PAA - Aula 04
Análise Assintótica

Análise assintótica é um ferramental matemático que

- nos dá vocabulário para descrever a eficiência de algoritmos
- e permite formalizar vários princípios da análise.

Princípios da análise de algoritmos:

- análise de pior caso,
- pouca atenção para constantes multiplicativas e termos de menor ordem,
- foco no comportamento dos algoritmos para entradas grandes.

Ela é simples o bastante para ignorar aspectos sobre os quais

- não temos controle ou interesse quando analisamos um algoritmo, como
  - detalhes de implementação, linguagem, compilação e arquitetura.

Mas é precisa o bastante para diferenciar algoritmos,

- e permitir escolhas embasadas ao resolver um problema,
  - particularmente quando as entradas são grandes.

Notação O

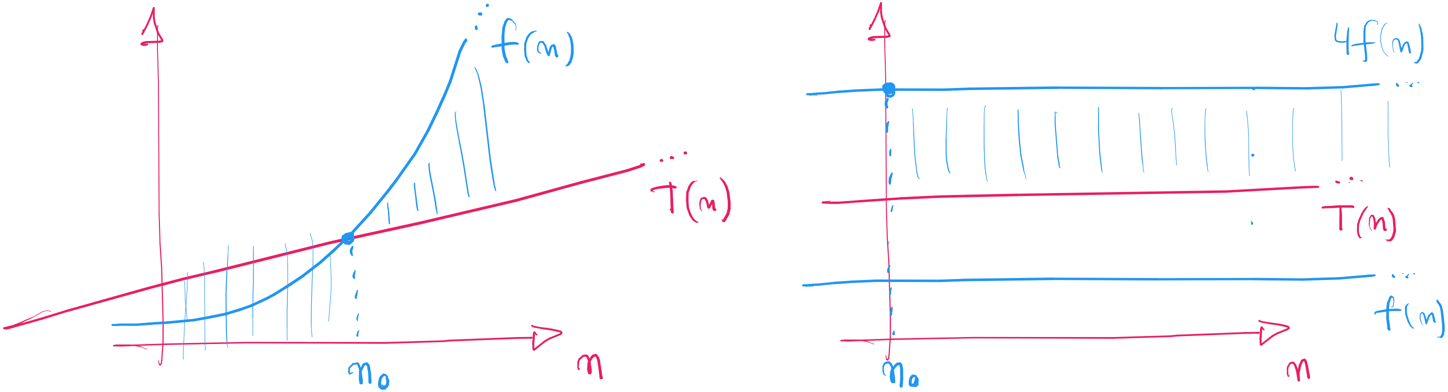
$T(n) \in O(f(n)) \iff T(n)$ é $O(f(n)) \iff T(n) = O(f(n))$

Considere duas funções T e f, definidas nos inteiros positivos.

- $T(n) = O(f(n))$ significa que, para um n suficientemente grande,
  - uma constante vezes f(n) limita superiormente T(n).
- A intuição é que, assintoticamente, T cresce no máximo tão rápido quanto f.

↳ o termo dominante de T sem as constantes multiplicativas

Exemplos gráficos:



Definição formal:

- $T(n) = O(f(n))$ se existem constantes n_0 e c tais que
  $T(n) \leq c f(n)$ para todo $n \geq n_0$
- Note que n_0 e C não podem depender de n.

Definição formal:

- $T(n) = O(f(n))$ se existem constantes n_0 e $c > 0$
  - tais que $T(n) <= c * f(n)$ para todo $n >= n_0$.
- Note que n_0 e c não podem depender de n.

Exemplo 1: $T(n) = 10\, n \lg n + 10\, n$ é $O(n \lg n)$?

Prova: Queremos encontrar constantes c e n_0 tais que,

- para todo $n \geq n_0$ temos $T(n) = 10 n \lg n + 10 n \leq c\, n \lg n$

Para tanto, vamos manipular T(n) usando limitantes superiores

$$\Rightarrow \boxed{T(n) = 10 n \lg n + 10 n \underset{p/\, n \geq 2}{\leq} 10 n \lg n + 10 n \lg n = 20 n \lg n}$$

Como, para $c = 20$ e $n_0 = 2$ temos $T(n) \leq c\, n \lg n$, então $T(n) \in O(n \lg n)$

Exemplo 2: T(n) = a_k n^k + a_k-1 n^k-1 + ... + a_1 n + a_0 é O(n^k)?

$$T(n) = \sum_{i=0}^{k} a_i n^i$$

Prova: Queremos encontrar constantes c e n_0 tais que,

- para todo $n \geq n_0$ temos $T(n) = \sum_{i=0}^{k} a_i n^k \leq c \cdot n^k$
- Para tanto, vamos manipular T(n) usando limitantes superiores

$$\boxed{T(n)} = \sum_{i=0}^{k} a_i n^i \leq \sum_{i=0}^{k} |a_i| n^i \underset{p/\, n \geq 0}{\leq} \sum_{i=0}^{k} |a_i| \cdot n^k = n^k \left(\sum_{i=0}^{k} |a_i| \right)$$

- Como, para $c = |a_0| + |a_1| + \ldots + |a_k|$ e $n_0 = 1$
  - temos $T(n) \leq c \cdot n^k$, concluímos que $T(n) \in O(n^k)$

Exemplo 3: T(n) = n^k não é O(n^k-1).

Prova: Por contradição, vamos supor que T(n) = n^k é O(n^k-1), ou seja,

- supomos que existem c e n_0 tais que, para $n \geq n_0$
  - temos $T(n) = n^k \leq c\, n^{k-1}$
- Como $n \geq 0$ temos $\frac{n^k}{n^{k-1}} \leq \frac{c \cdot n^{k-1}}{n^{k-1}} \Rightarrow n \leq c$
- Como o resultado deve valer para todo $n \geq n_0$
  - $c \geq n$ não é constante, o que é um absurdo.

Exemplo 4: T(n) = 2^(n+10) é O(2^n)? $T(n) = 2^{n+10}$

Prova: Queremos encontrar constantes c e n_0 tais que,

- para todo $n \geq n_0$ temos $T(n) = 2^{n+10} \leq c\,2^n$

$T(n) \leq 2^{n+10} = 2^n \cdot 2^{10}$

↳ constante

- Portanto, para $c = 2^{10}$ e $n_0 = 1$ temos $T(n) \leq c \cdot 2^n \Rightarrow T(n) \in O(2^n)$

Exemplo 5: T(n) = 2^(10n) não é O(2^n). $T(n) = 2^{10n}$

Prova: Por contradição, vamos supor que T(n) = 2^(10n) é O(2^n), ou seja,

- supomos que existem c e n_0 tais que, para $n \geq n_0$
  - temos $T(n) = 2^{10n} \leq c \cdot 2^n$
- Como $2^{a.b} = (2^a)^b$ temos

$T(n) = 2^{10n} = (2^n)^{10} = 2^n \cdot (2^n)^9 = 2^n \cdot 2^{9n} \leq c \cdot 2^n \Rightarrow c \geq 2^{9n}$

- Como o resultado deve valer para todo $n \geq n_0$,
  - $c \geq 2^{9n}$ não é constante, o que é uma contradição.

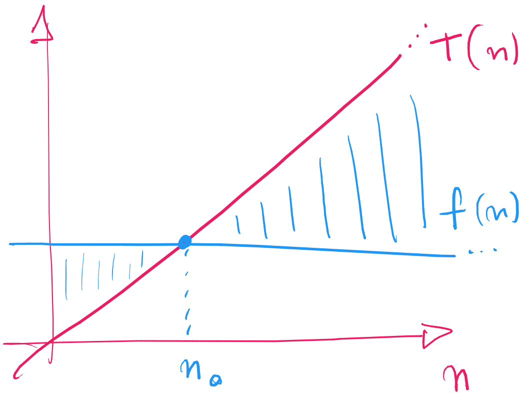
$T(n) = 2^{10n} = (2^{10})^n$

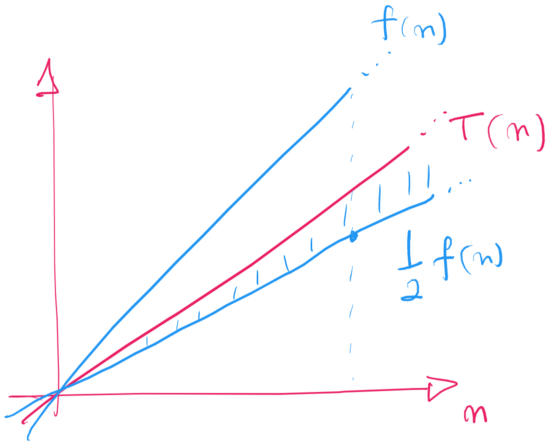
Notação Omega Ω

Considere duas funções T e f, definidas nos inteiros positivos.

- $T(n) = \Omega(f(n))$ significa que, para um n suficientemente grande,
    - uma constante vezes f(n) limita inferiormente T(n).
- A intuição é que, assintoticamente, T cresce pelo menos tão rápido quanto f.

Exemplos gráficos:





Definição formal:

- $T(n) = \Omega(f(n))$ se existem constantes n_0 e c tais que
  $T(n) \geq c \cdot f(n)$ para todo $n \geq n_0$
- Note que n_0 e c não podem depender de n.

Exemplo 1: T(n) = 2^(n+10) é Omega(2^n)? $T(n) = 2^{n+10}$

Prova: Queremos encontrar constantes c e n_0 tais que,

- para todo $n \geq n_0$ temos $T(n) = 2^{n+10} \geq c \cdot 2^n$

$$\Rightarrow \boxed{T(n) = 2^{n+10} = 2^n \cdot 2^{10}} \geq 2^n$$

- Portanto, para $c = 2^{10}$ e $n_0 = 1$ temos $T(n) \geq c \cdot 2^n$

$c = 1$

Exemplo 2: T(n) = 2^(10n) é Omega(2^n)? $T(n) = 2^{10n} = (2^{10})^n$

Prova: Queremos encontrar constantes c e n_0 tais que,

- para todo $n \geq n_0$ temos $T(n) \geq c \cdot 2^n$

$$\boxed{T(n) = 2^{10n} = (2^n)^{10} = 2^n \cdot 2^{9n} \geq 2^n}$$

p/ $n \geq 0$

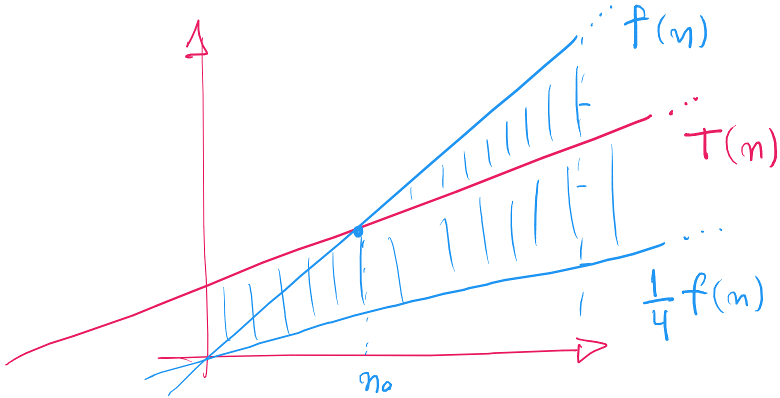
- Portanto, para $c = 1$ e $n_0 = 1$ temos $T(n) \geq c \cdot 2^n$

Notação Theta — Θ

Considere duas funções T e f, definidas nos inteiros positivos.

- $T(n) = \Theta(f(n))$ significa que, para um n suficientemente grande,
    - f(n) limita T(n) tanto inferiormente quanto superiormente.
- A intuição é que, assintoticamente, T e f crescem igualmente rápido.

Exemplo gráfico:



Definição formal:

- $T(n) = \Theta(f(n))$ se existem constantes n_0, $c_1 > 0$, $c_2 > 0$ tais que $c_1 f(n) \leq T(n) \leq c_2 f(n)$ para todo $n \geq n_0$
- Note que n_0, c_1, c_2 não podem depender de n.

Exemplo 1: $T(n) = \max\{f, g\}$ é Theta(f(n) + g(n)).

Prova: Queremos encontrar constantes n_0, c_1, c_2 tais que,

- para todo $n \geq n_0$ temos $c_1(f(n)+g(n)) \leq T(n) \leq c_2(f(n)+g(n))$
- Buscando primeiro o limitante superior temos

$$T(n) = \max\{f(n), g(n)\} \leq 1(f(n)+g(n))$$

  - Portanto, para $c_2 = 1$ e $n_0 = 1$ temos $\boxed{T(n) \leq c_2(f(n)+g(n))}$
- Buscando agora o limitante inferior observamos que

$$f(n)+g(n) = \max\{f(n), g(n)\} + \underbrace{\min\{f(n), g(n)\}}_{\leq \max\{f(n), g(n)\}} \leq 2\max\{f(n), g(n)\}$$

- Assim $T(n) = \max\{f(n), g(n)\} \geq (f(n)+g(n))/2$
  - Portanto, para $c_1 = 1/2$ e $n_0 = 1$ temos $\boxed{T(n) \geq c_1(f(n)+g(n))}$

Exemplo 2: Para $T(n) = (1/2)n^2 + 3n$ responda

- T(n) é O(n)? Não
- T(n) é Omega(n)? Sim
- T(n) é O(n^3)? Sim
- T(n) é Theta(n^2)? Sim → Provar!!!

$$T(n) \in o(f(n))$$

$$T(n) \in \omega(f(n))$$

$$T(n) = \frac{1}{2}n^2 + 3n^1$$

(Bônus) Exemplo 3: $T(n) = 2^n + 15n^2$ é $\Theta(2^n)$?

Prova: Queremos encontrar constantes c_1, c_2 e n_0 tais que,

- para todo $n \geq n_0$, temos $c_1 \cdot 2^n \leq T(n) \leq c_2 \cdot 2^n$

- Buscando primeiro o limitante inferior temos

$$T(n) = 2^n + 15n^2 \geq 2^n$$

  - Portanto, para $c_1 = 1$ e $n_0 = 0$ temos $c_1 \cdot 2^n \leq T(n)$.

- Buscando agora o limitante superior temos

$$\begin{aligned} T(n) &= 2^n + 15n^2 \\ &\leq 2^n + 2^n \quad \text{(para } n \geq 16\text{, pois } 2^n \geq 15n^2 \rightarrow \\ &\qquad\qquad \rightarrow n \geq \lg (15\, n^2) \rightarrow n \geq \lg 15 + 2 \lg n \rightarrow \\ &\qquad\qquad \rightarrow n - 2 \lg n \geq \lg 15) \\ &= 2 \cdot 2^n \end{aligned}$$

  - Portanto, para $c_2 = 2$ e $n_0 = 16$ temos $T(n) \leq c_2 \cdot 2^n$.

Note que, quando demonstramos que uma função é Theta de outra,

- podemos chegar a n_0s diferentes para os limitantes inferior e superior.
  - No final usamos o maior dentre eles.

(Bônus) Notação little omicron

Considere duas funções T e f, definidas nos inteiros positivos.

- $T(n) = \text{littleomicron}(f(n))$ significa que, para qualquer constante $c > 0$,
    - existe $n_0 > 0$ tal que $T(n) <= c * f(n)$ para todo $n >= n_0$.
- Intuitivamente, quer dizer que T cresce assintoticamente mais devagar que f.

Exemplo: para qualquer $k >= 1$ temos $T(n) = n^{(k-1)} = \text{littleomicron}(n^k)$?

Prova: Tome uma constante $c > 0$ arbitrária.

- Queremos encontrar um limitante inferior n_0 para n tal que

$$T(n) = n^{(k-1)} <= c * n^k \rightarrow 1 <= c * n \rightarrow n >= 1/c$$

- Escolhendo $n_0 = 1/c$ temos $T(n) <= c * n^k$.
- Como c foi escolhido arbitrariamente a prova está concluída.

(Bônus) Notação little omega

Considere duas funções T e f, definidas nos inteiros positivos.

- T(n) = littleomega(f(n)) significa que, para qualquer constante c > 0
    - existe n_0 > 0 tal que T(n) >= c * f(n) para todo n >= n_0.
- Intuitivamente, quer dizer que T cresce assintoticamente mais rápido que f.

Exemplo: para qualquer k >= 1 temos T(n) = n^k = littleomega(n^(k-1))?

Prova: Tome uma constante c > 0 arbitrária.

- Queremos encontrar um limitante inferior n_0 para n tal que

  T(n) = n^k >= c * n^(k-1) → n >= c

- Escolhendo n_0 = c temos T(n) >= c * n^(k-1).
- Como c foi escolhido arbitrariamente a prova está concluída.