UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
Bacharelado em Ciência da Computação
/ Engenharia da Computação
INF 01203 – Estruturas de Dados

Exercícios
Árvores AVL

01 – Insira em uma árvore AVL, itens com as chaves apresentadas nos itens a seguir (na ordem em que aparecem). Desenhe a árvore resultante da inserção, sendo que uma nova árvore deve ser desenhada quando houver uma rotação. Indique qual a rotação que foi executada.

a) 30, 40, 24, 58, 48, 26, 11, 13, 14
b) 20, 15, 25, 10, 30, 24, 17, 12, 5, 3
c) 40, 30, 50, 45, 55, 52
d) 20, 15, 25, 12, 17, 24, 30, 10, 14, 13
e) 20, 15, 25, 12, 17, 30, 26

Resolver utilizando a applet fornecida em aula

02 – Quantas árvores binárias de pesquisa (**ABP**) diferentes podem armazenar as chaves {1,2, 3, 4}?

14

03 – Um certo professor Amongus afirma que a ordem pela qual um conjunto fixo de elementos é inserido em uma árvore AVL não interessa – sempre resulta na mesma árvore. Apresente um pequeno exemplo que prove que ele está errado.

A ordem dos valores implica em arvores resultantes diferentes.

04 – Analise uma árvore **T** que armazena 100.000 itens. Qual a situação para o pior e melhor casos em relação à altura de **T** das seguintes árvores:

- **T** é uma árvore binária de pesquisa (ABP)
- **T** é uma árvore AVL.

ABP - pior - altura 100.000
ABP - *melhor* - altura *17*
AVL - pior - altura *18*
AVL - *melhor* - altura *17*

05 – Para qual ordem de inserção dos elementos o caminhamento pré-fixado à esquerda se iguala ao caminhamento central à esquerda?

todos os valores ordenados (ordem crescente)

06 – Analise os dois trechos de códigos a seguir, identificando o que faz cada um deles (os dois trechos de código utilizam a estrutura de dados apresentada abaixo e a função que calcula a altura de nodo na árvore).

```
struct TNodoA{
char info;
int FB;
struct TNodoA *esq;
struct TNodoA *dir;
};

typedef struct TNodoA *pNodoA;
```

```
int Altura (pNodoA a)
{
    int Alt_Esq, Alt_Dir;
    if (a == NULL)
      return 0;
    else
    {
       Alt_Esq = Altura (a->esq);
       Alt_Dir = Altura (a->dir);
       if (Alt_Esq > Alt_Dir)
         return (1 + Alt_Esq);
       else
         return (1 + Alt_Dir);
     }
}
```

(A)

```
int funcaoA(pNodoA a)
{
    return (Altura(a->esq) - Altura(a->dir));
}
```

calcula o fator de um nodo

(B)

```
int funcaoB(pNodoA a)
{
  int alt_esq, alt_dir;

  if (a!=NULL)
  {
     alt_esq = Altura(a->esq);
     alt_dir = Altura(a->dir);
     return ( (alt_esq - alt_dir <2) && (alt_dir - alt_esq <2) &&
(funcaoB (a->esq)) && (funcaoB (a->dir)) );
  }
  else
  return 1;
}
```

Verifica se uma árvore é AVL