# Estrutura de Dados - 1o. período de 2011

## Gabarito da Primeira Avaliação à Distância

1. (1,0) Escrever as seguintes funções em notação $O$:
   $n^3 + 10$; $n^2 - 2\log n$; $8n^n - 5.4^n$; $(n-1)^n + n^{n-1}$; $51$.

   Resposta: $n^3 + 10 = O(n^3)$; $n^2 - 2\log n = O(n^2)$; $8n^n - 5.4^n = O(n^n)$; $(n-1)^n + n^{n-1} = O(n^n)$; $51 = O(1)$.

2. (1,5) Para cada item abaixo, responda "certo" ou "errado", justificando:

   a. Se a complexidade de caso pior de um algoritmo for $f$, então o número de passos que o algoritmo efetua, qualquer que seja a entrada, é $O(f)$.

   Resposta: Certo, pois qualquer que seja o número de passos, estará limitado superiormente por $f$, que é a função do pior caso.

   b. Se a complexidade de melhor caso de um algoritmo for $f$, então o número de passos que o algoritmo efetua, qualquer que seja a entrada, é $\Theta(f)$.

   Resposta: Errado, pois o pior caso pode corresponder a um número de passos dado por uma função $g$ que não seja $\Theta(f)$.

   c. A complexidade de pior caso de um algoritmo para um certo problema é estritamente maior do que qualquer limite inferior para o problema.

   Resposta: Errado, pois o pior caso de um algoritmo pode ser igual ao limite inferior do problema.

3. (1,5) Seja $f_1, f_2, \ldots, f_n$ uma sequência de elementos definida do seguinte modo: $f_1 = 0, f_2 = 1, f_3 = 1, f_j = f_{j-1} - f_{j-2} + f_{j-3}$ para $j > 3$.

   (i) Elabore um algoritmo *não recursivo* para determinar o elemento $f_n$ da sequência. Calcule sua complexidade em função de $n$.

   Resposta:

   $f[1] := 0;$
   $f[2] := 1;$
   $f[3] := 1;$
   para $j = 4 \cdots n$ faça
   $\quad f[j] := f[j-1] - f[j-2] + f[j-3];$

   A complexidade do algoritmo acima é $O(n)$.

(ii) Em seguida, elabore um algoritmo *recursivo* para resolver este mesmo problema. Comparar as complexidades dos algoritmos recursivo e não recursivo.

Resposta:

função $seq(j)$
    se $j = 1$ então
        *retornar* 0
    senão se $j = 2$ então
            *retornar* 1
        senão se $j = 3$ então
                *retornar* 1
            senão
                *retornar* $(seq(j-1) - seq(j-2) + seq(j-3)))$;

Chamada externa: $seq(n)$

Complexidade: É dada pela seguinte equação de recorrência:
$T(1) = T(2) = T(3) = 1$
$T(j) = T(j-1) + T(j-2) + T(j-3)$

Resolvendo esta recorrência, verificamos que a complexidade deste algoritmo é $O(3^n)$. Comparando as complexidades obtidas, vemos que o algoritmo não recursivo é linear, enquanto o recursivo é exponencial.

4. (1,5) Determinar a expressão da complexidade média de uma busca não ordenada de $n$ chaves, em que a probabilidade de busca da chave $i$ é o dobro da probabilidade de busca da chave $i-1$, para $i = 2, ...n$. Supor, ainda, que a probabilidade de a chave procurada se encontrar na lista é igual a 1.

   Resposta: Sejam $E_1, E_2, \cdots, E_n$ as entradas correspondentes ao sucesso, e $p(E_1) = p$ a probabilidade de busca da entrada $E_1$.

   Temos então que $p + 2p + 4p + \cdots + 2^{n-1}p = 1$. Utilizando a fórmula para soma dos termos de uma PG, temos $p = 1/(2^n - 1)$.

   O número de passos necessários para cada entrada $E_i$ é $t(E_i) = i$, $1 \leq i \leq n$.

   Logo, a expressão da complexidade média é dada por:

$$C.M. = (1.1 + 2.2 + 3.4 + 4.8 + 5.16 + \cdots + n.2^{n-1})\frac{1}{(2^n - 1)}$$

5. (1,5) Sejam $L_1$ e $L_2$ duas listas ordenadas, em alocação sequencial. Apresentar um algoritmo que construa uma lista ordenada contendo os elementos comuns a $L_1$ e $L_2$ (supor que tanto $L_1$ como $L_2$ não contêm elementos repetidos.) Determinar a complexidade do algoritmo, em função dos tamanhos $|L_1|$ e $|L_2|$ de $L_1$ e $L_2$, respectivamente.

   Resposta: Sejam $|L_1| = m$, $|L_2| = n$, e $L_3$ a lista resultante, com tamanho suficiente para armazenar todos os elementos comuns a $L_1$ e $L_2$.

```
i := 1;          // utilizado para percorrer L1
j := 1;          // utilizado para percorrer L2
k := 1;          // utilizado para percorrer L3
enquanto i ≤ m e j ≤ n faça
      se L1[i] < L2[j] então
            i := i + 1;
      senão se L1[i] > L2[j] então
            j := j + 1;
      senão
            L3[k] = L1[i];
            i := i + 1;
            j := j + 1;
            k := k + 1;
```

Como o algoritmo percorre $L_1$ e $L_2$ apenas uma vez, executando um número fixo de instruções a cada elemento lido, sua complexidade é linear no tamanho das listas, ou seja, $O(m+n)$.

6. (1,5) Para cada item abaixo, responda "certo" ou "errado", justificando:

   (a) Suponha que $n$ elementos $e_i$ foram incluídos em uma pilha, na ordem $e_1, e_2, \ldots, e_n$, de tal forma que ocorreram remoções de alguns desses elementos, entre as inclusões. Então a ordem em que os elementos foram retirados é $e_n, e_{n-1}, \ldots, e_1$.

   Resposta: Errado. A única maneira de $e_n$ ser o primeiro elemento retirado da pilha é todas as inclusões ocorrerem sem nenhuma remoção entre elas.

   (b) Suponha que $n$ elementos $e_i$ foram incluídos em uma fila, na ordem $e_1, e_2, \ldots, e_n$, de tal forma que ocorreram remoções de alguns desses elementos, entre as inclusões. Então a ordem em que os elementos foram retirados é $e_n, e_{n-1}, \ldots, e_1$.

   Resposta: Errado. Em uma fila, a ordem de remoção dos elementos é sempre igual à ordem de inserção, independentemente da ocorrência ou não de remoções entre as inclusões. Logo, se os elementos foram inseridos na ordem $e_1, e_2, \ldots, e_n$, então podem ser retirados somente nesta mesma ordem.

7. (1,5) Seja $1, 2, \ldots, n$ uma seqüência de elementos que serão inseridos e posteriormente retirados de uma pilha $P$ uma vez cada. A ordem de inclusão dos elementos na pilha é $1, 2, \ldots, n$, enquanto a ordem de remoção depende das operações realizadas. Por exemplo, com $n = 3$, a seqüência de operações

   "incluir em $P$, incluir em $P$, retirar de $P$, incluir em $P$, retirar de $P$, retirar de $P$"

   produzirá a permutação $2, 3, 1$ a partir da entrada $1, 2, 3$. Representando por $I, R$, respectivamente, as operações de inserção e remoção da pilha, a permutação $2, 3, 1$ pode ser denotada por $IIRIRR$. De um modo geral, uma permutação é chamada *admissível* quando ela puder ser obtida mediante uma sucessão de inclusões e remoções em uma pilha a partir da permutação $1, 2, \ldots, n$. Assim, por exemplo, a permutação $2, 3, 1$ é admissível. Pede-se:

(i) Determinar a permutação correspondente a $IIIRRIRR$, $n = 4$.

Resposta: 3241.

(ii) Dê um exemplo de permutação não admissível.

Resposta: 4123.

(iii) Se o mesmo problema for aplicado a uma fila, ao invés de uma pilha, quais serão exatamente as permutações admissíveis ?

Resposta: Neste caso, a única permutação admissível será $1, 2, \ldots, n$.