Lista de Exercícios árvores AVL

1) Inserir em AVL as chaves:

   30, 50, 70, 60, 80, 40, 20, 10, 5, 35, 55, 1

2) Com a árvore construída no exercício 1, retirar da árvore as maiores chaves até ela ficar vazia.

3) Dar exemplo de inserção de chaves que provoque:
   a. uma rotação simples à direita
   b. uma rotação dupla à esquerda
   c. duas rotações simples à esquerda
   d. duas rotações duplas à direita

4) Implementar os algoritmos de busca e inclusão em árvore AVL.

5) Descrever em pseudocódigo o algoritmo de exclusão em árvore AVL.

6) Responda se Falso ou Verdadeiro, justificando.
   a. Na inserção de um nó na árvore AVL, basta uma regulagem para a árvore fique balanceada.
   b. Na remoção de um nó na árvore AVL, basta uma regulagem para a árvore fique balanceada.

7) Provar ou dar contraexemplo:
   Toda árvore estritamente binária é balanceada.
   Toda árvore binária de busca é balanceada.
   Toda árvore AVL é balanceada.

8) Qual o número máximo e mínimo de nós em uma árvore AVL de altura h?

9) Seja uma árvore AVL ***T***. Considere, no algoritmo de inserção apresentado no curso, que a inserção de um nó ***q*** em ***T*** tornou ***T*** desregulada. Seja ***p*** o nó desregulado mais próximo das folhas.

   a) Qual o valor exato de |he(p) - hd(p)| ? Por que não pode ser nem mais nem menos?

   b) Supondo hd(p) > he(p) então existe um filho direito u de p. Por que necessariamente temos |hd(u) - he(u)| = 1? Por que não pode ser 2 ou 0?

   c) Quando hd(p) > he(p) existem dois subcasos a serem considerados:
   he(u) = hd(u) + 1 ou hd(u) = he(u) + 1.

   Para cada um dos subcasos acima, explique a transformação que regula p (apresente também um esquema). Mostre que todos os nós originalmente em T ficam regulados (através da análise das alturas das sub-árvores).

   d) Por que a regulagem de p (nó desregulado mais próximo das folhas) regula toda a árvore?

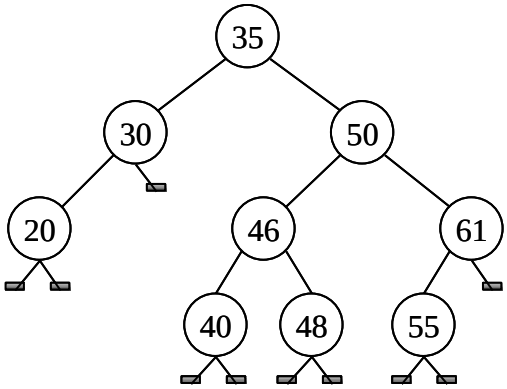
10) Desenhe o passo a passo de como ficam as árvores AVL após as operações indicadas:
Inserção das chaves 8, 4, 10, 2, 6, 5, 7
Remoção da chave 6

11) Partindo de uma árvore AVL vazia, realize a inserção da seguinte sequência de chaves:

99, 44, 71, 80, 74, 63, 59, 120, 98, 150.

Redesenhe a árvore após cada inserção. Indique para cada rotação feita, o nome da rotação e o nó desregulado. Desenhe também as árvores resultantes da exclusão dos nós 59 e 63.

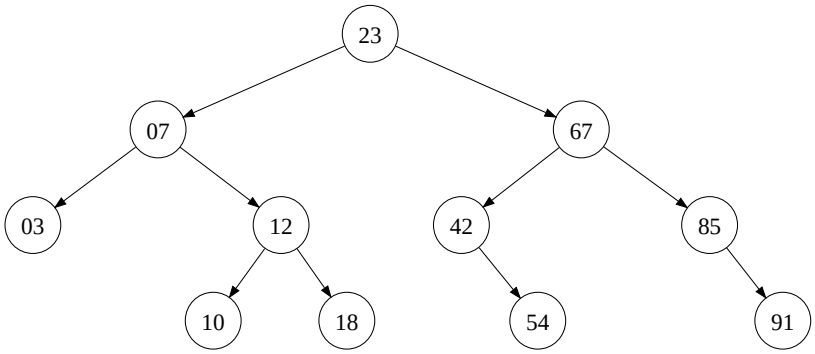
12) Considere a árvore AVL a seguir:



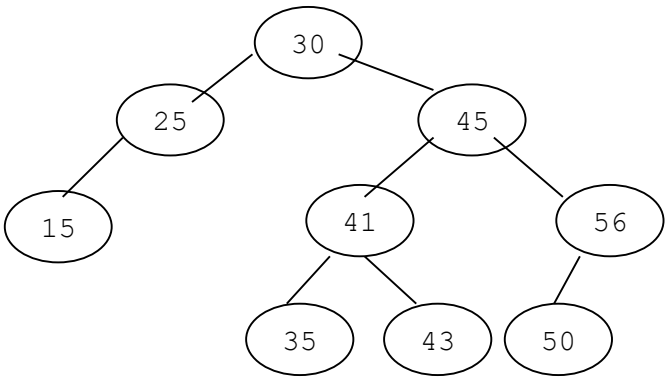
Realize, na árvore dada, a inserção das chaves 65, 70, 38, 44, 49 e 42, atualizando o fator de equilíbrio dos nós a cada inserção. Quando necessário, indique o nó desregulado e a rotação utilizada para regulá-lo. Redesenhe a árvore a cada passo.

13) Desenhe as árvores resultantes das operações de remoção na seguinte árvore seguindo os algoritmos de árvore balanceada AVL. Execute as operações sempre na árvore original.

a) remover 10
b) remover 18
c) remover 42
d) remover 76



14) Considere a árvore AVL a seguir:



a) Insira na árvore acima, as seguintes chaves 49, 60, 65, e em seguida a remova as chaves 45 e 41. Use na remoção sempre o antecessor (ao invés do sucessor, apresentado em aula). Mostre todas as rotações e a estrutura da árvore após cada operação.

b) Seja q um nó recém inserido e p o seu ancestral mais próximo que se tornou desregulado. Quais os possíveis valores para o fator de balanço de p após a inserção? O fator de balanço de p é suficiente para concluir se a inserção foi à esquerda ou à direita de p? Porque?

15) Escreva em C uma função para determinar se uma árvore é AVL. Use a estrutura:

```
typedef struct no t_no;
struct no {
        int chave;
   int bal; /* fator de balanço: hdir - hesq */
   t_no *esq, *dir;
};
t_no *raiz;
```

A função deve receber como parâmetro o endereço do nó raiz da árvore e retornar o valor TRUE se for AVL e FALSE caso contrário.

16) Uma árvore binária de busca AVL é implementada através de:

typedef struct avl Avl;
struct avl {int info; int fb; Abb* pai; Abb* esq; Abb* dir; };

Escreva uma função que determina o número de nós menores que um dado valor, visitando o menor número de nós possível.

int menor_que(Avl* raiz, int valor);

17) Sem utilizar estruturas auxiliares (pilhas, filas, etc), implemente de forma não recursiva a função que calcula a altura de um nó de uma AVL, com o seguinte protótipo:

int node_height(AvlNode* node);

visitando o menor número de nós possível. 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;
};
```

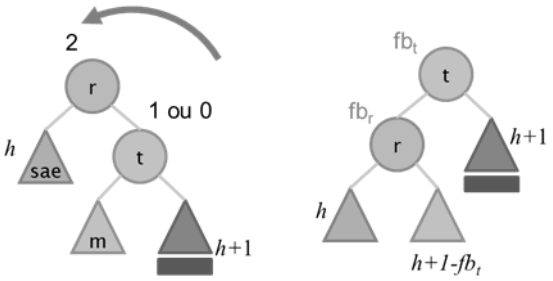
18) Considere a seguinte estrutura de dados de árvore AVL abaixo:

```
typedef struct _avl Avl;
struct _avl {
   int chave;
   int fb; /*fator de balanceamento hd-he*/
   Avl *pai;
   Avl *esq;
   Avl *dir;
};
```

Desenvolva uma função de rotação simples a esquerda num nó, corrigindo todos os fatores de balanceamento e os links de pai, esq, dir de todos os nós envolvidos menos a raiz. Protótipo da função:

```
Avl* rotacao_esquerda(Avl* r);
```

Dica:



19) Desenhar a árvore rubro-negra obtida pela sequência de inserção das chaves: 19, 18, 16, 15, 17, 2, 6, nesta ordem.

20) Mostre a menor árvore binária que não seja rubro-negra.

21) Provar ou dar contraexemplo:
   a) Toda árvore rubro-negra é AVL.
   b) Toda árvore AVL é rubro-negra.