Armando Matos, 2008-2009

Tópicos Avançados em Algoritmos - Folha prática de exercícios 01

1. Determine da forma mais simples possível as somas

   (a) $1+a+a^2+\cdots+a^n$ como função de a e n. Para o caso $a=2$: memorize este resultado; faça uma interpretação desta igualdade com base do número total de nós (internos e "folhas") de uma árvore binária completa.

   (b) $a+(a+1)+(a+2)+\cdots+b$ onde a e b são inteiros com $a \leq b$.

   (c) $1+2^2+3^2+\cdots+n^2$, como função de n.

2. Alguns exercícios[1] de recorrências

   (a) Mostre que a solução da recorrência $a_1=1$, $a_{n+1}=2a_n+1$ satisfaz $a_n \geq 2^n$.

   (b) Determine a solução de $a_0=0$, $a_1=1$, $a_n=a_{n-1}+a_{n-2}$ para $a \geq 2$.

   (c) Determine a solução de $a_0=0$, $a_1=0$, $a_n=a_{n-1}+a_{n-2}+1$ para $a \geq 2$.

3. • Usando o método "tabelar – suspeitar – demonstrar" determine a solução da seguinte recorrência

$$f_n = \begin{cases} f_1 = 0 \\ f_{2n} = 2f_n + 2n \quad \text{para } n \geq 1 \end{cases}$$

   Notas. A sucessão só está definida para valores de n que sejam potências de 2. Comece por tabelar n e $\log n$ para $n=1$, 2, 4, 8, 16.

4. • Eficiência de um programa para calcular a sucessão de Fibonacci...
   Considere o seguinte programa que calcula a sucessão de Fibonacci

```
def fib(n):
  if n=0: return 0
  if n=1: return 1
  return fib(n-1)+fib(n-2)
```

   (a) Exprima através de uma recorrência o número de adições $s(n)$ que são efectuadas quando se execura `fib(n)`.

   (b) Resolva essa recorrência.

5. • Eficiência do *mergesort*.
   O "mergesort" é um método de ordenação muito eficiente, mesmo no pior caso. Descrição, supondo que n é uma potência de 2:

   `mergesort(v[0...n-1],n]:`

   (a) Se $n=1$: nada se faz

   (b) Se $n \geq 2$:

   - `mergesort`$(\mathtt{v}[0\cdots\mathtt{n}/2-1],\mathtt{n}/2)$ `//` ordena $\mathtt{v}[0\cdots\mathtt{n}/2-1]$
   - `mergesort`$(\mathtt{v}[\mathtt{n}/2\cdots\mathtt{n}-1],\mathtt{n}/2)$ `//` ordena $\mathtt{v}[\mathtt{n}/2\cdots\mathtt{n}-1]$
   - `merge`$(\mathtt{v}[0\cdots\mathtt{n}/2-1],\mathtt{v}[\mathtt{n}/2\cdots\mathtt{n}-1]) \longrightarrow \mathtt{v}[0\cdots\mathtt{n}-1]$

   Definiu-se nas aulas teóricas o que é o `merge`. O maior número de comparações efectuadas quando se faz o `merge` de 2 vectores de com $n/2$ elementos cada é $n-1$.

   (a) Ilustre a execução do mergesort para o vector

$$v[\,] = \boxed{9}\boxed{7}\boxed{8}\boxed{5}\boxed{1}\boxed{3}\boxed{6}\boxed{2}$$

[1] Encontra muitos mais exercícios de recorrências nos apontamentos fornecidos e nas folhas da disciplina de Complexidade disponíveis a partir de `www.ncc.up.pt/~acm/aulas/cpx07`.

(b) Indique uma recorrência para um majorante do número de comparações efectuadas (são todas efectuadas nas chamadas de `merge`) pelo algoritmo.

(c) Resolva a recorrência pelo método "tabelar / suspeitar / demonstrar".

6. O que está errado com a seguinte demonstração baseada no método da indução finita?
Teorema (errado!). Considerem-se n rectas concorrentes no plano (qualquer das rectas intersecta todas as outras; diremos simplesmente: n rectas "não paralelas"). Então as n rectas intersectam-se todas num único ponto.
Prova (errada!).
Caso base $n = 1$: verdadeiro (não há intersecções a considerar).
Caso base $n = 2$ (por segurança consideramos também este caso base): 2 rectas não paralelas intersectam-se num ponto.
Passo indutivo: propriedade válida para n implica propriedade válida para $n+1$: consideremse $n + 1$ rectas não paralelas, sejam as rectas 1, 2,. . . , $n + 1$.

   Pela hipótese indutiva as n primeiras rectas (1, 2,. . . , n) intersectam-se num só ponto, seja P. E também, pela hipótese indutiva as n últimas rectas (2, 3,. . . , $n+1$) intersectam-se num só ponto, seja P'. Como as $n - 1$ rectas (2, 3,. . . , n) são comuns aos 2 conjuntos, concluimos que P e P' são necessariamente o mesmo ponto e que esse ponto é a intersecção das $n + 1$ rectas.

7. • Os valores de uma sequência `a[0..n-1]` são todos distintos. Considere o seguinte programa para determinar o maior e o menor valor da sequência

```
dados: n, a[0..n-1]
resultado: (mínimo,máximo)
    função maxmin(n,a):
      imin=0
      for i=1 to n-1:
        if a[i]<a[imin]:        (*)
          imin=i
      imax=0
      for i=1 to n-1:
        if a[i]>a[imax]:        (*)
          imax=i
      return (a[imin],a[imax])
```

(a) Quantas comparações envolvendo valores de `a` são efectuadas (como função de `n`)? Essas comparações ocorrem nas linhas `(*)`.

(b) Determine um algoritmo que efectua um número significativamente menor de comparações (envolvendo valores de `a`) ou, em alternativa, mostre que tal algoritmo não existe.

Equação característica não homogénea
Suponhamos que a equação geral da recorrência tem a forma

$$a_0 f_n + a_1 f_{n-1} + \cdots + a_k f_{n-k} = b^n p(n)$$

onde b é uma constante e $p(n)$ é um polinómio em n; seja d o seu grau. Então as soluções da recorrência com a equação geral indicada podem obter-se a partir das raízes da equação

$$(a_0 r^k + a_1 r^{k-1} + \cdots + a_k)(r - b)^{d+1} = 0$$

usando o método que foi explicado para o caso das equações homogéneas.