Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados II (Turmas M, N, W, F)
1º Semestre de 2012
Profs. Camilo Oliveira, Gisele Pappa, Ítalo Cunha, Loïc Cerf, William Schwartz

1ª Lista de Exercícios

- **Todos os programas devem ser feitos em C.**

1. São dados 2*n* números distintos distribuídos em dois vetores com *n* elementos A e B ordenados de maneira tal que

A[1] > A[2] > A[3] > ... > A[*n*] e B[1] > B[2] > B[3] > ... > B[*n*].

Apresente um algoritmo linear para encontrar o *n*-ésimo maior número dentre estes 2*n* elementos.

2. Considere o problema de encontrar a posição de inserção de um novo elemento em um conjunto ordenado:

A[1] > A[2] > A[3] > ... > A[*n*].

a) Apresente a situação e/ou entrada de dados em que ocorre o melhor caso e o pior caso.
b) Apresente um algoritmo para resolver o problema acima.

3. Considere a função abaixo:

```
int X(int a)
{
    if(a<=0) {
        return 0;
    } else {
        return (a + X(a-1));
    }
}
```

a) O que essa função faz?
b) Calcule a sua ordem de complexidade. Mostre como você chegou a esse resultado.
c) Escreva uma função não-recursiva que resolve o mesmo problema. Qual é a ordem de complexidade da sua função? Explique.
d) Qual implementação é mais eficiente? Justifique.

4. Considere que a multiplicação de matrizes é $O(n^3)$. Se você tivesse a opção de utilizar um algoritmo exponencial $O(2^n)$ para multiplicar duas matrizes, qual algoritmo você iria preferir? Justifique.

5. O **Casamento de Padrões** é um problema clássico em Ciência da Computação e é aplicado em áreas diversas como pesquisa genética, editoração de textos, buscas na internet, etc. Basicamente, ele consiste em encontrar as ocorrências de um **padrão P** de tamanho **m** em um **texto T** de tamanho **n**. Por exemplo, no texto T = "BELO HORIZONTE" o padrão P = "ORI" é encontrado na posição 6 enquanto o padrão P = "ORA" não é encontrado. O algoritmo mais simples para o casamento de padrões é o algoritmo da "Força Bruta", mostrado abaixo. Analise esse algoritmo e responda: qual é a função de complexidade do número de

comparações de caracteres efetuadas no melhor caso e no pior caso. Dê exemplos de entradas que levam a esses dois casos. **Explique sua resposta!**

```
typedef char TipoTexto[MaxTexto];
typedef char TipoPadrao[MaxPadrao];

void ForcaBruta(TipoTexto T, int n, TipoPadrao P, int m)
{
    // Pesquisa o padrao P[0..m-1] no texto  T[0..n-1]
    int i, j, k;
    for(i = 0; i < n; i++) {
        k = i;
        j = 0;
        while((j < m) && (T[k] == P[j])) {
            k++;
            j++;
        }
        if (j == m) {
            printf("Casamento na posicao %d\n", i);
            break;  // sai do for
        }
    }
}
```

6. Vários algoritmos em computação usam a técnica de "Dividir para Conquistar": basicamente eles fazem alguma operação sobre todos os dados, e depois dividem o problema em sub-problemas menores, repetindo a operação. Uma equação de recorrência típica para esse tipo de algoritmo é mostrada abaixo. Resolva essa equação de recorrência.

$$\begin{cases} T(n) = 2T(n/2) + n; \\ T(1) = 1; \end{cases}$$

7. Indique para cada par de expressões (A,B) na tabela abaixo, se A é O, o, Ω, ω ou Θ de B. Assuma que $k \geq 1$ e $0 < \varepsilon < 1 < c$ são constantes. Sua resposta deve ser da forma SIM ou NÃO.

Nota: $\log^k n = \underbrace{\log\log\cdots n}_{k}$ e $n! \approx \left(\frac{n}{e}\right)^n$.

	A	B	O	o	Ω	ω	Θ
(i)	$\log^k n$	n^ε					
(ii)	n^k	c^n					
(iii)	c^n	$c^{n/2}$					
(iv)	$\log(n!)$	$\log(n^n)$					
(v)	$\log^{k+1} n$	$\log^k n$					
(vi)	c^ε	$(c+1)^\varepsilon$					

8. Qual algoritmo você preferiria: um que requer n^5 passos ou um que requer 2^n passos? Justifique sua resposta.

9. Indique se as afirmativas a seguir são verdadeiras e justifique sua resposta:

a) $2^{n+1} = O(2^n)$

b) $2^{2n} = O(2^n)$

c) $f(n) = O(u(n))$ e $g(n) = O(v(n)) \rightarrow f(n) + g(n) = O(u(n) + v(n))$

d) $f(n) = O(u(n))$ e $g(n) = O(v(n)) \rightarrow f(n) - g(n) = O(u(n) - v(n))$

10. Considerando que a operação relevante é o número de vezes que a operação soma é executada, apresente a função de complexidade de tempo para:

a)
```
for i ← 1 to n do
   for j ← 1 to n do
      for k ← 1 to n do
         temp ← temp + i + j + k
```

b)
```
for i ← 1 to n do
   for j ← 1 to i do
      for k ← 1 to j do
         temp ← temp + i + j + k
```

c)
```
for i ← 1 to n do
   for j ← 1 to n do
      for k ← i to n do
         temp ← temp + i + j + k
```

d)
```
for i ← 1 to n do
   for j ← i to n do
      for k ← i to n do
         temp ← temp + i + j + k
```

e)
```
for i ← 1 to n do
   for j ← i to n do
      for k ← i to j do
         temp ← temp + i + j + k
```

11. Resolva as seguintes equações de recorrência:

a) $\begin{cases} T(n) = T(n-1) + c & c \text{ constante, } n > 1 \\ T(1) = 0 \end{cases}$

b) $\begin{cases} T(n) = T(n-1) + 2^n & n \geq 1 \\ T(0) = 1 \end{cases}$

c) $\begin{cases} T(n) = cT(n-1) & c, k \text{ constantes}, n > 0 \\ T(0) = k \end{cases}$

d) $\begin{cases} T(n) = 3T(n/2) + n & n > 1 \\ T(1) = 1 \end{cases}$

e) $\begin{cases} T(n) = 3T(n-1) - 2T(n-2) & n > 1 \\ T(0) = 0 \\ T(1) = 1 \end{cases}$

12. Considere o algoritmo a seguir, supondo que a operação crucial é "inspecionar elemento". O algoritmo inspeciona os n elementos de um conjunto e, de alguma forma, consegue descartar 2/5 dos elementos e fazer uma chamada recursiva sobre os 3n/5 elementos restantes.

a) Escreva uma equação de recorrência que descreva esse comportamento
b) Converta a equação de recorrência para um somatório
c) Dê a fórmula fechada para esse somatório

```
void Pesquisa (int n)
{
   if (n < 1)
         'inspecione elemento';
         termine;
   else {
         'para cada um dos elementos, inspecione elemento';
         Pesquisa(3 * n / 5);
   }
}
```

13. Torre de Hanói. Em 1883, o matemático francês Edouard Lucas criou um jogo chamado Torre de Hanói. O jogo começa com um conjunto de oito discos empilhados em tamanho decrescente em uma das três varetas, conforme mostrado na Figura 1. O objetivo do jogo é transferir toda a torre para uma das outras varetas, movendo um disco de cada vez, mas nunca movendo um disco maior sobre um menor. Esboçe uma função recursiva para resolver este problema e calcule sua complexidade.

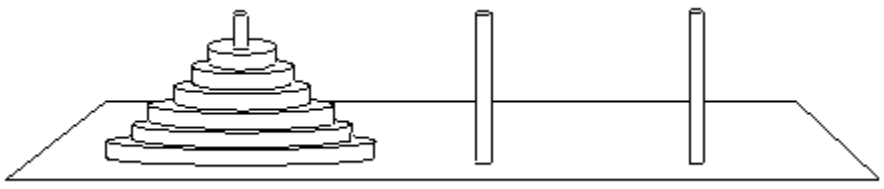
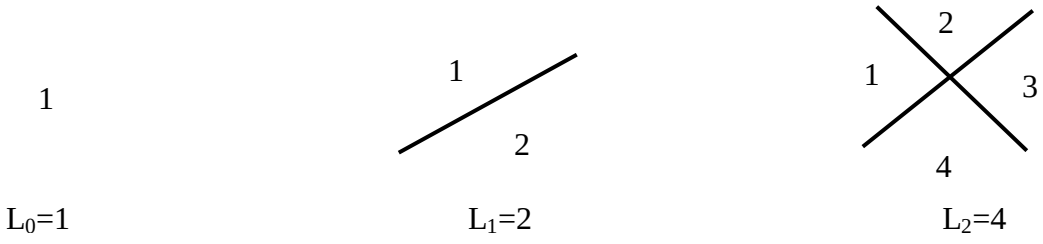


Figura 1: Configuração inicial da Torre de Hanói.

14. Linhas no plano ou Cortando a sua pizza favorita. Quantas fatias de pizza uma pessoa pode obter ao fazer *n* cortes retos com uma faca? Ou, expressando de outra forma, qual é o número máximo de regiões L_n determinado por *n* retas no plano? Lembre-se que um plano sem nenhuma reta tem uma região, com uma reta tem duas regiões e com duas retas tem quatro regiões, conforme mostrado na figura 2.



$L_0=1$ $L_1=2$ $L_2=4$

Figura 2: Regiões no plano.

15. Use o teorema mestre para derivar um limite assintótico Θ para as seguintes recorrências:

a) $T(n) = 2T(n/2) + n - 1$

b) $T(n) = 3T(n/2) + n$

c) $T(n) = 4T(n/2) + n^2$

d) $T(n) = 4T(n/2) + n^3$

Fórmulas Úteis: $\sum_{i=1}^{n} k = n.k$ $\sum_{i=1}^{n} i = \frac{n.(n+1)}{2}$ $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ $\sum_{i=1}^{n} a^i = a\frac{a^n - 1}{a - 1}$