

ELEMENTOS BÁSICOS DE ALGORITMOS

Problema : multiplicar dois inteiros $a, b : a \times b$

Instância (Instance) : multiplicar 981×1234

EFICIÊNCIA DE ALGORITMOS

- Escolha do melhor algoritmo para resolver um problema
Enfoque teórico de escolha : determinar matematicamente a quantidade de recursos necessários para cada algoritmo, em função do tamanho das instâncias.
- Fator principal da eficiência de um algoritmo: velocidade de execução
- Outros fatores: memória e número de processadores em paralelo

A) TAMANHO DE UMA INSTÂNCIA

- Formal: número de bits necessários para representar a instância usando esquema de codificação compacto

- Menos formal

Problema de ordenação: número de itens ordenados

Problema de grafos: número de nós e/ou arcos

B) PRINCÍPIO DA INVARIÂNCIA

Se duas implementações do mesmo algoritmo gastam $t_1(n)$ e $t_2(n)$ segundos para resolver uma instância de tamanho n então existe uma constante positiva c tal que $t_1(n) \leq c t_2(n)$ para n suficientemente grande

C) ORDEM DE TEMPO

Um algoritmo, para um problema, gasta um tempo da ordem de $t(n)$, para uma dada função t , se existe uma constante positiva c e uma implementação do algoritmo capaz de resolver toda instância de tamanho n em não mais de $c t(n)$ segundos (pior caso).

Exemplos para $t(n)$: $n, n^3, 2^n$

D) OPERAÇÃO ELEMENTAR

É tal que seu tempo de execução é limitado superiormente por uma constante que depende da implementação, máquina, linguagem de programação, etc. A constante não depende do tamanho e parâmetros da instância.

Exemplos de operações elementares: adição, subtração, multiplicação, divisão, operação de módulo, comparação, designação.

Exemplo: Ordenação de Números

Algoritmo Bubble-Sort (n, a_1, a_2, \dots, a_n)

```
Para i = 1 até n - 1 faça
  Para j = n até i + 1 faça
    Se  $a_{j-1} > a_j$  então
      Início
        temp =  $a_{j-1}$  ;
         $a_{j-1} = a_j$ 
         $a_j = temp$ 
      fim ;
  fim ;
Fim.
```

- Laço interno requer no pior caso $4(n - i)$ operações.
- Número total de operações:

$$t(n) = \sum_{i=1}^{n-1} [4(n - i)]$$

$$= 4[(n - 1) + (n - 2) + \dots + 1] = 2n(n - 1) = 2n^2 - 2n$$

$$\leq 2n^2, \quad \forall n \geq 0$$

E) COMPLEXIDADE DE TEMPO ASSINTÓTICA

$t(n)$: número de operações no pior caso da instância de tamanho n

- $t(n)$ é da ordem de $f(n)$, $t(n) \in O(f(n))$ se existe uma constante real positiva c tal que $t(n) \leq c f(n)$, $\forall n \geq n_0$

Exemplo

$$t(n) = 27n^2 + \frac{355}{113}n + 12 \leq 27n^2 + \frac{355}{113}n^2 + 12n^2$$

$$= 42 \frac{16}{113} n^2 = 42 \frac{16}{113} f(n)$$

$$c = 42 \frac{16}{113} \quad ; \quad n_0 = 1$$

- $t(n)$ é da ordem de n^2

Nota : $O(n^2)$ é a complexidade de qualquer implementação pelo princípio da invariância

Preposição 1 : Regra das Somas

Se um determinado algoritmo A se divide em partes independentes, por exemplo, A_1 e A_2 e sendo $t_1(n)$ e $t_2(n)$ de ordem $O(f(n))$ e $O(g(n))$, então $t(n) = t_1(n) + t_2(n)$ e A é de ordem $O(\max\{f(n), g(n)\})$.

Demonstração

$$t_1(n) \leq c_1 f(n) \quad n \geq n_1$$

$$t_2(n) \leq c_2 g(n) \quad n \geq n_2$$

Seja $n_0 = \max\{n_1, n_2\}$. Para $n \geq n_0$

$$t_1(n) + t_2(n) \leq c_1 f(n) + c_2 g(n) \leq (c_1 + c_2) \max\{f(n), g(n)\}$$

Preposição 2 : Regra dos Produtos

Se $t_1(n)$ e $t_2(n)$ são da ordem $O(f(n))$ e $O(g(n))$ então $t(n) = t_1(n) t_2(n)$ é da ordem $O(f(n) g(n))$.

ALGORITMO EM TEMPO POLINOMIAL

- função complexidade de tempo é $O(p(n))$ onde $p(n)$ é um polinômio e n o tamanho da instância

ALGORITMO EM TEMPO EXPONENCIAL

- função complexidade de tempo não é limitada por polinômio
Exemplos: 2^n , $n!$

TEORIA DE COMPLEXIDADE COMPUTACIONAL

- classes de problemas: P e NP

Comparação entre Funções Complexidade de Tempo com 10^6 operações/segundo

função tamanho	n	n^2	n^5	2^n
10	10^{-5} s	10^{-4} s	10^{-1} s	10^{-3} s
20	2×10^{-5} s	4×10^{-4} s	3,2 s	1,0 s
30	3×10^{-5} s	9×10^{-4} s	24,3 s	17,9 min
40	4×10^{-5} s	16×10^{-4} s	1,7 min	12,7 dias
50	5×10^{-5} s	25×10^{-4} s	5,2 min	35,7 anos
60	6×10^{-5} s	36×10^{-4} s	13 min	366 séculos

Tamanho da Instância x Velocidade Computacional

Função	Computador atual	Computador 100 vezes mais rápido	Computador 1000 vezes mais rápido
n	A	100 A	1000 A
n^2	B	10 B	31,6 B
n^5	C	2,5 C	3,98 C
2^n	D	$D + 6,64$	$D + 9,97$