MC202 - Estruturas de Dados
Semestre 2001-2 - Turmas EF
Prova 2: 13/nov/2001

RA	Nome
Assinatura	Notas

A prova é individual e sem consulta.
Não são permitidos computadores ou calculadoras.
Não separe as folhas deste caderno de prova.
Não é permitido o uso de outro rascunho além destas folhas.
Escreva seu nome completo, e assine a tinta.
Valem apenas as respostas nos espaços indicados.

1. [2.5 pontos] Escreva um procedimento $Busca(A, x, \textbf{var}\ d)$ para localizar um elemento com chave x numa árvore de busca 2-3. Cada nó p da árvore tem contador de chaves $p\uparrow.m \in \{1, 2\}$, chaves $p\uparrow.ch[0..m-1]$, dados associados $p\uparrow.dado[0..m-1]$ e sub-árvores $p\uparrow.sub[0..m]$. Como de costume, o procedimento deve devolver um resultado booleano **true** se encontrou o elemento.

resposta

2. [2.5 pontos] Escreva procedimentos *Insere*(d, **var** H, **var** n) e *RemoveMin*(**var** H, **var** n, **var** d) para as operações básicas de um *heap* armazenado nos elementos $H[0..n-1]$ de um vetor H, com espaço para até *nmax* elementos. Suponha que cada elemento $H[j]$ é um apontador para um registro de tipo *Dado*, que contém, entre outros, um campo *prioridade* $H[j]\uparrow.pr$ (um número real). O *heap* está organizado de modo que $H[0]$ é o elemento de *menor* prioridade.

   No procedimento *Insere*, o parâmetro d é um apontador para um registro de tipo *Dado*, já alocado e preenchido. O procedimentro *RemoveMin* devolve em d o endereço do registro removido. Os dois procedimentos devem levar tempo $O(\log n)$, e devem verificar a validade das chamadas. Para esse fim, suponha definido um procedimento *Erro*(m) que imprime a cadeia m e pára o programa.

resposta

3. [2.5 pontos] Uma alternativa para a árvore de busca 2-3 é a de *árvore binária de busca auto-ajustável*. A estrutura é a tradicional árvore binária de busca; a diferença é que, cada vez que um nó p é percorrido numa busca, inserção, ou remoção, a estrutura é modificada de modo a trazer esse nó para mais perto da raiz. Com isso, os nós mais procurados tendem a ficar mais perto da raiz; e, mesmo que a árvore fique desbalanceada, assim que a parte mais funda começar a ser acessada mais que o resto, ela rapidamente se torna mais rasa.

   A modificação é chamada de *rotação*, e pode ser para a *esquerda* ou para a *direita*. Suponha um nó r com sub-árvore esquerda A e filho direito p, que por sua vez tem sub-árvores B e C. A rotação para a esquerda aplicada em r faz com que p passe a ser o pai, r o filho esquerdo de p, com sub-árvores A e B, e C a sub-árvore direita de p. A rotação para a direita é a operação inversa.

   (a) Escreva um procedimento para fazer a rotação à esquerda:

   resposta

   (b) Suponha definidos os procedimentos *RotParaEsquerda*(q) e *RotParaDireita*(q), que implementam as duas rotações (onde q é sempre o mais alto dos dois nós envolvidos). Escreva um procedimento de busca numa árvore binária que, ao mesmo tempo que devolve o elemento encontrado, aplica rotações necessárias para trazer esse elemento para a raiz.

   resposta

4. [2.5 pontos] Escreva um procedimento *Constroi*(**var** d, n), que, dado um vetor $d[0..n-1]$ de dados, em ordem crescente de chave, constrói uma árvore binária de busca com altura $O(\log n)$ para esses dados. Suponha que $d[j]$ é um inteiro, ao mesmo tempo chave e dado (ou seja, a informação a ser armazenada na estrutura é apenas um conjunto de chaves, sem dados adicionais associados às mesmas). O procedimento *Constroi* deve criar os nós da árvore, e devolvê-la como resultado.

resposta