ESTRUTURAS DE DADOS - 2o. período de 2010

Gabarito da Primeira Avaliação à Distância

1. Para cada item abaixo, responda "certo" ou "errado", justificando cada resposta. (Valor de cada item: 0,5)

   a. Se as complexidades de melhor e pior caso de um algoritmo corresponderem, exatamente, ao mesmo número de passos efetuados, então podemos concluir que todas as entradas do algoritmo são idênticas entre si.

   Resposta: Falso. Podemos concluir apenas que o algoritmo executa sempre o mesmo número de passos (em função do tamanho da entrada), independente dos dados de entrada. Por exemplo, considere o problema de selecionar o maior elemento em um vetor não ordenado. O melhor caso e o pior caso do algoritmo são $\Theta(n)$ pois, para qualquer entrada, é necessário sempre percorrer todo o vetor. Mas há infinitas possibilidades de entradas distintas.

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

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

   Resposta: Falso. Se a complexidade de caso médio do algoritmo for $\Theta(n \log n)$ e a de melhor caso $\Theta(n)$, por exemplo, temos que o algoritmo não é $\Omega(n \log n)$, mas sim $\Omega(n)$.

   d. Se um algoritmo possui a menor complexidade, dentre todos os conhecidos que resolvem um certo problema P, então podemos concluir que este algoritmo é ótimo para P.

   Resposta: Falso. Só podemos concluir que um algoritmo é ótimo quando sua complexidade de pior caso é igual ao limite inferior do problema.

   e. A complexidade de pior caso de um algoritmo para um certo problema P é necessariamente maior do que um limite inferior para P.

   Resposta: Falso. A complexidade de pior caso de um algoritmo pode ser igual ao limite inferior. Neste caso, o algoritmo é ótimo.

   f. Dois algoritmos que sejam ótimos, para um certo problema, possuem necessariamente a mesma complexidade de pior caso e melhor caso, respectivamente.

   Resposta: Falso. Podemos afirmar apenas que os dois algoritmos ótimos têm a mesma complexidade de pior caso.

2. (Valor 1,5) Considere uma sequência de elementos $f_1, f_2, \ldots, f_n$, definida do seguinte modo: $f_1 = 1$, $f_2 = 3$, $f_j = 2.f_{j-1} + f_{j-2}$ para $j > 2$. Escrever dois algoritmos para determinar o elemento f_n da sequência, o primeiro recursivo e o segundo não recursivo. Qual dos dois é mais eficiente ? Justificar a resposta.

   Resposta:

   <u>Algoritmo recursivo:</u>

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

   Chamada externa: $seq(n)$

   Complexidade: É dada pela seguinte equação de recorrência:
       $T(1) = T(2) = 1$
       $T(j) = T(j-1) + T(j-2)$
   Resolvendo esta recorrência, verificamos que a complexidade deste algoritmo é $O(2^n)$.

   <u>Algoritmo iterativo:</u>

   $f[1] := 1;$
   $f[2] := 3;$
   para $j = 3 \ldots n$ faça
       $f[j] := 2 \times f[j-1] + f[j-2];$

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

   Logo, o algoritmo iterativo é mais eficiente.

3. (Valor 1,0) Escreva um algoritmo que inverte uma lista simplesmente encadeada. Exemplo: se a lista contém os elementos x, y, z, t, w, nesta ordem, então a lista resultante deverá conter os elementos w, t, z, y, z. Você deve utilizar uma pilha auxiliar para resolver este problema. Se a pilha for substituída por uma fila, qual será o resultado do algoritmo ?

   Resposta:

   $pont := ptLista \uparrow .prox$
   $topo := 0$ % inicialmente a pilha P está vazia
   enquanto $pont \neq \lambda$ faça
       $topo := topo + 1$
       $P[topo] := pont \uparrow .info$ % isere o dado em P
       $pont := pont \uparrow .prox$

ocupar(L) % cria a lista resultante L
$L \uparrow .prox := \lambda$
$ultimo := L$
enquanto $topo > 0$ faça % remove dados da pilha e os insere em L
    ocupar(pt)
    $pt \uparrow .info := P[topo]$
    $pt \uparrow .prox := \lambda$
    $ultimo \uparrow .prox := pt$
    $ultimo := pt$
    $topo := topo - 1$

Se substituirmos a pilha por uma fila no algoritmo, então a lista gerada será igual à lista original.

4. (Valor 2,0) As listas simplesmente encadeadas L_1 e L_2 contêm dados de clientes de uma empresa. A lista L_1 corresponde ao cadastro dos clientes. Para cada cliente, L_1 contém um registro que compreende o código identificador do cliente em questão (COD-L1) e um valor numérico que corresponde ao total da dívida deste cliente (DIVIDA) para com a empresa. A lista L_2 contém dados de pagamento de clientes da empresa: para cada cliente que efetuou o pagamento em questão, há um registro com o código do cliente (COD-L2) e o valor pago (PAGAMENTO). Em ambas as listas os registros estão ordenados por código. Escrever um algoritmo para atualizar o cadastro de clientes, lançando os respectivos pagamentos. Isto é, o algoritmo deve ler as duas listas, e comparar, sucessivamente, os códigos dos clientes em L_1 e L_2. Se esses códigos forem idênticos (COD-L1 = COD-L2), o algoritmo deve debitar o valor pago pelo cliente (em L_2), do total da dívida (em L_1), isto é, efetuando DIVIDA := DIVIDA - PAGAMENTO. Se não houve pagamento por parte de um cliente, então ele não aparece na lista L_2 e o algoritmo deve simplesmente repetir o valor de sua dívida em L_1. Note que somente a lista $L1$ é eventualmente alterada.

   Resposta:

   $pt1 := L_1 \uparrow .prox$
   $pt2 := L_2 \uparrow .prox$
   enquanto $pt2 \neq \lambda$ faça
       se $pt1 \uparrow$ .COD-L1< $pt2 \uparrow$ .COD-L2 então % cliente não efetuou pgto
           $pt1 := pt1 \uparrow .prox$
       senão
           $div := pt1 \uparrow$ .DIVIDA
           $div := div -$ ($pt2 \uparrow$ .PAGAMENTO)
           $pt1 \uparrow$ .DIVIDA := div
           $pt1 := pt1 \uparrow .prox$
           $pt2 := pt2 \uparrow .prox$

5. (Valor 2,5) Determinar a expressão da complexidade média de uma busca não ordenada de n chaves, n par, em que as probabilidades de busca das chaves 1 a $n/2$ são iguais entre si, sendo esse valor igual ao triplo da probabilidade de qualquer chave entre $n/2+1$ a n. Supor, ainda, que a probabilidade de a chave se encontrar na lista é igual a 30%.

   Resposta:

   Seja p a probabilidade de cada chave dentre 1 e $n/2$. Temos então que a probabilidade de cada chave dentre $(n/2+1)$ e n é $p/3$. Distribuindo a probabilidade de 30% pelas n chaves, temos que

$$(n/2)p + (n/2)(p/3) = 0,3$$

   Concluímos que $p = 0,45/n$. Logo, a expressão da complexidade média neste caso é dada pela expressão:

$$\begin{aligned} C.M. &= \frac{0,45}{n} \sum_{i=1}^{\frac{n}{2}} i \ + \ \frac{0,15}{n} \sum_{i=\frac{n}{2}+1}^{n} i \ + \ 0,7n \\ &= \frac{0,45}{n} \cdot \frac{(n/2)(1+n/2)}{2} + \frac{0,15}{n} \cdot \frac{(n/2)(n/2+1+n)}{2} \ + \ 0,7n \\ &= \frac{6,5n+1,2}{8} \end{aligned}$$