AULA e13
Algoritmos e Estruturas de Dados I

Árvores AVL – Inserção e Exclusão (códigos)

Prof. Luciano Antonio Digiampietri

Rotação L – Código Completo

```
PONT rotacaoL(PONT p){
      PONT u, v;
      u = p->esq;
      if(u->bal == -1) {    // LL
            p->esq = u->dir;
            u->dir = p;
            p->bal = 0;
            u->bal = 0;
            return u;
      } else if (u->bal == 1) {  // LR
            v = u->dir;
            u->dir = v->esq;
            v->esq = u;
            p->esq = v->dir;
            v->dir = p;
            if(v->bal == -1) p->bal = 1;
            else p->bal = 0;
            if(v->bal == 1) u->bal = -1;
            else u->bal = 0;
            v->bal = 0;
            return v;
      }else{    // caso necessario apenas para a exclusao (u->bal == 0)
            p->esq = u->dir;
            u->dir = p;
            // p->bal = -1;
            u->bal = 1;
            return u;
      }
}
```

LL
(u->bal = -1)

LR
(u->bal = 1)

L0
(u->bal = 0)

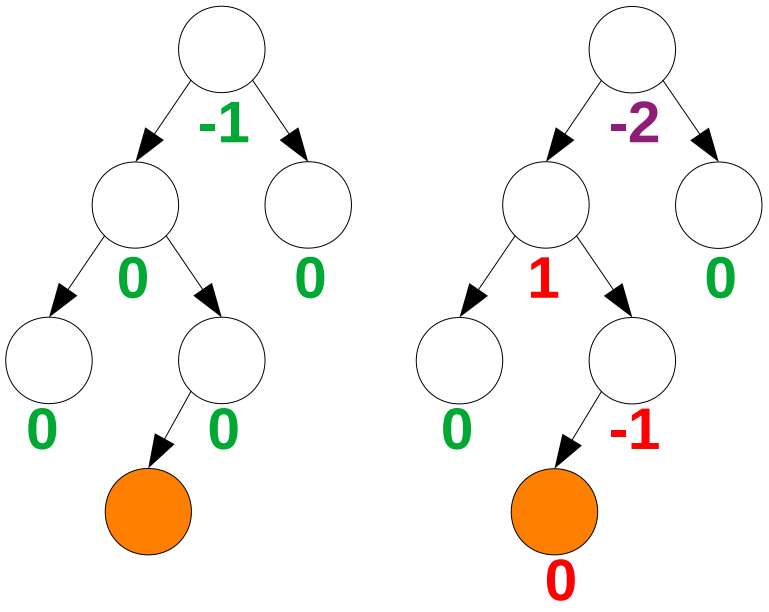
Inserção – Código Completo

```
void inserirAVL(PONT* pp, TIPOCHAVE ch, bool* alterou){
      PONT p = *pp;
      if(!p){
            *pp = criarNovoNo(ch);
            *alterou = true;
      } else {
            if(ch == p->chave) *alterou = false;
            else if(ch < p->chave) {
                  inserirAVL(&(p->esq), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case 1 : p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = -1;
                              break;
                              case -1: *pp = rotacaoL(p);
                              *alterou = false;
                              break;
                        }
            } else {
                  inserirAVL(&(p->dir), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case -1: p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = 1;
                              break;
                              case 1 : *pp = rotacaoR(p);
                              *alterou = false;
                              break;
                        }
            }
      }
}
```

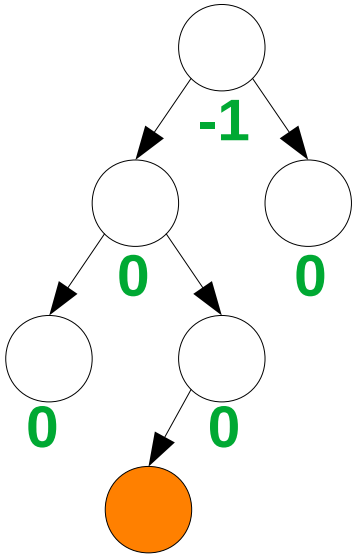
```
void inserirAVL(PONT* pp, TIPOCHAVE ch, bool* alterou){
      PONT p = *pp;
      if(!p){
            *pp = criarNovoNo(ch);
            *alterou = true;
      } else {
            if(ch == p->chave) *alterou = false;
            else if(ch < p->chave) {
                  inserirAVL(&(p->esq), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case 1 : p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = -1;
                              break;
                              case -1: *pp = rotacaoL(p);
                              *alterou = false;
                              break;
                        }
            }
```

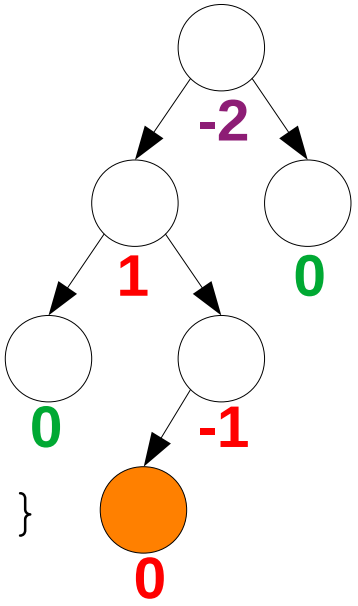
```
        } else {
            inserirAVL(&(p->dir), ch, alterou);
            if(*alterou)
                switch (p->bal) {
                    case -1: p->bal = 0;
                    *alterou = false;
                    break;
                    case 0 : p->bal = 1;
                    break;
                    case 1 : *pp = rotacaoR(p);
                    *alterou = false;
                    break;
                }
        }
    }
}
```

```
void inserirAVL(PONT* pp, TIPOCHAVE ch, bool* alterou){
      PONT p = *pp;
      if(!p){
            *pp = criarNovoNo(ch);
            *alterou = true;
      } else {
            if(ch == p->chave) *alterou = false;
            else if(ch < p->chave) {
                  inserirAVL(&(p->esq), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case 1 : p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = -1;
                              break;
                              case -1: *pp = rotacaoL(p);
                              *alterou = false;
                              break;
                        }
            }
```



```
} else {
        inserirAVL(&(p->dir), ch, alterou);
        if(*alterou)
                switch (p->bal) {
                        case -1: p->bal = 0;
                        *alterou = false;
                        break;
                        case 0 : p->bal = 1;
                        break;
                        case 1 : *pp = rotacaoR(p);
                        *alterou = false;
                        break;
                }
        }
}
```





Exclusão – Código Completo

```
bool excluirAVL(PONT* raiz, TIPOCHAVE ch, bool* alterou){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) { // tem zero ou um filho
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      *alterou = true;
      return true;
    }
    else {   // tem dois filhos
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  bool res;
  if (ch > atual->chave) {
    res = excluirAVL(&(atual->dir), ch, alterou);
    if (*alterou){
       switch (atual->bal){
          case 1: atual->bal = 0;
          return true;
          case 0: atual->bal = -1;
          *alterou = false;
          return true;
          case -1: *raiz = rotacaoL(atual);
          if (atual->bal != 0) *alterou = false;
          return true;
       }
    }
  }else{
    res = excluirAVL(&(atual->esq), ch, alterou);
    if (*alterou){
       switch (atual->bal){
          case -1: atual->bal = 0;
          return true;
          case 0: atual->bal = 1;
          *alterou = false;
          return true;
          case 1: *raiz = rotacaoR(atual);
          if (atual->bal != 0) *alterou = false;
          return true;
       }
    }
  }
  return res;
}
```

```
bool excluirAVL(PONT* raiz, TIPOCHAVE ch, bool* alterou){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) { // tem zero ou um filho
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      *alterou = true;
      return true;
    }
    else {   // tem dois filhos
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
```

```
bool res;
if (ch > atual->chave) {
   res = excluirAVL(&(atual->dir), ch, alterou);
   if (*alterou){
      switch (atual->bal){
         case 1: atual->bal = 0;
         return true;
         case 0: atual->bal = -1;
         *alterou = false;
         return true;
         case -1: *raiz = rotacaoL(atual);
         if (atual->bal != 0) *alterou = false;
         return true;
      }
   }
}
```

```
else{
    res = excluirAVL(&(atual->esq), ch, alterou);
    if (*alterou){
        switch (atual->bal){
            case -1: atual->bal = 0;
            return true;
            case 0: atual->bal = 1;
            *alterou = false;
            return true;
            case 1: *raiz = rotacaoR(atual);
            if (atual->bal != 0) *alterou = false;
            return true;
        }
    }
}
```

AULA e13
Algoritmos e Estruturas de Dados I

Árvores AVL – Inserção e Exclusão (códigos)

Prof. Luciano Antonio Digiampietri