# Exemplo de Prova – ACH2002 – Introdução à Análise de Algoritmos

1) Demonstre que $n^3 - 10n^2 \in \omega(n^2)$, com base na seguinte definição:

Uma função $g(n) \in \omega(f(n))$ se, para toda constante positiva c existe uma constante positiva m, tal que $0 \leq c*f(n) < g(n)$, para todo $n \geq m$.

2) Resolva a seguinte equação de recorrência (isto é, encontre a equação fechada equivalente):
T(n) = 1 para n = 1
T(n) = 2*T(n/2) + 1 para n>1

3) Identifique a equação de recorrência para as seguintes funções, considerando o número de vezes que a operação hachurada será executada no pior caso (isto é, identifique a equação de recorrência que representa de maneira exata o número de vezes que a operação hachurada será executada em relação ao valor de n).

```
long fib(int n){
  if (n<=1) return n;
  return fib(n-1) + fib(n-2);
}
```

```
void hanoi(char ori, char dst, char aux, int n) {
  if(n == 1) {
    printf("Mova de %c para %c.\n", ori, dst);
  } else {
    hanoi(ori, aux, dst, n-1);
    hanoi(ori, dst, aux, 1);
    hanoi(aux, dst, ori, n-1);
  }
}
```

```
int buscaBin(int arr[], int el, int ini, int fim) {
  int meio;
  if (ini > fim) return -1;
  meio = (fim + ini)/2;
  if (arr[meio] < el)
    return(buscaBin(arr,el,meio+1, fim));
  if (arr[meio] > el)
    return(buscaBin(arr,el,ini, meio-1));
  return meio;
}

* para esta função, considere que n = fim-ini+1
```

4) Identifique a função de custo (complexidade temporal) da seguinte função iterativa, considerando as operações hachuradas. Isto é, o número exato de vezes que as operações serão executadas, no pior caso, em relação ao valor de n.

```
int subSequenciaSomaMaxima2(int A[], int n){
  int ini, fim, x, atual, max;
  max = 0;
  int soma[n];
  soma[0] = A[0];
  for(x=1;x<n;x++) soma[x] = soma[x-1] + A[x];

  for (ini=0; ini<n; ini++){
    for (fim=ini; fim<n; fim++){
      if (ini==0) atual = soma[fim];
      else atual = soma[fim]-soma[ini-1];
      if (atual > max) max = atual;
    }
 }
 return max;
}
```

5) Implemente três versões de uma função que tem por objetivo encontrar o menor valor entre os elementos de um arranjo. Você deve fazer uma implementação iterativa, uma versão recursiva usando princípios da indução fraca e uma versão recursiva utilizando princípios da indução forte (divisão e conquista). Assinatura das funções:
`int min(int A[], int n)` – para a implementação iterativa
`int minRec(int A[], int n)` – para a implementação baseada na indução fraca
`int minDC(int A[], int ini, int fim)` – para a implementação com Divisão e Conquista

6) Complete as relações abaixo com **o** (ó pequeno), **ω** ou **Θ**. Para esta questão, cada valor preenchido incorretamente anulará um valor preenchido corretamente. Em caso de dúvida, sugere-se deixar o item em branco.

a) $n \in$ ____ $(n*\log(n))$

b) $n \in$ ____ $(n + \log(n))$

c) $\log_{10}(n) \in$ ____ $(\log_2(n))$

d) $2^n \in$ ____ $(4^n)$

e) $2*(2^n) \in$ ____ $(2^n)$

7) Utilize o Teorema Mestre para identificar a complexidade assintótica da seguinte equação de recorrência:

$T(n) = 1$ para $n=1$

$T(n) = 4*T(n/2) + n^3$ para $n>1$

**Teorema Mestre (CLRS)**

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n)$$

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para $n$ suficientemente grande, então $T(n) \in \Theta(f(n))$