① 2ª Aula (1ª Aula: Apresentação da disciplina)

Objetivos de Análise de Algoritmos (AA):

- Prova de corretude de um algoritmo
- Análise da quantidade de recursos para executar um algoritmo (Complexidade)
- Projeto de algoritmos usando diferentes estratégias

Definição (informal) de algoritmo:
Algoritmo é um procedimento bem definido para resolver um problema computacional específico.

② Busca e Ordenação são exemplos de problemas computacionais clássicos.

Algoritmos e Tecnologia

Dois cenários diferentes:

- Mundo ideal: computadores possuem capacidades de processamento e memória infinitas
- Mundo real: computadores tem poder de processamento e armazenamento finitos

③

Problema:

Dada uma sequência de n números $\langle a_1, a_2, \ldots, a_n \rangle$, determinar uma permutação $\langle a'_1, a'_2, \ldots, a'_n \rangle$ da sequência de entrada tal que $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Instância:

Conjunto de valores de entrada necessários para que se possa resolver o problema.

④

Algoritmos eficientes são mais importantes que aspectos tecnológicos de hardware/software.

Exemplo de ordenação de um vetor com n elementos:

- Algoritmo 1 (Insertion-Sort): $K_1 n^2$ operações, com $K_1 > 0$
- Algoritmo 2 (Merge-Sort): $K_2 n \lg(n)$ operações, com $K_2 > 0$

⑤ Suponha computadores A e B que executam 10G e 10M instruções por segundo, i.é, o computador A é 1000 vezes mais rápido que o computador B.

→ O computador A implementa o Algoritmo 1 com $2n^2$ operações e o melhor programador assembly do planeta.

→ O computador B implementa o Algoritmo 2 com $50 n \lg(n)$ operações e um bom programador Java.

⑥ Qual abordagem é mais rápida para $n = 10^7$?

→ Algoritmo 1 (Insertion-Sort): $\frac{2 \cdot (10^7)^2 \text{ instruções}}{10^{10} \text{ operações/s}} = 2000\text{s}$

→ Algoritmo 2 (Merge-Sort): $\frac{50 \cdot 10^7 \cdot \lg(10^7) \text{ instruções}}{10^7 \text{ operações/s}} = 1163\text{s}$

Portanto, a execução no computador B foi aproximadamente 20 vezes mais rápida que no computador A!

(7)

```
Insertion-Sort (A)
1  for j = 2 to n
2      chave = A[j]
3      // insere chave na sequência ordenada A[1..j-1]
4      i = j - 1
5      while i > 0 e A[i] > chave
6          A[i+1] = A[i]
7          i = i - 1
8      A[i+1] = chave
```

→ Algoritmo iterativo

(8)

Tempo de processamento	Nº de vezes
c_1	n
c_2	$n-1$
0 (Zero)	$n-1$
c_4	$n-1$
c_5	$\sum_{j=2}^{n} t_j$
c_6	$\sum_{j=2}^{n} (t_j - 1)$
c_7	$\sum_{j=2}^{n} (t_j - 1)$
c_8	$n-1$

do while

t_j = Nº de vezes que o teste da linha 5 é executado para cada valor de j.

(9)

$T(n)$ = Tempo de execução do algoritmo com uma instância de tamanho n

$$T(n) = c_1 \cdot n + (c_2 + c_4 + c_8)(n-1) + c_5 \cdot \sum_{j=2}^{n} t_j + (c_6 + c_7) \sum_{j=2}^{n} (t_j - 1) =$$

$$= c_1 \cdot n + (c_2 + c_4 + c_8 - c_6 - c_7)(n-1) + (c_5 + c_6 + c_7) \sum_{j=2}^{n} t_j$$

Melhor caso: $t_j = 1, \forall_j$ (sequência já ordenada)

$$T(n) = (c_1 + c_2 + c_4 + \overset{c_5}{\overbrace{\quad}} + c_8)n - (c_2 + c_4 + c_5 + c_8) = an - b$$

(linear em n)

Pior caso: $t_j = j, \forall_j$ (sequência em ordem decrescente)

$$T(n) = (c_5 + c_6 + c_7)n^2 + \left(c_1 + c_2 + c_4 + c_8 + \frac{c_5}{2} - \frac{(c_6 + c_7)}{2}\right)n$$

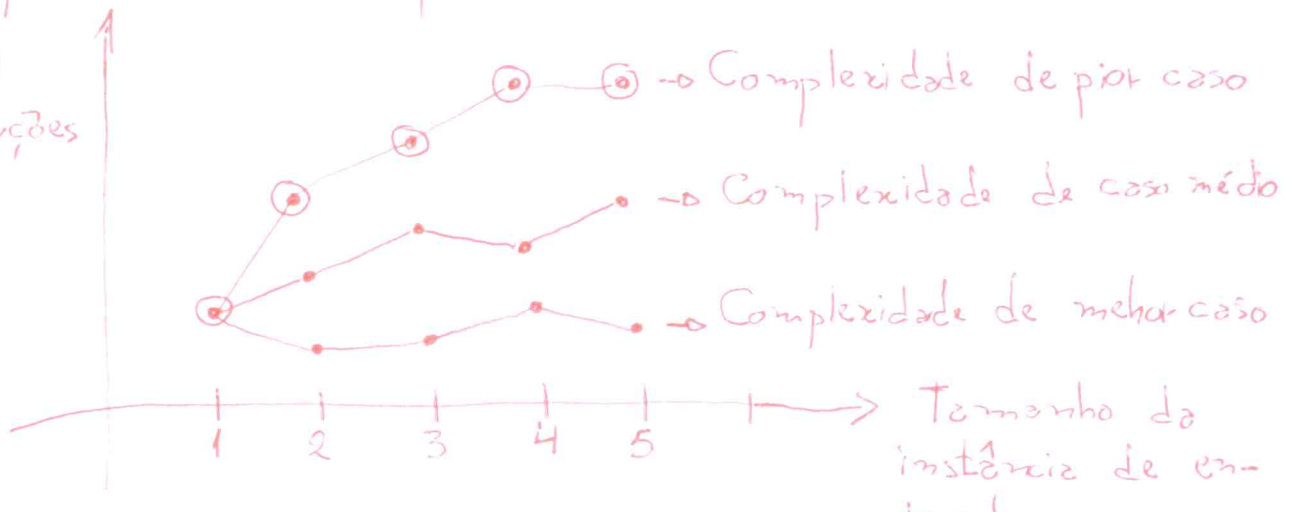
(10)

$$- (c_2 + c_4 + c_5 + c_8) = an^2 + bn + c$$

(quadrático em n)

Lembre-se que $\boxed{\sum_{j=1}^{n} j = \frac{n(n+1)}{2}}$

Complexidade de pior caso é a mais utilizada.

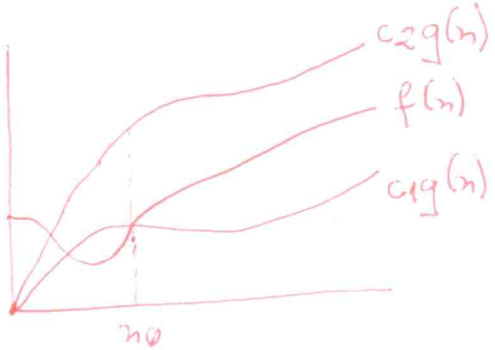


(11) Notação assintótica

(1) Notação Θ (theta)

Para uma função $g(n)$, denotamos por $\Theta(g(n))$ o conjunto de funções,

$$\Theta(g(n)) = \{ f(n) : \exists\, c_1, c_2 > 0 \text{ e } n_0 > 0 \text{ t.q. } c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall\, n \geq n_0 \}$$



Em vez de escrever $f(n) \in \Theta(g(n))$ escrevemos $f(n) = \Theta(g(n))$

A notação Θ é usada para delimitações exatas de funções.

(12) Exemplo

$$\frac{1}{2} n^2 - 3n = \Theta(n^2)$$

$c_2 = \frac{1}{2}$ (fácil) $\Rightarrow \frac{1}{2} n^2 - 3n \leq \frac{1}{2} n^2, \quad \forall\, n \geq 0$

$c_1 = \frac{1}{4} \Rightarrow \frac{1}{4} n^2 \leq \frac{1}{2} n^2 - 3n \Rightarrow \frac{1}{4} n^2 = 3n \Rightarrow n = 12$

$n \geq 12$

$n_0 = \max\{0, 12\} = 12$

(13)

(2) Notação O (oh-grande)

A notação O serve para atribuir delimitações superiores para uma função $g(n)$.

$O(g(n)) = \{f(n) : \exists\ c, n_0 > 0$ t.q. $f(n) \leq c\,g(n), \forall n \geq n_0\}$

Novamente escrevemos $f(n) = O(g(n))$ em vez de $f(n) \in O(g(n))$.

(14)

Exemplos:

(a) $n = O(n^2)$?

Sim, tome $c = 1$ e $n_0 = 1$.

(b) $\frac{1}{2}n^2 - 3n = O(n^2)$?

Sim, tome $c = \frac{1}{2}$ e $n_0 = 7$

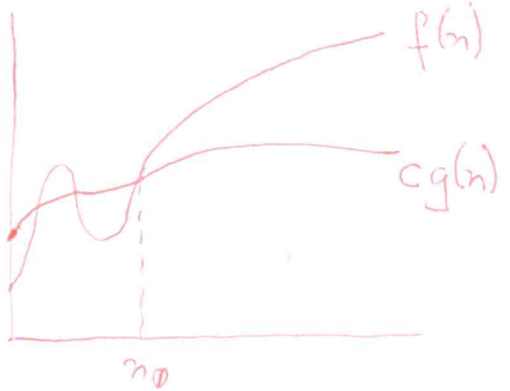
(c) $an + b = O(n^2)$

(15)

(3) Notação Ω (omega)

A notação Ω serve para atribuir delimitações <u>inferiores</u> para uma função $g(n)$.

$$\Omega(g(n)) = \{f(n) : \exists\, c, n_0 > 0 \text{ t.q. } f(n) \geq c g(n), \forall n \geq n_0\}$$



$f(n) = \Omega(g(n))$

(16) Exemplo:

$\frac{1}{3} n^2 - 3n = \Omega(n^2)$

Tome $c = \frac{1}{14}$ e $n_0 = 7$

<u>Outras notações</u>:

$o(g(n)) = O(g(n)) \setminus \Theta(g(n))$

$\omega(g(n)) = \Omega(g(n)) \setminus \Theta(g(n))$

<u>Observação 1</u>:

$f(n) = O(g(n)) \approx a \leq b$

$f(n) = \Omega(g(n)) \approx a \geq b$

$f(n) = \Theta(g(n)) \approx a = b$

$f(n) = o(g(n)) \approx a < b$

$f(n) = \omega(g(n)) \approx a > b$

(17)

Observação 2:

$f(n) = O(g(n))$ se $\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$

$f(n) = \Omega(g(n))$ se $\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0$

$f(n) = \Theta(g(n))$ se $0 < \lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$

(18) Propriedades

- Reflexiva:
  - $f(n) = O(f(n))$
  - $f(n) = \Omega(f(n))$
  - $f(n) = \Theta(f(n))$
- Simetria:
  - $f(n) = \Theta(g(n))$ se e somente se $g(n) = \Theta(f(n))$
  - $f(n) = O(g(n))$ se e somente se $g(n) = \Omega(f(n))$
  - $f(n) = o(g(n))$ se e somente se $g(n) = \omega(f(n))$

(19)

→ Transitiva:

$f(n) = \Theta(g(n))$ e $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

$f(n) = O(g(n))$ e $g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$

$f(n) = \Omega(g(n)) \dots$

$f(n) = o(g(n)) \dots$

$f(n) = \omega(g(n)) \dots$

(20) Notação assintótica em expressões

Como interpretar, por exemplo $2n^2 + n - 1 = 2n^2 + \Theta(n)$?

Existe uma função $f(n) = \Theta(n)$ tal que $2n^2 + n - 1 = 2n^2 + f(n)$.

Neste caso, $f(n) = n - 1$.

É a expressão $2n^2 + \Theta(n) = \Theta(n^2)$?

Para toda função $f(n) = \Theta(n)$, existe uma função $g(n) = \Theta(n^2)$ tal que $2n^2 + f(n) = g(n)$

Por exemplo, $f(n) = n - \lg n + 5 = \Theta(n)$ e $g(n) = 2n^2 + n - \lg n + 5 = \Theta(n^2)$