Gabarito da Primeira Avaliação à Distância

1. (1,0) Escrever as seguintes funções em notação O:
   $n^3 + n \log n$; $n - 5$; $\log^2 n$; $n^8 + 8n!$; $(n+1)^n$; 7; $\sqrt{n}$; $\log n$.

   Resposta:
   $O(n^3)$
   $O(n)$
   $O(\log^2 n)$
   $O(n!)$
   $O(n^n)$
   $O(1)$
   $O(\sqrt{n})$
   $O(\log n)$

2. (2,0) Para cada item abaixo, responda "certo" ou "errado", justificando em ambos os casos.

   a. Se a complexidade de melhor caso de um algoritmo for $O(f)$, então o número de passos que o algoritmo efetua no pior caso é $\Omega(f)$.

   Resposta: Errado. Sendo a complexidade de melhor caso de um algoritmo $\Theta(g)$, $g = O(f)$ e $f \neq O(g)$ então a complexidade de melhor caso deste algoritmo será $\Omega(g)$ e não $\Omega(f)$.

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

   Resposta: Certo. Se a complexidade de pior caso de um algoritmo for $\Omega(f)$, então o número de passos que o algoritmo efetua, qualquer que seja a entrada, é limitado superiormente por $O(f)$.

   c. Se um limite inferior para um problema P é n^3, então todo algoritmo para P tem complexidade de pior caso $\Omega(n^3)$.

   Resposta: Certo. Pela definição de limite inferior, se n^3 é um limite inferior para P, então qualquer algoritmo para P, ótimo ou não, tem complexidade de pior caso $\Omega(n^3)$.

   d. Se dois algoritmos A_1 e A_2 têm complexidades de pior caso $O(n^2)$ e $O(n^3)$, respectivamente, então A_2 não é ótimo.

   Resposta: Um algoritmo ótimo tem a menor complexidade de pior caso possível. Como A_2 tem complexidade de pior caso maior que A_1, então A_2 não pode ser ótimo.

3. (1,5) Escreva um algoritmo recursivo para encontrar os dois maiores elementos de uma lista com n elementos, baseado no seguinte princípio: divide-se a lista ao meio e encontra-se recursivamente os dois maiores elementos das duas metades; a seguir combina-se as duas soluções parciais na solução final.

   Resposta:

   função $dois\_maiores(i, j)$
   se $i = j$ então
   retornar $(V[i], V[i])$
   senão
   se $j = i + 1$ então
   retornar $(MAX(V[i], V[j]), MIN(V[i], V[j]))$
   senão
   $meio = (i + j)/2$
   $(maior1, segMaior1) = dois\_maiores(i, meio)$
   $(maior2, segMaior2) = dois\_maiores(meio + 1, j)$
   aux1:= $MAX(maior1, maior2)$
   aux2:= $MAX(MIN(maior1, maior2), MAX(segMaior1, segMaior2))$
   retornar (aux1,aux2)

4. (1,5) Dentre as estrutura de dados estudadas (levando em conta inclusive se estão ou não ordenadas), indique qual(is) apresenta(m) a menor e maior complexidade(s), respectivamente, para cada operação abaixo. Justifique:

Estruturas de dados estudadas	**Complexidade para busca de um elemento**
Lista sequencial não ordenada	$O(n)$
Lista sequencial ordenada	$O(\log n)$
Lista encadeada ordenada	$O(n)$
Lista encadeada não ordenada	$O(n)$

Estruturas de dados estudadas	**Complexidade para remoção de um elemento**
Lista sequencial não ordenada	$O(n)$
Lista sequencial ordenada	$O(n)$
Lista encadeada ordenada	$O(1)$
Lista encadeada não ordenada	$O(1)$

   (a) Busca de um elemento

   Resposta: Lista sequencial ordenada. Neste caso, podemos usar busca binária, e o pior caso da busca terá tempo $O(\log n)$. Todas as demais estruturas, no pior caso, necessitarão de um tempo $O(n)$.

   (b) Remoção de um elemento (desconsiderando a necessidade de uma busca prévia)

   Resposta: Listas encadeadas ordenada e não ordenada. Nestas estruturas, basta alterar os ponteiros vizinhos ao nó a ser removido e liberar seu espaço de memória.

5. (1,5) Seja V um vetor com n posições. Escreva um algoritmo que construa uma lista encadeada L, com nó cabeça, a partir de V de forma que os elementos de L sejam os de V, em ordem inversa. Por exemplo, se V contiver os elementos 1 7 3 5 8, nesta ordem, a lista L deverá conter os elementos 8 5 3 7 1, nesta ordem.

   Resposta: Seja V o vetor com n elementos, indexado de 1 a n.

   Algoritmo:
   ocupar(PTLISTA)
   PONT := PTLISTA;

   para $i := n \ldots 1$ faça
   ocupar(PT)
   PT↑.info := $V[i]$
   PT↑.prox := λ
   PONT↑.prox := PT
   PONT:= PONT↑.prox

6. (1,5) Sejam L_1 e L_2 duas listas ordenadas, simplesmente encadeadas com nó-cabeça. Apresentar um algoritmo que construa uma lista ordenada contendo os elementos que pertencem exclusivamente a L_2.

   Resposta:

   Algoritmo:

   $ocupar(ptlista3)$ % ponteiro para a nova lista L_3

   $ptlista3 \uparrow .prox := \lambda$

   $pont1 := ptlista1 \uparrow .prox$ % ponteiro para a lista L_1

   $pont2 := ptlista2 \uparrow .prox$ % ponteiro para a lista L_2

   $ptaux := ptlista3$

   enquanto $pont1 \neq \lambda$ e $pont2 \neq \lambda$ faça

   se $pont1 \uparrow .info = pont2 \uparrow .info$ então

   $pont1 := pont1 \uparrow .prox$

   $pont2 := pont2 \uparrow .prox$

   senão

   se $pont1 \uparrow .info < pont2 \uparrow .info$ então

   $pont1 := pont1 \uparrow .prox$

   senão % o elemento pertence exclusivamente a L_2

   $incluir\_no(pont2, ptaux)$

   enquanto $pont2 \neq \lambda$ faça

   $incluir\_no(pont2, ptaux)$

procedimento $incluir\_no(pont2, ptaux)$

$ocupar(pt)$

$pt \uparrow .info := pont2 \uparrow .info$

$pt \uparrow .prox := \lambda$

$ptaux \uparrow .prox := pt$

$ptaux := pt$

$pont2 := pont2 \uparrow .prox$

7. (1,0) 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 $IIRIRRIR$, $n = 4$.
   Resposta: 2 3 1 4.

   (ii) Dê um exemplo de permutação não admissível.
   Resposta: 4 1 2 3. Após a remoção do elemento 4 (primeiro a ser removido), o elemento 3 encontra-se no topo da pilha, sendo portanto impossível retirar o elemento 1 na proxima remoção.