2 Recursão

Cf. Secção 3.5 na 5a edição do livro *Data Structures and Algorithms* do Goodrich&Tamassia e secção 5.3 na 6a edição.

1. (a) Recorrendo à fórmula

$$x^n = \begin{cases} 1 & \text{se } n = 0 \\ x^{n-1} \cdot x & \text{caso contrário} \end{cases}$$

escreveu-se a seguinte função recursiva para calcular x^n:

```
double exp(double x, int n) {
  if (n=)0
    return 1;
  else
    return x*exp(x,n-1);
}
```

Siga a execução duma chamada $exp(2,3)$.

(b) Quantas chamadas recursivas a *exp* são efectuadas para calcular x^n?

(c) Escreva uma versão que seja *tail recursive* e compare-a com a anterior.

(d) Escreva uma função recursiva que calcule x^n com um número de chamadas $\mathcal{O}(\log n)$.

2. (a) Escreveu-se o seguinte procedimento recursivo para imprimir a representação binária do número natural n:

```
//@requires n>=0
void binRep(int n) {
  if (n<2)
    print(n);
  else{
    binRep(n/2);
    print(n%2);
  }
}
```

Siga a execução duma chamada $binRep(12)$.

(b) Quantas chamadas recursivas a *binRep* são efectuadas para imprimir a representação binária do número n?

(c) Escreva uma versão que seja *tail recursive* e compare-a com a anterior em termos de números de operações realizadas e espaço ocupado.

3. (a) Escreva uma função recursiva para calcular o n-ésimo número de Fibonacci, o qual é definido como se segue:

$$\mathrm{fib}(n) = \begin{cases} 1 & \text{se } n = 1 \text{ ou } n = 2 \\ \mathrm{fib}(n-1) + \mathrm{fib}(n-2) & \text{caso contrário} \end{cases}$$

(b) Argumente que esta função efectua $\mathcal{O}(2^n)$ chamadas para calcular $fibn$.

(c) Escreva uma função recursiva que calcule $fibn$ em $\mathcal{O}(n)$ chamadas e memória constante.

4. Um vector v é *bitonic* se é constituído por uma sequência crescente de inteiros seguida de uma sequência decrescente. Considere o problema de encontrar o maior elemento de um destes vectores.

   (a) Conceba um algoritmo que resolva este problema e indique qual a sua complexidade temporal assintótica.

   (b) Considere o seguinte procedimento que implementa um algoritmo recursivo que resolve este problema.

```
//@requires v is bitonic
//@requires lo>=0 && lo<v.length
//@requires hi>=0 && hi<v.length
//@requires lo<=hi
int max(int[] v, int lo, int hi) {
    if (hi == lo)
      return v[hi];

    int mid = lo + (hi - lo) / 2;

    if (v[mid] < v[mid + 1])
      return max(v, mid+1, hi);
    if (v[mid] > v[mid + 1])
      return max(v, lo, mid);
    else
      return v[mid];
}
```

   Siga a execução duma chamada a max com um vector com os inteiros $2, 5, 7, 12, 9, 8, 7, 6, 2, 1, 0$ e $lo = 0$ e $hi = 10$.

   (c) Estime o número de chamadas recursivas na chamada a $max(a, 0, a.length)$.

5. (a) Considere o seguinte algoritmo recursivo para imprimir todas as permutações dos n primeiros elementos de x, que se assume não ter repetições.

```
Algorithm permutations(char[] x, int n){
  if n=1 then
    print(x)
  else
    for i:=0 to n-1
      swap x[i] and x[n-1]
  permutations(x,n-1)
  swap x[i] and x[n-1]
}
```

   Siga a execução duma chamada do procedimento com um vector com os caracteres a,b,c,d e $n = 4$.

   (b) Estime o número de chamadas recursivas na chamada a

$$permutations(x, n)$$

(c) Conceba um algoritmo recursivo para imprimir todas as permutações de tamanho k dos elementos de x. Note que existem $\frac{n!}{(n-k)!}$ permutações de tamanho k dos elementos de x, onde n é o tamanho de x.

6. (a) Considere a seguinte função recursiva para calcular $\binom{n}{k}$ (com $n \geq k$):

```
long binomial(int n, int k) {
   if (n = 0) || (k = 0) ||(n = k)
     return 1
   else
     return binomial(n-1, k) +
            binomial(n-1, k-1)
}
```

Siga uma execução de $binomial(10, 5)$ e estime o número de chamadas recursivas.

(b) Apresente um algoritmo mais eficiente para calcular esta função recorrendo à técnica da memorização.

7. O problema da permeabilidade surge em diferentes domínios, mas é mais facilmente descrito através de um caso particular: quando se deita algum líquido sobre um material poroso, será o liquido capaz de, percorrendo os poros existentes, chegar à base do material?

O sistema em causa pode ser modelado através de uma rede de nós de dois tipos: nós abertos ou fechados. No início todos os nós estão vazios. Mais tarde ou mais cedo, um nó enche se existe um caminho (uma cadeia de vizinhos na rede) através de nós abertos desde um nó do topo. O sistema é permeável se há algum nó na base do material que encha.

Se restringirmos o problema a 2 dimensões, com nós de forma quadrada organizados em n linhas e n colunas, o sistema pode ser modelado como uma matriz como mostrado abaixo. Neste caso, o problema reduz-se a saber se existe um nó aberto na última linha que possa ser ligado a um nó aberto na primeira linha através de nós abertos vizinhos (o de cima, o de baixo, o da direita e o da esquerda).

Escreva um algoritmo recursivo que resolva o problema da permeabilidade que não faça excessivas computações ou excessivo uso de espaço. Analise os recursos consumidos pelo seu algoritmo.