Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP1 - Segundo Semestre de 2006

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. Considere o *Problema da Torre de Hanói* para 3 pinos e n discos.

   (1,5) Descreva um algoritmo *recursivo* que resolva este problema.

   Resposta:

   procedimento $hanoi(n, A, B, C)$

   se $n > 0$ então

   $hanoi(n-1, A, C, B)$

   mover o disco do topo de A para B

   $hanoi(n-1, C, B, A)$

   (1,5) Calcule o número $T(n)$ de movimentos de discos realizados pelo algoritmo. Sugestão: use uma recorrência.

   Resposta:

   Temos a seguinte fórmula de recorrência:

$$T(n) = \begin{cases} 0 & \text{se } n = 0 \\ 2\,T(n-1) + 1 & \text{se } n > 0 \end{cases}$$

   Resolvendo esta recorrência, temos:

$$\begin{array}{rcll} T(n) & = & 2\,T(n-1) + 1 & \\ & = & 2(2\,T(n-2) + 1) + 1 & = \quad 4\,T(n-2) + 2 + 1 \\ & = & 4(2\,T(n-3) + 1) + 2 + 1 & = \quad 8\,T(n-3) + 4 + 2 + 1 \\ & = & \ldots & \\ & = & 2^k\,T(n-k) + \displaystyle\sum_{i=0}^{k-1} 2^i & \end{array}$$

   Fazendo $n - k = 0$, temos $k = n$. Logo,

$$\begin{array}{rcl} T(n) & = & \underbrace{2^n\,T(0)}_{=0} + \displaystyle\sum_{i=0}^{n-1} 2^i \\ & = & 2^n - 1 \end{array}$$

2. Considere a técnica de *Busca Binária* numa lista ordenada com n elementos.

   (1,5) Descreva um algoritmo *não recursivo* de busca binária.

Resposta:

função *busca-bin*(x)

$inf := 1$

$sup := n$

busca-bin $:= 0$

enquanto $inf \leq sup$ faça

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

se $L[meio] = x$ então

busca-bin $:= meio$

$inf := sup + 1$

senão

se $L[meio] < x$ então

$inf := meio + 1$

senão $sup := meio - 1$

(1,5) Calcule a complexidade de pior caso do algoritmo acima.

Resposta:

O pior caso ocorre quando o elemento procurado é o último a ser encontrado, ou quando não está na lista. Nestes casos, a busca prossegue até a tabela se resumir a um único elemento. Na primeira iteração, a dimensão da tabela é n, e algumas operações são realizadas para situar o valor procurado. Na segunda, a dimensão se reduz a $\lfloor n/2 \rfloor$, e assim sucessivamente. Então, no pior caso, temos:

1ª iteração: a dimensão da tabela é n,

2ª iteração: a dimensão da tabela é $\lfloor n/2 \rfloor$,

3ª iteração: a dimensão da tabela é $\lfloor (\lfloor n/2 \rfloor)/2 \rfloor$,

...

mª iteração: a dimensão da tabela é 1.

Ou seja, o número de iterações é, no máximo, $1 + \lfloor log_2 n \rfloor$. O tempo consumido pelas operações em cada iteração é constante. Logo, a complexidade de pior caso da busca binária é $\Theta(log\, n)$.

3. Considere a estrutura de dados *Lista Duplamente Encadeada não Ordenada*.

   (1,5) Descreva o algoritmo de remoção de um elemento nesta estrutura.

   Resposta:

   Considere x o elemento a ser removido e *ptlista* um ponteiro para o nó-cabeça da lista.

   Algoritmo:

   $pont := ptlista \uparrow .post$

   enquanto $pont \uparrow .chave \neq x$ e $pont \neq ptlista$

   $pont := pont \uparrow .post$

   se $pont \uparrow .chave = x$ então

   $anterior := pont \uparrow .ant$

   $posterior := pont \uparrow .post$

   $anterior \uparrow .post := posterior$

   $posterior \uparrow .ant := anterior$

   $valor\text{-}recuperado := pont \uparrow .info$

   $desocupar(pont)$

   senão "elemento não se encontra na lista"

   (1,5) Calcule a complexidade de pior caso do algoritmo acima.

   Resposta:

   No pior caso, é necessário percorrer todos os nós da lista. Como o número de operações necessárias para a remoção do nó que contém x (caso exista) é constante, a complexidade de pior caso do algoritmo é $\Theta(n)$, onde n é o número de nós da lista.

4. (1,0) Determine quantos nós e quantas folhas possui uma árvore binária cheia com k níveis. Justifique sua resposta.

   Resposta:

   Em uma árvore binária cheia T, cada nível i possui 2^{i-1} nós. Logo, quando o número de níveis é k, o total de nós de T é:

   $$\text{nós}(T) = \sum_{i=1}^{k} 2^{i-1} = 2^k - 1$$

   As folhas de uma árvore cheia correspondem a todos os nós do último nível. Logo, T possui 2^{k-1} folhas.