Algoritmos para multilistas

Como já vimos, a busca na multilista consiste em fazer a busca binária em cada uma das suas partes

A coisa fica assim

```
função  busca-Multista ( k, L[1..n], m)
 {
     Para  j <-- 0  Até  piso ( m / raiz(n) )
      {
           i  <--  j * raiz(n) + 1                // início da (j+1)-ésima sublista

           f  <--  Min { m, (j+1) * raiz(n) }     // fim da (j+1)-ésima sublista

        resp  <--  busca-Binária ( k, L[i,f] )

        Se ( resp = SIM )  Retorna (SIM)
      }
     Retorna (NÃO)
 }
```

Como já vimos, a inserção sempre ocorre na última parte da lista.

E a coisa fica assim

```
função  inserção-Multista ( k, L[1..n], m)
 {
    i  <--  piso ( m / raiz(n) ) * raiz(n)  +  1    // início da última sublista

    m ++;   L[m]  <--  k                            // inserção no fim

    j  <--  m
    Enquanto ( j > i )                              // deslocamento
     {
        Se ( L[j] < L[j-1] )
         {
               aux  <--  L[j]
             L[j]  <--  L[j-1]
          L[j-1]  <--  aux
         }
        Senão  Break
 }   }
```

Finalmente, a remoção envolve primeiro uma busca pelo elemento a ser removido.

Daí, nós movemos o último elemento da última sublista para a sua posição, e depois fazemos o deslocamento.

A coisa fica assim

```
função  remoção-Multista ( k, L[1..n], m)
 {
    pos  <--  busca-Multilista ( k, L[1..n], m )

    Se ( pos = 0 )  Retorna                        // o elemento não está lá


    L[pos]  <--  L[m];    m --

    Se ( L[m] < k )                                    // deslocamento para trás
     {
        i  <--  piso ( pos / raiz(n) ) * raiz(n) + 1    // início da sublista de k

        j  <--  pos

        Enquanto ( j > i )
         {
            Se ( L[j] < L[j-1] )
             {
                   aux  <--  L[j]
                 L[j]  <--  L[j-1]
              L[j-1]  <--  aux
             }
            Senão  Break
     }   }

    Se ( L[m] > k )                                    // deslocamento para frente
     {
        f  <--  piso ( pos / raiz(n) + 1 ) * raiz(n)    // fim da sublista de k

        j  <--  pos

        Enquanto ( j < f )
         {
            Se ( L[j] < L[j+1] )
             {
                   aux  <--  L[j]
                 L[j]  <--  L[j+1]
              L[j+1]  <--  aux
             }
            Senão  Break
     }   }
 }
```