Teoria da Complexidade

Notação Assintótica

Prof. Diego Noble

diegovnoble@gmail.com

12 de julho de 2018

Introdução

- Analisar algoritmos significa determinar os recursos computacionais que o algoritmo requer conforme o tamanho da entrada aumenta.

$\longrightarrow$ O objetivo desta aula é "concretizar" esta ideia de análise.

Introdução

- Analisar algoritmos significa determinar os recursos computacionais que o algoritmo requer conforme o tamanho da entrada aumenta.

$\longrightarrow$ O objetivo desta aula é "concretizar" esta ideia de análise.
$\longrightarrow$ Esse é passo inicial para compreender o conceito de *tratabilidade*.

Conteúdo

Conceito de Eficiência

Notação Assintótica

Conceito de Eficiência

Complexidade

- *Encontrar algoritmos eficientes para solucionar problemas computacionais.*

Mas o que é "*executar rapidamente*"?

Eficiência I

Definição 1. Um algoritmo é eficiente se, quando implementado, executa rapidamente para instâncias reais como entrada.

Eficiência II

Definição 2. Um algoritmo é eficiente se, qualitativamente e em seu pior caso, tem um desempenho superior a um algoritmo de força-bruta.

Eficiência III

Definição 3. Um algoritmo é eficiente se o seu tempo de execução é polinomial.

Eficiência III

Definição 3. Um algoritmo é eficiente se o seu tempo de execução é polinomial.

Mas n^{100} é melhor que $n^{1+0.02(\log n)}$?

Notação Assintótica

Big-oh $\mathcal{O}$

Definição 4. *Limite assintótico superior* $T(n)$ é $O(f(n))$ *se* existem constantes $c > 0$ e $n_0 \geq 0$ tal que $\forall n \geq n_0$, é o caso que $T(n) \leq c.f(n)$.

Big-oh $\mathcal{O}$

Definição 4. *Limite assintótico superior* $T(n)$ é $O(f(n))$ *se* existem constantes $c > 0$ e $n_0 \geq 0$ tal que $\forall n \geq n_0$, é o caso que $T(n) \leq c.f(n)$.

Dizemos neste caso que $T(n)$ é limitada superiormente por $f(n)$.

Exemplo

$T(n) = 10n + 8, f(n) = n^2, c = 5, n_0 = 2$

Big-oh $\mathcal{O}$

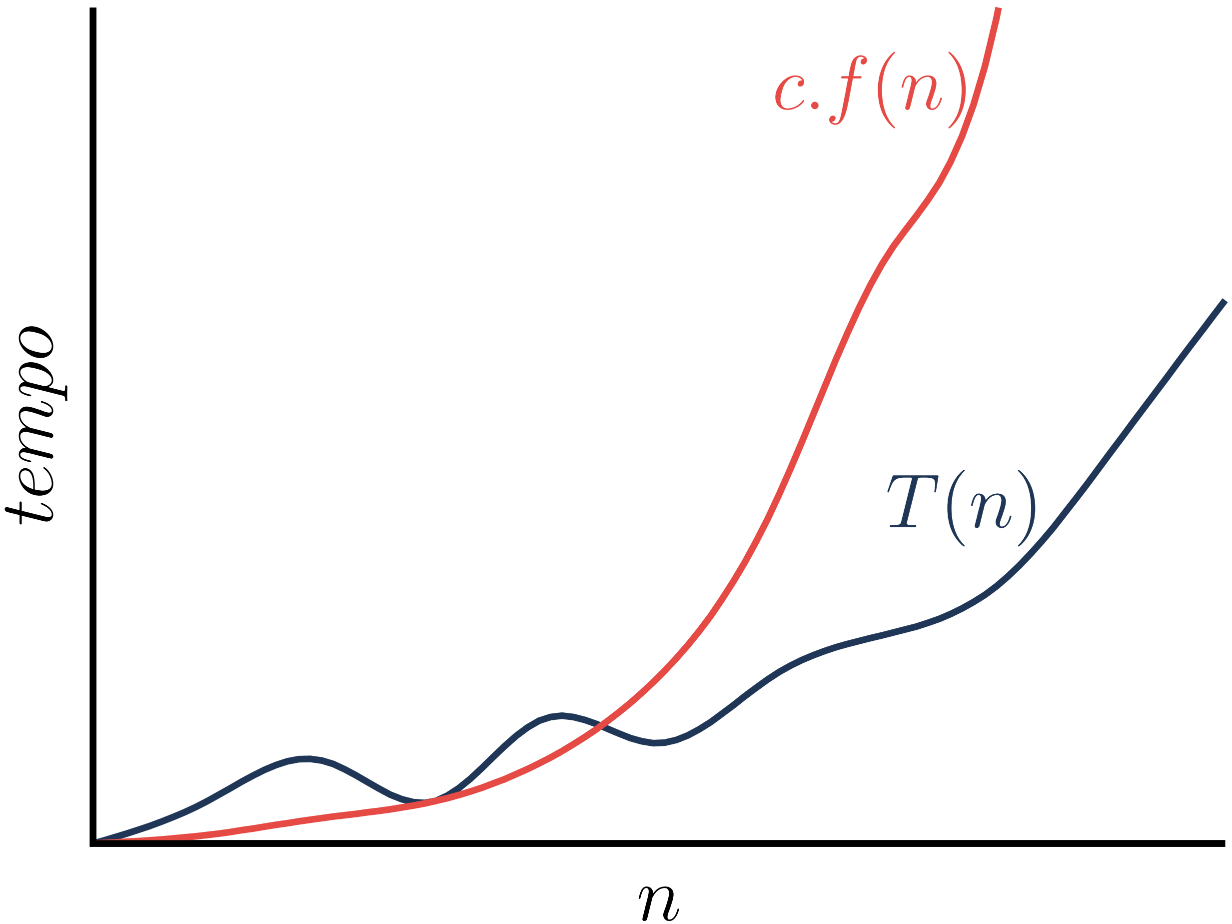

Figura 1: $T(n) \in \mathcal{O}(f(n))$

Ω

Definição 5. *Limite assintótico inferior* $T(n)$ é $\Omega(f(n))$ *se* existem constantes $c > 0$ e $n_0 \geq 0$ tal que $\forall n \geq n_0$, é o caso que $T(n) \geq c.f(n)$.

Ω

Definição 5. *Limite assintótico inferior* $T(n)$ é $\Omega(f(n))$ *se* existem constantes $c > 0$ e $n_0 \geq 0$ tal que $\forall n \geq n_0$, é o caso que $T(n) \geq c.f(n)$.

Dizemos neste caso que $T(n)$ é limitada inferiormente por $f(n)$.

Ω

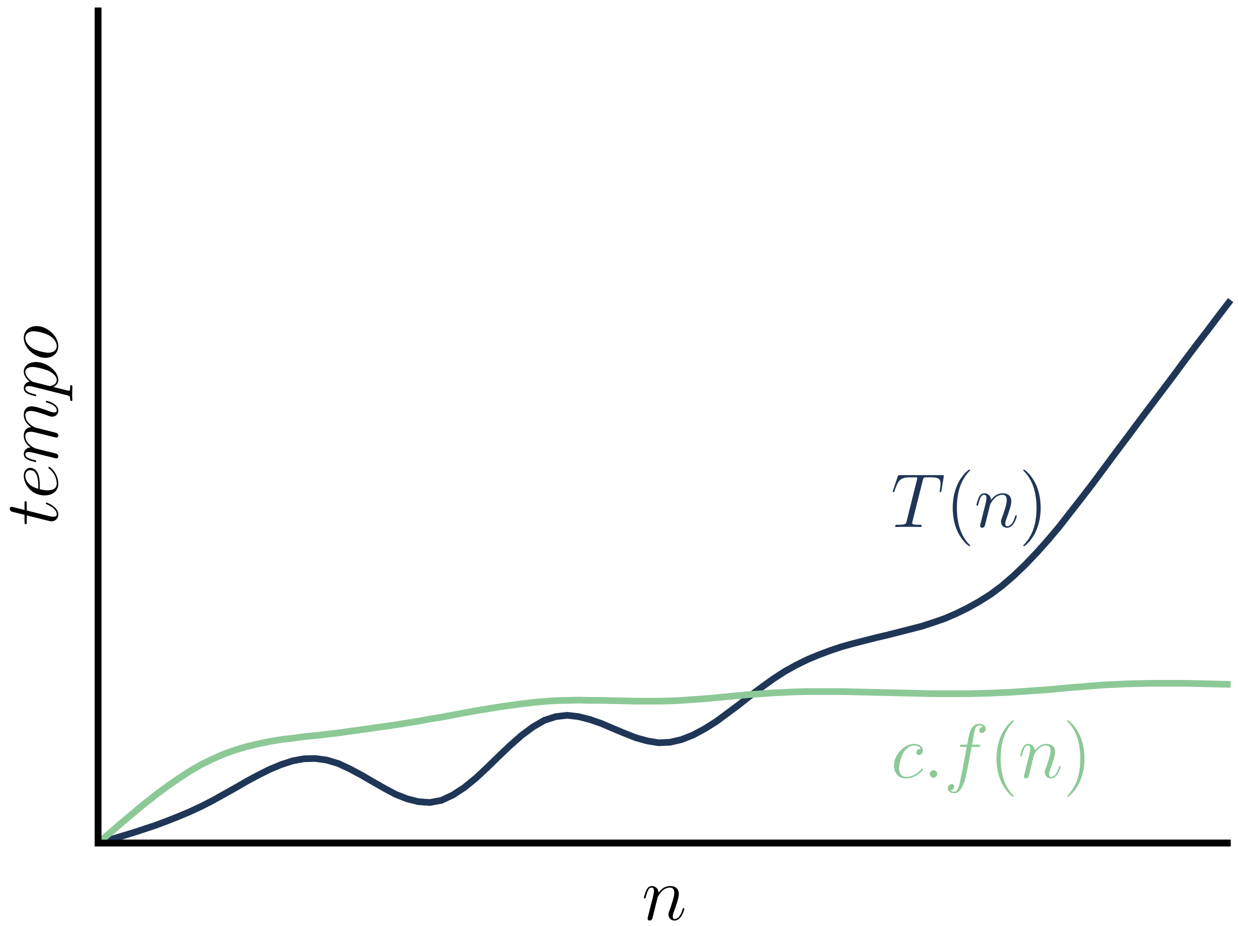

Figura 2: $T(n) \in \Omega(f(n))$

Θ

Definição 6. *Limite assintótico estrito* $T(n)$ é $\Theta(f(n))$ *se* $T(n)$ é tanto $O(f(n))$ quanto $\Omega(f(n))$.

Θ

Definição 6. *Limite assintótico estrito* $T(n)$ é $\Theta(f(n))$ *se* $T(n)$ é tanto $O(f(n))$ quanto $\Omega(f(n))$.

Dizemos neste caso que $T(n)$ é limitada estritamente por $f(n)$.

Definição 6. *Limite assintótico estrito* $T(n)$ é $\Theta(f(n))$ *se* $T(n)$ é tanto $O(f(n))$ quanto $\Omega(f(n))$.

Dizemos neste caso que $T(n)$ é limitada estritamente por $f(n)$.

- Também conhecido por limite restrito.
- A função $T(n)$ cresce dentro de um fator constante multiplicado por $f(n)$.

Θ

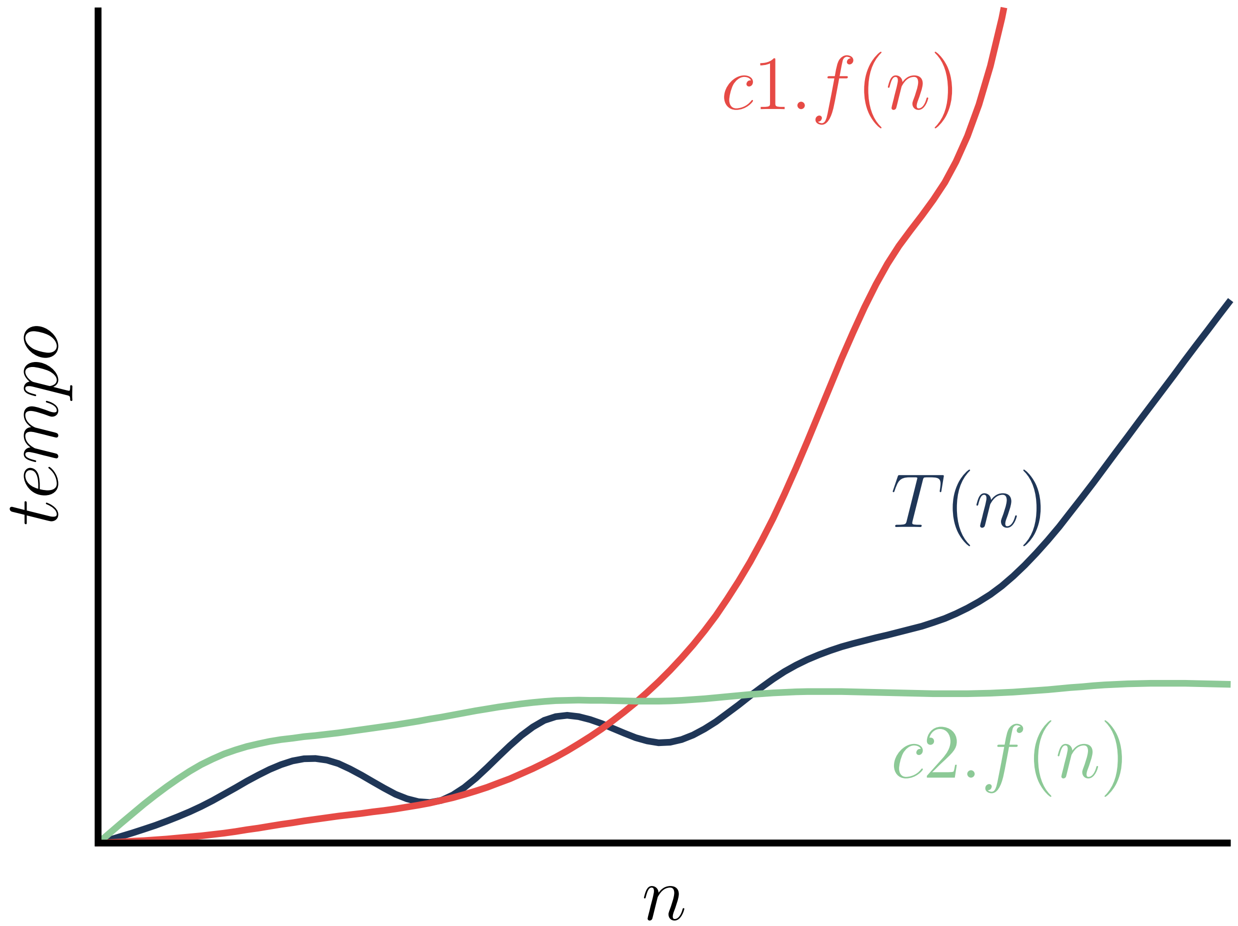

Figura 3: $T(n) \in \Theta(f(n))$

Observações

- Dadas duas funções $g = n^2$ e $f = n + 32$, anotamos que $f(n) \in \mathcal{O}(g(n))$ usando a seguinte notação: $f(n) = \mathcal{O}(g(n))$.
- Neste caso, lemos $f(n)$ *é* $\mathcal{O}(g(n))$ ao invés de nos referirmos ao sentido de igualdade usual.
- Isto porque $\mathcal{O}(g(n))$ é um conjunto de funções que tem o mesmo limite assintótico superior que $g(n)$.
- Portanto $f(n)$ é uma função que pertence a esse conjunto.

Bibliografia consultada

[1] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. *Algoritmos: Teoria e Prática*, 3 ed. Elsevier, 2012.

[2] Kleinberg, J., and Tardos, E. *Algorithm Design*, 1 ed. Addison-Wesley, 2005.