01-C1

绪论

渐近复杂度：大O记号

Any time you are stuck on a problem, introduce more notation.

\- Chris Skinner

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

\- A. Turing

邓俊辉

deng@tsinghua.edu.cn

渐近分析

❖ 回到原先的问题：

随着问题规模的增长，计算成本如何增长？

❖ 这里更关心：

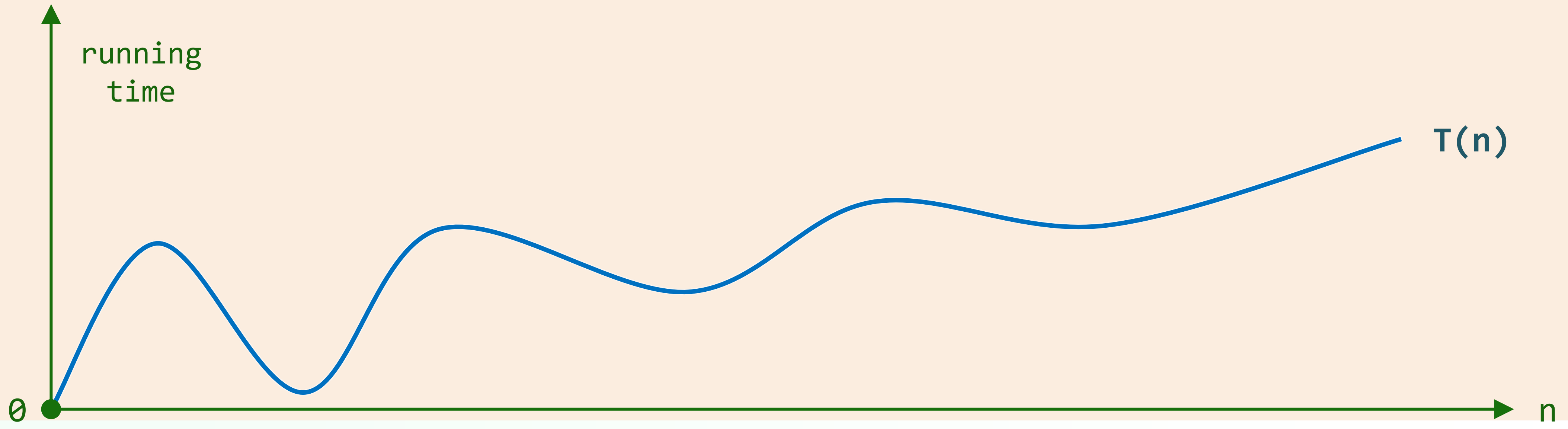
问题规模**足够大**之后，计算成本的**增长趋势**

❖ 当输入规模 $n \gg 2$ 后，算法

需执行的基本操作次数 $T(n) = ?$

❖ 如欲更为精确地估计，还可考查

需占用的存储单元数 $S(n) = ?$



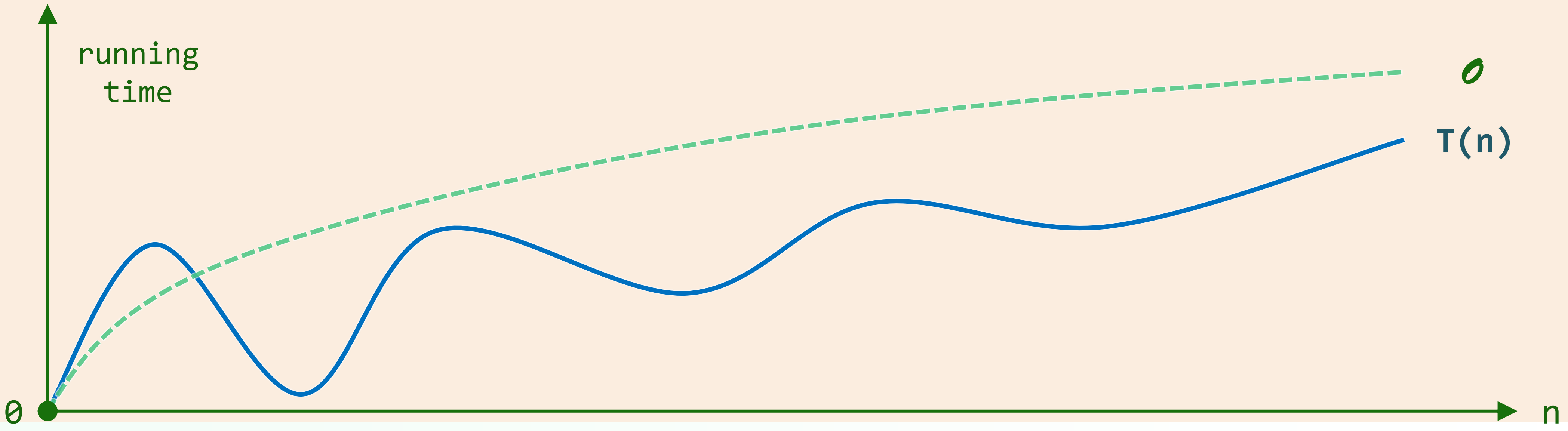
Big-O notation

- **Paul Bachmann, 1894:** $T(n) = \mathcal{O}(f(n)) \quad \text{iff} \quad \exists\, c > 0 \quad \text{s.t.} \quad T(n) < c \cdot f(n) \quad \forall\, n \gg 2$

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

- **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**

$$\mathcal{O}(f(n)) \;=\; \mathcal{O}(c \cdot f(n)) \qquad\qquad \mathcal{O}(n^a + n^b) \;=\; \mathcal{O}(n^a),\; a \geq b > 0$$



其它记号

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

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