Aluno(a):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Matrícula:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| 1a) | 1.0 |  |
| 2a) | 2.5 |  |
| 3a) | 2.5 |  |
| 4a) | 2.5 |  |
| 5a) | 1,5 |  |
|  | 10.0 |  |

1. A prova é individual e sem consulta.
2. A interpretação faz parte da questão. Em caso de dúvida escreva a dúvida e a sua interpretação na resposta.
3. O tempo de prova é 1:45 h.
4. As respostas devem seguir as questões. Caso precise de rascunho use o verso da folha.
5. A prova pode ser feita a lápis.
6. (1.0 pontos)
   1. (0,5 pontos) No pior caso, qual a altura de uma árvore de busca binária com *n* nós. Explique e ilustre a sua resposta.
7. (0,5 pontos) No melhor caso, qual a altura de uma árvore de busca binária com *n* nós. Qual a relação entre *h* e *n* para atingir o melhor caso. Explique e ilustre a sua resposta.

Resp:

1. No pior caso, a árvore é degenerada. Terá altura h=n-1.
2. No melhor caso, n = 2h+1 -1, ou seja, h = log2(n+1)-1
3. (2.5 pontos) O *índice* de uma árvore AVL é definido como o maior número de nós que devem ser visitados para localizar uma chave na árvore. Sem utilizar estruturas auxiliares (pilhas, filas, etc...), implemente de forma não recursiva uma função que calcule o índice de uma AVL, visitando o menor número possível de nós. A função deve ter o seguinte protótipo:

int indice(AvlNode\* node)**;**

Utilize a estrutura:

typedef struct \_avl\_node AvlNode;

struct \_avl\_node {

void\* info;

int bf; /\* balance factor = h\_right - h\_left \*/

AvlNode\* parent;

AvlNode\* left;

AvlNode\* right;

};

Resp:

O índice nada mais é do que a altura da árvore.

int indice(AvlNode\* node) {

int i=0;

while(node!=NULL) {

node = (node->bf>0) ? node->right : node->left;

i++;

}

return i;

}

1. a) (1.5 pontos) Em cada um dos casos abaixo indique qual ou quais rotações devem ser feitas para rebalancear a árvore AVL após a sequencia de operações indicada. Explique a sua resposta, usando os fatores de balanceamento.

(i) (0,5 ponto) Operação: **Inserção da chave 5**.

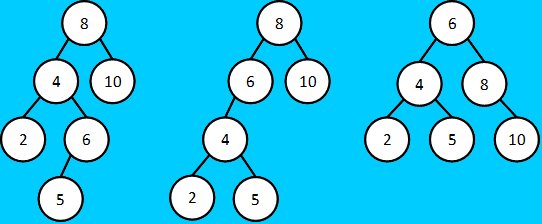
(ii) (0,5 ponto) Operação: **Remoção da chave 6**.

(iii) (0,5 ponto) Operações: **Remoção da chave 6**, seguida de **Remoção da chave 11**.

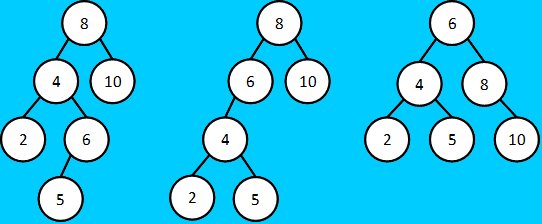
**b) (1.0 ponto) Qual estratégia você sugeriria para diminuir o número de rotações em uma árvore AVL, à luz dos exemplos acima? Explique sua resposta.**

Resp:

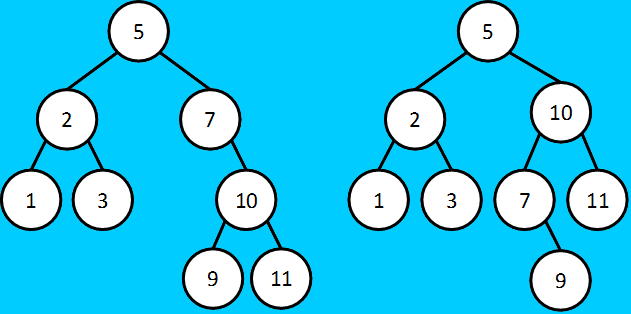
(i) A raiz possui fator de balanceamento -2 e o nó rotulado com 4 tem fator de balanceamento +1. Logo, deve-se fazer uma rotação à esquerda na sub-árvore à esquerda:



seguida de uma rotação à direita:



ii) O nó rotulado com 7 possui fator de balanceamento +2 e o nó rotulado com 10 tem fator de balanceamento 0. Logo, deve-se fazer uma rotação à esquerda no nó rotulado com 7.



iii) A árvore estará balanceada após as duas operações. Não há nada a fazer

b) Adie o rebalancemento o máximo possível. No exemplo (iii) acima, a árvore final estava balanceada.

1. (2.5 pontos) Remova a chave 70 da árvore B abaixo. Explique e ilustre como a remoção foi processada. Use e explique decisões ótimas em cada passo.

10

20

60

90

100

120

**70**

50

80

Resp:

10

20

60

90

100

120

50

80

Hipótese 1: A árvore sofrerá remoções sucessivas a partir da remoção da chave 70.

A melhor maneira de prosseguir consiste em concatenar o primeiro e o segundo nós de nível 1. Desta forma, diminuiremos o número de nós. O espaço do novo nó deverá conter a chave do pai, 50, para mantermos a estrutura íntegra.

10

20

50

60

90

100

120

80

Hipótese 2: A árvore sofrerá inserções e remoções sucessivas a partir da remoção da chave 70.

A melhor maneira de prosseguir consiste em redistribuir as chaves entre o segundo e o terceiros nós de nível 1. Os nós reorganizados conterão ainda espaço para 1 inserção.

10

20

60

100

120

80

50

90

1. (1.5 pontos) Considere uma família de árvores definida de forma semelhante a árvores B, exceto que:

* Os nós possuem tamanho fixo em 136 bytes
* As chaves possuem tamanho variável, entre 4 bytes e 12 bytes
* Os ponteiros ocupam 4 bytes
* Em cada nó, há um campo a mais, de 4 bytes, indicando o número de chaves que o nó efetivamente armazena
  1. (0.5 pontos) Qual será o maior número de chaves, todas de menor tamanho, que é possível acomodar em uma árvore de altura 2? Explique cuidadosamente sua resposta.
  2. (1.0 ponto) Qual será o número mínimo de chaves, todas de maior tamanho, que uma árvore de altura 2 armazena? Explique cuidadosamente sua resposta.

Resp:

* 1. Seja m o número de chaves que podem ser acomodadas em um nó, assumindo que todas as chaves tem o menor tamanho. Cada chave é acompanhada por um ponteiro, exceto a primeira, que é acompanhada por 2 ponteiros. Há ainda 4 bytes adicionais para indicar o número de chaves. Logo, m será:

136 = 4+4+(4+4)\*m

ou seja,

m = (136-8)/8 = 128/8 = 16 chaves

Logo, cada nó terá 17 filhos.

Portanto, como a árvore possui altura 2, o número máximo de chaves de menor tamanho será:

CM = 16 + 17\*16 + 17\*17\*16 = 4.912

b) O número mínimo de chaves de maior tamanho é obtido da seguinte forma.

Cada nó terá no máximo p chaves, onde p é dado por:

136 = 4+4+(12+4)\*p

p = (136-8)/16 = 128/16 = 8

O número mínimo de chaves de maior tamanho será então:

* A raiz possui no mínimo 1 chave e 2 filhos
* Cada nó no primeiro nível terá no mínimo p/2=4 chaves de maior tamanho e p/2+1=5 filhos
* Cada nó no terceiro e último nível terá no mínimo p/2=4 chaves de maior tamanho

Logo, o número mínimo de chaves de maior tamanho em uma árvore de altura 2 será:

Cm = 1 + 2\*4 + 5\*2\*4 = 49