INF 1721
Análise de Algoritmos

Leitura recomendada:

Capítulos 1 a 3 do Cormen

Algoritmo

"Um procedimento para resolver problemas matemáticos em um número finito de passos que envolve frequentemente a repetição de uma operação, ou, no sentido amplo, um procedimento passo a passo para atingir um objetivo ou resolver um problema" (Webster)

Matemático árabe *Al-Khowarizmi* (825 DC)
"Rules of Restoration and Reduction"

Algoritmo

Algoritmo para computadores:

- os procedimentos para resolução de problemas precisam ser traduzidos em uma linguagem compreendida pelo computador.
- Nessa disciplina: ± C

Algoritmo

Finito:

Termina em um número finito de passos. Se não termina ou não tem garantia que termina, podemos chamar de *método computacional*.

Sem ambiguidade:

Todo passo é preciso e rigorosamente definido através de linguagens computacionais.

Algoritmo

Entrada:

Valores, símbolos, ... que são fornecidos antes do início do algoritmo.

Saída:

Valores, símbolos, ... que possuem uma relação específica com a *entrada*.

Algoritmo

Eficácia:

Capacidade de realizar todos os passos com sucesso.

Eficiência:

Característica que mede a capacidade do algoritmo em realizar a tarefa (resolver o problema) a que se propõe.

Análise da eficiência de algoritmos

Prever o comportamento do algoritmo em relação aos recursos necessários na execução:

- Memória
- Largura de banda
- Tempo de execução (mais comum)
  - Proporcional ao número de passos executados
  - Proporcional ao número de instruções que o computador executa (temos uma idéia do tempo de cada instrução)

Análise da eficiência de algoritmos

Prever exatamente não é sempre possível (quase nunca é). A análise é apenas uma *aproximação*, mas a melhor possível.

O objetivo da análise é obter uma "estimativa" do crescimento do número de passos executados em função do *tamanho da entrada*: *n*

Problema P

A definição do tamanho da entrada *n* nem sempre é óbvia ou mesmo clara:

- É suficiente considerar o número de elementos em um problema de otimização?
- Seja *P* o problema de determinar se um dado número inteiro positivo *k* é primo. Quanto vale *n*?

Análise da eficiência de algoritmos

Prever exatamente não é sempre possível (quase nunca é). A análise é apenas uma *aproximação*, mas a melhor possível.

O objetivo da análise é obter uma "estimativa" do crescimento do número de passos executados em função do *tamanho da entrada*: *n*

Algoritmo A para P: A(P)

- $f(n) \rightarrow$ estima o crescimento do número de passos de *A* executado sobre uma entrada de *P* de tamanho *n*.
- Idealmente gostaríamos de conhecer $f^x(n)$ que exprimiria *exatamente* o número de instruções executadas para uma entrada de tamanho *n*.

Exemplo: Ordenação

Como varia o tempo de execução de um algoritmo que ordena um vetor de inteiros?

- Ordenar 3 elementos é mais rápido que ordenar 1000 elementos (f(3) < f(1000)).
- Duas sequências distintas com o mesmo número de elementos sempre executa o mesmo número de passos?

Exemplo: Ordenação

InsertionSort (*A*)

```
for  j=1  to  length(A) – 1
    i = j
    elem = A[ i ]
    while i > 0 and A[ i -1 ] > elem
        A[ i ] = A[ i – 1 ]
        i = i – 1
    A[ i ] = elem
```

Quantas vezes a comparação
A[i -1] > elem
é feita ao ordenar...

... [3 5 1 4 2] ? 7
... [1 2 3 4 5] ? 4
... [5 4 3 2 1] ? 10

Tempo de execução varia mesmo com entradas de mesmo tamanho!

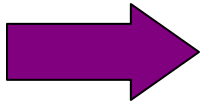
Algoritmo A para P: A(P)

Três casos possíveis considerando todas as possíveis entradas de tamanho *n:*

- Melhor caso
- Caso médio (depende de uma distribuição de probabilidade)
- Pior caso

Algoritmo A para P: A(P)

- Como não é possível determinar $f^x(n)$, buscamos conhecer o máximo possível desta função (relevante as necessidades existentes).
- Se $f^x(n)$ é monótona crescente (hipótese), podemos estimar o crescimento de $f^x(n)$ a partir do crescimento de n.



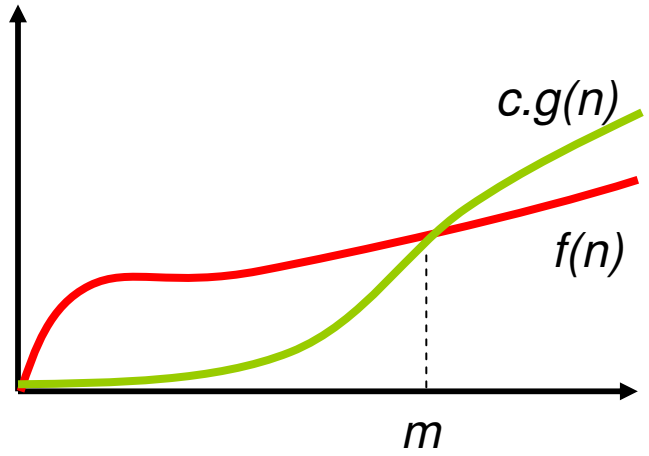
Crescimento assintótico

Comportamento assintótico

Comparação de $f^x(n)$ com outras funções para obter:

- Limite superior assintótico
- Limite inferior assintótico

Comportamento assintótico: limite superior



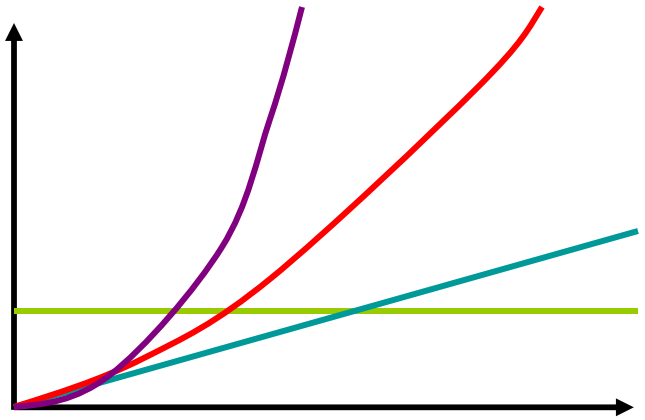
$f(n) = O(g(n)) \Leftrightarrow \exists\, c, m > 0$ tais que

$0 \leq f(n) \leq c.g(n) \quad \forall\ n \geq m$

ou

$\lim_{n \to \infty} f(n) / g(n) \leq c', \quad c' \geq 0$

Comportamento assintótico: limite superior



$f(n) = 2$
$g(n) = n$
$h(n) = n^2$
$u(n) = n^3$

$f(n) = O(g(n))$
$f(n) = O(u(n))$
$g(n) = O(h(n))$
$h(n) = O(u(n))$

Propriedades

$f(n) = O(f(n))$

$c\,O(f(n)) = O(f(n)) \quad \forall c$

$O(f(n)) + O(f(n)) = O(f(n))$

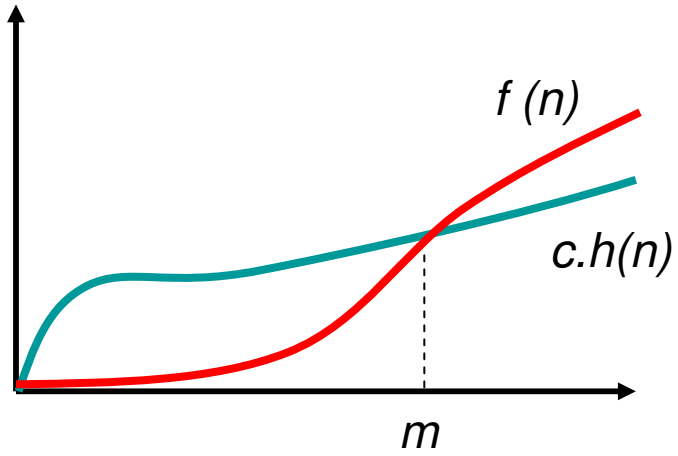
$O(f_1(n)) + O(f_2(n)) = O(\max\{f_1(n), f_2(n)\})$

$O(f_1(n)) \,.\, O(f_2(n)) = O(f_1(n) \,.\, f_2(n))$

$f_1(n) \,.\, O(f_2(n)) = O(f_1(n) \,.\, f_2(n))$

Se $f(n) = a_k n^k + \ldots + a_0 n^0$, então $f(n) = O(n^k)$

Comportamento assintótico: limite inferior



$f(n) = \Omega(\, h(n)\,) \Leftrightarrow \exists\ c,\ m > 0$ tais que

$0 \leq c.h(n) \leq f(n) \quad \forall\ n \geq m$

ou

$\lim_{n \to \infty} f(n) / h(n) \geq c',\ \ c' \geq 0$

Comportamento assintótico

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \text{ e } f(n) = \Omega(g(n))$$

$$f(n) = \Theta(g(n)) \Leftrightarrow \exists\ c_1, c_2, m > 0 \text{ tais que}$$
$$c_1.g(n) \leq f(n) \leq c_2.g(n)\ \forall\ n \geq m$$
ou
$$\lim_{n \to \infty} f(n) / g(n) = c',\ c' \geq 0$$

Comportamento assintótico

$$f(n) = o(g(n)) \Leftrightarrow \exists\, c, m > 0 \text{ tais que } 0 \le f(n) < c.g(n) \;\forall\; n \ge m$$

ou

$$\lim_{n \to \infty} f(n) / g(n) = 0$$

$$f(n) = \omega(h(n)) \Leftrightarrow \exists\, c, m > 0 \text{ tais que } 0 \le c.h(n) < f(n) \;\forall\; n \ge m$$

ou

$$\lim_{n \to \infty} f(n) / h(n) = +\infty$$

Complexidade polinomial

$f(n) = O(n^k)$ onde k é uma constante

Por que é importante?

Complexidade polinomial

Considerando que uma operação leva 10^{-9} segundos…

	Valor de n						
	20	50	100	200	500	1000	5000
10^5 n	< 0.01 s	< 0.01 s	0,01 s	0,02 s	0,05 s	0,1 s	0,5 s
10^5 n log n	< 0.01 s	0,02 s	0,06 s	0,15 s	0,44 s	0,99 s	6,14 s
10^4 n^2	< 0.01 s	0,02 s	0,1 s	0,4 s	2,5 s	10 s	4,1 min
10^3 n^3	< 0.01 s	0,12 s	1 s	8 s	2 min	16,6 min	1,4 dias
$n^{\log n}$	< 0.01 s	3,87 s	5,3 h	12,3 anos	> 1000 séc	> 1000 séc	> 1000 séc
$2^{n/3}$	< 0.01 s	< 0.01 s	10,82 s	37,1 séculos	> 1000 séc	> 1000 séc	> 1000 séc
2^n	< 0.01 s	13 dias	> 1000 séc	> 1000 séc	> 1000 séc	> 1000 séc	> 1000 séc
3^n	3,48 s	> 1000 séc	> 1000 séc	> 1000 séc	> 1000 séc	> 1000 séc	> 1000 séc

Complexidade polinomial