

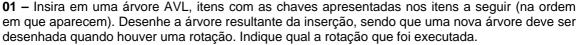
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL INSTITUTO DE INFORMÁTICA

Bacharelado em Ciência da Computação / Engenharia da Computação

INF 01203 - Estruturas de Dados

Professores: Renata de Matos Galante

Exercícios Árvores AVL



```
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

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

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.

04 – Analise uma árvore **T** que armazena 100.000 itens. Quais são o <u>pior</u> e o <u>melhor</u> casos em relação à altura de **T** das seguintes árvores:

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

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

06 – Analise os trechos de código apresentados a seguir, identificando a rotação que eles realizam e corrigindo-os quando for o caso.

```
procedure rotacaoA(var pt: PNodo);
                                           procedure rotacaoB(var pt: PNodo);
   var ptu: PNodo;
                                               var ptu: PNodo;
begin
                                          begin
   ptu:= pt^.esq;
                                              ptu:= pt^.dir;
   pt^.esq:= ptu^.dir;
                                              pt^.dir:= ptu^.esq;
   ptu^.dir:= pt;
                                              ptu^.esq:= pt;
   pt^.fator:= 0;
                                              pt^.fator:= 0;
   pt:= ptu;
                                              pt:= ptu;
end;
                                           end;
```

Informática UFRGS

```
procedure rotacaoD (var pt: Pnodo);
procedure rotacaoC (var pt: PNodo);
   var ptu, ptv: Pnodo ;
                                              var ptu, ptv: PNodo;
begin
                                           begin
   ptu:= pt^.dir;
                                              ptu:= pt^.esq;
   ptv:= ptu^.esq;
                                              ptv:= ptu^.dir;
   ptu^.esq:= ptv^.dir;
                                              ptu^.dir:= ptv^.esq;
   ptv^.dir:= ptu;
                                              ptv^.esq:= ptu;
   pt^.dir:= ptv^.esq;
                                              pt^.esq:= ptv^.dir;
                                              ptv^.dir:= pt;
   ptv^.esq:= pt;
   if ptv^.fator = 1
                                              if ptv^*.fator = -1
      then pt^.fator:= -1
                                                  then pt^.fator:= 1
     else pt^.fator:= 0;
                                                 else pt^.fator:= 0;
                                              if ptv^.fator = 1
   if ptv^{-}.fator = -1
      then ptu^.fator:= 1;
                                                 then ptu^.fator:= -1;
      else ptu^.fator:= 0;
                                                 else ptu^.fator:= 0;
   pt:= ptv;
                                              pt:= ptv;
end;
                                           end;
```

3 – A seguir, é apresenta o procedimento que faz o balanceamento e atualização dos fatores para os dois casos de inserção (à direita e à esquerda) apresentados em aula.

```
Proc InsereAVL (var a:Arvore; x:Info; ok:lógico);
{ Insere nodo em uma árvore AVL, onde A representa a raiz da árvore,
 x, a chave a ser inserida e h a altura da árvore }
se a = nil
então início
        InicioNo(a,x); {aloca o novo nodo e insere a informação}
        ok:=verdadeiro
fim;
senão
        se x = a^. Info então pare {se o nodo já existe então não será inserido
novamente }
        se x < a^.chave então início {o nodo será inserido à esquerda}
                 InsereAVL(a^.esq,x,ok);
                 se ok então
                          caso a^.bal seja
                            -1: a^.bal := 0; ok := falso;
                             0: a^.bal := 1;
                             1: Caso1(a,ok); fim;
                                                    {faz o balanceamento}
        senão início {o nodo será inserido à direita}
                 InsereAVL(a^.dir,x,ok);
                 se ok então
                         caso a^.bal seja
                           1: a^.bal := 0; ok := falso;
                          0: a^.bal := -1
                          -1: Caso2(a,ok); fim;
                                                     {faz o balanceamento}
fim;
```

Para os casos 01 e 02 identifique nos pontos de interrogação quais são as rotações que devem ser feitas. Justifique sua resposta.

```
Proc Casol (var a:Arvore, ok:lógico);
                                           Proc Caso2 (var a: Arvore, ok:lógico);
var ptu: Arvore;
                                           var ptu: Arvore;
início
                                            início
         ptu = a^.esq;
                                                    ptu = a^.dir;
         se ptu^.bal = 1
                                                    se ptu^.bal = -1
         então rotacao???;
                                                    então rotacao_???;
         senão rotacao???;
                                                    senão rotacao_???;
         a^*.bal := 0;
                                                    a^*.bal := 0;
        ok := falso;
                                                    ok := falso;
fim;
                                            fim;
```