Estrutura de Dados - 2o. período de 2005

Segunda Avaliação à Distância

A questão 5 vale dois pontos, e as demais valem um ponto cada.

1. Uma palavra é um *palíndromo* se a sequência de letras que a forma é a mesma, quer seja lida da esquerda para a direita ou da direita para a esquerda (exemplo: *radar*). Escrever um algoritmo eficiente para reconhecer se uma dada palavra é um palíndromo. Escolher a estrutura de dados conveniente para representar a palavra.

   Resposta:

   Podemos armazenar a palavra de entrada numa *lista duplamente encadeada.*

   Nesse caso, o algoritmo que reconhece se uma palavra é palíndromo fica assim (o nó-cabeça é apontado por *ptlista*):

   $p1 := ptlista \uparrow .post$ % ponteiro que anda para a frente

   $p2 := ptlista \uparrow .ant$ % ponteiro que anda para trás

   se $p1 = \lambda$

   então "palavra vazia"

   senão repita

   se $p1 \uparrow .info \neq p2 \uparrow .info$

   então "não é palíndromo – abandonar o *repita*!! "

   senão $p1 := p1 \uparrow .post$; $p2 := p2 \uparrow .ant$

   até que $p1 = p2$ ou $p1 \uparrow .post = p2$

2. Apresentar o algoritmo de alteração do campo $info$ de uma lista circular encadeada com nó-cabeça.

   Resposta:

   Devemos inicialmente localizar o nó com chave x na lista, e depois trocar a informação corresponde a este nó. Assim fica o algoritmo (o nó-cabeça é apontado por *ptlista*):

   $ptlista \uparrow .chave := x$ % sentinela colocado no nó-cabeça

   $pont := ptlista \uparrow .prox$ % ponteiro para percorrer a lista

   enquanto $pont \uparrow .chave \neq x$ faça $pont := pont \uparrow .prox$

   se $pont \neq ptlista$

   então $pont \uparrow .info := novo - valor$ % chave localizada

   senão "chave não localizada" % paramos na sentinela

3. Prove ou dê contra-exemplo: Uma árvore binária pode ser construída, de forma única, a partir dos seus percursos em pré-ordem e ordem simétrica.

   Resposta:

A afirmação é verdadeira. Suponha que o percurso em pré-ordem é dado pela sequência de nós $x_1 x_2 \ldots x_n$, e que o percurso em ordem simétrica é dado pela sequência de nós $y_1 y_2 \ldots y_n$. Sabemos que a raiz da árvore é o nó x_1. Suponha que $x_1 = y_j$ no percurso em ordem simétrica. Sabemos portanto que os nós $y_1 \ldots y_{j-1}$ pertencem à sub-árvore esquerda T_E da raiz, e que os nós $y_{j+1} \ldots y_n$ pertencem à sub-árvore T_D direita da raiz. Aplicando recursivamente o mesmo raciocínio a T_E e T_D, conseguimos reconstruir a árvore binária original de forma única.

4. Prove ou dê contra-exemplo: Uma árvore binária pode ser construída, de forma única, a partir dos seus percursos em pré-ordem e pós-ordem.

   Resposta:

   A afirmação é falsa. Considere duas árvores binárias T_1 e T_2, onde cada uma delas contém apenas dois nós A e B de forma que:

   - em T_1, B é filho esquerdo de A;

   - em T_2, B é filho direito de A.

   Para ambas as árvores acima, o percurso em pré-ordem é AB e o percurso em pós-ordem é BA. No entanto, elas são distintas.

5. Determinar a árvore binária de custo mínimo relativa às seguintes frequências: $f_1 = 1, f_2 = 3, f_3 = 2, f'_0 = 2, f'_1 = 2, f'_2 = 1, f'_3 = 0$.

   Resposta:

   As matrizes do algoritmo de cálculo da árvore ótima são:

   Matriz dos custos $c[i, j]$:

0	5	14	19
-	0	6	11
-	-	0	3
-	-	-	-

   Matriz dos valores $F[i, j]$:

2	5	9	11
-	2	6	8
-	-	1	3
-	-	-	0

   Matriz dos valores minimizantes k:

-	1	2	2
-	-	2	2
-	-	-	3
-	-	-	-

   Da última matriz acima, segue que a árvore binária de custo ótimo tem raiz s_2, e portanto filho esquerdo s_1 e filho direito s_3.

6. Prove ou dê contra-exemplo: Toda árvore AVL é rubro-negra.

Resposta:

A afirmação é verdadeira. (Fica implícito, é claro, que a árvore AVL já está munida de nós externos.)

Para provar esta afirmação, precisamos do seguinte fato: "Duas árvores rubro-negras T_1 e T_2 cujas alturas diferem em no máximo uma unidade podem ser coloridas de modo que suas raízes recebam valores de posto idênticos".

(Lembre-se de que $posto(v)$ é igual ao número de nós negros encontrados em *qualquer* caminho de v a uma folha.)

O fato acima pode ser provado do seguinte modo. Colora dois caminhos de comprimento máximo c_1 e c_2 pertencentes a T_1 e T_2, respectivamente, alternando nós negros e rubros, de baixo para cima. Suponha sem perda de generalidade que o comprimento de c_1 seja menor ou igual que o comprimento de c_2. Se a raiz v de T_1 tornou-se rubra e a raiz w de T_2 tornou-se negra, troque a cor de v. Em qualquer outro caso, mantenha a coloração obtida. Dessa forma, garantiremos que em ambos os caminhos c_1 e c_2 o número k de nós negros é o mesmo. A seguir, colora os demais nós de T_1 e T_2 adequadamente, de modo que todos os caminhos ascendentes da forma nó externo$\rightarrow$raiz tenham k nós negros, tanto em T_1 como em T_2. No final, teremos $posto(v) = posto(w)$.

Provado este fato, fica fácil provar a afirmação, também por indução. Incialmente, pode-se mostrar por verificação exaustiva que todas as árvores AVL com altura no máximo 4 são rubro-negras.

Seja agora T uma árvore AVL com altura $h \geq 5$. Removendo-se a raiz de T, obtemos duas árvores AVL T_1 e T_2 com alturas menores ou iguais a 4. Pela hipótese de indução, T_1 e T_2 são rubro-negras. Além disso, pela definição de árvore AVL, suas alturas diferem em no máximo uma unidade. Logo, pelo fato enunciado no início, T_1 e T_2 podem ser coloridas de modo que suas raízes recebam postos idênticos. Basta agora colorir a raiz de T como negra para concluir que T é realmente uma árvore rubro-negra.

7. Determinar os valores dos números mínimo e máximo de nós (chaves) que uma árvore B de ordem d pode armazenar.

   Resposta:

   Nos cálculos abaixo, a altura h da árvore B satisfaz $h \geq 1$.

   O número mínimo de chaves é atingido quando a árvore tem o menor número possível de páginas, que é (veja a pg. 162 do livro-texto):

$$1 + \frac{2}{d}[(d+1)^{h-1} - 1].$$

   Além disso, cada página neste caso tem d chaves, exceto a raiz, que possui uma única chave. Logo, o número mínimo de chaves é dado pela expressão.

$$n_{\min} = 1 + d\left(\frac{2[(d+1)^{h-1} - 1]}{d}\right) = 2(d+1)^{h-1} - 1.$$

Já o número máximo de chaves é atingido quando a árvore tem o maior número possível de páginas, que é (veja a pg. 162 do livro-texto):

$$\frac{(2d+1)^h - 1}{2d}.$$

Além disso, cada página neste caso tem $2d$ chaves. Logo, o número máximo de chaves é dado pela expressão.

$$n_{\max} = 2d\left[\frac{(2d+1)^h - 1}{2d}\right] = (2d+1)^h - 1.$$

8. Determine o heap obtido pela aplicação do algoritmo de construção às seguintes prioridades: $18, 25, 41, 34, 14, 10, 52, 50, 48$.

   Resposta:

   Os passos do algoritmo de complexidade $O(n)$ são os seguintes:

   Início: $18, 25, 41, 34, 14, 10, 52, 50, 48$

   Descer 34: $18, 25, 41, 50, 14, 10, 52, 34, 48$

   Descer 41: $18, 25, 52, 50, 14, 10, 41, 34, 48$

   Descer 25: $18, 50, 52, 48, 14, 10, 41, 34, 25$

   Descer 18: $52, 50, 41, 48, 14, 10, 18, 34, 25 \rightarrow$ heap final!

9. Descrever um algoritmo de inserção em uma tabela de dispersão por encadeamento aberto, supondo a não-existência de remoções.

   Resposta:

   Suponha que x é a chave a ser incluída na tabela $T[0 \ldots m-1]$, de tamanho m.

   O algoritmo é:

   $end := h(x)$ mod m

   $pont := T[end]$ % ponteiros para percorrer a lista

   $achou := falso$

   enquanto $pont \neq \lambda$ faça

       $ant := pont$

       se $pont \uparrow .chave = x$

           então $achou := verdadeiro$; $pont := \lambda$ % x já está na lista

           senão $pont := pont \uparrow .prox$

se $achou = falso$ então

$ocupar(pt)$

$pt \uparrow .chave := x$

$pt \uparrow .prox := \lambda$

se $T[end] = \lambda$

então $T[end] := pt$

senão $ant \uparrow .prox := pt$