Estrutura de Dados - 2o. período de 2012

Gabarito da Primeira Avaliação à Distância

1. (1,0) O tamanho da entrada de um problema computacional está em função das variáveis n e m. Coloque as seguintes funções em ordem crescente de complexidade, sabendo que $m = O(n \log n)$: $O(m^2 + n)$, $O(m \log n)$, $O(mn^2)$, $O(m^2 n)$, $O(m^2 + \log n)$.

   Resposta: $O(m \log n) < O(m^2 + n) = O(m^2 + \log n) < O(mn^2) < O(m^2 n)$.

2. (1,5) Todas as afirmações abaixo são falsas. Explique o porquê.

   a. Se um limite inferior para um problema P é n^2, então nenhum algoritmo ótimo para P pode ter complexidade de caso médio $O(n \log n)$.

   Resposta: O limite inferior de um problema P diz respeito somente ao pior caso dos algoritmos que resolvem P. Nada podemos afirmar a respeito da complexidade média de um algoritmo com base em um limite inferior de P.

   b. Se a entrada de um problema P tem tamanho $O(m + n)$, então qualquer algoritmo para P tem complexidade de melhor caso $\Theta(m + n)$.

   Resposta: O tamanho da entrada de um problema P é um limite inferior natural para P. No entanto, o limite inferior se aplica somente ao pior caso dos algoritmos para P. Mesmo que exista um algoritmo ótimo para P com pior caso $\Theta(m + n)$, não podemos afirmar que o melhor caso deste algoritmo também será $\Theta(m + n)$. Neste caso, poderíamos apenas afirmar que o melhor caso do algoritmo é $O(m + n)$.

   c. Se todos os algoritmos para um problema P têm complexidades de pior caso $\Omega(n^2)$, e um certo algoritmo A para P tem complexidade $O(n^2)$, então A é ótimo.

   Resposta: Só podemos afirmar que um algoritmo é ótimo quando sua complexidade de pior caso for da mesma ordem de grandeza que o limite inferior do problema correspondente.

3. (1,5) Faça um algoritmo para uma cadeia de supermercados. Este algoritmo recebe como entrada uma lista L desordenada com N elementos, correspondendo às vendas efetuadas em um mês. Cada elemento $L[i]$ corresponde a uma venda e contém dois campos: $L[i].nome$ (contendo o nome do cliente para o qual a venda foi efetuada), e $L[i].valor$ (contendo o valor daquela venda). O algoritmo deve ordenar as vendas de acordo com as seguintes regras: na ordenação final, os nomes dos clientes crescem em ordem alfabética, e se um mesmo cliente efetuou várias compras, estas devem estar em ordem crescente de valor.

   Resposta:

```
para i := 1 até n − 1 faça
        menor := i
        para j := i + 1 até n faça
                se L[j].nome < L[menor].nome então
                        menor := j
                senão
                        se L[j].nome = L[menor].nome então
                                se L[j].valor < L[menor].valor então
                                        menor := j
        nome_aux := L[i].nome
        valor_aux := L[i].valor
        L[i].nome := L[menor].nome
        L[i].valor := L[menor].valor
        L[menor].nome := nome_aux
        L[menor].valor := valor_aux
```

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 vale metade da probabilidade de busca da chave $i+1$, $i = 1, ..., n-1$. Supor, ainda, que a probabilidade de a chave procurada se encontrar na lista é igual a 50%. Sua resposta deve vir em função de n. Faça uma breve discussão sobre a fórmula obtida para a complexidade média: por que ela faz sentido? (Reveja a transparência 11 da aula 7: ali você encontra um exemplo de como fazer uma discussão sobre a fórmula final obtida).

   Resposta:

   Seja p a probabilidade de busca da chave 1. Temos então que $p + 2p + 2^2p + \cdots + 2^{n-1}p = 0,5$. Logo,

$$\begin{aligned} p\sum_{i=1}^{n} 2^{i-1} &= \frac{1}{2} \\ p\,(2^n - 1) &= \frac{1}{2} \\ p &= \frac{1}{2(2^n - 1)} \end{aligned}$$

   A expressão da complexidade média neste caso é dada por:

$$\begin{aligned} C.M. &= \sum_{i=1}^{n}(t_i \,.\, p_i) + 0,5n \\ &= \sum_{i=1}^{n}\left(i\,.2^{i-1}p\right) + 0,5n \\ &= p.\sum_{i=1}^{n}\left(i\,.\,2^{i-1}\right) + 0,5n \\ &= \frac{1}{2(2^n-1)}\sum_{i=1}^{n}(i\,.\,2^{i-1}) + 0,5n \end{aligned}$$

   Para analisarmos a fórmula obtida para a complexidade média em relação a n, consideremos alguns valores de n e os valores correspondentes de complexidade média dados pela fórmula obtida: para $n = 10$, temos $C.M. \approx 9,5$; para $n = 50$, $C.M. \approx 49,5$ e para $n = 100$, $C.M. \approx 99,5$. O valor bem alto obtido para $C.M.$ (entre $n-1$ e n) é condizente com a distribuição das probabilidades. Além de a entrada correspondente ao fracasso (que custa n passos) ser de 50%, a probabilidade de sucesso da entrada E_n é praticamente igual à soma das probabilidades de todas as demais entradas de sucesso.

5. (1,5) Escreva um algoritmo RECURSIVO baseado no método de divisão e conquista para encontrar o maior e o menor elementos de uma lista com n elementos, baseado no seguinte princípio: divide-se a lista ao meio e encontra-se recursivamente o maior e menor elementos de cada uma das metades; a seguir, combina-se as duas soluções parciais na solução final. Qual é a complexidade do seu algoritmo?

   Resposta: Sejam V o vetor com n elementos, indexado de 1 a n, $MAX(a,b)$ e $MIN(a,b)$ funções que calculam, respectivamente, o maior e o menor valor entre $\{a,b\}$. A complexidade do algoritmo é $\Theta(n)$.

   Chamada externa: $maior\_menor(1,n)$

função $maior\_menor(i, j)$
    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
            $meio = (i + j)/2$
            $(maior1, menor1) = maior\_menor(i, meio)$
            $(maior2, menor2) = maior\_menor(meio + 1, j)$
            retornar $(MAX(maior1, maior2), MIN(menor1, menor2))$

6. (1,5) Sejam F_1 e F_2 duas filas, onde cada elemento da fila corresponde a uma pessoa, e o valor armazenado na fila é a idade da pessoa em questão. As filas estão em ordem, no sentido de que o primeiro elemento de F_1 (ou F_2) é a pessoa mais idosa desta fila, o segundo é a segunda pessoa mais idosa desta fila, etc. Apresentar um algoritmo que construa uma fila única por esta ordem de idades. Se F_1 contém n elementos e F_2 contém m elementos, o seu algoritmo deve ter complexidade $\Theta(n + m)$.

   Resposta: Seja F_3 a fila resultante, $|F_3| = n + m$.

$i := 1;$ // utilizado para percorrer F_1
$j := 1;$ // utilizado para percorrer F_2
$k := 1;$ // utilizado para percorrer F_3
enquanto $i \leq n$ e $j \leq m$ faça
    se $F_1[i] < F_2[j]$ então
        $F_3[k] := F_1[i]$
        $i := i + 1$
    senão
        $F_3[k] := F_2[j]$
        $j := j + 1$
    $k := k + 1$
enquanto $i \leq n$ faça // caso F_1 não tenha chegado ao fim
    $F_3[k] := F_1[i]$
    $i := i + 1$
    $k := k + 1$
enquanto $j \leq m$ faça // caso F_2 não tenha chegado ao fim
    $F_3[k] := F_2[j]$
    $j := j + 1$
    $k := k + 1$

7. (2,5) Faça um algoritmo não recursivo para resolver o problema da Torre de Hanói, utilizando pilhas. Este problema é mais difícil, então a pontuação dele vale um bônus na nota. É claro que se a soma dos pontos das suas questões ultrapassar 10,0, então a sua nota será apenas 10,0.

   Resposta: Considere uma pilha P, que armazena uma estrutura de dados da seguinte forma: (X, Y, Z, n), tal que X, Y, Z indicam pinos e n armazena um inteiro positivo. Sejam A, B, C os pinos de origem, trabalho e destino, respectivamente, e n o número de discos do problema.

   Algoritmo:

$topo := 1$
$P[topo] := (A, B, C, n)$
enquanto $topo > 0$ faça
    $(X, Y, Z, n) := P[topo]$
    $topo := topo - 1$
    se $n = 1$ então $mover(X, Z)$ % move o disco do topo de X para Z
    senão
        $topo := topo + 1$
        $P[topo] := (Y, X, Z, n - 1)$
        $topo := topo + 1$
        $P[topo] := (X, Y, Z, 1)$
        $topo := topo + 1$
        $P[topo] := (X, Z, Y, n - 1)$