ICMC-USP

Tipos e Estruturas de Dados - Pós-Graduação

Profa. Graça Nunes

Lista de Exercícios sobre Árvores

1. Num sistema de arquivos, um catálogo de todos os arquivos é organizado como uma árvore de busca binária. Cada nó denota um arquivo e especifica seu nome e, entre outras coisas, a data de seu último acesso, codificada como um inteiro. Escreva um programa que percorra a árvore e apague todos os arquivos cujos últimos acessos tenham sido anteriores a uma certa data. As chaves do catálogo são os nomes dos arquivos.

2. (a) Assuma que as chaves 1, 2, 3, ... são inseridas, nesta ordem, numa árvore-B de ordem 2. Quais chaves produzem divisão de páginas? Quais chaves fazem aumentar a altura da árvore? (b) Se as chaves são eliminadas na mesma ordem acima, quais chaves produzem concatenação de páginas e quais chaves fazem decrescer a altura da árvore?

3. (a) Escreva a declaração, em C ou Pascal, da estrutura que descreve um nó de uma árvore-B. (b) Escreva o algoritmo (estilo C ou Pascal) de uma função *localiza (chave, raiz)* que, dada uma chave *(chave)* que está na raiz de uma árvore-B, e dado um ponteiro para o nó raiz *(raiz)*, retorna a chave que é a sucessora imediata da chave dada na árvore, ou *nil*, caso a chave seja a maior delas. Por exemplo, para a árvore do exercício anterior, a sucessora de 90 é 100.

4. Escrever uma função recursiva que calcule a altura de uma árvore binária dada. A altura de uma árvore é igual ao máximo nível de seus nós.

5. Ache uma regra de composição para a seqüência de n números que, se aplicada ao procedimento de busca com inserção em uma árvore binária de busca, constrói uma árvore perfeitamente balanceada.

6. Numa árvore binária de busca a freqüência de acesso de cada elemento é medida empiricamente, atribuindo-se a cada nó um número de acessos. A cada certos intervalos de tempo, a organização da árvore é atualizada, percorrendo-se a árvore e gerando-se uma nova árvore usando o procedimento de busca com inserção, inserindo as chaves em ordem decrescente de sua freqüência de acesso. Escreva um programa que realize esta reorganização.

7. Compute o número de nós ancestrais em uma árvore binária para um dado nó a um nível *K*.

8. Liste 3 aplicações fora da ciência da computação onde a estrutura de árvores é útil. Para cada aplicação, desenhe a árvore típica, rotulando cada nó em termo das variáveis relevantes para a área de aplicação.

9. Faça um algoritmo para remover um nó de uma árvore binária.

10. Escreva o procedimento cópia:

```
procedure cópia (t: tree; var c: tree);
```

(* cria uma árvore, c, que é a mesma de t *)

11. Escreva o procedimento espelho:

```
procedure espelho(t: tree; var e: tree);
```

(* cria uma árvore, e, que é a imagem no espelho de t *)

12. Considere uma árvore binária que representa uma expressão aritmética, definida pela estrutura:

```
type  ponto-no= ^ no;
      no =  record
            esq, dir, pont-no;
            case  boolean of
                  fase: (info-op: char);
                  breve: (infor-num: real)
            end;
```

Assuma que os únicos operadores usados são os operadores binários *, +, -, /. A função a seguir pretende calcular, recursivamente, o valor da expressão representada por uma árvore binária. Desenvolva os trechos (a) e (b).

```
function avalia (raiz: pont-no) : real;
  var result1, result2 : real;
  begin
   if (a) raiz é operando then
      avalia := raiz^.info-num
   else /* Raiz é operador */
      (b) cálculo de avalia neste ponto
  end;
```

13. Escreva uma função recursiva **remove_arvore** que dado um ponteiro para a raiz da árvore, remova recursivamente todos os nós dessa árvore.

14. Supondo que cada nó em uma árvore binária possua as informações do pai, filho da esquerda e do filho da direita, escreva procedimentos que recebam como parâmetro um nó *A* da árvore e:

(a) Faça uma rotação LL em *A*;

(b) Faça uma rotação RR em *A*;

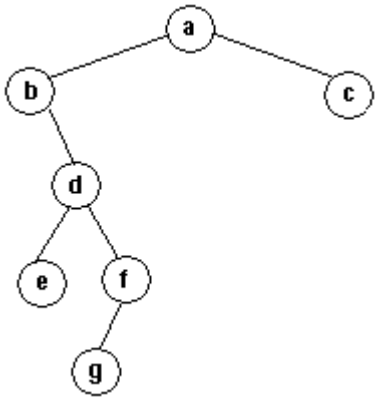
(c) Faça uma rotação RL em *A*;

(d) Faça uma rotação LR em *A*;

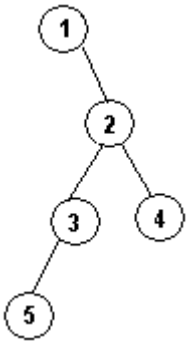
(Suponha que as rotações possam ser efetuadas)

15. Dada as seguintes árvores binárias abaixo, indique os passos para torná-las uma árvore binária balanceada (AVL).

(a)



(b)



16. Escreva um procedimento que verifique se uma árvore é AVL.

17. Insira os números 35, 39, 51, 20, 13, 28, 22, 32, 25, 33 (nesta ordem) em uma árvore AVL.

18. Defina árvore AVL.

19. Dê um exemplo de inserção de um elemento em uma árvore AVL que cause rearranjo da estrutura da árvore.

20. Dê um exemplo de remoção de um elemento de uma árvore AVL que cause rearranjo da estrutura da árvore.

21. Por que nos damos ao trabalho de procurar trabalhar com árvores binárias balanceadas? Justifique.

22. Descreva uma representação de árvores (não necessariamente binárias).

(a) Escreva um método `ImprimePorNivel()`

(b) Escreva um método `ElementoMaiorGrau()` que retorna o elemento de maior grau.

23. Escreva um programa que recebe a lista de nós de uma árvore quando percorrida em in-ordem e pré-ordem e devolve a árvore.

24. Considere uma árvore de busca binária com os elementos 1,2, ...,1000. As seqüências a seguir poderiam ser a seqüência de nós visitados quando estamos procurando o nó 374?

(a) 423 – 118 – 256 – 450 – 354 – 375 – 374

(b) 423 – 231 – 394 – 300 – 350 – 382 – 374