26) TEMPO DE EXECUÇÃO

Os limites da decidibilidade

→ Infelizmente, os limites da computação não se resumem apenas entre o que é decidível e o que não é.

→ Dentro do conjunto de ~~linguagens~~ problemas decidíveis, temos o que é tratável e o que é intratável.

→ Resolver um problema exige o uso de recursos, como memória e tempo.

Análise de algoritmos

→ Nos permite verificar corretude, prever desempenho, comparar soluções sem que seja necessário implementá-los.

→ Importante pois, em geral, não existe um único algoritmo que resolve um problema, ou porque a primeira solução pode ser muito ruim.

Tempo de execução

→ Vários fatores afetam o tempo de execução.
↳ Mais fitas, entrada pequena, representação, implementação...

→ Precisamos de um conceito simples e geral.

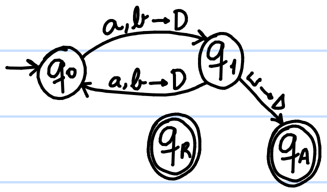
→ Para descrever o tempo de execução, levamos em conta o tamanho da entrada (o comprimento da cadeia) e contamos o número de passos feitos no pior caso.

Tempo de execução

DEFINIÇÃO: Seja M uma máquina de Turing determinística decisora. O tempo de execução de M é a função $f: \mathbb{N} \to \mathbb{N}$ onde $f(n)$ é o número máximo de passos que M usa sobre entradas de comprimento n.

DEFINIÇÃO: Seja N uma máquina de Turing não-determinística decisora. O tempo de execução de N é a função $f: \mathbb{N} \to \mathbb{N}$, onde $f(n)$ é o número máximo de passos que N usa sobre qualquer ramo de sua computação sobre entradas de tamanho n.

→ Note como isso implica em ter, a qualquer momento, no máximo $f(n)$ posições não-brancas na fita.



$q_0\, a b b a b$

$a\, q_1\, b b a b$

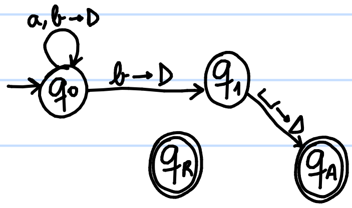
$a b\, q_0\, b a b$

$a b b\, q_1\, a b$

$a b b a\, q_0\, b$

$a b b a b\, q_1$

$a b b a b \sqcup q_A$



$q_0\, a b a a b$

$a\, q_0\, b a a b$

$a b\, q_0\, b a a b$ | $a b\, q_1\, b a a b$

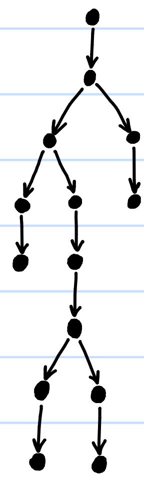
$a b b\, q_1\, a a b$ | $a b b\, q_0\, a a b$ | $a b b\, q_R\, a a b$

$a b b a\, q_R\, a b$ | $a b b a\, q_0\, a b$

$a b b a a\, q_0\, b$

$a b b a a b\, q_0$ | $a b b a a b\, q_1$

$a b b a a b \sqcup q_R$ | $a b b a a b \sqcup q_A$



Exemplo: $A = \{a^k b^k : k \geq 0\}$

M_1 = " Sobre a cadeia de entrada w:

(1) Faça uma varredura na fita e rejeite se for encontrado um a à direita de um b. $\leq 2n$

(2) Repita enquanto existem a's e b's na fita: $\leq n/2$

Faça uma varredura na fita, marcando um único a e um único b. $\leq 2n$ } n^2

(3) Se ainda há a's após todos os b's serem marcados ou se ainda há b's após todos os a's serem marcados, rejeite. Caso contrário, aceite. " $\leq n$

$= 2n + n^2 + n = n^2 + 3n$

M_2 = " Sobre a cadeia de entrada w:

(1) Faça uma varredura na fita e rejeite se for encontrado um a à direita de um b. $\leq 2n$

(2) Repita enquanto existem a's e b's na fita: $\leq \lg n$

Faça uma varredura na fita, verificando se o número total de a's e b's restantes é par ou ímpar. Se for ímpar, rejeite. $\leq 2n$

Faça outra varredura na fita, cortando alternadamente um a sim e outro não começando com o primeiro a, e então cortando um b sim e outro não começando com o primeiro b. $\leq 2n$ } $4n \lg n$

(3) Se nenhum a e nenhum b permanecerem na fita, aceite. Caso contrário, rejeite. " $\leq n$

$= 2n + 4n \lg n + 2n = 4n \lg n + 4n$

M_3 = " Sobre a cadeia de entrada w :

(1) Faça uma varredura na fita e rejeite se for encontrado um a à direita de um b. $\leqslant 2n$

(2) Faça uma varredura nos a's até encontrar o primeiro b. Ao mesmo tempo, copie os a's para a fita 2. $\leqslant n$

(3) Faça uma varredura nos b's sobre a fita 1 até o final da entrada. Para cada b lido na fita 1, marque um a na fita 2. Se todos os a's estiverem marcados antes que todos os b's sejam lidos, rejeite. $\leqslant n$

(4) Se todos os a's tiverem sido marcados, aceite. Se restar algum a, rejeite. " $\leqslant n$

$= 2n + n + n + n = 5n$

27) NOTAÇÃO ASSINTÓTICA

Notação assintótica

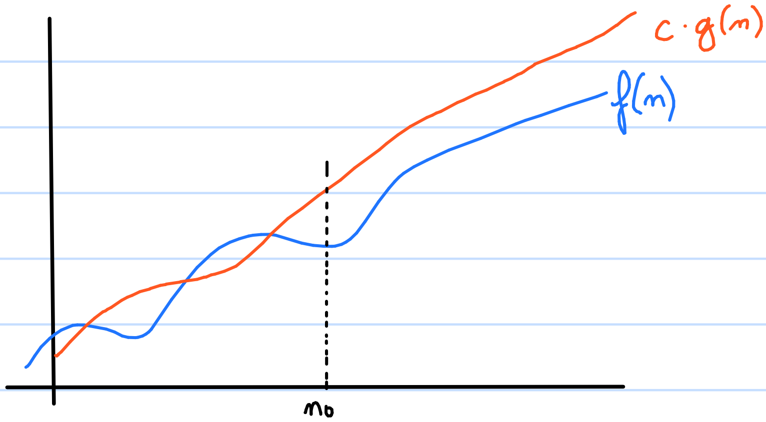
→ É uma abstração matemática que ajuda muito na análise do tempo de execução

↳ Em uma expressão, o termo de mais alta ordem domina os outros e os coeficientes (ex: $5n^4 + 3n^2 - 10n + 100$).

Notação O

DEFINIÇÃO: Sejam f e g funções $f, g : \mathbb{N} \to \mathbb{R}^+$.

Dizemos que $f(n) = O(g(n))$ se existem inteiros positivos c e n_0 tais que para todo $n \geq n_0$, $f(n) \leq c \cdot g(n)$.

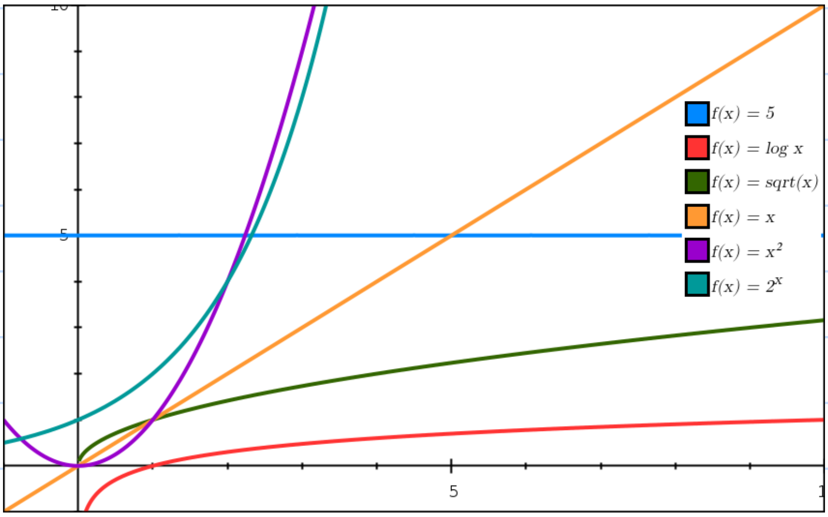


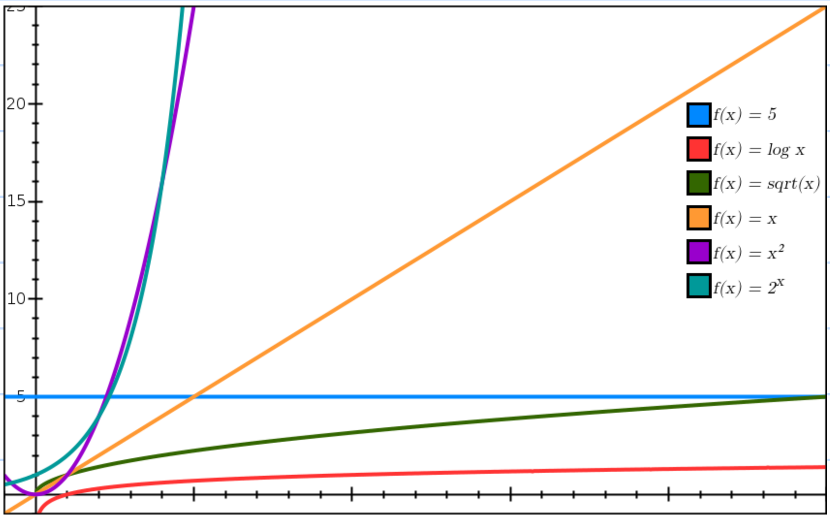
Exemplos

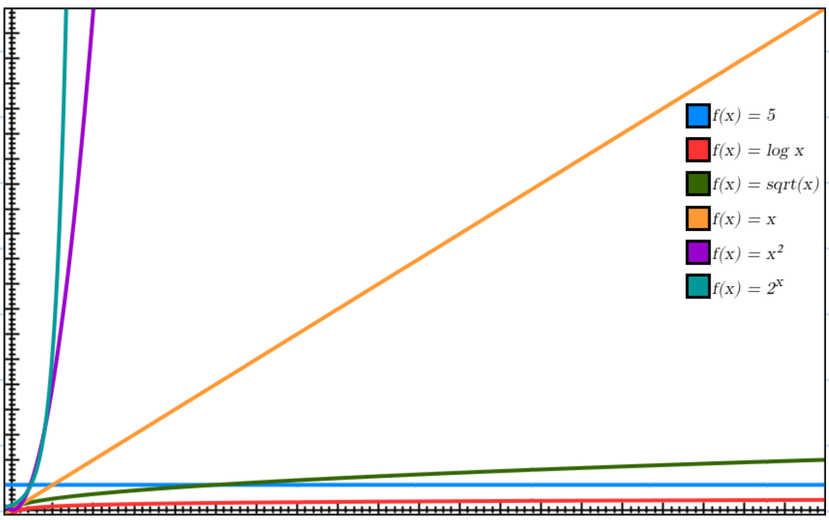
$f(n) = 4n^3 + 5n^2 - 3n + 15$ $\qquad$ $f(n) = O(n^2) + O(\lg n)$

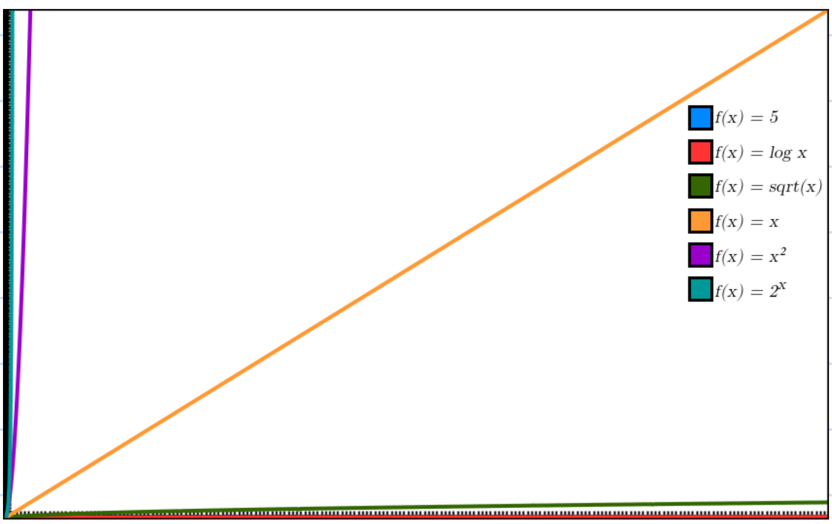
Revendo o tempo das máquinas

→ Observe como o tempo para resolver A depende do modelo de computação


$f(x) = 5$
$f(x) = \log x$
$f(x) = \mathrm{sqrt}(x)$
$f(x) = x$
$f(x) = x^2$
$f(x) = 2^x$
5
5


$f(x) = 5$
$f(x) = \log x$
$f(x) = \mathrm{sqrt}(x)$
$f(x) = x$
$f(x) = x^2$
$f(x) = 2^x$
20
15
10
5


$f(x) = 5$
$f(x) = \log x$
$f(x) = \mathrm{sqrt}(x)$
$f(x) = x$
$f(x) = x^2$
$f(x) = 2^x$

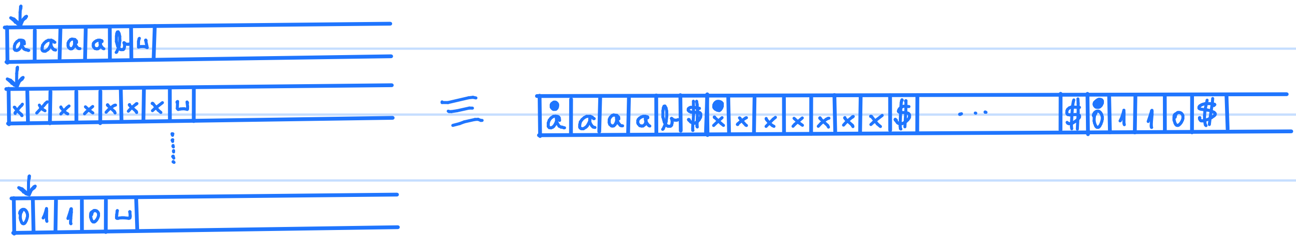

$f(x) = 5$
$f(x) = \log x$
$f(x) = \mathrm{sqrt}(x)$
$f(x) = x$
$f(x) = x^2$
$f(x) = 2^x$

28) TEMPO NOS DIFERENTES MODELOS DE COMPUTAÇÃO

Tempo nos diferentes modelos de computação

TEOREMA: Toda máquina de Turing multifita de tempo $t(n) \geq n$ tem uma máquina de Turing de única fita equivalente de tempo $O(t^2(n))$.

IDEIA DA DEMONSTRAÇÃO: Dada uma máquina M com k fitas que roda em tempo $t(n)$, construímos S com uma fita que simule M e mostramos o tempo levado na simulação.



A máquina S tem em sua fita o conteúdo das k fitas de M, com cada posição da cabeça de M marcada nas células.

Um passo de M: $\delta(q, s_1, s_2, \dots, s_k) = (p, t_1, t_2, \dots, t_k, d_1, d_2, \dots, d_k)$

Para simular, S percorre sua fita para encontrar os símbolos que estão sob as cabeças e depois percorre a fita para aplicar as mudanças: como cada uma das k porções tem $\leq t(n)$ posições, isso leva tempo $O(t(n))$.

Tempo total = inicializar a fita + simular os $t(n)$ passos de M

$$= O(n) + t(n) \cdot O(t(n))$$

$$= O(t^2(n))$$

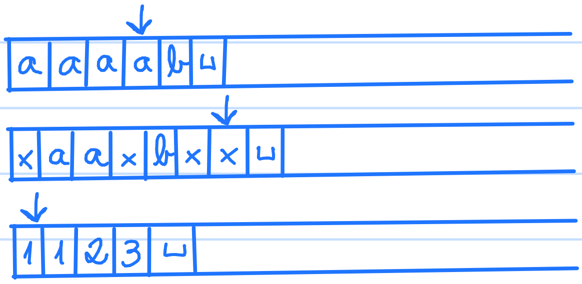
CQD

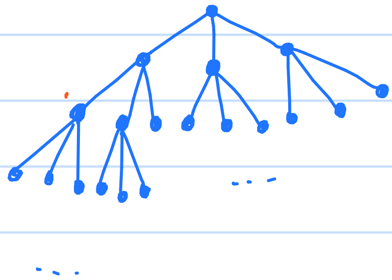
RESUMO DA IDEIA:

Há $t(n)$ passos para serem simulados e cada um leva tempo $\leq k \cdot t(n) = O(t(n))$ para ser simulado.

TEOREMA: Toda máquina de Turing não-determinística de uma única fita de tempo $t(n) \geq n$ tem uma máquina de Turing determinística de única fita equivalente de tempo $2^{O(t(n))}$.

IDÉIA: Dada uma máquina não-determinística N que roda em tempo $t(n)$, construímos D determinística que simule N e mostramos o tempo levado na simulação.





A máquina D tem 3 fitas e simula N ao explorar a árvore de computação de N no estilo busca em largura: todo nó dessa árvore tem $\leq b$ filhos (b é o maior nº de escolhas dado pela função de transição de N) e essa árvore tem altura $t(n)$

$\Rightarrow$ o total de folhas da árvore é $\leq b^{t(n)}$

O que D faz ao visitar cada nó é simular a computação da raiz até o nó, o que leva tempo $O(t(n))$.

Como há $\leq c \times$ nº folhas nós, o tempo de D é $O(t(n)\, b^{t(n)})$.

Note que $O(t(n)\, b^{t(n)}) = 2^{O(t(n))}$.

Agora, D pode ser simulada em uma única fita, o que nos dá tempo total $(2^{O(t(n))})^2 = 2^{O(t(n))}$.

CQD

RESUMO DA IDÉIA:

Há $\leq b^{t(n)}$ folhas = caminhos a serem simulados, cada um levando tempo $O(t(n))$. E $O(t(n) \cdot b^{t(n)})$ é $2^{O(t(n))}$.★

Essa simulação é numa máquina de 3 fitas, mas $O((2^{O(t(n))})^2)$ é $2^{O(t(n))}$.

★
$$c\,t(n)\,b^{t(n)} = 2^{\log_2(c\,t(n)\,b^{t(n)})} = 2^{\log_2 c + \log_2 t(n) + \log_2 b^{t(n)}}$$
$$= 2^{\log_2 c + \log_2 t(n) + t(n)\log_2 b} = 2^{O(1) + O(\log t(n)) + O(t(n))} = 2^{O(t(n))}$$