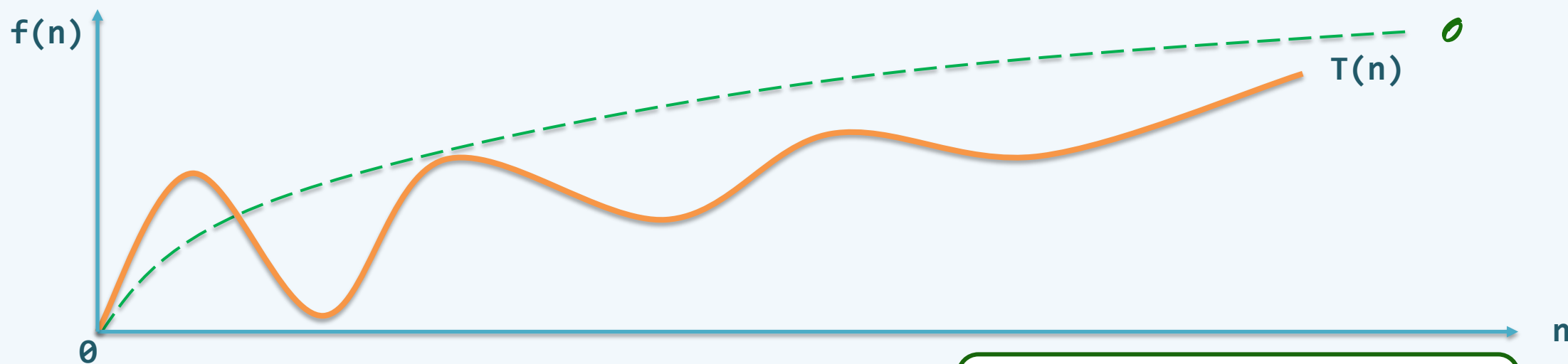
大O记号

❖ big-O notation

//Paul Bachmann, 1894

$$T(n) = \mathcal{O}(f(n)) \quad \text{iff} \quad \exists c > 0 \quad \text{s.t.} \quad T(n) < c \cdot f(n) \quad \forall n \gg 2$$

$$\text{Ex: } \sqrt{5n \cdot [3n \cdot (n+2) + 4] + 6} < \sqrt{5n \cdot [\boxed{6n^2} + 4] + 6} < \sqrt{\boxed{35n^3} + 6} < 6 \cdot n^{1.5} = \mathcal{O}(n^{1.5})$$

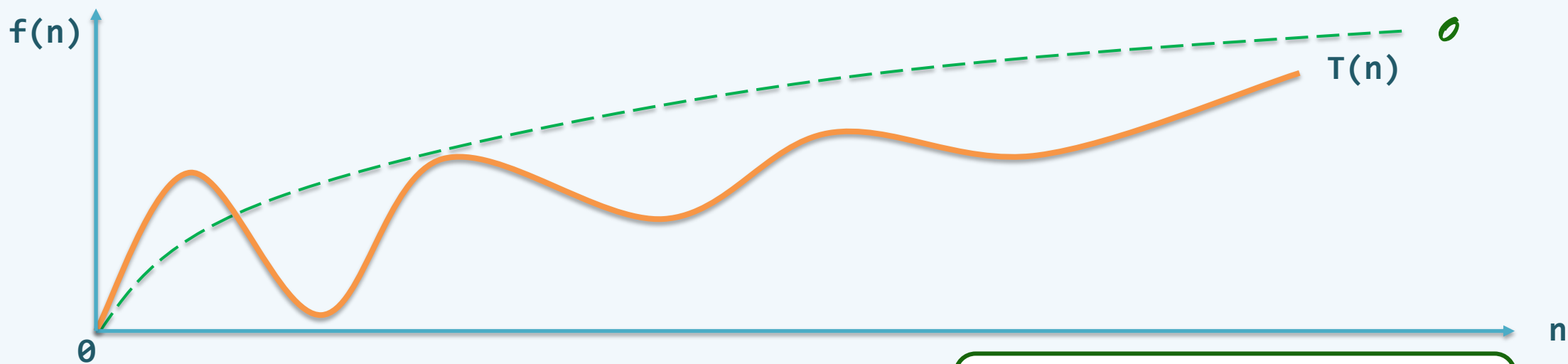


大O记号

❖ 与 $T(n)$ 相比, $f(n)$ 在形式上更为简洁, 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: $O(n^a + n^b) = O(n^a), a \geq b > 0$



其它记号

$$T(n) = \Omega(f(n)) \quad \text{iff} \quad \exists c > 0 \quad \text{s.t.} \quad T(n) > c \cdot f(n) \quad \forall n \gg 2$$

$$T(n) = \Theta(f(n)) \quad \text{iff} \quad \exists c_1 > c_2 > 0 \quad \text{s.t.} \quad c_1 \cdot f(n) > T(n) > c_2 \cdot f(n) \quad \forall n \gg 2$$

