1ª Lista de exercícios

1.1 Responder se é certa ou errada cada afirmativa abaixo:

(i) O algoritmo que calcula o fatorial de forma recursiva requer apenas uma quantidade constante de memória.

(ii) O algoritmo abaixo requer o armazenamento do vetor fat, com n+1 elementos.

```
fat[0] = 1
for(j = 1; j <= n; j = j+1)
  fat[j] = j * fat[j-1];
```

1.2 Desenvolver um algoritmo não recursivo para o cálculo do fatorial de inteiro $n \geq 0$, de tal forma que prescinda do armazenamento de qualquer vetor.

1.3 Mostrar que o algoritmo para o problema das Torres de Hanói, requer exatamente 2^n-1 movimentos de disco para terminar.

1.4 Reescrever o algoritmo do problema das Torres de Hanói, de forma que a recursividade pare no nível correspondente a n = 1, e não a n = 0, como no algoritmo do texto. Há alguma vantagem em realizar essa modificação? Qual?

1.5 Escrever as seguintes funções em notação O:

$n^3 - 1$
$n^2 + 2 \log n$
$3n^n + 5.2^n$
$(n - 1)^n + n^{n-1}$
302

1.6 A seqüência de Fibonacci é uma seqüência de elementos $f_1,\cdots, f_n$, definida do seguinte modo:

$f_1 = 0,$
$f_2 = 1,$
$f_j = f_{j-1} + f_{j-2}, \; j > 2.$

Elaborar um algoritmo, não recursivo, para determinar o elemento f_n da seqüência, cuja complexidade seja linear em n.

1.7 Considere a seguinte seqüência de elementos $g_1, \ldots, g_n$ para um dado valor de k.

$g_j = j - 1, \; 1 \leq j \leq k;$
$g_j = g_{j-1} + g_{j-2}, \; j > k.$

Elaborar um algoritmo para determinar o elemento g_n da seqüência, cuja complexidade seja O(n).

1.8 Determine a complexidade dos algoritmos abaixo, justificando sua resposta:

a)

```
int MAIOR (int N, int A[])
{
  int i, max = A[0];
  for (i=1; i<N, i++)
    if (max < A[i])
      max = A[i];
  return max;
}
```

b)

```
void ORDENA (int N, int A[])
{
  int i;
  for(i=0; i<N-1; i++)
    for(j=0; j<N-1-i; j++)
      if (A[j] > A[j+1])
      {
        x = A[j];
        A[j] = A[j+1];
        A[j+1] = x;
      }
}
```

c)

```
n = 0;
while (n < 10)
{
  k = 1;
  while (k < 10)
  {
    ...
    k = k + 1;
  }
  n = n + 1;
}
```

1.9) Considere o problema de encontrar um elemento em um conjunto ordenado de dados: $a_1 < a_2 < a_3 < ....< a_n$. Elabore um algoritmo eficiente para este problema. Em que situações o algoritmo é ótimo? Em que situações o algoritmo apresenta o pior desempenho? Apresente um limite superior para este problema em função de n e exemplifique.