O algoritmo de inserção

Para implementar as árvores AVL, nós acrescentamos o campo `.alt` ao registro `RegInt`

```
registro RegInt
 {
    int       val;
    int       alt;
    RegInt   *esq, *dir, *pai;
 }
```

e mantemos as alturas de todas as subárvores pré-calculadas.

Daí, quando um novo elemento é inserido na árvore, nós recalculamos o campo `.alt` de todos os elementos no caminho da raiz até o novo elemento — (*percorrendo o caminho de baixo para cima*)

E sempre que nós encontramos um desbalanceamento nós aplicamos as rotações adequadas para consertar a situação.

As funções que implementam as rotações serão apresentadas no apêndice dessa nota — (*porque elas são semelhantes às que vimos na aula passada*)

Abaixo nós temos o algoritmo de inserção

```
Algoritmo  Inserção-AVL (k,p)
 {
     aux  <--  Novo (RegInt)
     aux->val,alt  <--  k, 1
 aux->esq,dir,pai  <--  Nulo, Nulo, Nulo

     Se ( p = Nulo )    Retorna (aux)

     // 1a ETAPA: descida na árvore e inserção

     q1  <--  p;    q2  <--  Nulo

     Enquanto ( q1 != Nulo )
      {
          q2  <--  q1
          Se ( k < q1->val )   q1  <--  q1->esq
          Senão                q1  <--  q1->dir
      }
     Se ( k < q2->val )   q2->esq  <--  aux
     Senão                q2->dir  <--  aux

     // 2a ETAPA: subida pelo caminho e ajustes

     x  <--  q2

     Repita ( p/ sempre )
      {
          Se ( x->esq->alt  >  x->dir->alt + 1 )
           {
               y  <--  x->esq

               Se ( y->esq->alt  >  y->dir->alt )
                  x  <--  Rdir(x)
               Senão
                {
                  y  <--  Resq(y);   x  <--  Rdir(x)
                }
           }
          Senão Se ( x->dir->alt  >  x->esq->alt + 1 )
           {
               ( ... análogo ao caso anterior ... )
           }
          Senão
             x->alt  <--  Max { x->esq->alt , x->dir->alt }  +  1

          Se ( x->pai = Nulo)   Retorna (x)
          Senão                 x  <--  x->pai
}    }
```