Algoritmos e Complexidade
LEI (2º ano)

7ª Ficha Prática

Ano Lectivo de 2011/12

O objectivo desta ficha é a análise do algoritmos recursivos e de equações de recorrência.

1. Considere a seguinte codificação recursiva em C do algoritmo *max sort*:

```
void maxsort (int v[], int n)
{
  int i;
  if (n > 1) {
    i = max(v, n);
    swap(v, i, n);
    maxsort(v, n-1);
  }
}
```

   em que a função `swap` troca os valores contidos em duas posições de um vector (em tempo constante) e a função `max` determina o elemento máximo de um vector (em tempo linear).

   Escreva uma relação de recorrência para o tempo de execução no pior caso da função `maxsort`. Encontre a respectiva solução.

2. Considere o seguinte algoritmo para o problema das *Torres de Hanoi*:

```
void Hanoi(int nDiscos, int esquerda, int direita, int meio)
{
  if (nDiscos > 0) {
    Hanoi(nDiscos-1, esquerda, meio, direita);
    printf("mover disco de %d para %d\n", esquerda, direita);
    Hanoi(nDiscos-1, meio, direita, esquerda);
  }
}
```

   Escreva uma relação de recorrência para este algoritmo. Desenhe a árvore de recursão do algoritmo e obtenha a partir dessa árvore um resultado sobre a sua complexidade.

3. Indique, justificando, a solução da seguinte recorrência:

$$T(n) = \begin{cases} c_1 & \text{se } n \leq 1 \\ c_2 + 5\ T(n/3) & \text{se } n > 1 \end{cases}$$

4. Considere a seguinte função recursiva:

```
void exemplo(int a[], int n) {
   int x = n/2;
   if (n>0) {
      exemplo(a,x);
      processa(a,n);
      exemplo(a+x,n-x);
   }
}
```

   Sabendo que $T_{processa}(n) = \Theta(n)$, escreva uma recorrência que descreva o comportamento temporal da função `exemplo` e indique, justificando, a solução dessa recorrência.

5. Considere a seguinte relação de recorrência que traduz a complexidade de um determinado algoritmo, em função do tamanho N do *input*.

$$T(N) = \begin{cases} c & \text{se} \quad N \leq 1 \\ \\ N + 2\ T(N/3) & \text{se} \quad N > 1 \end{cases}$$

   Apresente, justificando, uma expressão que descreva o comportamento assimptótico desse algoritmo.

6. Considere a seguinte função:

```
void example(int A[], int N) {
  int i;
  Node *p;

  for (i = 1; i <= N; i++)
      p = insert(A,N,i,p);
  convert(p,A,1);
}
```

   Sabendo que $T_{insert}(N) = \mathcal{O}(\log N)$ e que $T_{convert}(N) = \mathcal{O}(N^2)$, faça a análise assimptótica do tempo de execução da função `example` no pior caso de execução.

7. Considere o seguinte algoritmo para o cálculo de números de Fibonacci:

```
int fib (int n)
{
  if (n==0 || n==1) return 1;
  else return fib(n-1) + fib(n-2);
}
```

   Apesar de traduzir exactamente a definição da sequência de números de Fibonacci, este algoritmo é muito ineficiente.

   Escreva uma relação de recorrência para este algoritmo. Desenhe a árvore de recursão do algoritmo e obtenha a partir dessa árvore um resultado sobre a sua complexidade.