Estrutura de Dados - 2o. período de 2007

Gabarito da Primeira Avaliação à Distância

1. Para cada item abaixo, responda "certo" ou "errado", justificando:

   a. A função $n^2 + n^2 \log n$ é $O(n^2)$.
   Resposta: Errado. A função é $O(n^2 \log n)$. Podemos também dizer que ela é $\Omega(n^2)$.

   b. A função $n^2 + n^2 \log n$ é $\Theta(n^2 \log n)$.
   Resposta: Certo. Como ela é tanto $O(n^2 \log n)$ quanto $\Omega(n^2 \log n)$, podemos dizer que ela é $\Theta(n^2 \log n)$.

   c. A função $n^2 + n^2 \log n$ é $\Omega(n \log n)$.
   Resposta: Certo. Como ela é $\Omega(n^2 \log n)$, obviamente também é $\Omega(n \log n)$, já que $n^2 \log n$ limita superiormente $n \log n$, considerando valores assintóticos.

   d. Se a complexidade assintótica de pior caso de um algoritmo for $O(f)$, então o número de passos que o algoritmo efetua para uma entrada de tamanho n é no máximo igual a $f(n)$.
   Resposta: Errado. Nas notações de complexidade, as constantes aditivas e multiplicativas são desprezadas. Assim, se o algoritmo efetua, por exemplo, $3n$ passos para qualquer entrada de tamanho n, temos que este algoritmo é $O(n)$. No entanto, seu número de passos é superior a $f(n) = n$.

   e. Se as complexidades assintóticas de pior caso dos algoritmos A e B são iguais a $\Theta(f)$, então, dada uma entrada E de tamanho n, A e B executam o mesmo número de passos para resolver E.
   Resposta: Errado. Não necessariamente E é uma entrada que represente o pior caso tanto para A quanto para B. Logo, para uma mesma entrada, não podemos afirmar que exista uma função que represente o número de passos executados tanto por A quanto por B. Além disso, mesmo considerando que E seja uma entrada de pior caso para A e para B, podemos ter, por exemplo, o número de passos executados por A no pior caso igual a $2f(n)$, e por B igual a $3f(n)$ (já que as constantes foram desprezadas ao se afirmar que as complexidades de pior caso de A e B são $\Theta(f)$).

   f. Se um algoritmo ótimo que resolve um problema P tem complexidade de pior caso $O(f)$, então f é um limite inferior para P.
   Resposta: Errado. Se este algoritmo ótimo possui complexidade de pior caso $\Theta(n^2)$, por exemplo, podemos afirmar que seu pior caso é $O(n^3)$. No entanto, n^3 não é limite inferior para P, mas apenas n^2.

2. Elabore um algoritmo que resolva o seguinte problema: Dados dois números inteiros positivos m e n, onde $m \geq n$, achar o máximo divisor comum entre m e n. Calcule a complexidade do seu algoritmo em função de m e n.

   Resposta:

   enquanto $n \neq 0$ faça
   $\quad x := m \ mod \ n$
   $\quad m := n$
   $\quad n := x$
   imprimir('O mdc é:', m)

   Complexidade: Note que, para números inteiros $m \geq n > 0$, vale: $m \ mod \ n < m/2$. Temos então que, a cada duas iterações do loop, os valores de m e de n são reduzidos a menos de suas metades, totalizando $O(\log n)$ iterações. Logo, o algoritmo é $O(\log n)$.

3. Determinar a expressão da complexidade média de uma busca "ordenada" de 10 chaves, em que a probabilidade de busca da chave i é 50% maior que a probabilidade de busca da chave $i-1$, para $i = 2, ...10$. Supor, ainda, que a probabilidade de a chave procurada se encontrar na lista é igual a 50%.

   Resposta:

   Como a busca se dá em uma lista ordenada, temos 21 entradas distintas (10 entradas em que a chave é encontrada e 11 entradas correspondentes a fracasso). Sejam $E_1, \cdots, E_{10}$ as entradas correspondentes ao sucesso e $E'_0, \cdots, E'_{10}$ as entradas correspondentes ao fracasso (representando os "espaços" entre as chaves da lista).

   Considerando a probabilidade de sucesso, temos:

$$p(E_1) + p(E_2) + \cdots + p(E_{10}) = \frac{1}{2}$$

   Seja p a probabilidade de busca da chave 1 (entrada E_1). Logo:

$$p + \frac{3}{2}p + \frac{9}{4}p + \cdots + \left(\frac{3}{2}\right)^9 p = \frac{1}{2}$$

$$p \sum_{i=1}^{10} \left(\frac{3}{2}\right)^{i-1} = \frac{1}{2}$$

$$p \left[\frac{\left(\frac{3}{2}\right)^{10} - 1}{\frac{1}{2}} \right] = \frac{1}{2}$$

$$p = \frac{1}{4} \cdot \frac{1}{\left(\frac{3}{2}\right)^{10} - 1}$$

Considerando a probabilidade de fracasso, temos:

$$p(E'_0) + p(E'_1) + \cdots + p(E'_{10}) = \frac{1}{2}$$

Assumindo que as probabilidades de $E'_0, \cdots, E'_{10}$ são iguais entre si, temos:

$$p(E'_i) = \frac{1}{22}, \quad 0 \leq i \leq 10$$

O número de passos necessários para cada entrada é:

$t(E_i) = i \;, \qquad 1 \leq i \leq 10$
$t(E'_i) = i + 1 \;, \quad 0 \leq i \leq 10$

Logo, a expressão da complexidade média é dada por:

$$\begin{aligned}
C.M. &= \sum_{i=1}^{10} p(E_i)\, t(E_i) + \sum_{i=0}^{10} p(E'_i)\, t(E'_i) \\
&= \sum_{i=1}^{10} \left[\left(\frac{3}{2}\right)^{i-1} p \,.\, i \right] + \sum_{i=0}^{10} \left[\frac{1}{22}(i+1) \right] \\
&= p \left[1.1 + 2.\frac{3}{2} + 3.\left(\frac{3}{2}\right)^2 + \cdots + 10.\left(\frac{3}{2}\right)^9 \right] + \frac{1}{22}\,.\,66 \\
&= \frac{1}{4} \cdot \frac{1}{\left(\frac{3}{2}\right)^{10} - 1} \left[1.1 + 2.\frac{3}{2} + 3.\left(\frac{3}{2}\right)^2 + \cdots + 10.\left(\frac{3}{2}\right)^9 \right] + 3
\end{aligned}$$

4. Comparar as complexidades assintóticas de melhor e pior caso dos algoritmos de busca, inserção e remoção em *listas duplamente encadeadas não ordenadas com nó cabeça.*

   Resposta:

   Busca - $O(1)$ no melhor caso, $O(n)$ no pior caso.
   Inserção - $O(1)$ em ambos os casos.
   Remoção - $O(1)$ em ambos os casos.

   Obs: Na prática, a inserção e a remoção exigem um busca prévia. Portanto, na prática, elas se tornam $O(n)$ no pior caso.

5. Seja L uma lista ordenada, simplesmente encadeada, com nó-cabeça. Elabore um algoritmo que retire de L os elementos repetidos. Calcule sua complexidade.

   Resposta:

   $pt1 := ptLista \uparrow .prox$

   $pt2 := pt1 \uparrow .prox$

   enquanto $pt2 \neq \lambda$ faça

se $pt1 \uparrow .info = pt2 \uparrow .info$ então

$pt1 \uparrow .prox := pt2 \uparrow .prox$

desocupar($pt2$)

$pt2 := pt1 \uparrow .prox$

senão

$pt1 := pt2$

$pt2 := pt2 \uparrow .prox$

Complexidade: Como o algoritmo percorre a lista uma vez, sua complexidade é $O(n)$.

6. Elabore um algoritmo iterativo (não recursivo) de busca binária. Responda: a complexidade de pior caso deste algoritmo apresenta alguma diferença em relação à versão recursiva?

Resposta:

função busca-bin(x)

$inf := 1$; $sup := n$; $result := 0$

enquanto $inf \leq sup$ faça

$meio := \lfloor (inf + sup)/2 \rfloor$

se $L[meio] = x$ então

$result := meio$

$inf := sup + 1$

senão se $L[meio] < x$ então

$inf := meio + 1$

senão $sup := meio - 1$

Complexidade: A complexidade de pior caso deste algoritmo também é $O(\log n)$, não apresentando, portanto, diferença em relação à versão recursiva.

7. Elabore um algoritmo que utilize uma *fila* para resolver o seguinte problema. Clientes vão chegando ao banco e pegam uma senha eletrônica de atendimento, que automaticamente é armazenada no sistema para indicar um novo cliente em espera. Quando um cliente termina de ser atendido, sua senha é imediatamente retirada do sistema. Sendo os clientes atendidos por ordem de chegada, faça um relatório que imprima o estado da fila de atendimento a cada novo evento (chegada de um novo cliente, ou término do atendimento de um cliente.)

Resposta: Seja F uma fila circular de tamanho n que represente o sistema. Assume-se que n é grande o suficiente para que não ocorra overflow.

```
senha := 0
ini := fim := 0
enquanto (‘banco está aberto’) faça
    se (‘chegou novo cliente’) então
        senha := senha + 1
        se (fim = 0) então
            ini := fim := 1
        senão
            fim := (fim mod n) + 1
        F[fim] := senha
        imprimeFila()
    se (‘cliente foi atendido’) então
        F[ini] := 0
        se (ini = fim) então
            ini := fim := 0
        senão
            ini := (ini mod n) + 1
        imprimeFila()

procedimento imprimeFila()
    i := ini
    imprimir(‘Estado da fila:’)
    imprimir(F[i])
    enquanto (i ≠ fim) faça
        i := (i mod n) + 1
        imprimir(F[i])
```