Departamento de Engenharia Informática da Universidade de Coimbra

Estruturas Discretas 2013/14 | Folha 3 - TP

Algoritmos e complexidade

1. Exprima as seguintes expressões usando a notação abreviada de somatório:

   (a) $\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots+\frac{1}{n!}$ $(n \geq 2)$.

   (b) $\frac{1}{3!}+\frac{1}{3!}+\frac{1}{3!}+\cdots+\frac{1}{3!}$ (20 parcelas).

   (c) $1+4+9+16+25+36+49$.

   (d) $1^3-2^3+3^3-4^3+5^3-6^3+7^3$.

   (e) $\frac{1}{n}+\frac{2}{n+1}+\frac{3}{n+2}+\cdots+\frac{n+1}{2n}$.

   (f) $n-\frac{n+1}{2!}+\frac{n+2}{4!}-\frac{n+3}{6!}+\cdots+(-1)^n\frac{2n}{(2n)!}$.

2. Qual é o valor dos seguintes somatórios?

   (a) $\sum_{i=1}^{5} i^2$ (b) $\sum_{j=4}^{8}(-1)^j$ (c) $\sum_{i=1}^{4}\sum_{j=1}^{3} ij$ (d) $\sum_{s\in\{0,2,4\}} s$ (e) $\sum_{s\in\{1,3,5,7\}} \frac{1}{s}$ (f) $\sum_{k=0}^{4} k!$

3. Qual é o valor dos seguintes somatórios duplos?

   (a) $\sum_{i=1}^{2}\sum_{j=1}^{3}(i+j)$ (b) $\sum_{i=0}^{20}\sum_{j=0}^{30}(2i+3j)$ (c) $\sum_{i=1}^{300}\sum_{j=0}^{200} i$ (d) $\sum_{i=0}^{200}\sum_{j=1}^{300} ij$ (e) $\sum_{i=1}^{300}\sum_{j=1}^{200}(i-j)$

   (f) $\sum_{i=0}^{200}\sum_{j=0}^{300}(3i+2j)$ (g) $\sum_{i=1}^{300}\sum_{j=0}^{200} j$ (h) $\sum_{i=0}^{20}\sum_{j=1}^{30} i^2j^3$

4. Mostre que $\sum_{j=1}^{n}(a_j-a_{j-1})=a_n-a_0$ para qualquer sequência de números reais $a_0, a_1, \ldots, a_n$.

5. Use a identidade
$$\frac{1}{k(k+1)}=\frac{1}{k}-\frac{1}{k+1}$$
   e o exercício anterior para calcular
$$\sum_{k=1}^{n}\frac{1}{k(k+1)}.$$

6. Some ambos os membros da identidade $k^2-(k-1)^2=2k-1$ desde $k=1$ até $k=n$ e use o Exercício 4 para obter:

   (a) uma fórmula para $\sum_{k=1}^{n}(2k-1)$. (b) uma fórmula para $\sum_{k=1}^{n} k$.

7. (a) Indique uma fórmula para o somatório $\sum_{i=1}^{n} a_i$ onde $a_1, a_2, \ldots, a_n, \ldots$ é uma progressão aritmética de razão r.

(b) Calcule o valor da soma

$$\frac{1}{3} + \frac{2}{3} + 1 + \frac{4}{3} + \cdots + 10.$$

8. Considere o algoritmo seguinte que permite calcular o valor da função *soma* em cada inteiro positivo n dado.

```
procedure soma (n: inteiro positivo)
soma := 0    {valor inicial da soma}
for i := 1 to n
    for j := 1 to i
        soma := soma − 2;
```

(a) Calcule $soma(3)$.

(b) Determine $soma(n)$.

9. Quanto tempo demora um algoritmo a resolver um problema de comprimento n, para os seguintes valores de n, se efectuar $2n^2 + 2^n$ operações, cada uma demorando 10^{-9} segundos?

(a) $n = 10$ (b) $n = 20$ (c) $n = 50$ (d) $n = 100$

10. Qual é o comprimento máximo de um problema que pode ser resolvido num segundo usando um algoritmo que efectua $f(n)$ operações, onde cada uma demora 10^{-9} segundos, para os seguintes casos?

(a) $f(n) = \log_{10} n$ (b) $f(n) = n$ (c) $f(n) = n^2$ (d) $f(n) = 2^n$ (e) $f(n) = n!$

11. O algoritmo usual para calcular o valor de um polinómio $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ em $x = c$ pode ser expresso por

```
procedure polinomio(c, a_0, a_1, ..., a_n: números reais)
potencia := 1
y := a_0
for i := 1 to n
begin
    potencia := potencia * c
    y := y + a_i * potencia
end {y = a_n c^n + a_{n-1} c^{n-1} + ··· + a_1 c + a_0}
```

onde o valor final de y é o valor do polinómio em $x = c$.

(a) Calcule $3x^2 + x + 1$ em $x = 2$, percorrendo todos os passos do algoritmo.

(b) Quantas multiplicações e adições são feitas para determinar o valor de um polinómio de grau n em $x = c$? (Não conte as adições usadas para incrementar a variável do ciclo.)

12. Há um método mais eficiente (em termos do número de multiplicações e adições efectuada) para calcular valores de polinómios do que o descrito no exercício anterior (cf. apontamentos, p. 39). É o chamado *método de Horner*:

```
procedure Horner(c, a_0, a_1, ..., a_n: números reais)
y := a_n
for i := 1 to n
    y := y * c + a_{n-i}
{y = a_n c^n + a_{n-1} c^{n-1} + ··· + a_1 c + a_0}
```

(a) Calcule $3x^2 + x + 1$ em $x = 2$, percorrendo todos os passos do algoritmo.

(b) Quantas multiplicações e adições são feitas para determinar o valor de um polinómio de grau n em $x = c$? (Não conte adições usadas para incrementar a variável do ciclo.)

13. O problema de localizar um elemento numa lista ordenada (ou de determinar que ele não está na lista) pode ser resolvido pelos dois algoritmos seguintes (o primeiro, chamado *procura linear*, localiza um inteiro x numa lista de inteiros distintos de comprimento n; o segundo, chamado *procura binária*, localiza um inteiro x numa lista de inteiros ordenados por ordem crescente).

```
procedure procura linear(x: inteiro, a_1, a_2, ..., a_n: inteiros distintos)
i := 1
while (i ≤ n and x ≠ a_i)
i := i + 1
if i ≤ n then lugar := i
else lugar := 0
{lugar é o índice do termo igual a x, ou é 0 se x não for encontrado}
```

```
procedure procura binária(x: inteiro, a_1, a_2, ..., a_n: inteiros crescentes)
i := 1   {i é o limite esquerdo do intervalo de procura}
j := n   {j é o limite direito do intervalo de procura}
while i < j
begin
    m := ⌊(i + j)/2⌋
    if x > a_m then i := m + 1
    else j := m
end
if x = a_i then lugar := i
else lugar := 0
{lugar é o índice do termo igual a x, ou é 0 se x não for encontrado}
```

Liste todos os passos para procurar o número 9 na sequência 1, 3, 4, 5, 6, 8, 9, 11 usando

(a) procura linear (b) procura binária.

14. Suponha que se sabe que um determinado elemento está entre os primeiros quatro elementos de uma lista de 32 elementos. O que deveremos fazer para encontrar esse elemento mais rapidamente, uma procura linear ou uma procura binária?

15. Determine a complexidade dos algoritmos de procura linear e procura binária.