Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
Gabarito da AP3 - Segundo semestre de 2010

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. (Valor 4,0): Responder se cada afirmativa abaixo é falsa ou verdadeira, *justificando a resposta*, em cada caso:

   (a) Uma árvore binária com n nós, que possua o número máximo de folhas é necessariamente cheia.

   Resposta: Falso. Para $n = 5$, por exemplo, a árvore binária possui no máximo 3 folhas, mas não é possível obtermos uma árvore cheia com este número de nós.

   (b) A complexidade de pior caso do algoritmo Heapsort (que utiliza um heap para efetuar a ordenação de uma lista de elementos) é expressa por uma função f que satisfaz $f = \Omega(n^2)$.

   Resposta: Falso. A complexidade de pior caso do algoritmo Heapsort é $\Theta(n \log n)$. A função $f = n \log n$ não é $\Omega(n^2)$ (mas sim $\Omega(n \log n)$).

   (c) Seja um algoritmo cuja entrada consiste de um número inteiro n. Se o número de passos deste algoritmo for igual a n, e cada passo for executado em tempo constante, pode-se afirmar que o algoritmo possui complexidade linear no tamanho da entrada.

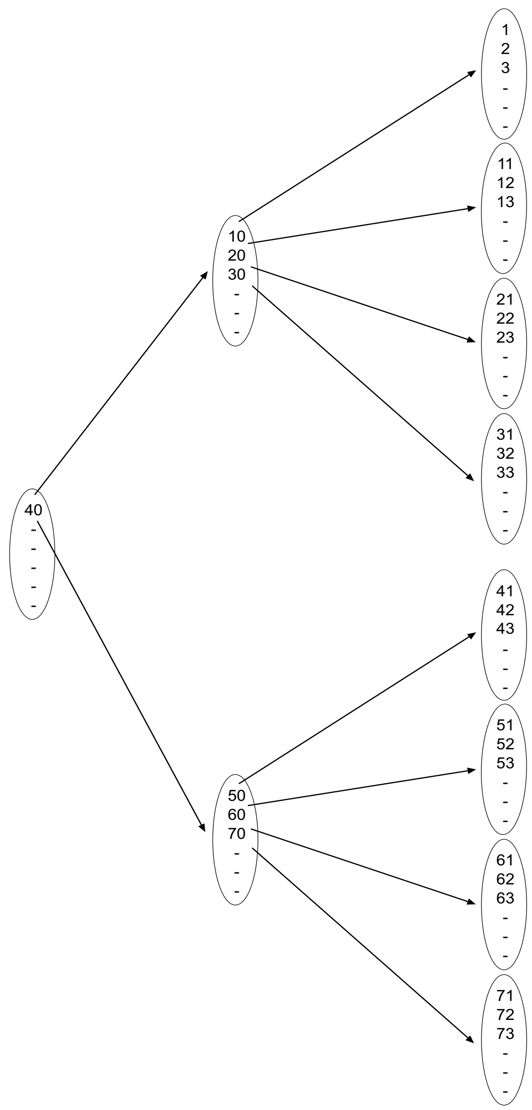
   Resposta: Verdadeiro. Neste caso, a complexidade do algoritmo é $O(n)$, sendo portanto linear no tamanho da entrada.

   (d) Sejam dois algoritmos distintos que resolvem um mesmo problema. Se ambos forem ótimos, pode-se concluir que ambos possuem complexidades idênticas de pior caso, e complexidades idênticas de melhor caso.

   Resposta: Falso. Sendo ambos os algoritmos ótimos, podemos apenas concluir que suas complexidades de pior caso são da mesma de grandeza. Nada podemos afirmar a respeito da complexidade de melhor caso destes algoritmos.

2. (Valor 1,0): Desenhar uma árvore B de ordem 3 e altura 3 que contenha um número mínimo de chaves. Indique os valores das chaves que escolheu, para cada nó da árvore. Além disso, desenhe cuidadosamente os ponteiros, de acordo com a definição.

   Resposta:



3. (Valor 2,0): Suponha um conjunto de n chaves x, formado pelos n primeiros múltiplos do número 7. Determinar o número de colisões que seriam obtidas mediante a aplicação das funções de dispersão h, seguintes. Justificar as respostas, em cada caso.

   (a) $h = x \bmod 7$

   Resposta: $n-1$ colisões. Neste caso, todas as chaves são sinônimas.

A primeira chave, 7, é colocada no endereço 0. A partir daí, as $n-1$ chaves restantes geram $n-1$ colisões.

(b) $h = x \bmod 14$

Resposta: $n-2$ colisões. A primeira chave é colocada no endereço 7, e a segunda chave, 14, no endereço 0. A partir da terceira, todas as chaves são colocadas alternadamente no endereço 7 e no endereço 0, gerando $n-2$ colisões.

(c) $h = x \bmod 5$

Resposta: $n-5$ colisões. A chave 7 é colocada no endereço 2, a chave 14 no endereço 4, a chave 21 no endereço 1, a chave 28 no endereço 3 e a chave 35 no endereço 0. A partir da sexta chave (42), todas geram colisões.

(d) $h = x$

Resposta: Nenhuma colisão. Neste caso, as n chaves ocupam n endereços distintos.

4. (Valor 3,0): Seja T uma árvore binária com n nós, representada por três vetores, E,D e R, onde para cada nó i, $1 \leq i \leq n$, $E(i)$, $D(i)$ e $R(i)$ informam o índice do filho esquerdo de i, o índice do filho direito de i, e o rótulo de i, respectivamente. Além disso, a variável *raiz* contém o índice da raiz de T. Pede-se:

   (a) Escrever um algoritmo para determinar um percurso em ordem simétrica para T.

   Resposta: Considere que os vetores são indexados de 1 a n. $E(i) = 0$ ($D(i) = 0$) indica que o nó i não possui filho esquerdo (direito). O algoritmo percorre a árvore T recursivamente em ordem simétrica.

   procedimento $simet(i)$
       se $E(i) \neq 0$ então
           $simet(E(i))$
       visita $R(i)$
       se $D(i) \neq 0$ então
           $simet(D(i))$

Chamada externa: $simet(raiz)$

(b) Escrever um algoritmo para determinar o número de folhas de T.

Resposta: Um nó i é folha quando não possui filhos ($E(i) = 0$ e $D(i) = 0$). O algoritmo percorre os vetores E e D, somando a quantidade de nós que atendem a essa condição.

$folhas = 0$
para $i = 1$ até n faça
    se $E(i) = 0$ e $D(i) = 0$ então
        $folhas = folhas + 1$
$imprimir\ folhas$

Observação: Os algoritmos devem ser acompanhados de um texto que explique a idéia utilizada.