Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA

Mestrado Decisão Económica e Empresarial
COMPUTAÇÃO

Sumário:

Apresentação.
Introdução ao estudo dos algoritmos.
Representação em binário e em vírgula flutuante.

LEONOR SANTIAGO PINTO

GAB 506 Quelhas (ext 3845)

Telefone 213925845

lspinto@iseg.utl.pt

Programa

- Introdução à Programação (VBA)
- Algoritmos de Ordenação e Pesquisa
- Estruturas de Dados
- Algoritmos de Grafos

Bibliografia

- *Introduction to Algorithms*, T. Cormen, C. Leiserson, R. Rivest and C. Stein (2001), 2nd ed., MIT, Massachusetts.
- *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Garey, M.R. and D.S. Jonhson (1979), W.H. Freeman and Company, San Francisco.
- *Algoritmia e Estruturas de Dados*, J.B.Vasconcelos e J.V. Carvalho (2005), Centro Atlântico.

Bibliografia

- *Mastering VBA for Office 2007*, R. Mansfield (2008), Wiley.
- *Developing Spreadsheet_Based Decision Support Systems. Using Excel and VBA Excel*, M.Seref, R. Ahuja and W. Winston (2007), Dynamic Ideas, Belmont, Massachusets.
- *Combinatorial Optimization: Algorithms and Complexity,* Papadimitriou, C. and K. Steiglitz (1998), 2nd ed., Dover, New York.

Noção Informal de Algoritmo

Sequência finita e não ambigua de instruções elementares bem definidas, conducente à solução de um determinado problema, cada uma das quais pode ser executada mecanicamente, numa quantidade finita de tempo e com uma quantidade finita de esforço.

exemplos:

substituir uma lâmpada fundida de um candeeiro,
encontrar um número de telefone na lista.

Definição de Algoritmo

Processo

discreto => sequência de acções indivisíveis,

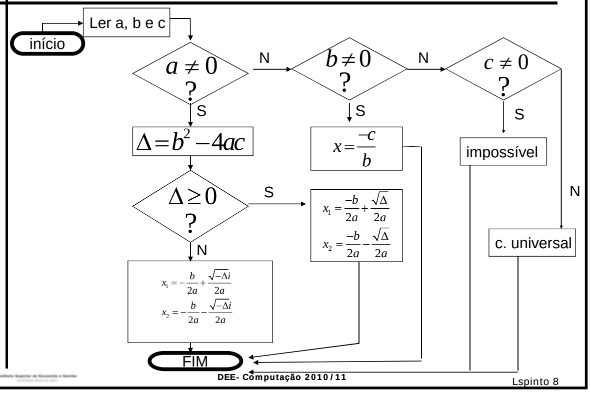
determinístico => para cada passo da sequência e para conjunto válido de dados, corresponde uma e uma só acção

que termina quaisquer que sejam os dados iniciais (pertencentes a conjuntos prédefinidos).

Exemplos :

- Determinar as raízes de uma equação de segundo grau;
- Determinar o máximo divisor comum entre dois inteiros;
- Atribuição de mandatos pelo método de Hondt.

Raizes da equação de 2ºgrau $ax^2 + bx + c = 0$



Algoritmo de "Euclides" (exemplo)

Dados os números inteiros m=38 e n=10, calcular o seu máximo divisor comum mdc(38,10).

Como 38=3*10+8

os divisores comuns a 38 e 10 são os comuns a 10 e 8;

Como 10=1*8+2

os divisores comuns a 10 e 8 são os comuns a 8 e 2;

Como 8=4*2+0

o máximo divisor comum entre 8 e 2 é 2 !

mdc(38,10)=2

Algoritmo de "Euclides" I

Dados 2 números inteiros m e n, calcular o seu máximo divisor comum mdc(m,n).

1. **Ler** m e n (inteiros diferentes de 0);
2. **Se** m< n **Então** torne maior=n e menor=m **Senão** maior=m e menor=n;
3. Torne resto=mod(maior,menor);
4. **Se** resto==0 **Escrever** mdc(m,n)=menor; **STOP**
5. **Enquanto** (resto!=0)
   Torne resto=mod(maior,menor), maior=menor e menor=resto;
6. **Escrever** mdc(m,n)=maior; **STOP**

Algoritmo de "Euclides" II

Dados 2 números inteiros m e n, calcular o seu máximo divisor comum mdc(m,n).

1. **Ler** m e n (inteiros diferentes de 0);
2. **Escrever** mdc(m,n)
3. **Enquanto** (n!=0)
   Torne resto=m mod n, m=n e n=resto;

5. **Escrever** =m; **STOP**

Escolha de algoritmo

Objectivos contraditórios influenciam a escolha.

Um algoritmo deve

- Ser fácil de compreender, codificar e depurar;
- Usar eficientemente os recursos do computador, em particular, ser o mais rápido possível.

Tendencialmente o segundo factor domina o primeiro mas usar um algoritmo simples pode ser útil para teste e avaliação de algoritmos mais sofisticados.

O tempo de execução depende

- Input do programa;
- Qualidade do código gerado pelo compilador para criar o .obj;
- Natureza e velocidade das instruções na máquina que é usada para correr o programa;
- Complexidade do algoritmo programado.

Do primeiro factor intui-se que o tempo de execução deve ser definido como função da dimensão do input.

Exemplo com ordenação

ordenar 2,1,3,1,5,8

por ordem crescente 1,1,2,3,5,8.

O tempo de execução do algoritmo para ordenar deve ser função do número de objectos a ordenar => quanto maior a sua quantidade mais tempo requer.

Neste caso, a dimensão do input é o comprimento da lista n, ie, o número de objectos a ordenar.

T(n)

T(n) => tempo de execução de um programa com dimensão de input n.

Exemplo

$$T(n) = cn^2$$

c é uma constante;

a unidade de T(n) é o número de instruções executadas num computador idealizado. (para evitar dependência da máquina e compilador).

Análise de pior caso

Em muitos casos o tempo de execução é função do input particular e não só da sua dimensão.

T(n) é definido como sendo o tempo correspondente à instância de dimensão n, que demora mais tempo.

Tempo médio

Alternativamente ao pior caso pode considerar-se o tempo médio

$$T_{avg}(n)$$

inconvenientes do tempo médio:

- nem todos os inputs ocorrem com igual probabilidade
- frequentemente é muito difícil de calcular

Notação O

$T(n) = O(n^2)$ Significa que existem constantes positivas c e n_0 tais que para $n \geq n_0$ se tem

$$T(n) \leq cn^2$$

Exemplos:

$T(0)=1, T(1)=4, T(n)=(n+1)^2$ é $O(n^2)$

$T(n) = 3n^3 + 2n^2$ é $O(n^3)$

$T(n) = 3^n$ não é $O(2^n)$

Notação O

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

se existirem constantes positivas c e n_0
Tais que para todo $n \geq n_0$ se tem

$$T(n) \leq cf(n)$$

Um programa com este tempo de execução diz-se que tem taxa de crescimento é f(n).

Notação Ω

$T(n) = \Omega(g(n))$ Significa que existe uma constante positiva c e tal que

$$T(n) \geq cg(n)$$

Para um número infinito de valores de n.

Traduz minorante no tempo, enquanto que notação O dá majorante.

Exemplo:

$T(n) = n^3 + 2n^2$ é $\Omega(n^3)$

Selecção

Os programas podem ser avaliados por comparação das suas funções de tempo de execução assumindo que as constantes de proporcionalidade podem ser desprezadas.

Nesta hipótese de trabalho

$O(n^2)$ é melhor do que $O(n^3)$

Desprezar constantes …

Para além dos factores constantes atribuídos ao compilador e computador existe um factor constante que depende da própria natureza do problema.

exemplo:

Programa 1 => T(n)= $100n^2$
Programa 2 => T(n)= $5n^3$

Com n = 10 o P1 demora 10000 para 5000 de P2
Mas quando o n aumenta P1 demora menos comparado com P2.

Dimensão máxima que é admissível....

Efeito dos tempos de computação

Tempo de execução T(n)	Dim max para 1000seg	Dim max para 10000seg	Aumento no max da dimensão
$100n$	10	100	10.0
$5n^2$	14	45	3.2
$n^3/2$	12	27	2.3
2^n	10	13	1.3

Complexidade e tempos

Suponhamos que cada operação é efectuada em 1 milisegundo.
A tabela seguinte dá o tempo gasto por cada um dos algoritmos.

	A1	A2	A3	A4	A5
n	*O(n)*	*O (nlogn)*	*O(n^2)*	*O(n^3)*	O(2^n)
16	0,016s	0,064s	0,256s	4s	1m4s
32	0,032s	0,16s	1s	33s	46 dias
512	0,512s	9s	4m22s	1 dia 13h	10^{137} séculos

ADVERTÊNCIAS

- Programas usados poucas vezes;
- usar só para instâncias de reduzida dimensão;
- Muito sofisticado é de difícil manutenção;
- Exigência de memória;
- Algorimos numéricos, precisão e estabilidade é tão importante como a eficiência.

Contagem de tempo - somas

Regra das somas

T1(n) e T2(n) são contagens de 2 fragmentos de um programa P1 e P2. T1 é O(f(n)) e T2 éO(g(n)) então

T1(n)+T2(n)=O(max{f(n),g(n)})

obs: se g(n) <= f(n) para todo n>n0 então

O(f(n)+g(n))=O(f(n))

Contagem de tempo - produtos

Regra do produto

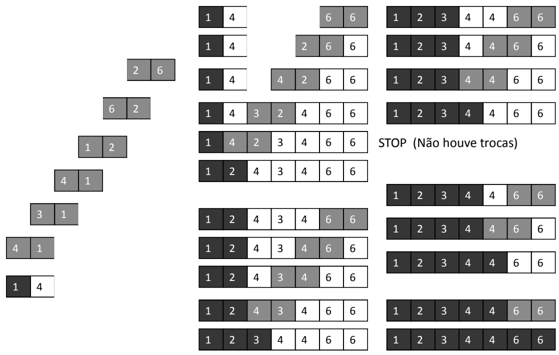
T1(n) e T2(n) são contagens de 2 fragmentos de um programa P1 e P2. T1 é O(f(n)) e T2 é O(g(n)) então

T1(n)*T2(n)=O(f(n)*g(n))

obs: para c constante positiva

O(cf(n))=O(f(n))

Exemplo bubble sort



Exemplo bubble sort

```
Prodedimento Bubble(vector de inteiros A[1..n]) {
//para ordenar por ordem crescente A
    inteiros i,j,temp;

    Para i=1 até n-1                              (1)
        Para j=n até i+1                          (2)
            Se A[ j -1 ] > A[ j ]  então {        (3)
            // trocar A[ j -1 ] com A[ j ]
                temp= A[ j -1 ] ;                 (4)
                A[ j - 1] = A[ j ];               (5)
                A[ j ] = temp;                    (6)

            }
}
```


Complexidade do bubble sort

$$O(n^2)$$

$$\sum_{i=1}^{n-1}(n-i)=\frac{n(n-1)}{2}=\frac{n^2}{2}-\frac{n}{2}$$

Representação em binário

Em decimal

$$125 = 100 + 20 + 5 = 1\times10^2 + 2\times10^1 + 5\times10^0$$

Em binário 01111101

$$125 = 64 + 32 + 16 + 8 + 4 + 0 + 1$$

$$125 = 1\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$$

0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

Números negativos em binário

1 Representar -100

$$2^8 - 100 = 156$$

Escrever em binário 156 10011100

2 Qual o número representado por 11100100 ?

como o bit da esquerda é 1 o número é negativo

128+64+32+4=228 228-256= -28

com sinal represento -8 a 7 (0)

0000	0	1000	-8 (8 -16)
0001	1	1001	-7 (9 -16)
0010	2	1010	-6 (10 -16)
0011	3	1011	-5 (11 -16)
0100	4	1100	-4 (12 -16)
0101	5	1101	-3 (13 -16)
0110	6	1110	-2 (14 -16)
0111	7	1111	-1 (15 -16)

$$\boxed{2^4 = 16}$$

Características desta representação

- O bit da esquerda indica o sinal
- O zero tem uma representação única
- A gama de valores representada (n bits)

signed $-2^{n-1}, 2^{n-1}-1$

unsigned $0, 2^n - 1$

Números reais

A memória é finita...

Aproximações implicam erros

Vírgula flutuante décimal

Exemplos

0.0003406 ou 3.406 E-4

3.406 E-4 = $(-1)^0 10^{-4}\left(3+4\frac{1}{10}+6\frac{1}{10^2}\right)$

-1.32 E-6 = $(-1)^1 10^{-6}\left(1+3\frac{1}{10}+2\frac{1}{10^2}\right)$

$$(-1)^s 10^e \left(d_0 + \sum d_i 10^{-i}\right) \quad d_i \in \{0,1,2,\ldots,9\}$$

Em binário

$$(-1)^s 2^e \left(1 + \sum d_i 2^{-i}\right) \quad d_i \in \{0,1\}$$

$$(-1)^s 10^e \left(d_0 + \sum d_i 10^{-i}\right) \quad d_i \in \{0,1,2,\ldots,9\}$$

(decimal)

103.0 em precisão simples

Como o número é positivo o 1º bit é zero

Escrever 103 em binário 1100111

Normalizar **1**.100111x 2^6

- Expoente 6
- Mantissa 1.100111

Valor a colocar na mantissa 1001110.........0

Expoente

- 6+127=133
- 133 em binário 10000101
- Resumindo:

  0 100 0010 1 100 1110 0000 0000 0000 0000

  sinal Expoente mantissa

Numeros em virgula flutuante

$$(-1)^s 2^e \left(1 + \sum d_i 2^{-i}\right) \quad d_i \in \{0,1\}$$

Precisão simples (32 bits)

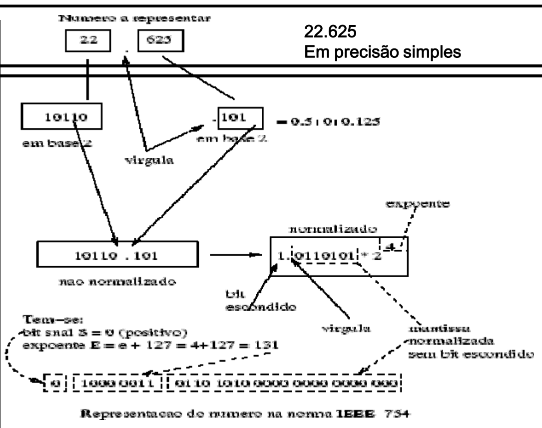
sinal: 1 bit

expoente: 8 bits (representado em excesso de 127)

mantissa: 23 bits

22.625

Em precisão simples



Representação de números reais

Exemplo:

Números em virgula flutuante

$$(-1)^s 2^e \left(1+\sum d_i 2^{-i}\right) \quad d_i \in \{0,1\}$$

Precisão dupla (64 bits)

sinal: 1 bit

expoente: 11 bits (representado em excesso de 1023)

mantissa: 52 bits

Exercícios

1. Escreva o algoritmo para atribuição de mandatos pelo método d'Hondt.
2. Represente em binário (com 8 bits) os inteiros 23, -63 e -1.
3. Qual o número representado por 11011011 se se tratar de um inteiro sem sinal ? E se se tratar de um inteiro com sinal ?
4. Represente em precisão simples 10.125.
5. Qual o número representado por 1100 0001 0100 0100 0000 0000 0000 0000 ?

Bom trabalho e até sexta-feira !

Método d'Hondt

O círculo eleitoral "X" elege 7 deputados e concorrem 4 partidos: A, B, C e D. Apurados os votos, a distribuição foi a seguinte: A - 12.000 votos; B - 7.500 votos; C - 4.500 votos; e D - 3.000 votos. Da aplicação do método de Hondt resulta a seguinte série de quocientes:

	Partido			
divisor	A	B	C	D
1	**12000**	**7500**	**4500**	**3000**
2	**6000**	**3750**	2250	1500
3	**4000**	2500	1500	1000
4	**3000**	1875	1125	750

http://pt.wikipedia.org/wiki/M%C3%A9todo_D'Hondt