Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
Gabarito da AP1 - Segundo Semestre de 2014

Nome -
Assinatura -

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1. Defina:

   (a) (1,0) Limite inferior de um problema.
   Resposta: O limite inferior de um problema P é uma função ℓ tal que a complexidade de pior caso de qualquer algoritmo que resolva P é $\Omega(\ell)$.

   (b) (1,0) Algoritmo ótimo.
   Resposta: Um algoritmo é ótimo quando sua complexidade de pior caso é igual ao limite inferior para o problema.

   (c) (1,0) Algoritmo recursivo.
   Resposta: Um algoritmo recursivo é aquele que contém, em sua descrição, uma ou mais chamadas a si mesmo (chamadas recursivas).

2. Considere os algoritmos **ordenação por seleção** e **ordenação pelo método da bolha**, aplicados a uma lista com n elementos em ordem inversa de ordenação (Ex: 8 7 6 5 4 3 2 1).

   (a) (1,0) Qual dos dois algoritmos efetua mais **comparações** entre elementos? Justifique sua resposta.
   Resposta: Ambos executam o mesmo número de comparações:

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

   (b) (1,0) Qual dos dois algoritmos efetua mais **trocas** entre elementos? Justifique sua resposta.
   Resposta: O algoritmo de ordenação por bolha, pois executa $n(n-1)/2$ trocas, enquanto o algoritmo de ordenação por seleção executa n trocas.

3. (1,5) Escreva um algoritmo que execute a seguinte tarefa: Dada uma lista sequencial não ordenada com n elementos ($n \geq 1$), encontre **o maior e o segundo maior elementos** desta lista. Seu algoritmo deverá percorrer a lista uma única vez.

Resposta:

se $L[1] > L[2]$ **então**
    $maior1 := L[1]$
    $maior2 := L[2]$
senão
    $maior1 := L[2]$
    $maior2 := L[1]$
para $i = 3$ **até** n **faça**
    se $L[i] > maior1$ **então**
        $maior2 := maior1$
        $maior1 := L[i]$
    senão
        se $L[i] > maior2$ **então**
            $maior2 := L[i]$
imprimir ("O maior elemento é:", $maior1$)
imprimir ("O segundo maior elemento é:", $maior2$)

4. (1,5) Suponha que duas pilhas P_1 e P_2 compartilhem a mesma memória, constituída de um vetor de tamanho n. Isto é, sobre o mesmo vetor X, com elementos $x_1, \ldots, x_n$, são projetadas duas pilhas, a primeira desenvolvendo-se de x_1 para x_n, e a segunda de x_n para x_1. Escreva algoritmos de inclusão e remoção de elementos em P_1 e P_2. Em particular quais seriam as condições de overflow para P_1 e P_2 ?

   Resposta: Seja b uma variável booleana que indica em qual pilha ocorrerá a inserção/remoção ($b = verdadeiro$ indica que a operação será feita em P_1). Inicialmente, temos $topo1 = 0$ e $topo2 = n + 1$, indicando que as pilhas P_1 e P_2 estão vazias.

   Inserção:

   se $topo2 \neq topo1 + 1$ **então**
       se b **então**
           $topo1 := topo1 + 1$
           $X[topo1] := novo\_valor$
       senão
           $topo2 := topo2 - 1$
           $X[topo2] := novo\_valor$

Remoção:

se b **então**
    se $topo1 \neq 0$ **então**
        $valor\_recuperado := X[topo1]$
        $topo1 := topo1 - 1$
    senão underflow em P_1
senão
    se $topo2 \neq n + 1$ **então**
        $valor\_recuperado := X[topo2]$
        $topo2 := topo2 + 1$
    senão underflow em P_2

A situação de overflow ocorre quando $topo2 = topo1 + 1$, e tentamos inserir um elemento em P_1 ou em P_2.

5. (2,0) Considere uma lista simplesmente encadeada L com n nós, que armazenam números inteiros. Elabore um algoritmo que crie uma nova lista L' contendo somente nós com os números pares que ocorrem em L. Os números devem aparecer em L' na mesma ordem em que aparecem em L. Por exemplo, se L contiver os números 1, 8, 4, 5, 7, 8, 6, 3, nesta ordem, então L' conterá os números 8, 4, 8, 6, nesta ordem. Qual a complexidade do seu algoritmo?

   Resposta: A complexidade do algoritmo a seguir é $\Theta(n)$, pois percorre a lista L apenas uma vez, e para cada nó de L executa um número constante de passos.

```
pt := L          % considerando que L não tem nó cabeça
L' := λ
ultimo := λ
enquanto pt ≠ λ faça
        se pt↑.info mod 2 = 0 então
                ocupar(novo)
                novo↑.info := pt↑.info
                novo↑.prox := λ
                se ultimo ≠ λ então          % L' já contém algum nó
                        ultimo↑.prox := novo
                senão
                        L' := novo
                ultimo := novo
        pt := pt↑.prox
```