Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP3 - Segundo semestre de 2008

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. Forneça as definições dos seguintes conceitos:

   (a) (1,0) Algoritmo ótimo para um problema.
   Resposta: Um algoritmo é ótimo quando sua complexidade de pior caso é igual ao limite inferior para o problema.

   (b) (1,0) Árvore binária de busca.
   Resposta: Seja $S = \{s_1, \cdots, s_n\}$ o conjunto de chaves satisfazendo $s_1 < \cdots < s_n$. Uma árvore binária de busca para S é uma árvore binária rotulada T, com as seguintes características:
   (i) T possui n nós. Cada nó v corresponde a uma chave distinta $s_j \in S$ e possui como rótulo o valor $r(v) = s_j$.
   (ii) Seja um nó v de T. Seja também v_1, pertencente à subárvore esquerda de v. Então, $r(v_1) < r(v)$. Analogamente, se v_2 pertence à subárvore direita de v, $r(v_2) > r(v)$.

   (c) (1,0) Complexidade de melhor caso de um algoritmo.
   Resposta: Sejam A um algoritmo, $E = \{E_1, \cdots, E_n\}$ o conjunto de todas as entradas possíveis de A e t_i o número de passos efetuados por A, quando a entrada for E_i. A complexidade de melhor caso de A é definida por $min_{E_i \in E} \{t_i \mid E_i \in E\}$.

2. (2,0) Escreva um algoritmo que execute a seguinte tarefa: Dada uma árvore binária T, com n nós, e dado um nó v de T, encontrar o nó que imediatamente sucede v, em um percurso em ordem simétrica.

   Resposta:

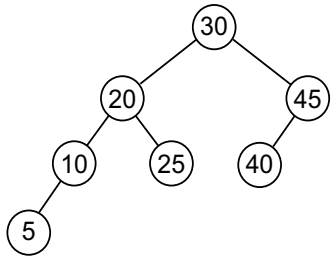
```
achou-v := falso                % indica se o nó v foi encontrado
achou-prox := falso             % indica se o sucessor de v foi encontrado
se ptraiz ≠ λ então
      simet(ptraiz)
se (achou-v = falso) então
      imprimir ("Nó v não encontrado.")
      senão
            se (achou-prox = falso) então
                  imprimir ("v é o último nó visitado em T pelo percurso
                              em ordem simétrica.")
```

procedimento simet(pt)
se ($pt \uparrow .esq \neq \lambda$) então
simet($pt \uparrow .esq$)
se $achou\text{-}v = falso$ então
se $pt \uparrow .chave = v$ então
$achou\text{-}v = verdadeiro$
senão
se $achou\text{-}prox = falso$ então
$achou\text{-}prox = verdadeiro$
imprimir ("Sucessor de v: ", $pt \uparrow .info$)
se $achou\text{-}prox = falso$ então
se ($pt \uparrow .dir \neq \lambda$) então
simet($pt \uparrow .dir$)

3. Desenhe a árvore T de altura 4 e que satisfaça às seguintes condições, em cada caso.

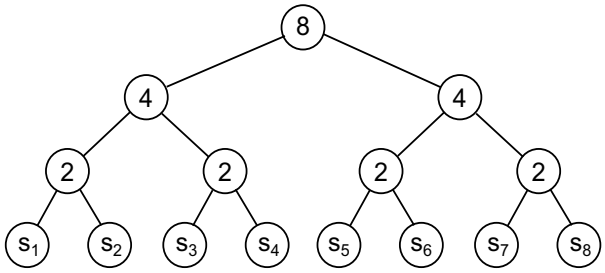
   (a) (1,5) T é uma árvore AVL e possui um número mínimo de nós. Não se esqueça de colocar os valores das chaves dentro de cada nó.

   Resposta:



   (b) (1,5) T é uma árvore de Huffman e possui um número máximo de nós. Não se esqueça de colocar os valores dos pesos dentro de cada nó.

   Resposta: Para $f_1 = f_2 = \cdots = f_8 = 1$, temos a seguinte árvore de Huffman:



4. (1,0) Para a seqüência abaixo, responda se ela corresponde ou não a um heap (lista de prioridade). Justifique.

   89 87 76 80 79 77 54 43 42 32 21

   Resposta: Não. Em um heap, temos que as chaves $s_1, \cdots, s_n$ satisfazem à propriedade $s_i \leq s_{\lfloor i/2 \rfloor}$, para $1 < i \leq n$. Na seqüência dada, temos $s_6 > s_3$.

5. (1,0) Dado um heap T com n nós, descrever o algoritmo para aumentar a prioridade de um dado nó i de T. Supor que as chaves de T sejam $s_1, \ldots, s_n$.

   Resposta: Ao aumentarmos a prioridade de um nó, este é realocado de forma correta "subindo-se" pelo caminho que leva à raiz da árvore através de trocas, até que sua prioridade seja menor ou igual à prioridade de seu pai. O algoritmo correspondente a esta operação é dado abaixo:

   procedimento subir(i)
   $j := \lfloor i/2 \rfloor$
   se $j \geq 1$ então
   se $\mathcal{T}[i].chave > \mathcal{T}[j].chave$ então
   $\mathcal{T}[i] \Leftrightarrow \mathcal{T}[j]$
   subir(j)