Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP3 - Primeiro 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. (2,5) Descreva um algoritmo que, tendo como entrada uma lista com n elementos distintos, encontra os dois maiores elementos da lista. Qual é a complexidade do seu algoritmo?

   Resposta:

   Seja L uma lista seqüencial com n elementos. Ao final da execução do algoritmo abaixo, a variável $maior1$ conterá o maior elemento de L e a variável $maior2$, o segundo maior.

   ```
   se n ≥ 2 então
       se L[1] > L[2]
           então maior1 := L[1]; maior2 := L[2]
           senão maior1 := L[2]; maior2 := L[1]
       para i = 3 ··· n faça
           se L[i] > maior1
               então maior2 := maior1; maior1 := L[i]
               senão
                   se L[i] > maior2 então
                       maior2 := L[i]
   ```

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

2. (2,5) Explique como efetuar a inclusão de um nó numa árvore AVL.

   Resposta:

   Após efetuar a inclusão de um nó q, percorre-se o caminho ascendente que vai de q até a raiz, e verifica-se se existe algum nó p que se tornou desregulado (isto é, tal que a diferença de altura entre as duas subárvores de p tornou-se maior que um.)

   Em caso afirmativo, podemos aplicar uma transformação apropriada para regulá-lo. Temos quatro casos, descritos a seguir. A notação para a compreensão dos casos é a seguinte: o nó u é o filho de p no caminho até q; $h_E(x)$ e $h_D(x)$ denotam as alturas das subárvores esquerda e direita do nó x, respectivamente. Para melhor visualização dos casos, reveja a aula 24.

Caso 1: $h_E(p) > h_D(p)$ e $h_E(u) > h_D(u)$.
Aplique a *rotação direita*.

Caso 2: $h_D(p) > h_E(p)$ e $h_D(u) > h_E(u)$.
Aplique a *rotação esquerda*.

Caso 3: $h_E(p) > h_D(p)$ e $h_D(u) > h_E(u)$.
Aplique a *rotação dupla direita*.

Caso 4: $h_D(p) > h_E(p)$ e $h_E(u) > h_D(u)$.
Aplique a *rotação dupla esquerda*.

3. (2,5) Desenhe e explique os passos intermediários do algoritmo de ordenação *Heapsort* ao seguinte vetor de entrada: 34, 23, 89, 12, 67, 58, 45.

   Resposta:

   Algoritmo Heapsort:

   arranjar(n)

   $m := n$

   enquanto $m > 1$ faça

   $T[1] \Leftrightarrow T[m]$

   $m := m - 1$

   descer($1, m$)

   Os passos do algoritmo são os seguintes:

Início:	$34, 23, 89, 12, 67, 58, 45$
Descer(3,7):	$34, 23, 89, 12, 67, 58, 45$
Descer(2,7):	$34, 67, 89, 12, 23, 58, 45$
Descer(1,7):	$89, 67, 58, 12, 23, 34, 45$
$T[1] \Leftrightarrow T[7]$:	$45, 67, 58, 12, 23, 34, 89$
Descer(1,6):	$67, 45, 58, 12, 23, 34, 89$
$T[1] \Leftrightarrow T[6]$:	$34, 45, 58, 12, 23, 67, 89$
Descer(1,5):	$58, 45, 34, 12, 23, 67, 89$
$T[1] \Leftrightarrow T[5]$:	$23, 45, 34, 12, 58, 67, 89$

Descer(1,4): $45, 23, 34, 12, 58, 67, 89$

$T[1] \Leftrightarrow T[4]$: $12, 23, 34, 45, 58, 67, 89$

Descer(1,3): $34, 23, 12, 45, 58, 67, 89$

$T[1] \Leftrightarrow T[3]$: $12, 23, 34, 45, 58, 67, 89$

Descer(1,2): $23, 12, 34, 45, 58, 67, 89$

$T[1] \Leftrightarrow T[2]$: $12, 23, 34, 45, 58, 67, 89 \rightarrow$ Ornenado!

4. (2,5) Explique como funciona uma tabela de dispersão em que as colisões são tratadas por encadeamento exterior.

   Resposta:

   O tratamento de colisões por encadeamento exterior consiste em manter m listas encadeadas, uma para cada possível endereço-base. Um campo para o encadeamento deve ser acrescentado a cada nó. Os nós correspondentes aos endereços-base são apenas nós-cabeça para essas listas. Para buscar uma chave x na tabela T, calcula-se $h(x) = x \bmod m$ e procura-se x na lista encadeada correspondente ao endereço-base $h(x)$. A inclusão de uma nova chave x é feita no final da lista encadeada correspondente ao endereço-base $h(x)$.