"A sorte favorece a mente preparada" (Louis Pasteur).

Exclusão em Árvores B

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Excluir

O algoritmo de exclusão é um pouco mais complicado que o de inserção.

Excluir

Incluir pode causar um overflow no tamanho máximo do nodo.

Precisamos fazer um split para criar mais espaço.

Uma exclusão pode causar um underflow no tamanho mínimo.

Precisamos transplantar chaves ou, no pior caso, fundir (*merge*) nodos.

Excluir

O algoritmo é uma modificação do *buscarArvoreB*.

Buscamos a chave a ser excluída.

A cada nível que descemos, precisamos garantir que o nodo tenha pelo menos *t* chaves.

Excluir

Atenção: a carga/descarga dos nodos será ocultada nos algoritmos.

```
função excluirArvoreB(T,x,k)
entrada: árvore B T, nodo x de onde começar a busca, e
chave a ser excluída k.
saída: a chave k é excluída da árvore B, caso exista.

i = 1
enquanto i ≤ x.n e k > x.chave[i]
     i = i+1
se i ≤ x.n e k == x.chave[i] //chave encontrada
     se x é folha
          remova k de x
          retorne
     senão
          //...
senão
     se x é folha
          retorne //chave não encontrada
retorne excluirArvoreB(x.filhos[i],k)
```

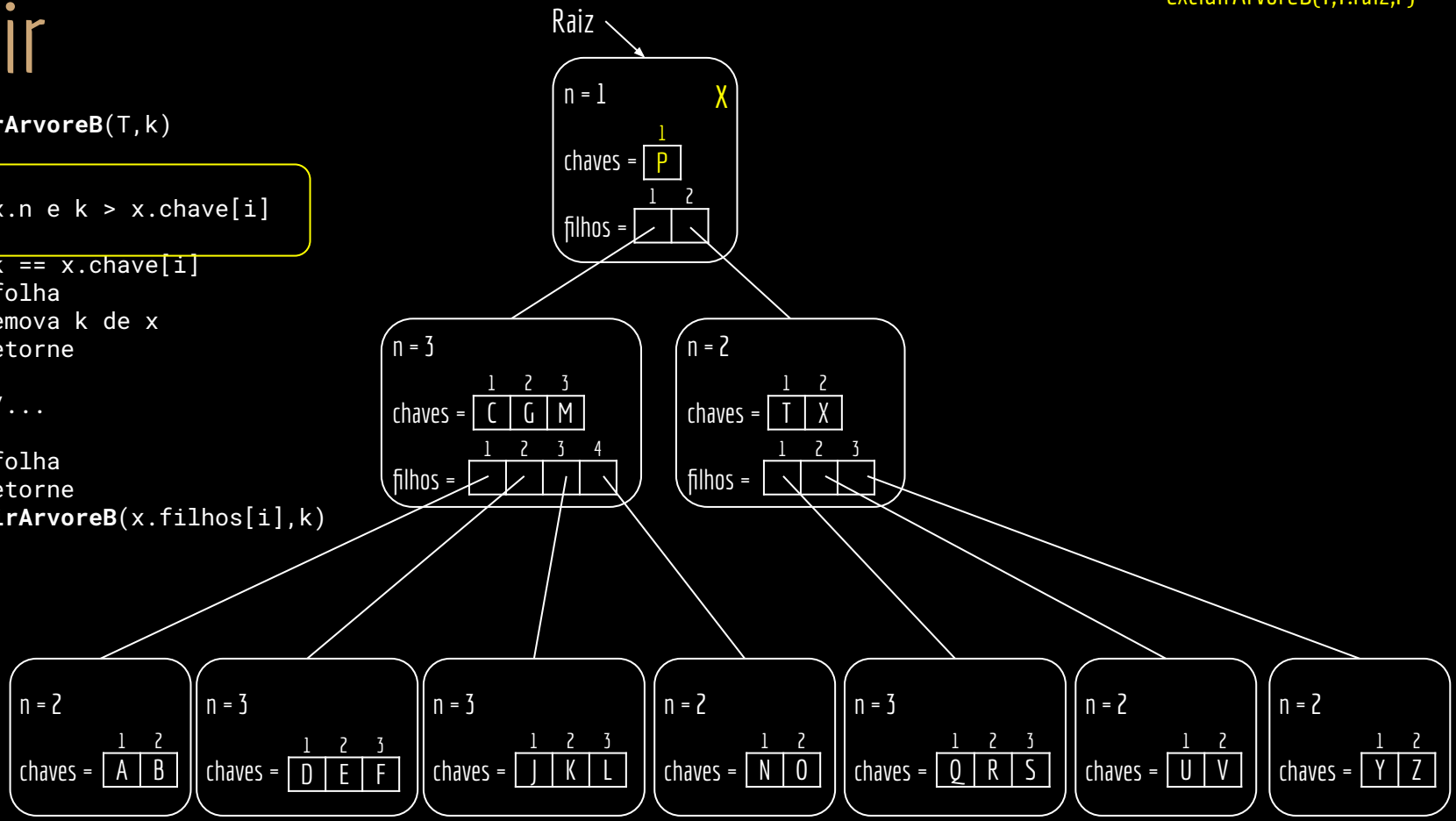
Caso 1, a chave está em uma folha, ou não está na árvore.

Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

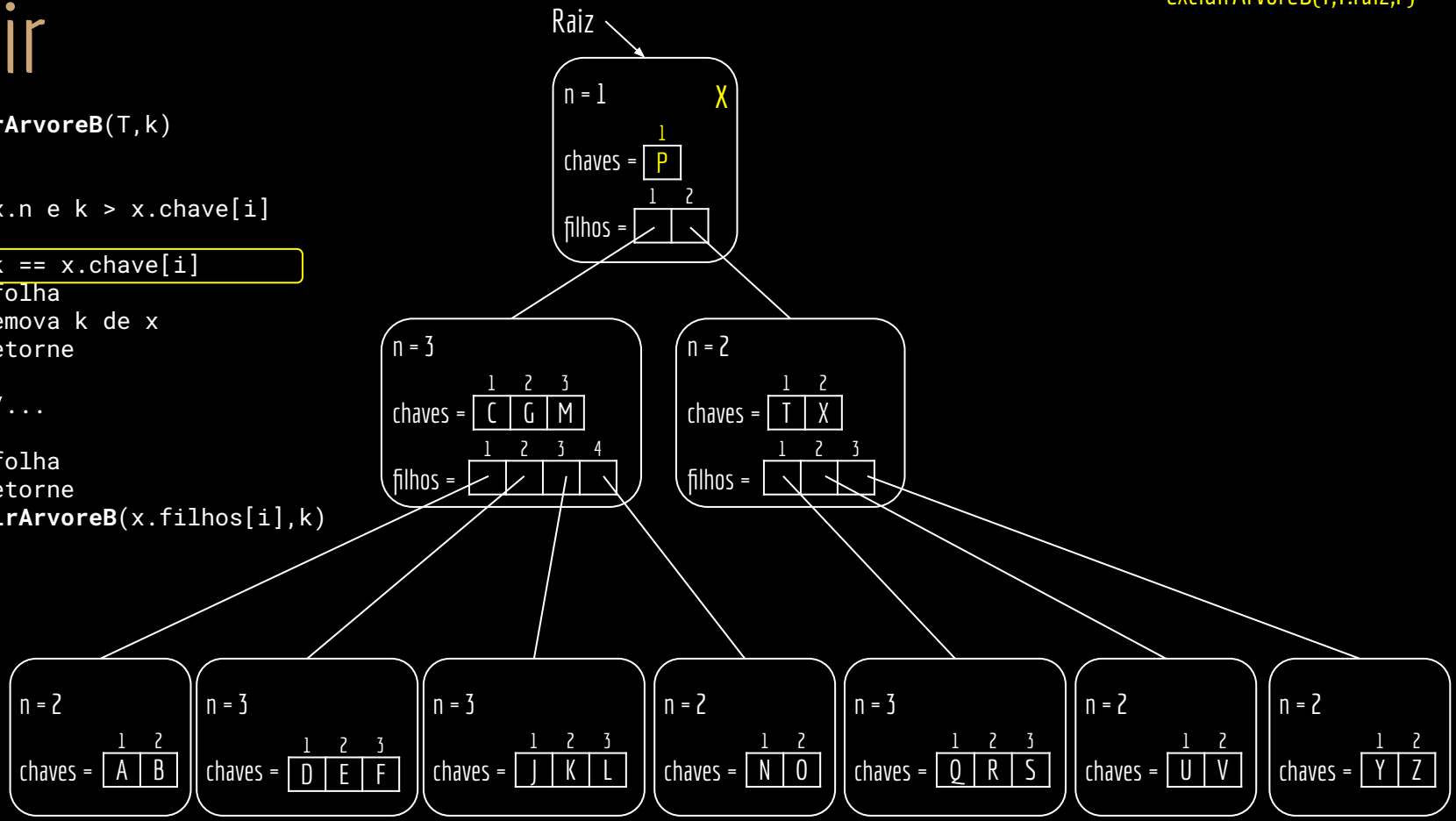


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

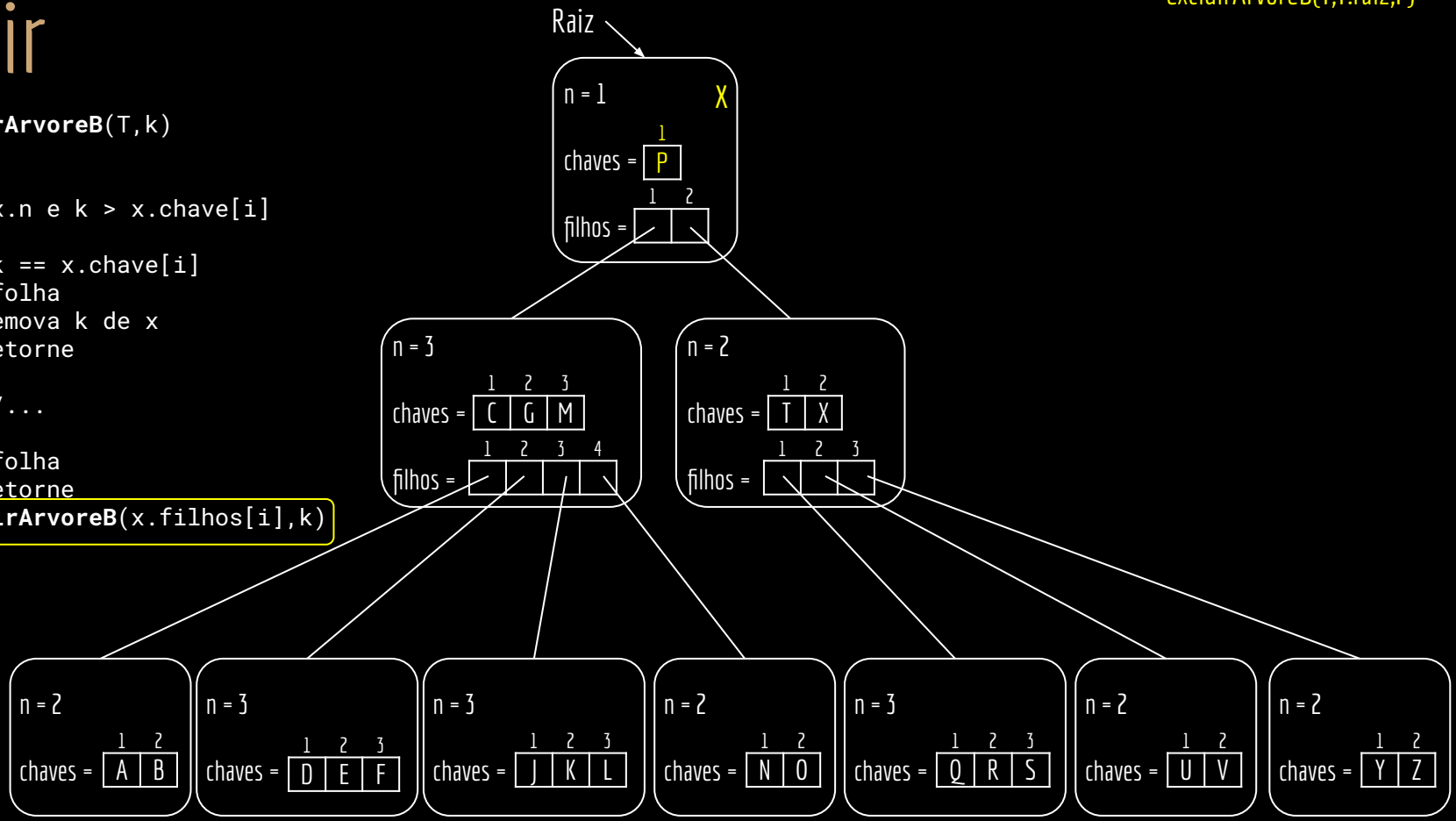


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

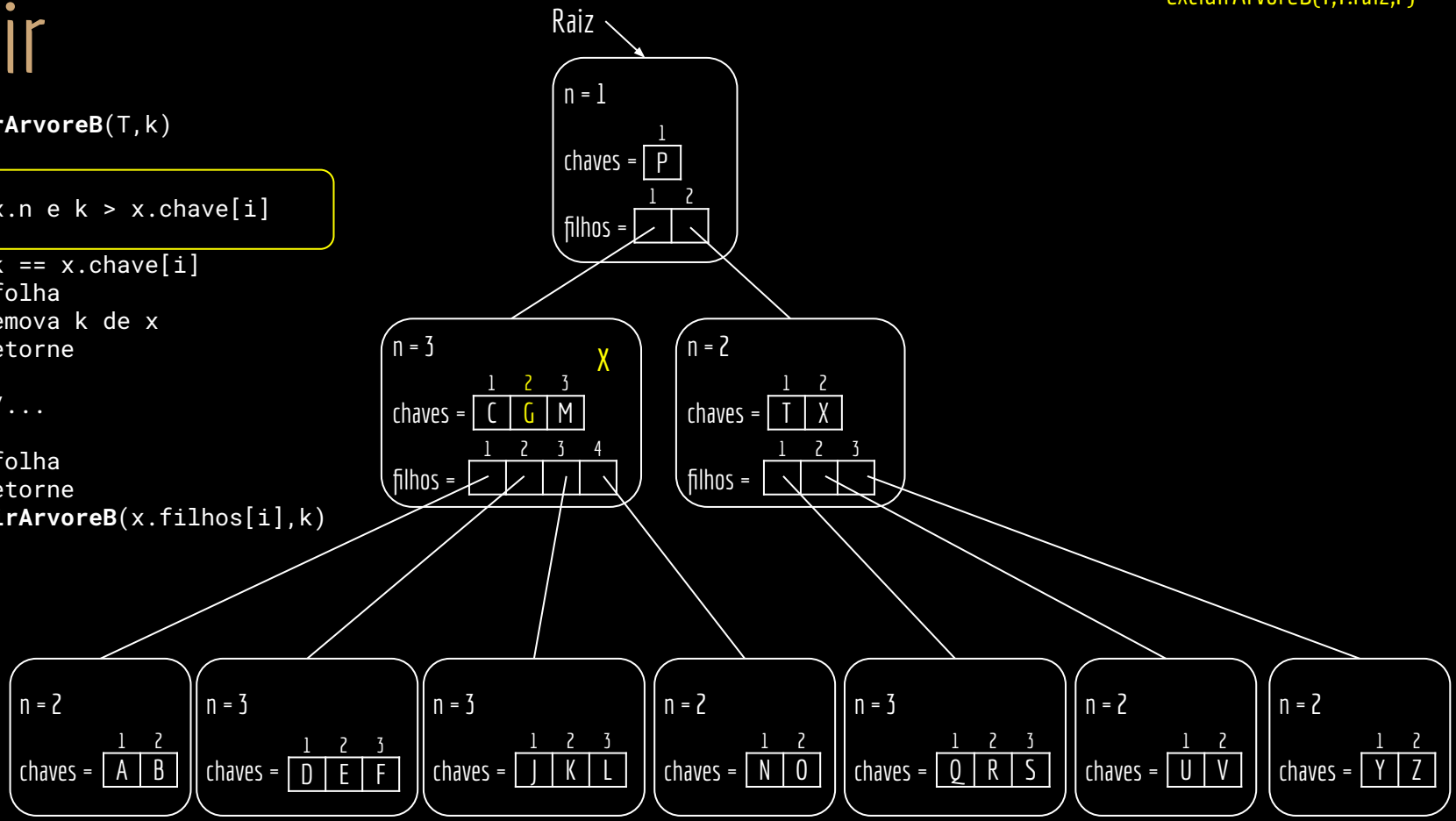


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

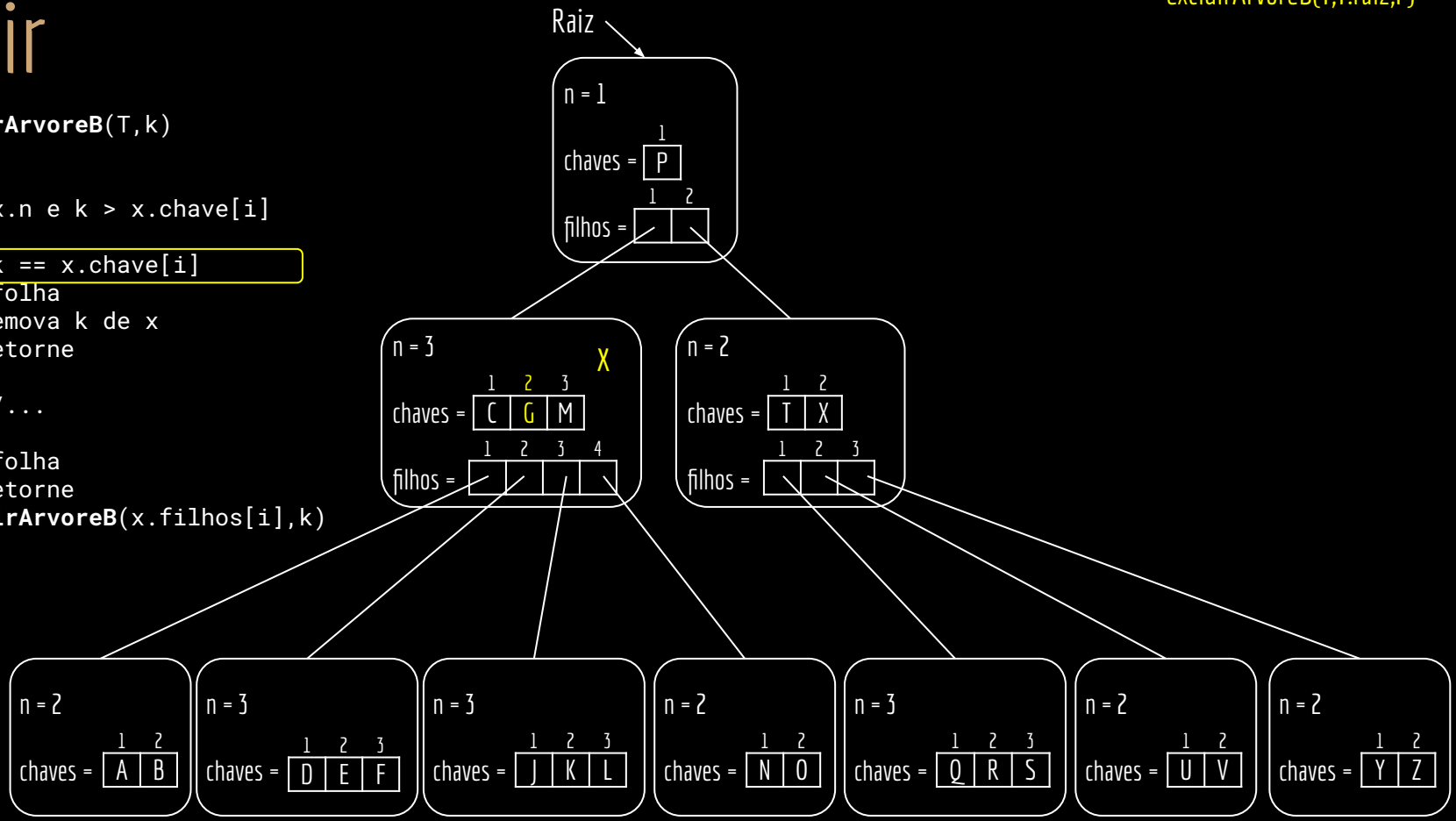


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

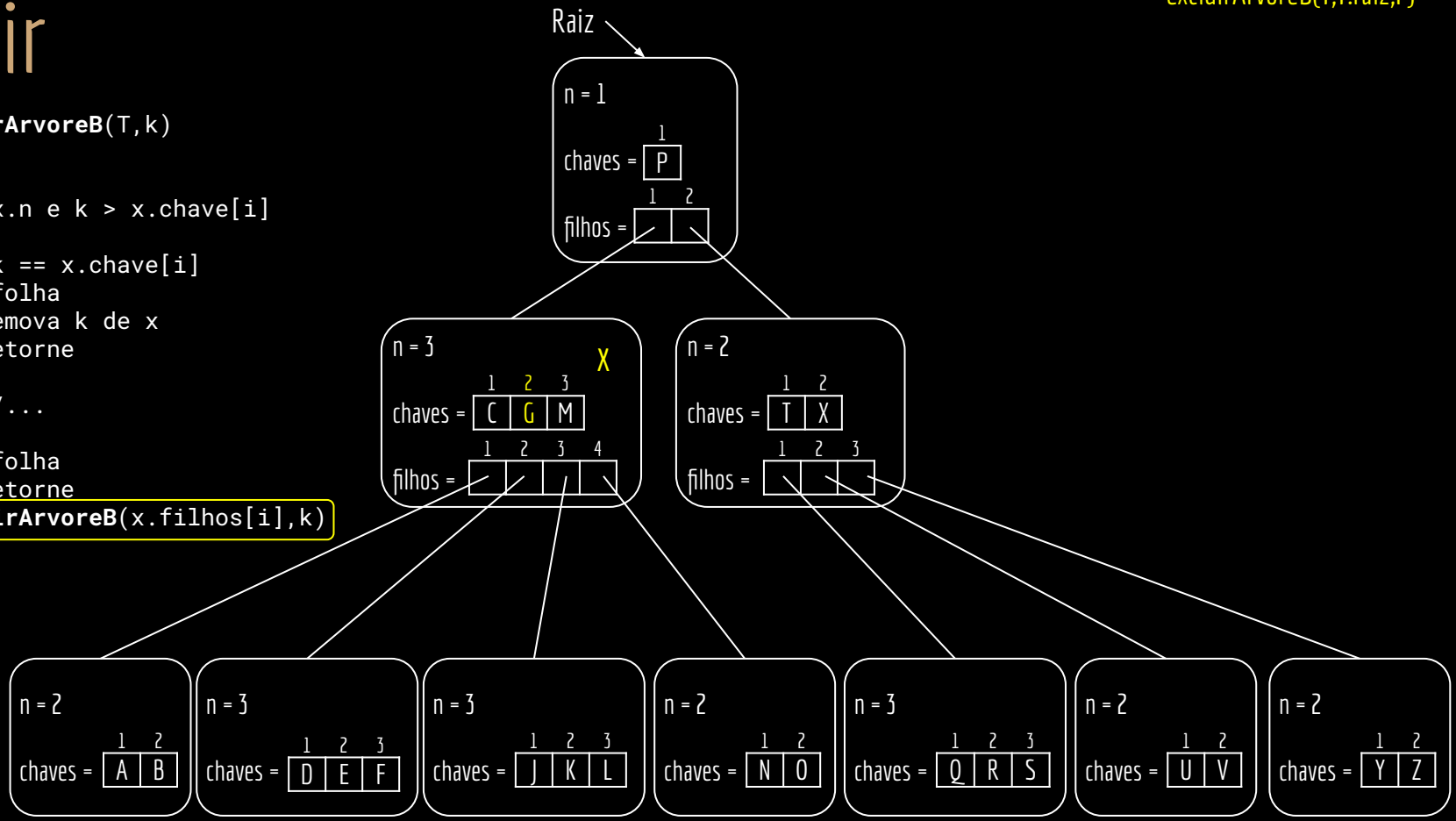


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

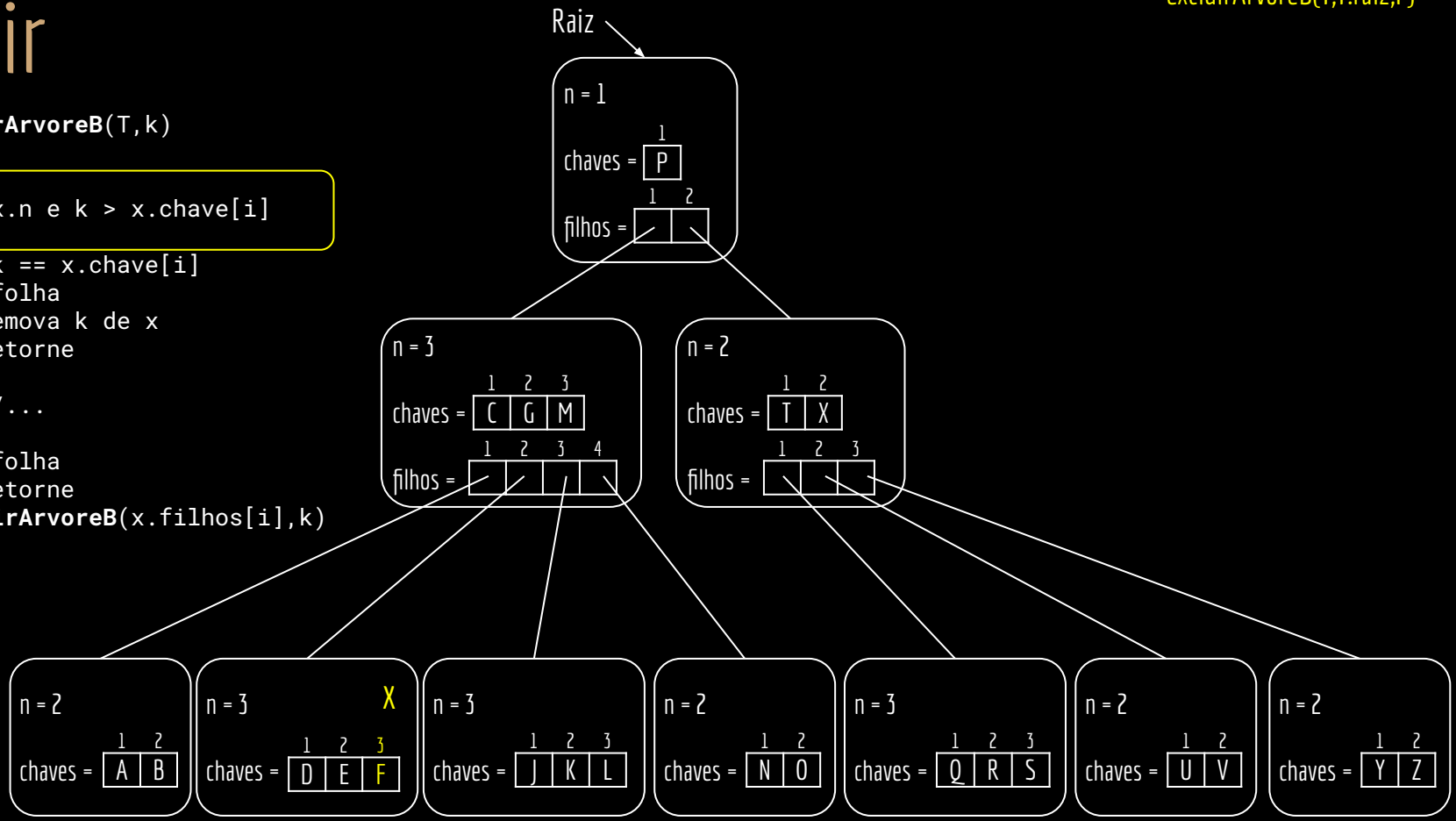


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

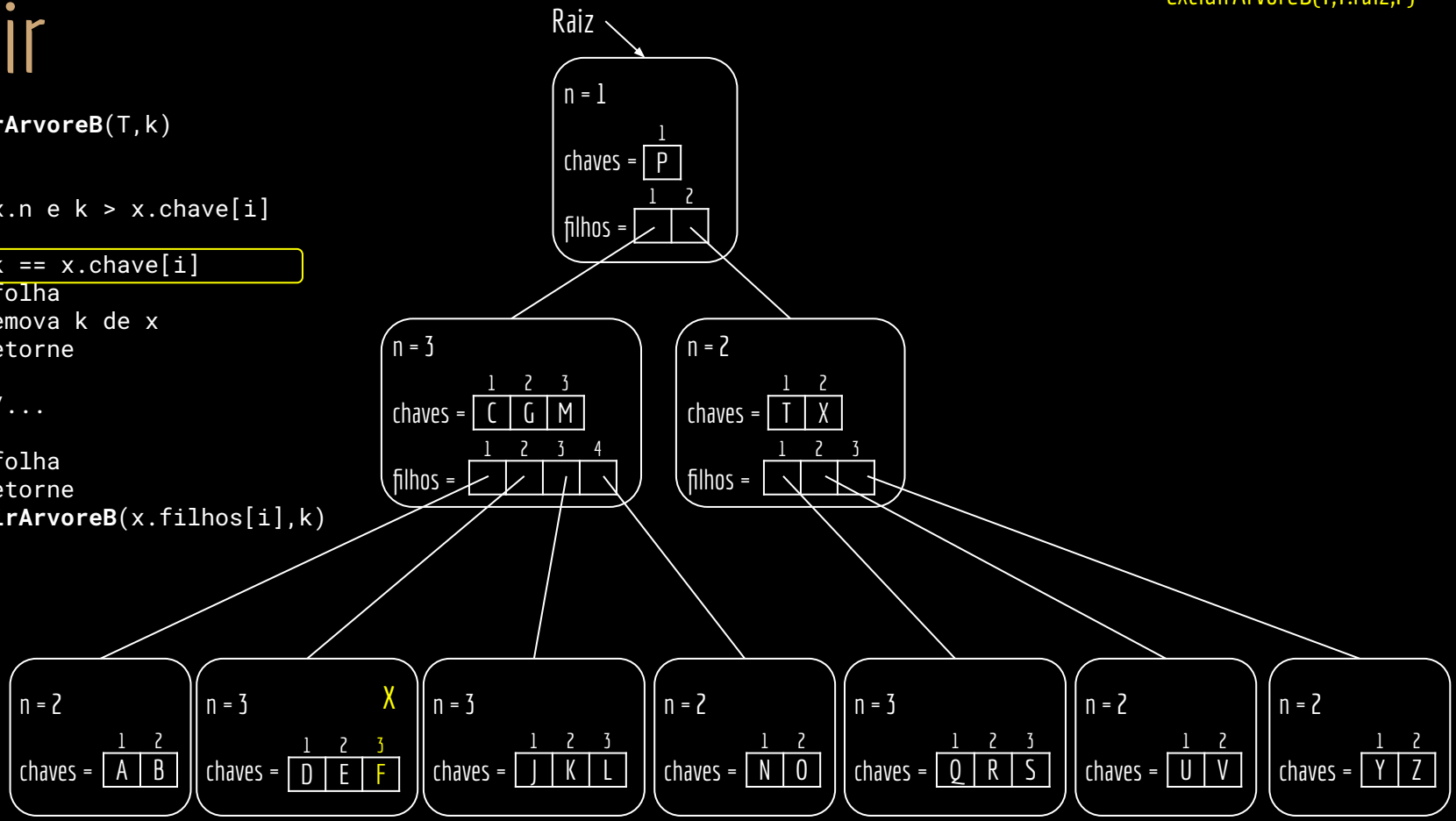


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

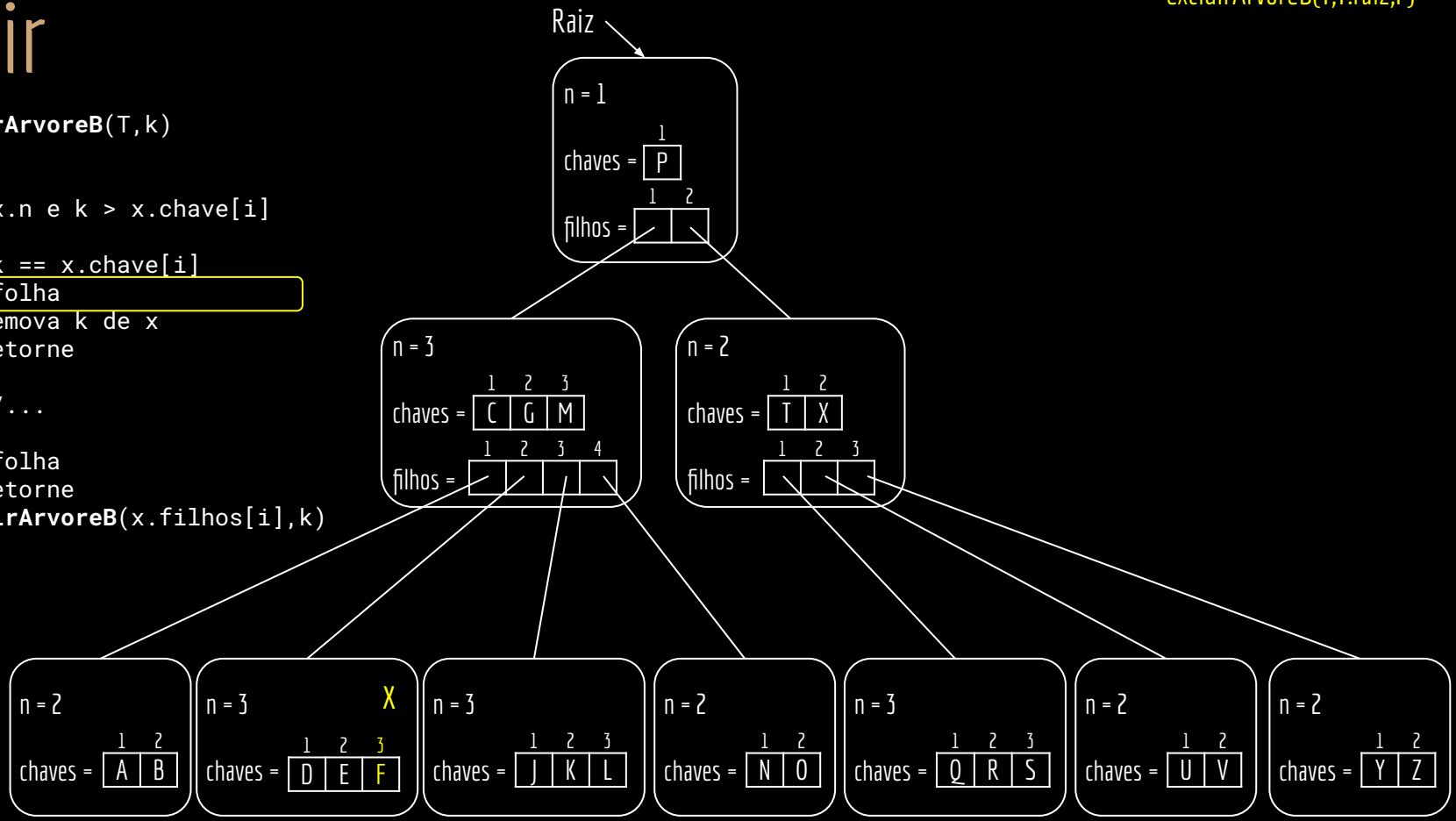


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

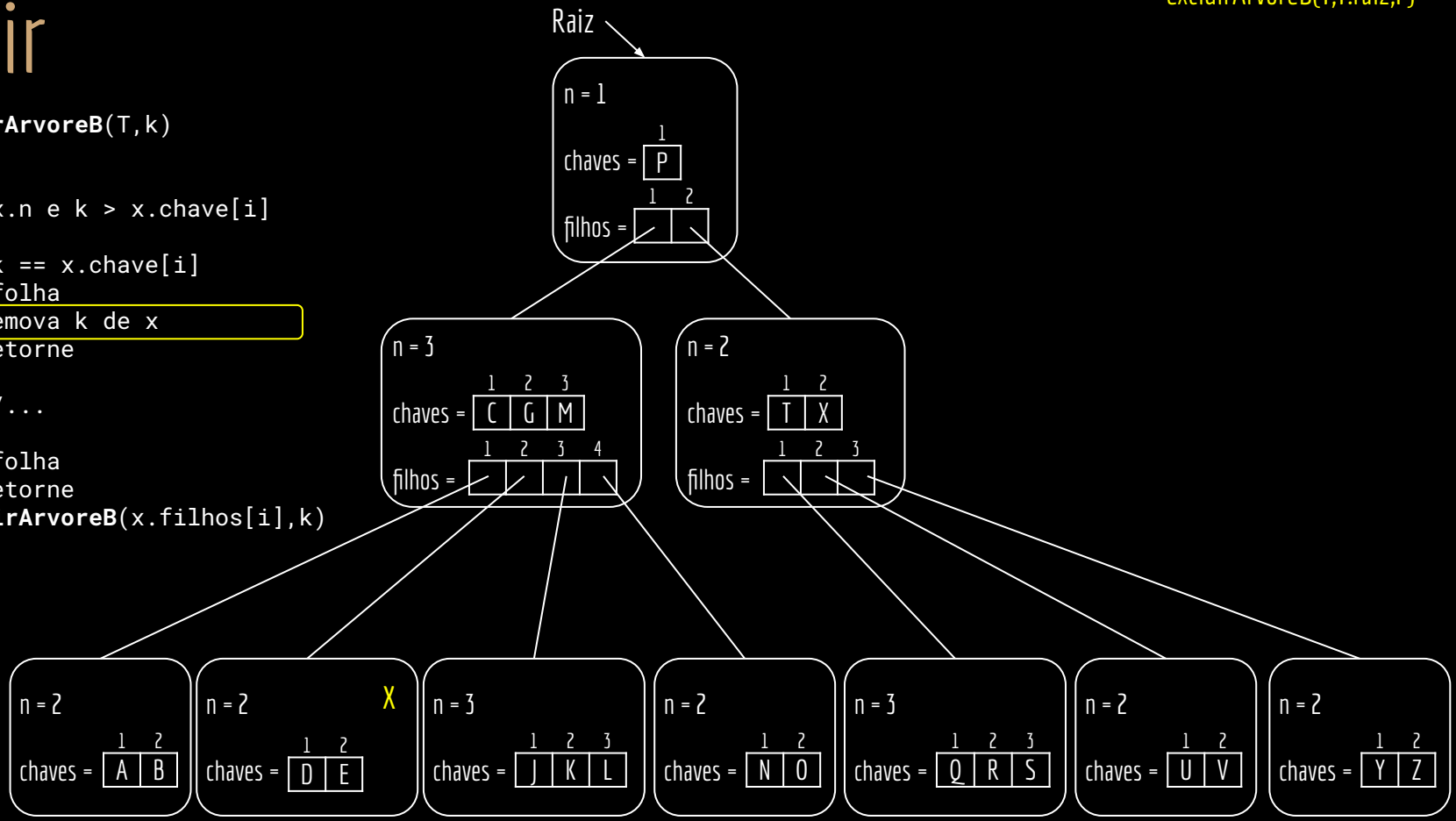


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

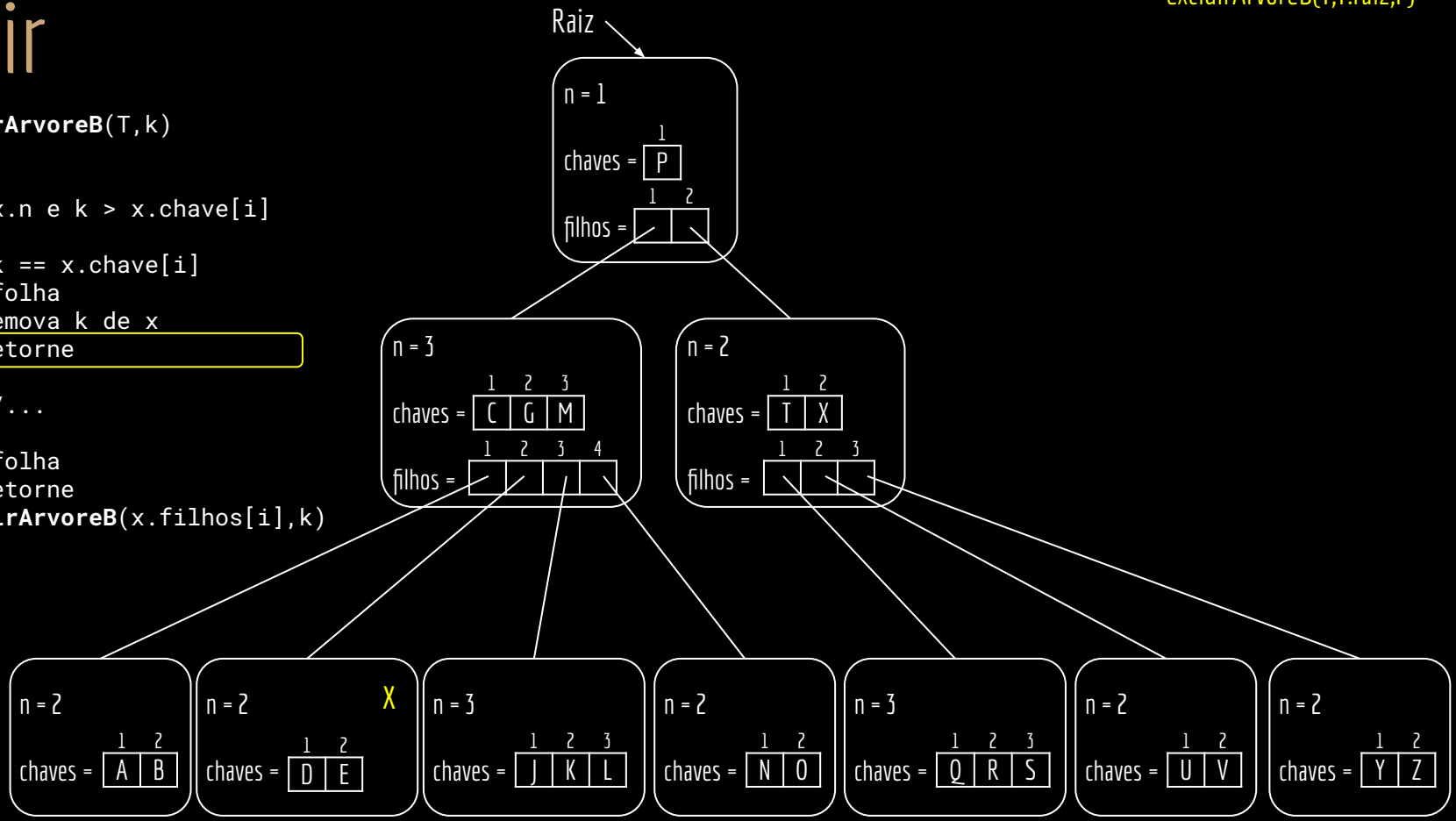


Excluir

t=3
excluirArvoreB(T,T.raiz,F)

```
função excluirArvoreB(T,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
     i = i+1
se i ≤ x.n e k == x.chave[i] //chave encontrada
     se x é folha
          remova k de x
          retorne
     senão
          se x.filhos[i].n ≥ t
               (p,j) = encontrarPred(x.filhos[i]) //Predecessor de k
               x.chave[i] = p.chaves[j]
               retorne excluirArvoreB(T,p,p.chaves[j]) // recursão
          senão
               se x.filhos[i+1].n ≥ t
                    //...
senão
     se x é folha
          retorne //chave não encontrada
retorne excluirArvoreB(x.filhos[i],k)
```

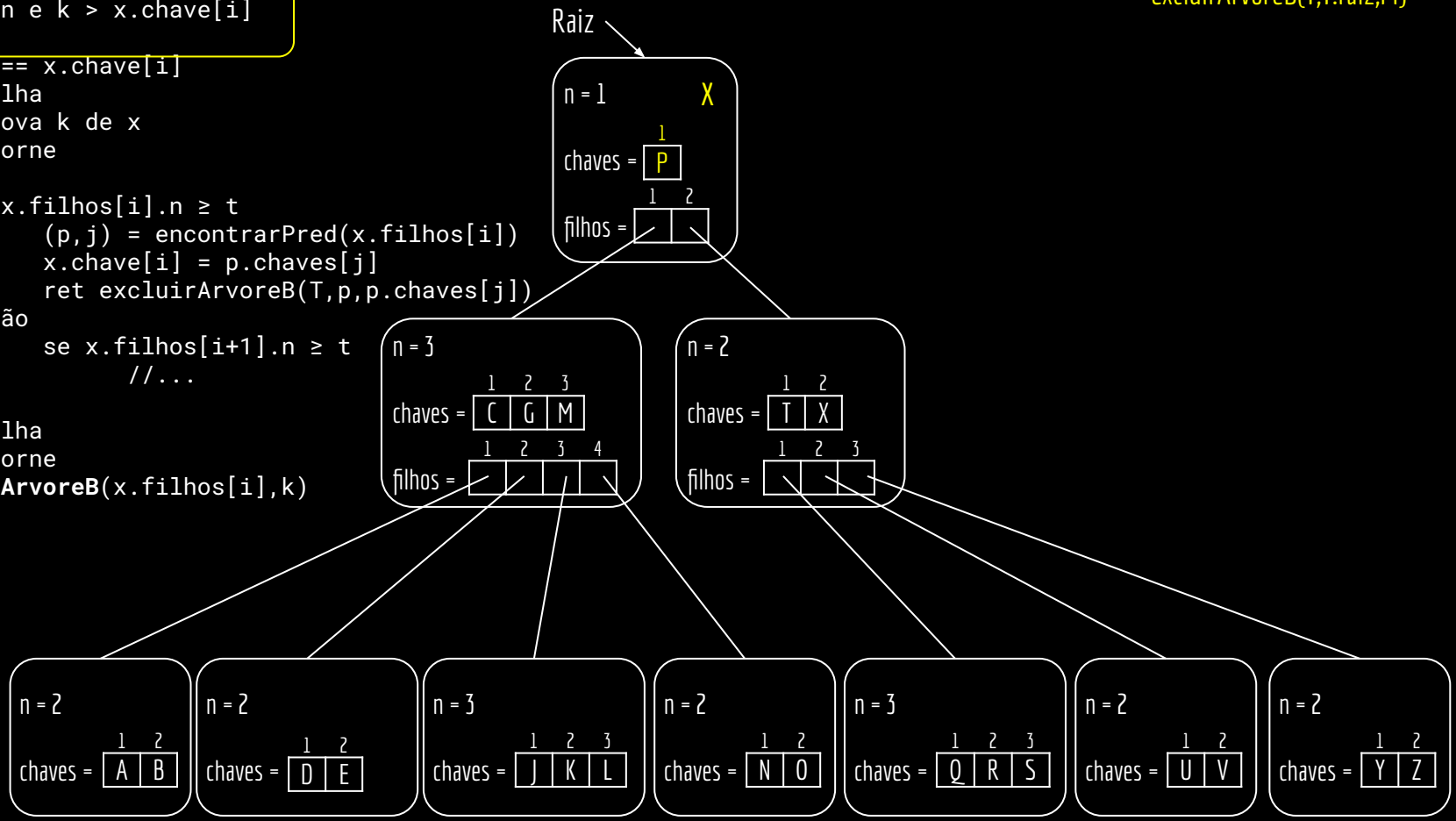
Caso 2, a chave está em um nodo interno.
2a. Subárvore esquerda possui pelo menos *t* chaves.
2b. Subárvore esquerda possui *t-1* chaves. Subárvore direita possui pelo menos *t* chaves. Simétrico a 2a.

```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

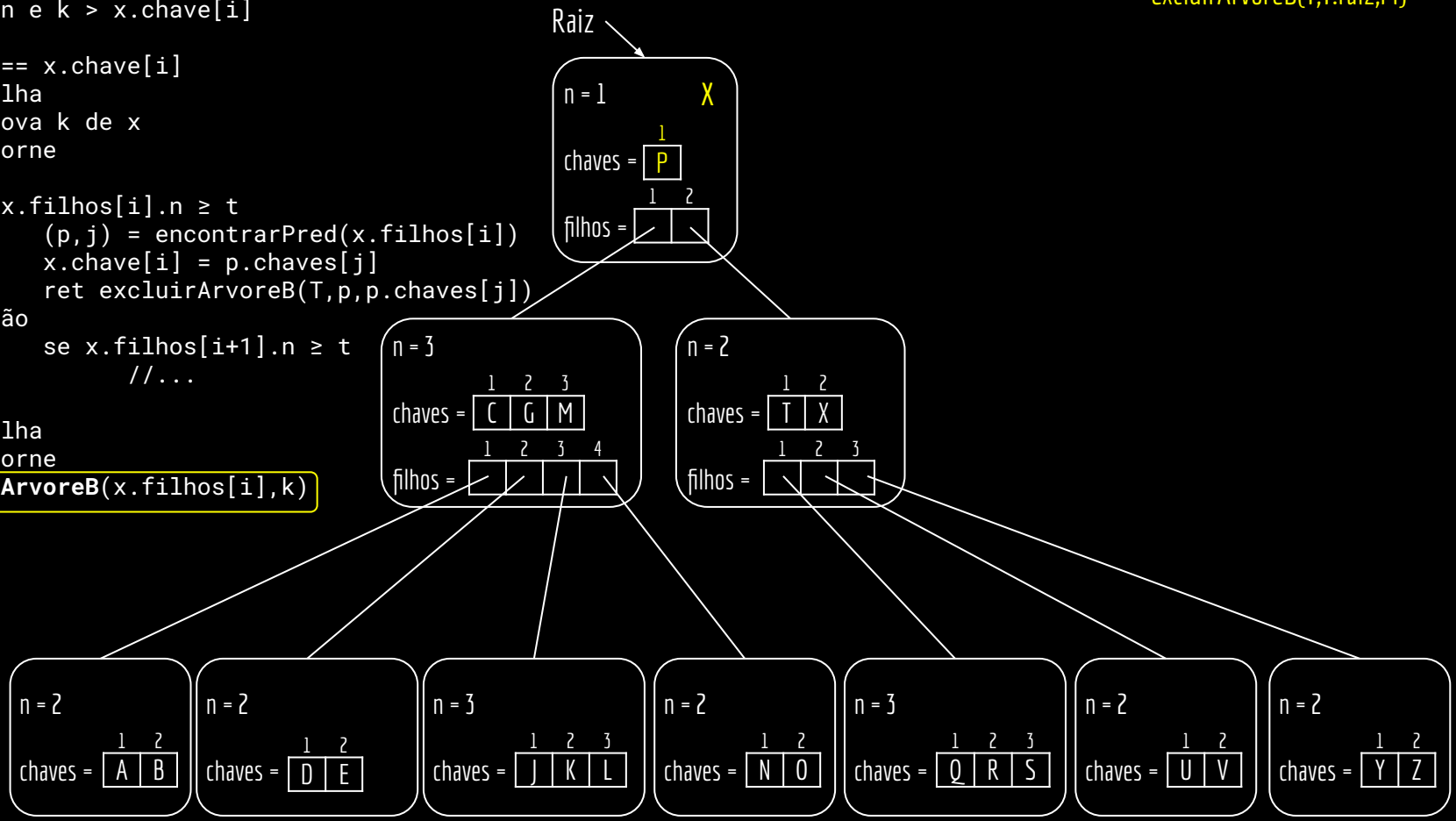
excluirArvoreB(T,T.raiz,M)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3
excluirArvoreB(T,T.raiz,M)

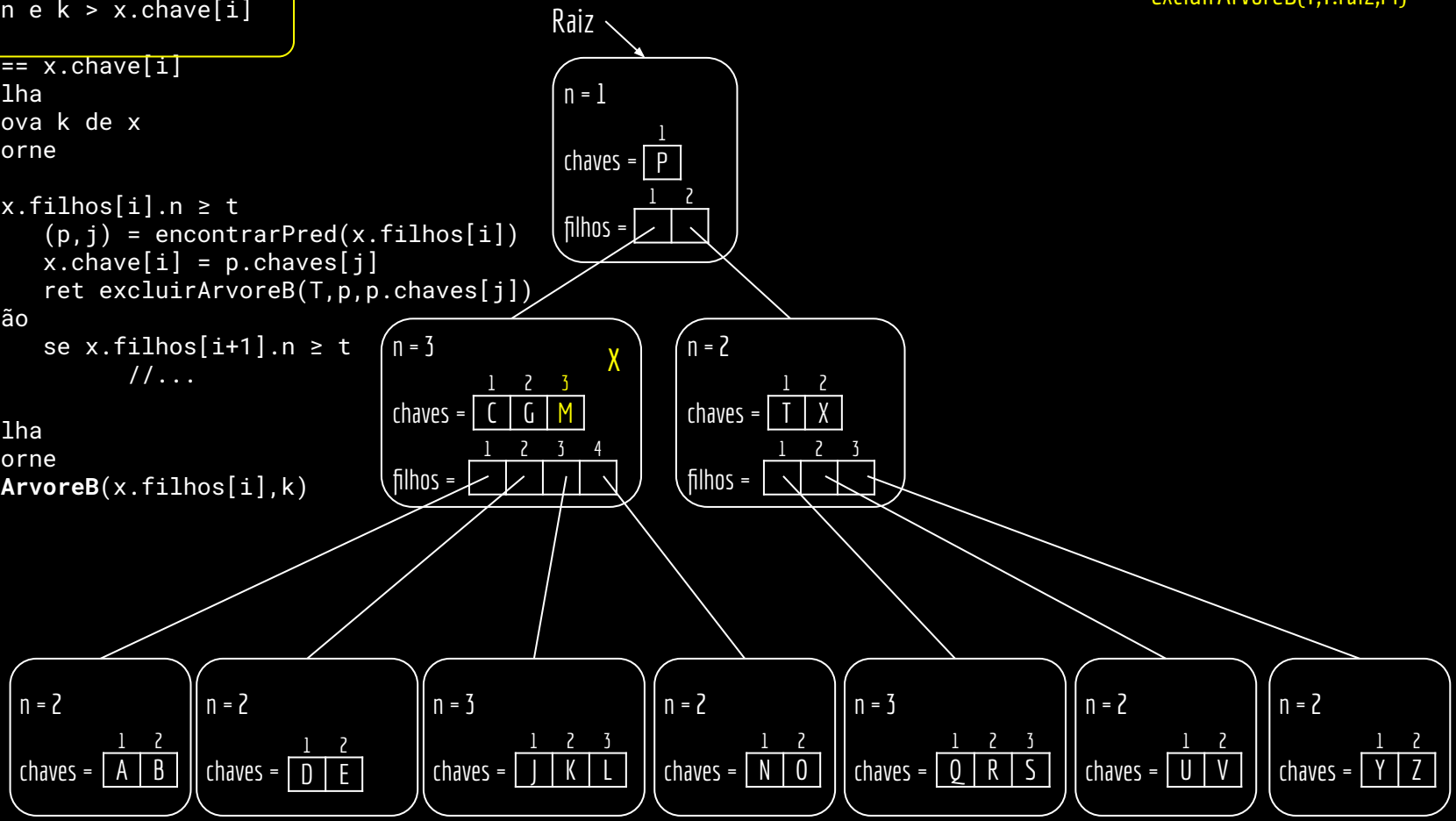


```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

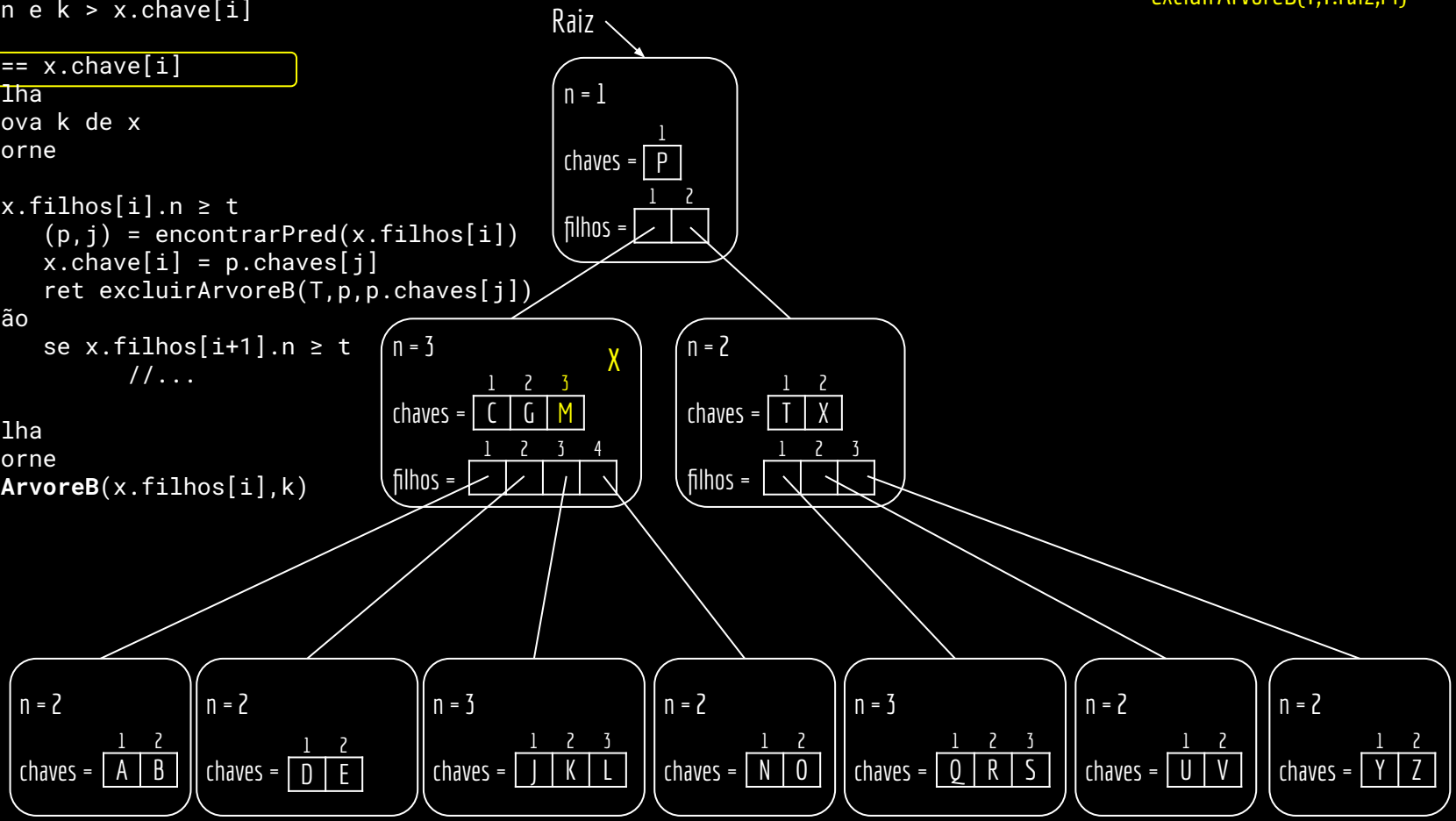
excluirArvoreB(T,T.raiz,M)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

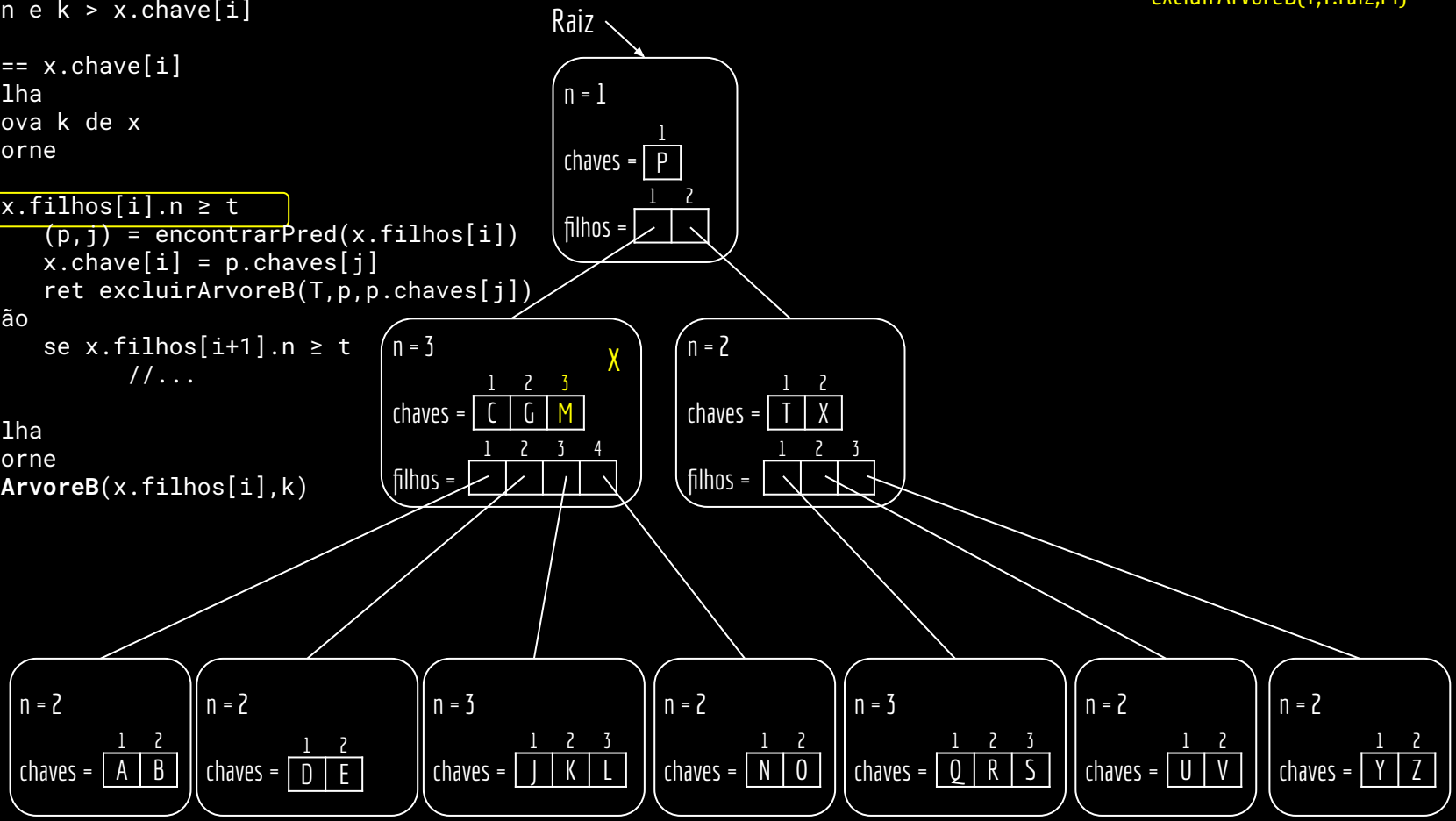
t=3
excluirArvoreB(T,T.raiz,M)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

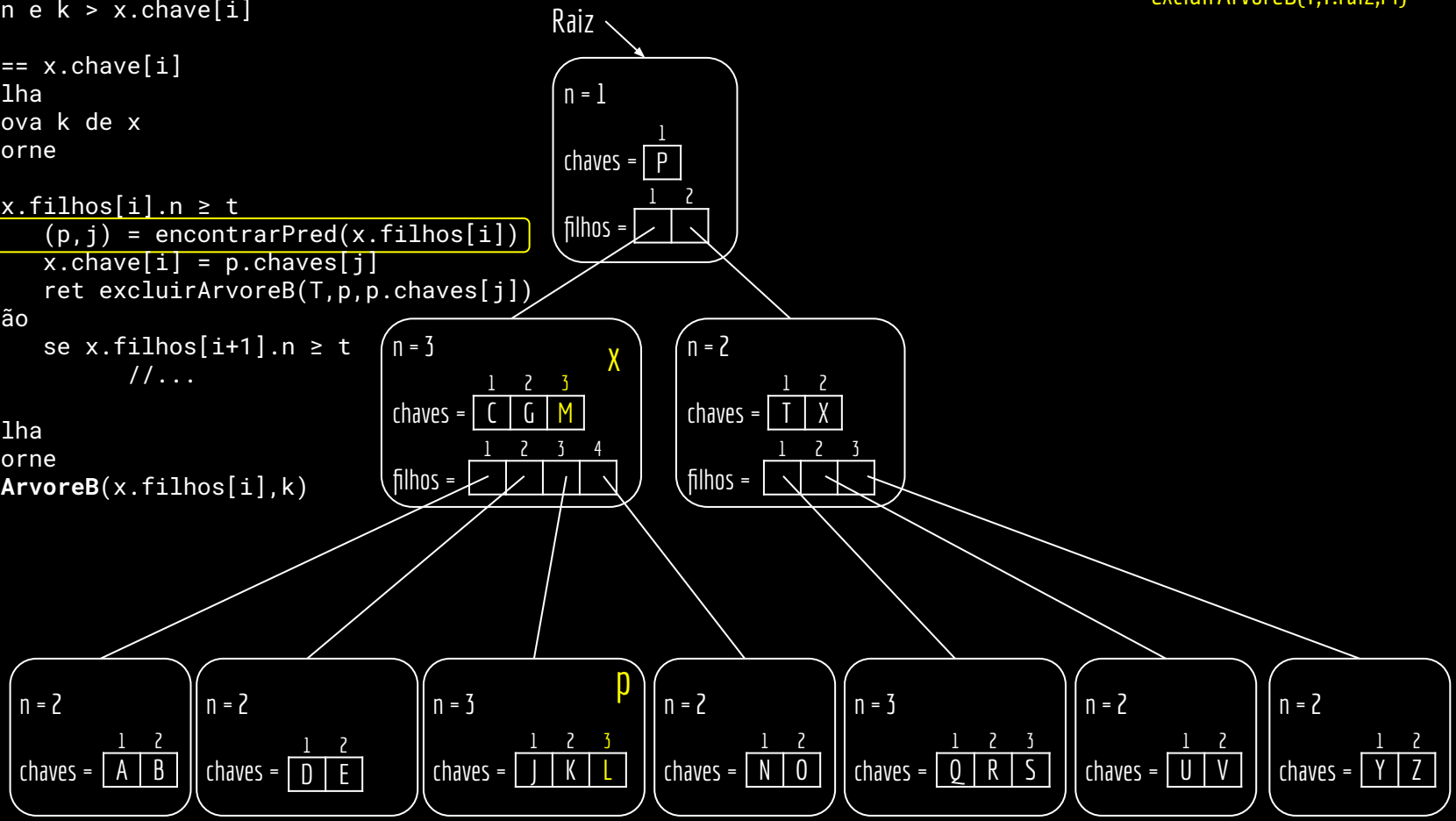
t=3
excluirArvoreB(T,T.raiz,M)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

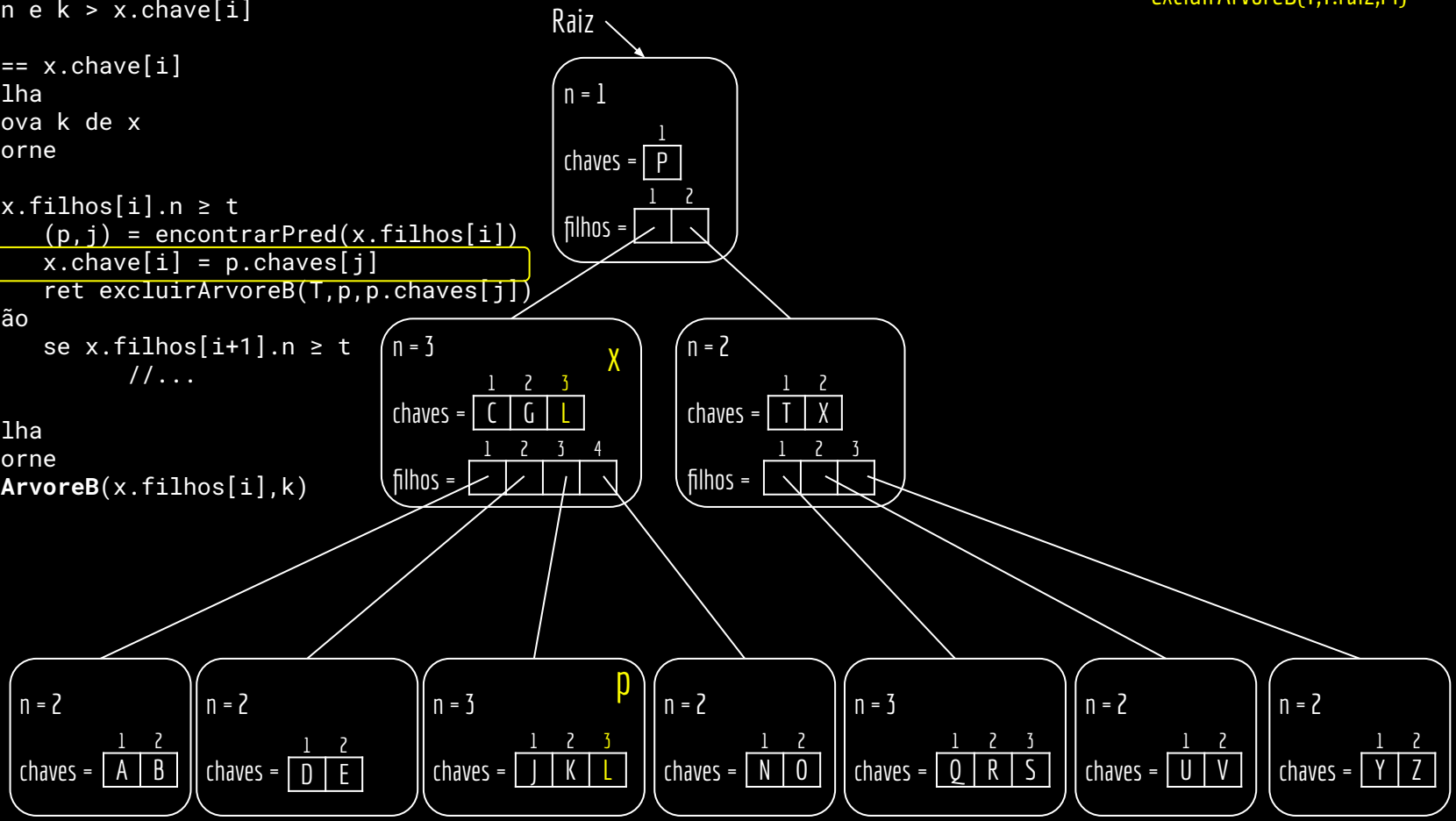
t=3
excluirArvoreB(T,T.raiz,M)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

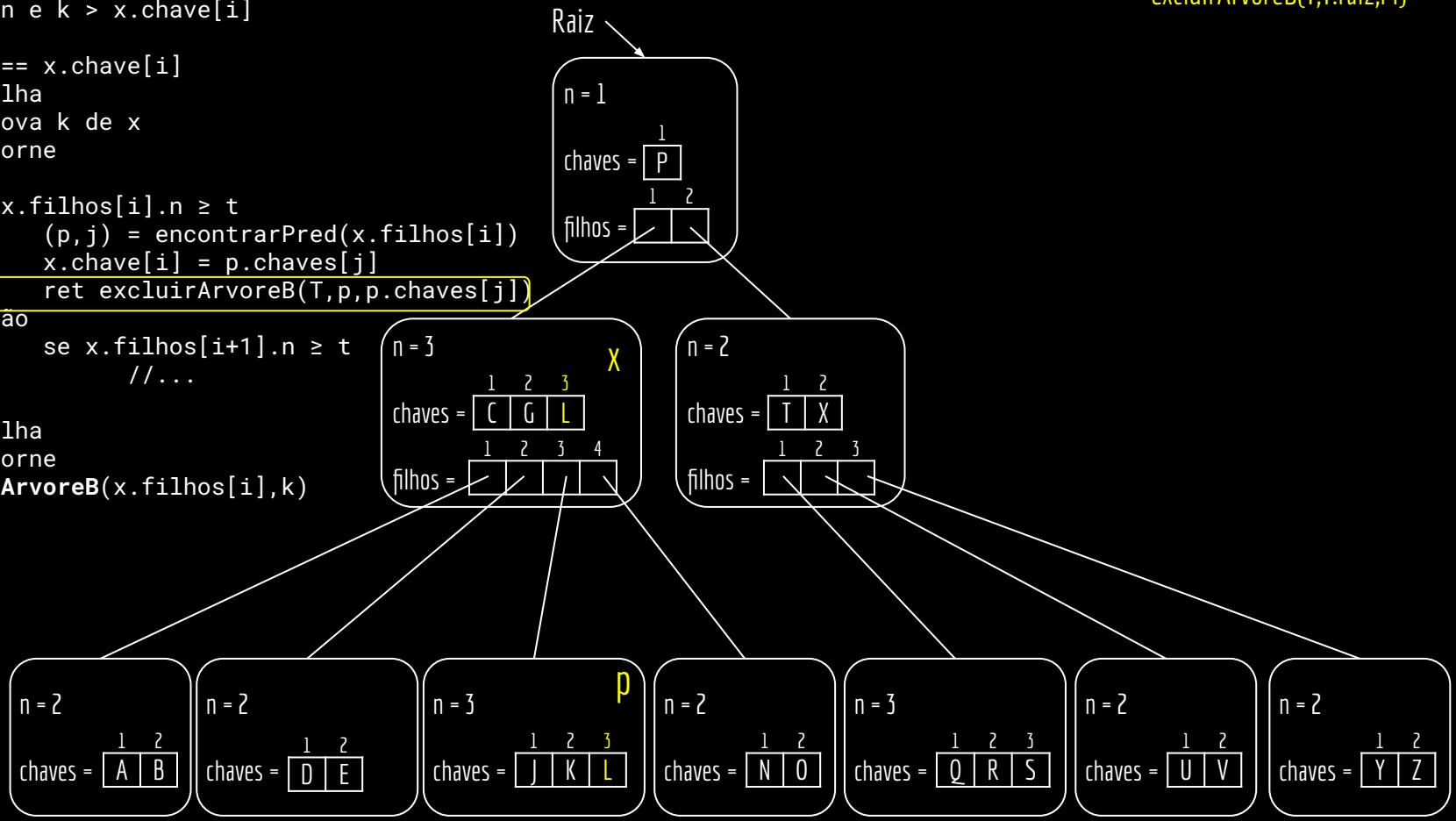
t=3
excluirArvoreB(T,T.raiz,M)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

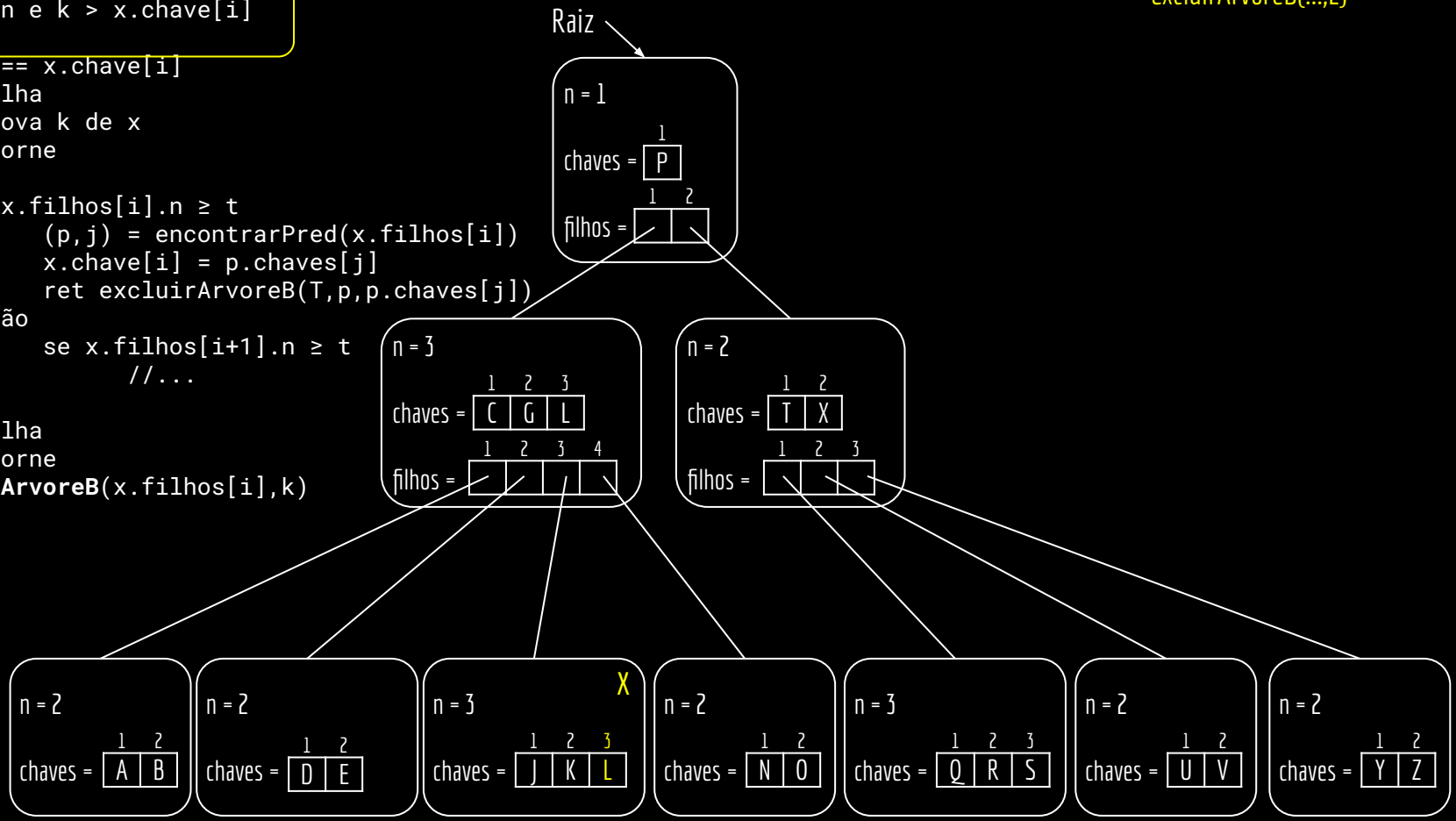
t=3
excluirArvoreB(T,T.raiz,M)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

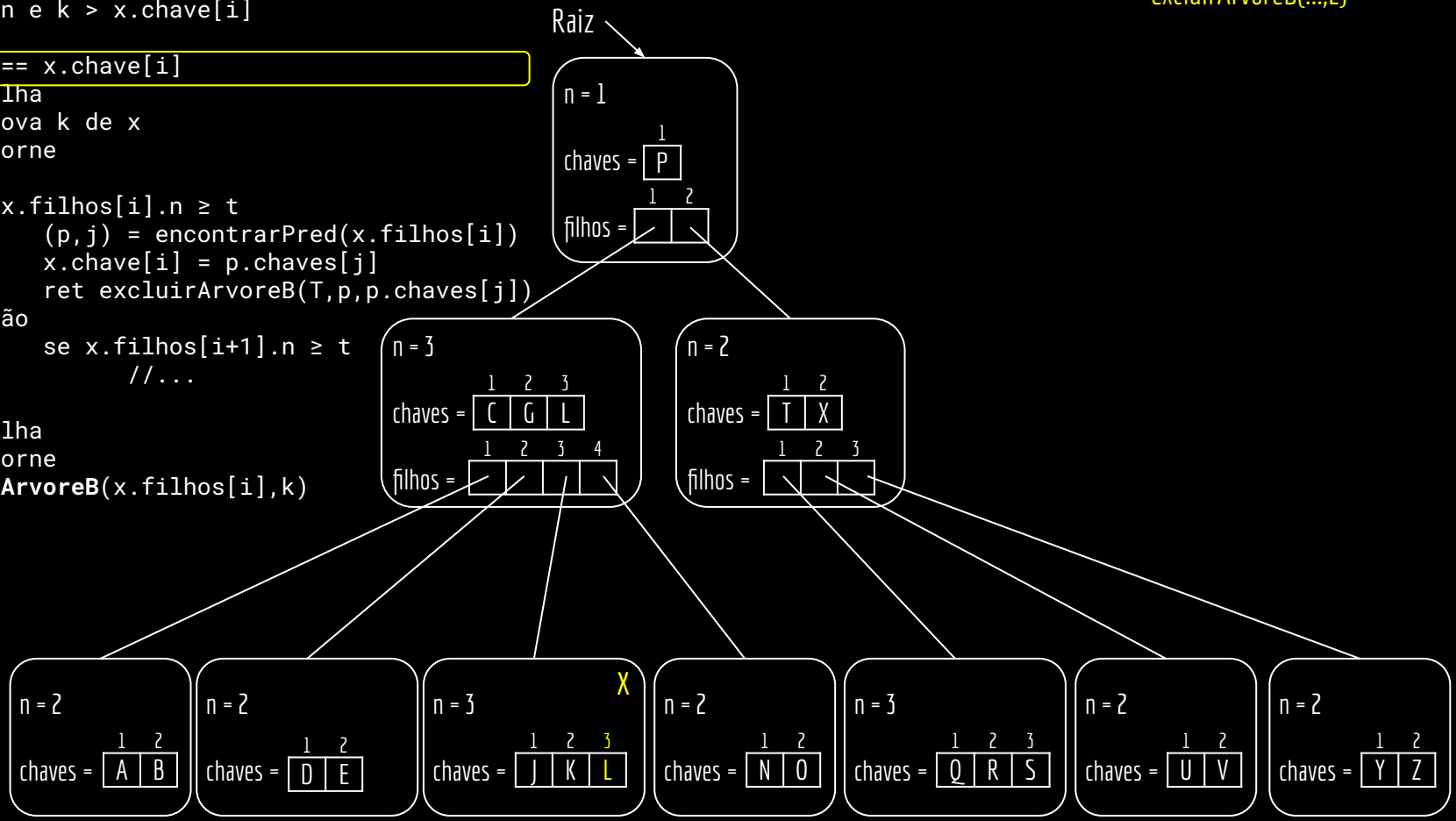
t=3
excluirArvoreB(...,L)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

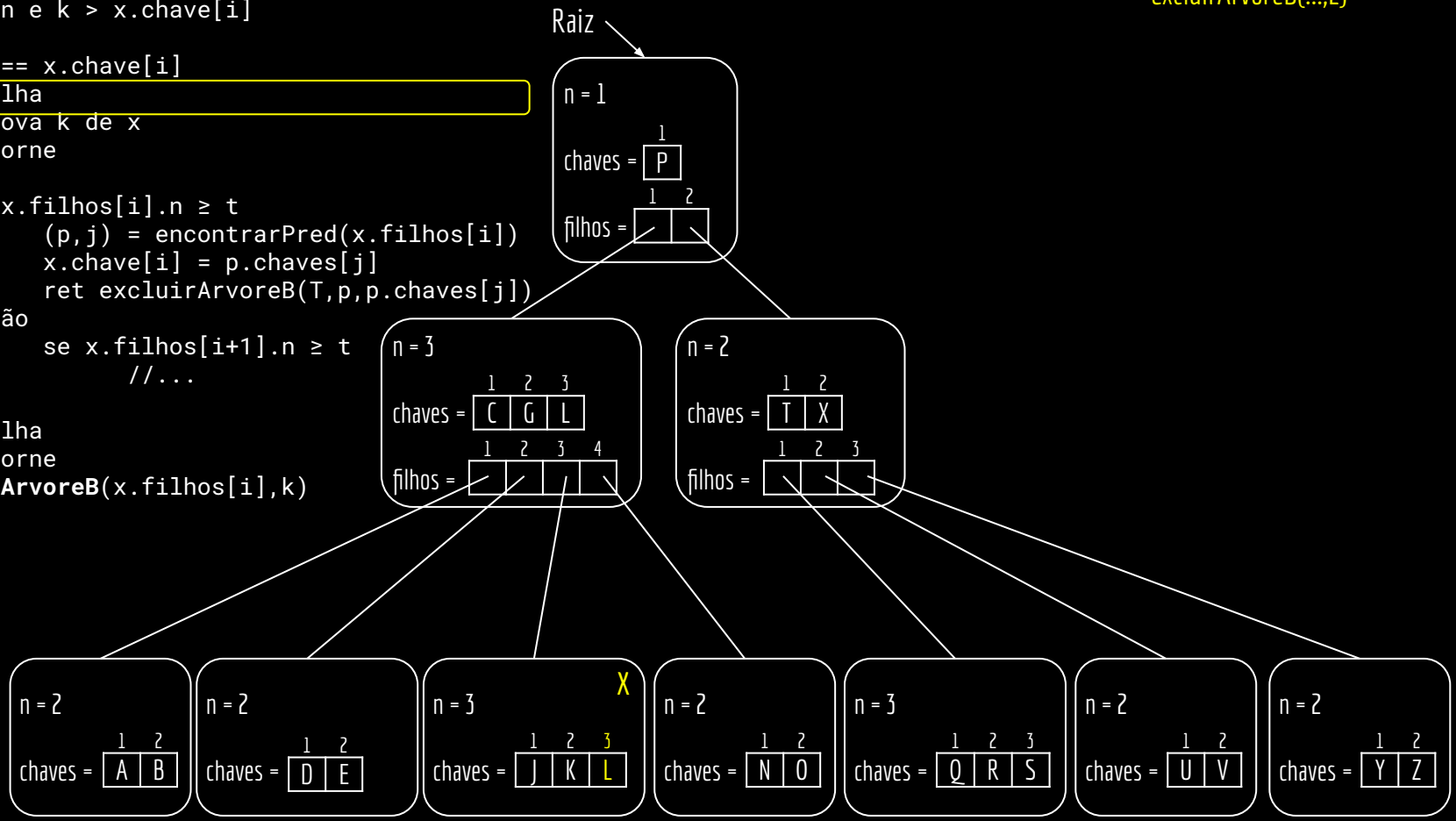
t=3
excluirArvoreB(...,L)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

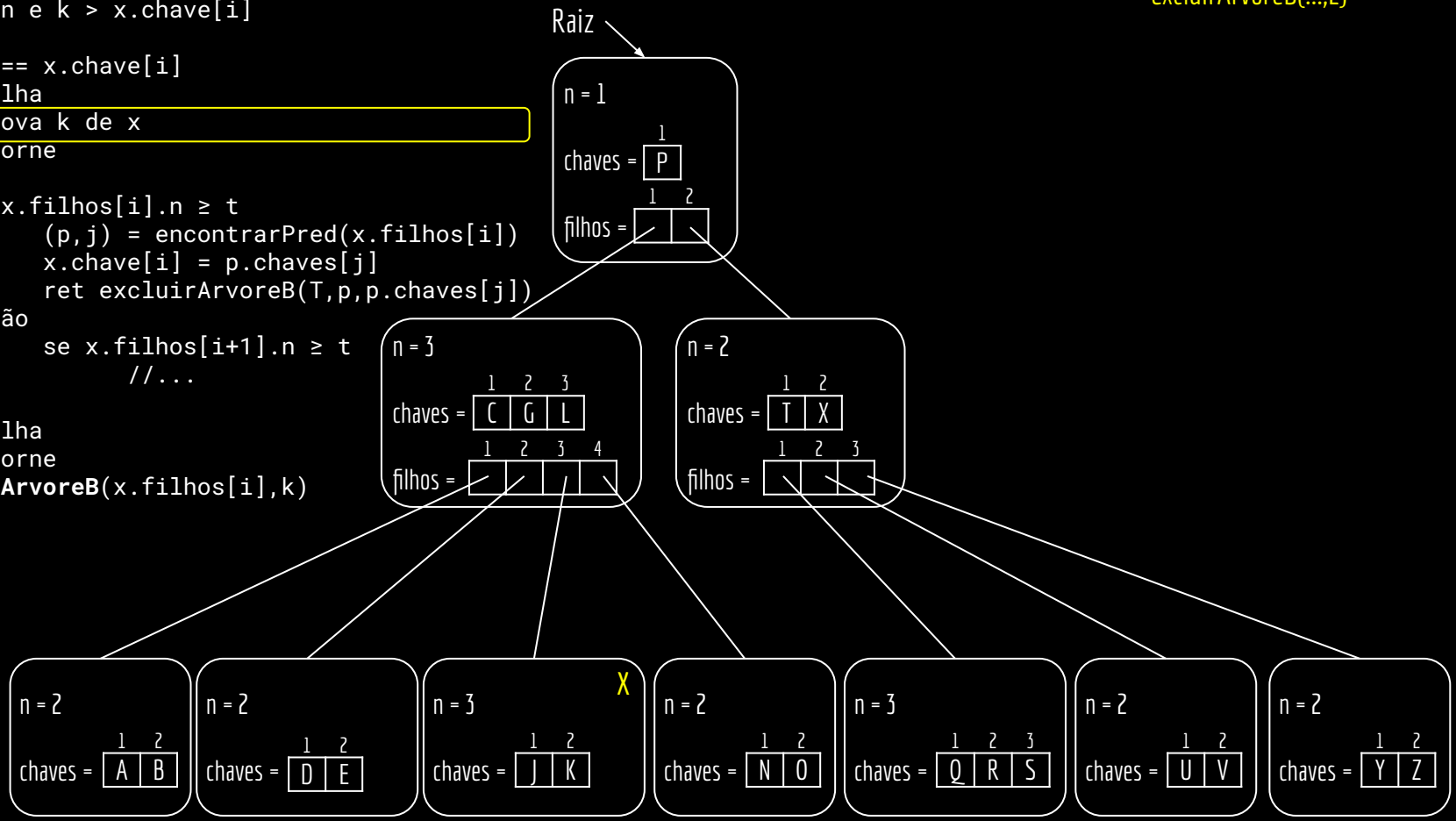
t=3
excluirArvoreB(...,L)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

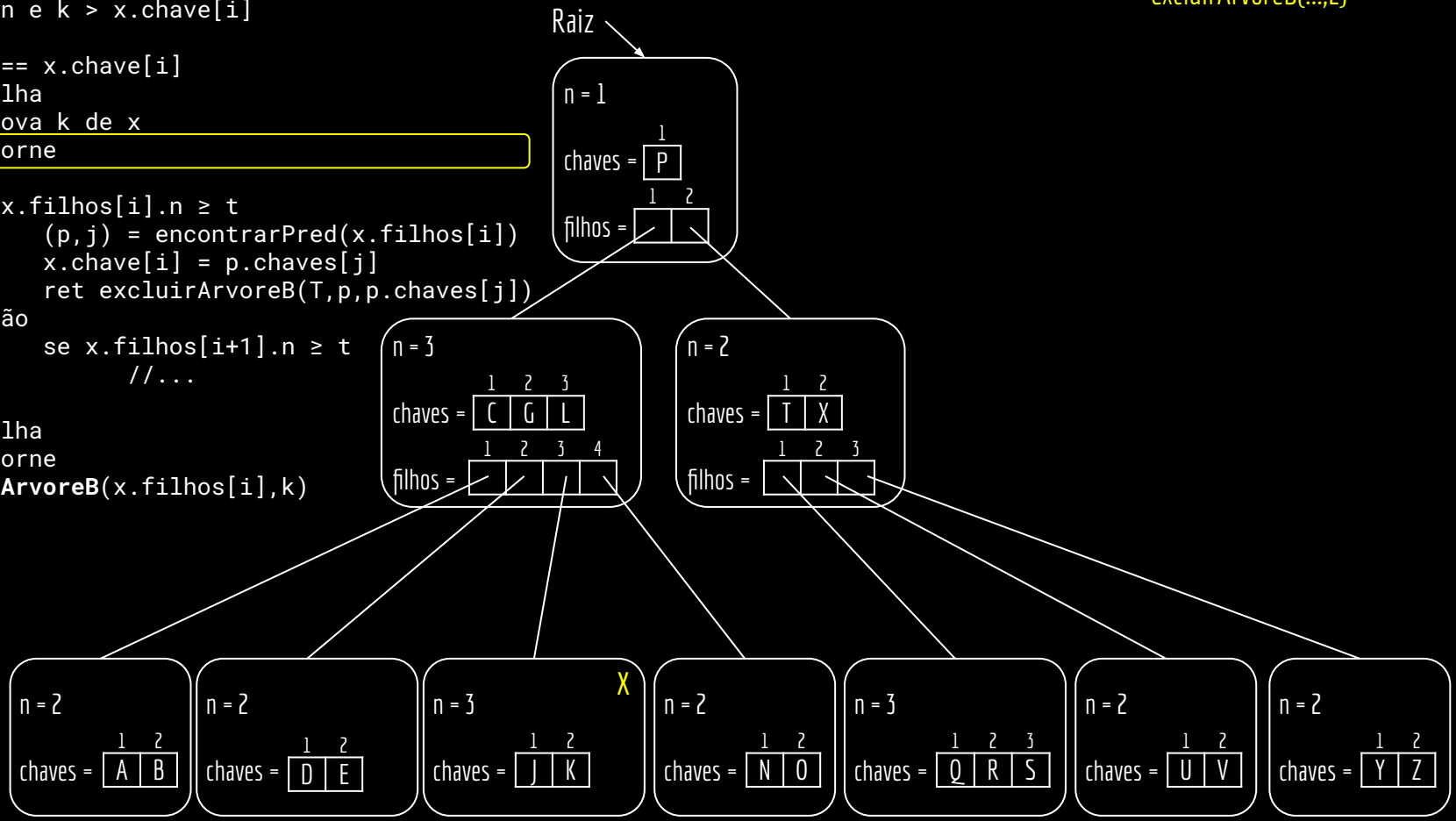
t=3
excluirArvoreB(...,L)



```
função excluirArvoreB(T,x,k)

i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  ret excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3
excluirArvoreB(...,L)



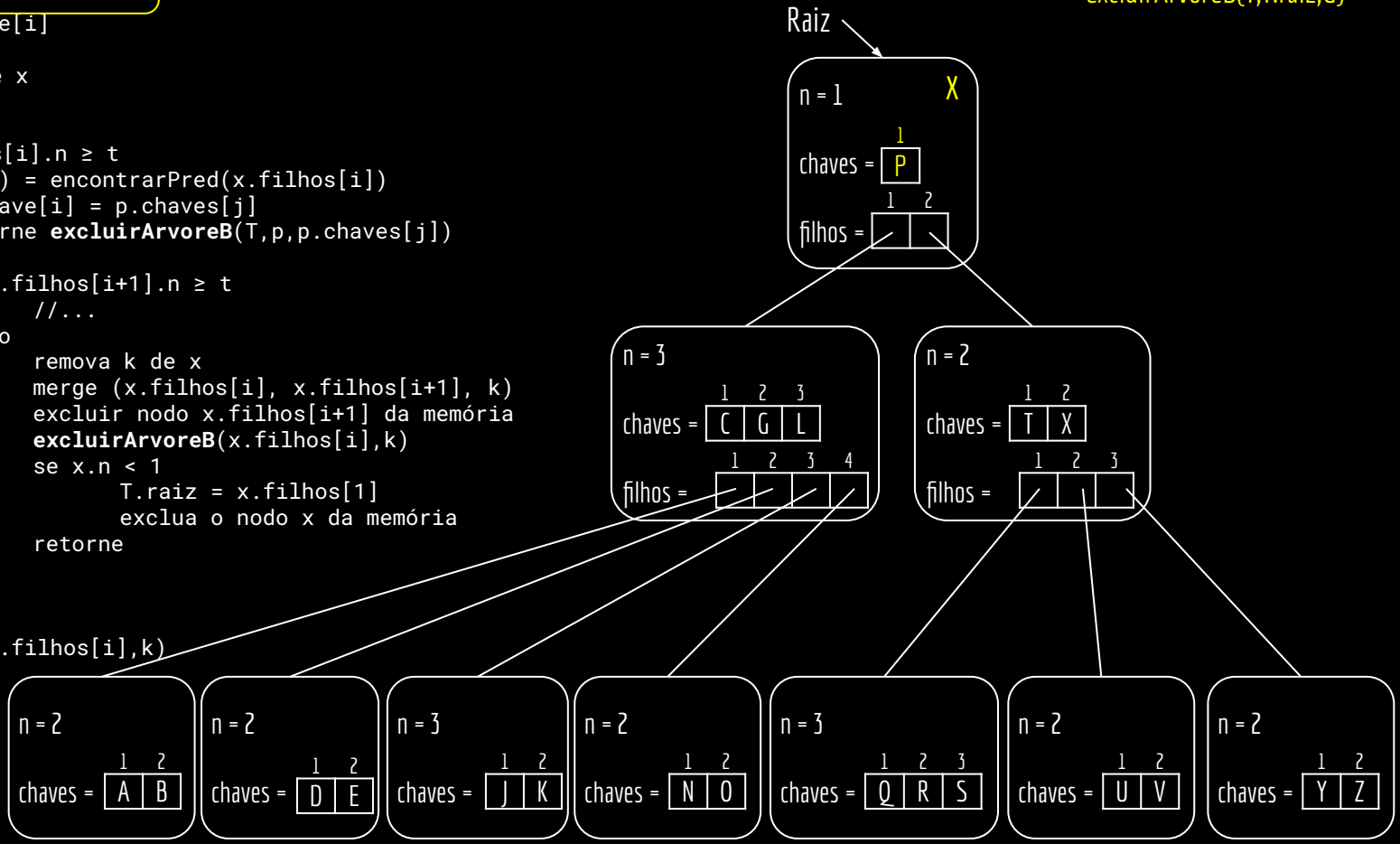
```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
     i = i+1
se i ≤ x.n e k == x.chave[i] //chave encontrada
     se x é folha
          remova k de x
          retorne
     senão
          se x.filhos[i].n ≥ t
               (p,j) = encontrarPred(x.filhos[i]) //Predecessor de k
               x.chave[i] = p.chaves[j]
               retorne excluirArvoreB(T,p,p.chaves[j]) // recursão
          senão
               se x.filhos[i+1].n ≥ t
                    //...
               senão
                    remova k de x
                    //x.filhos[i] vai receber tudo de x.filhos[i+1] e k
                    merge (x.filhos[i], x.filhos[i+1], k)
                    excluir nodo x.filhos[i+1] da memória
                    excluirArvoreB(x.filhos[i],k)
                    se x.n < 1 //x era raiz e pode não ter mais chaves
                         T.raiz = x.filhos[1]
                         exclua o nodo x da memória
                    retorne
senão
     se x é folha
          retorne //chave não encontrada
retorne excluirArvoreB(x.filhos[i],k)
```

Caso 2c: ambas subárvores têm t-1 chaves. Passar os dados (merge) de x.filhos[i+1] para x.filhos[i], juntamente com a chave k. Chamar a exclusão recursivamente.

```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

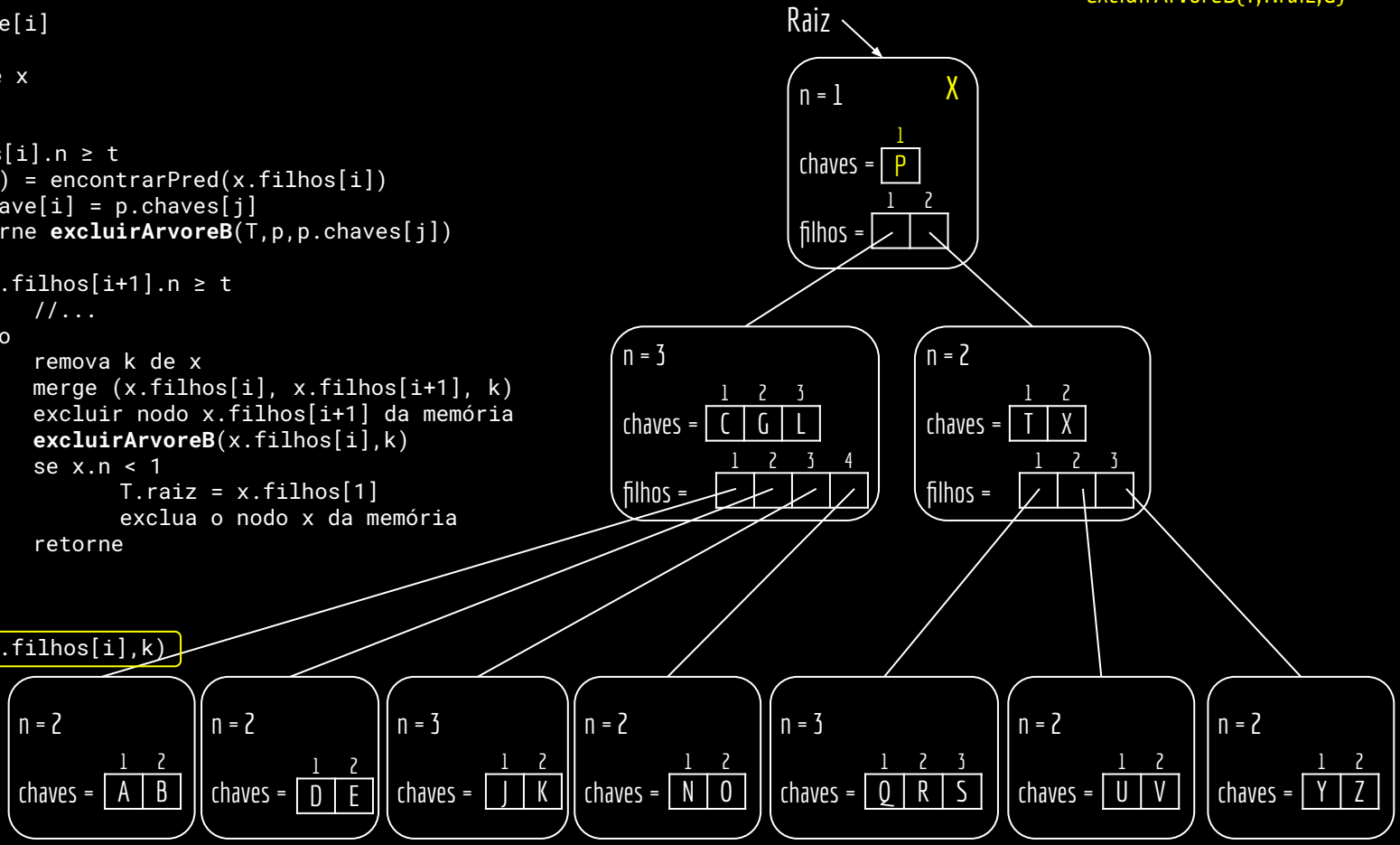
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

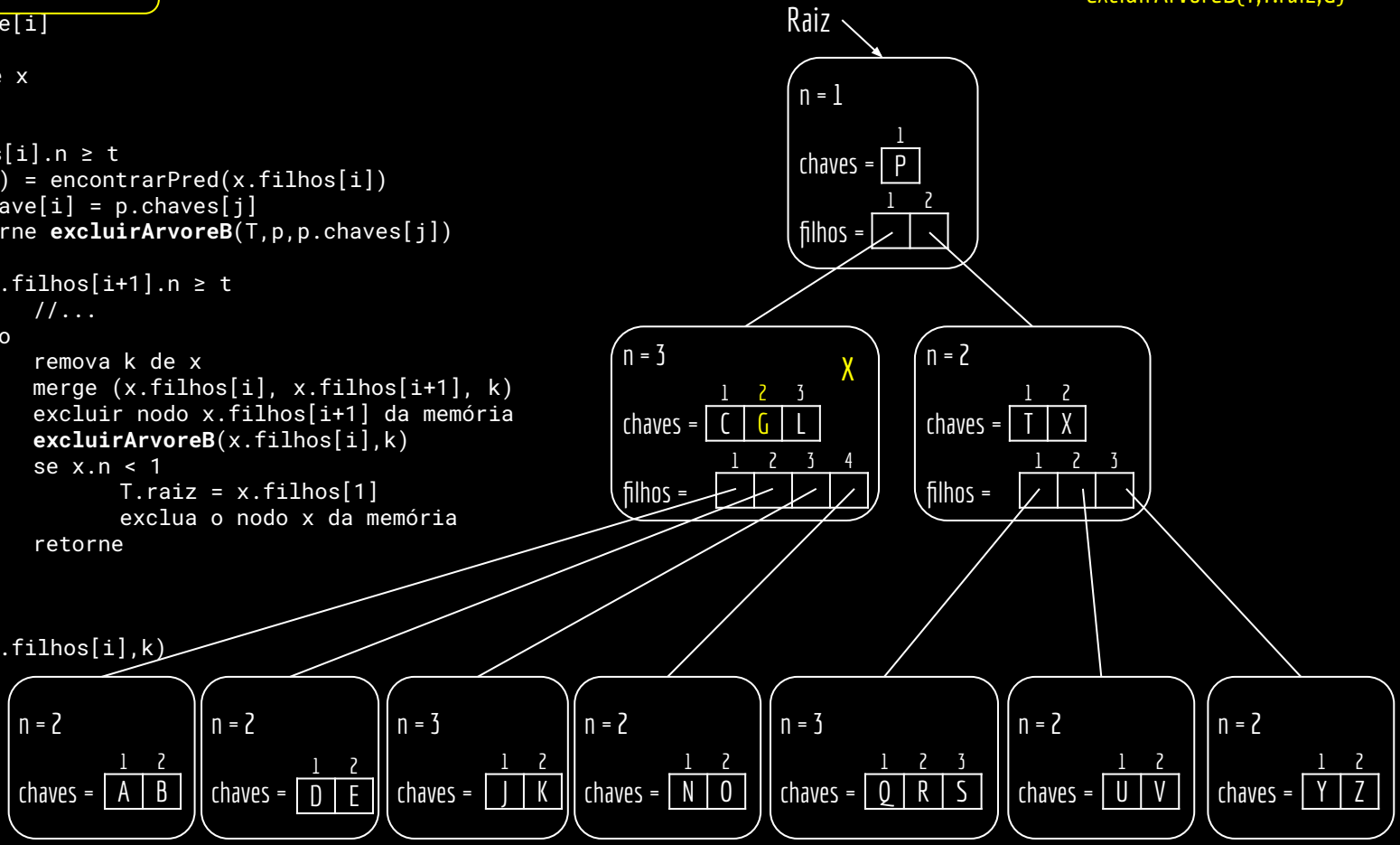
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

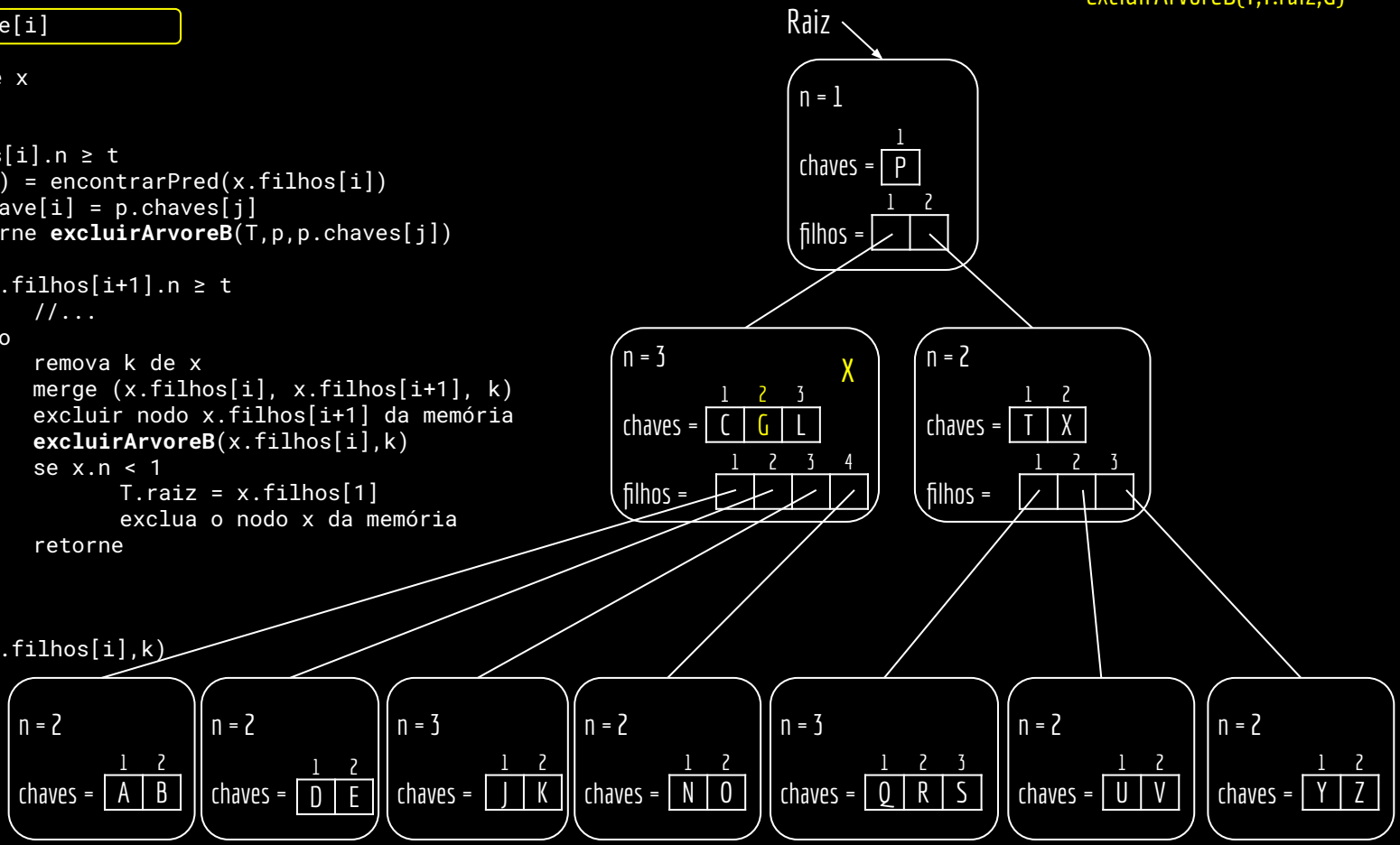
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

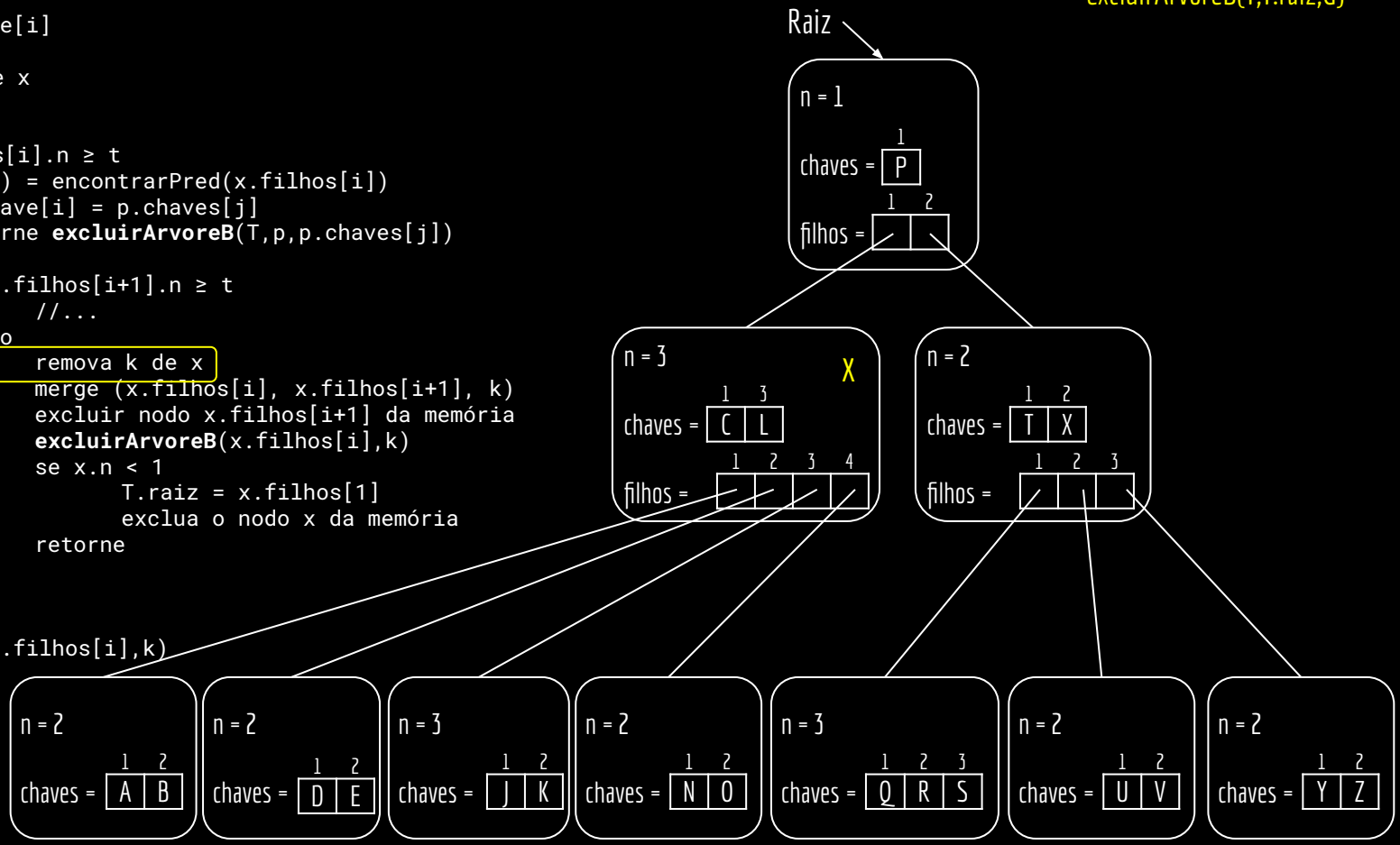
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

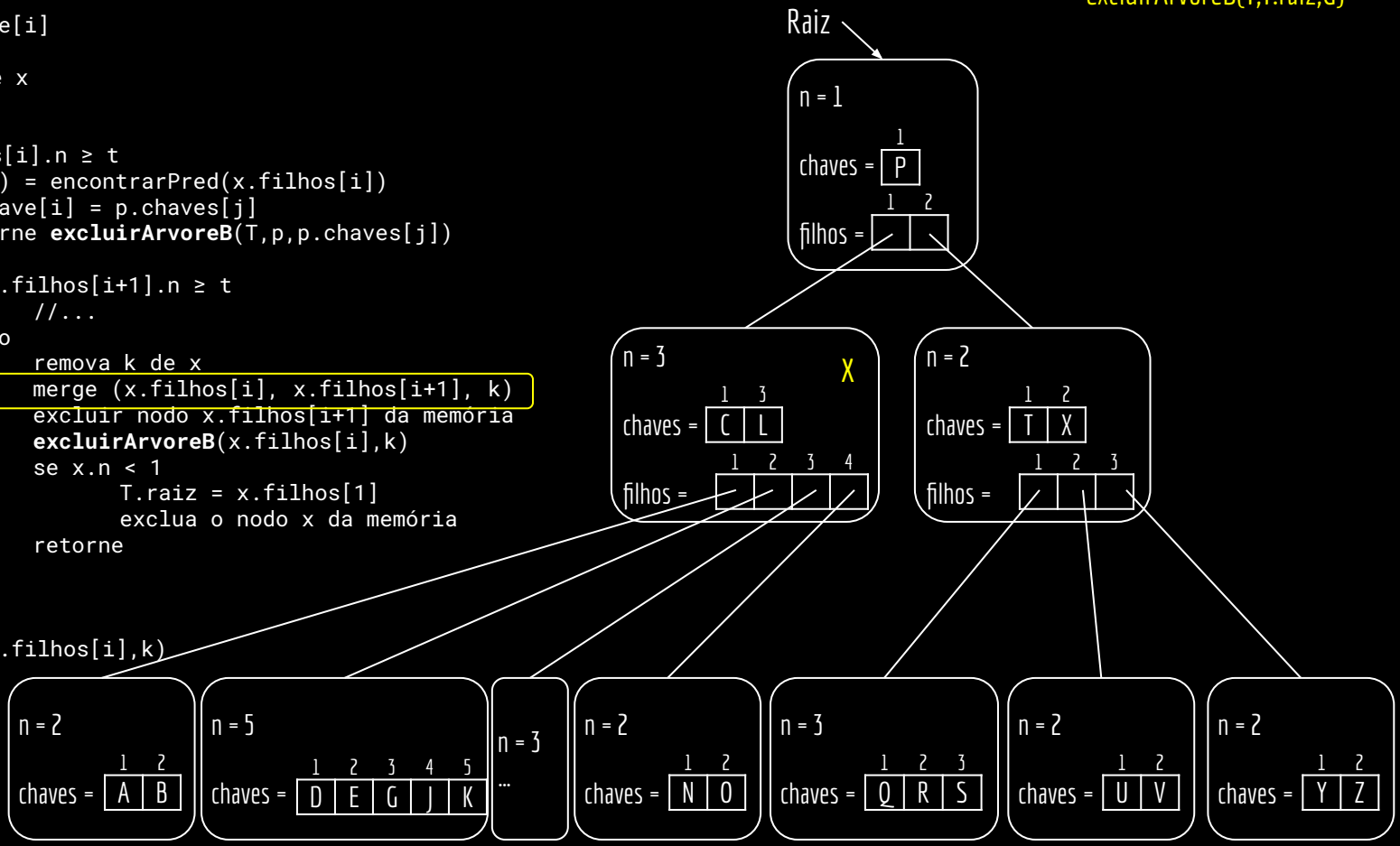
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

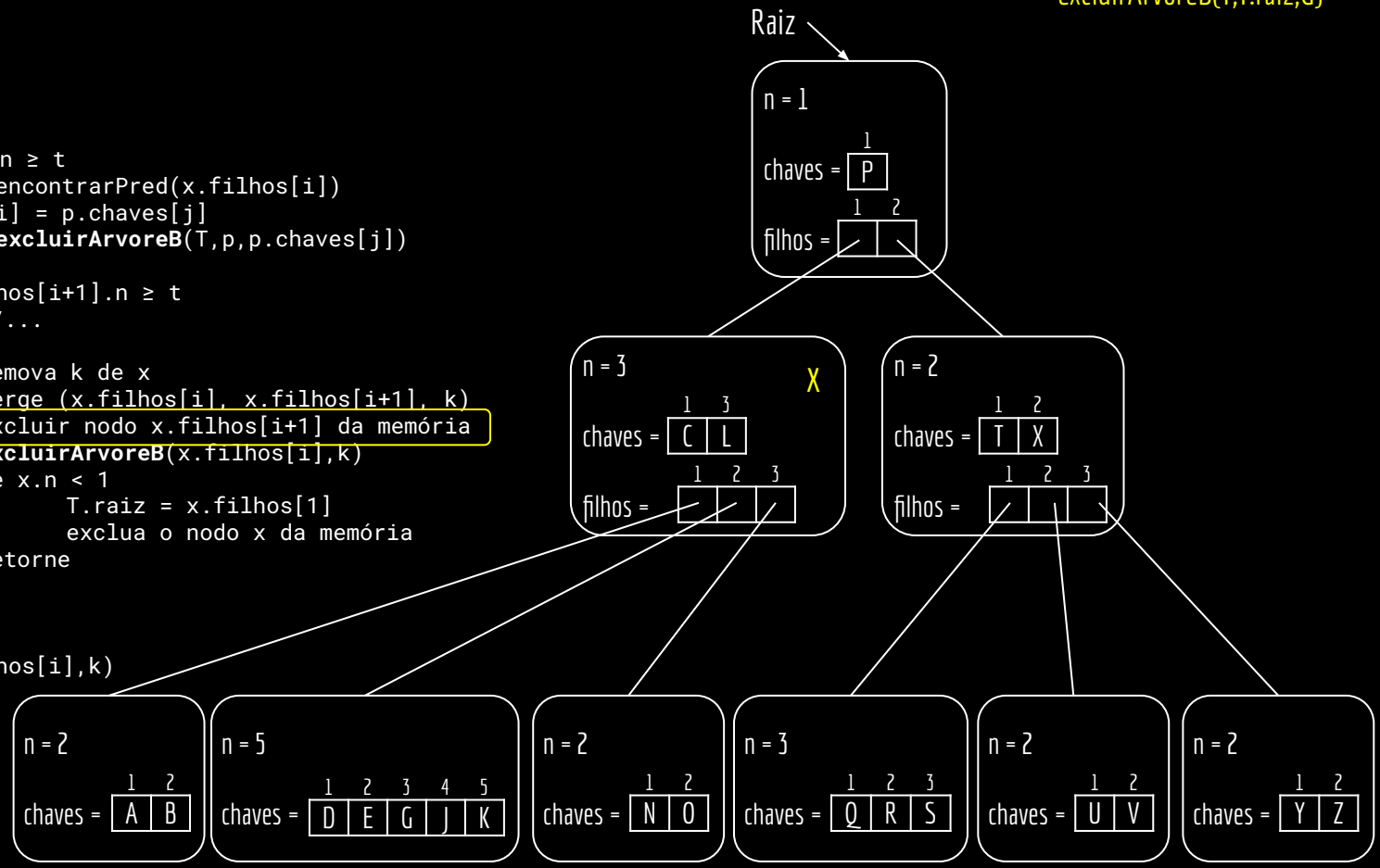
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

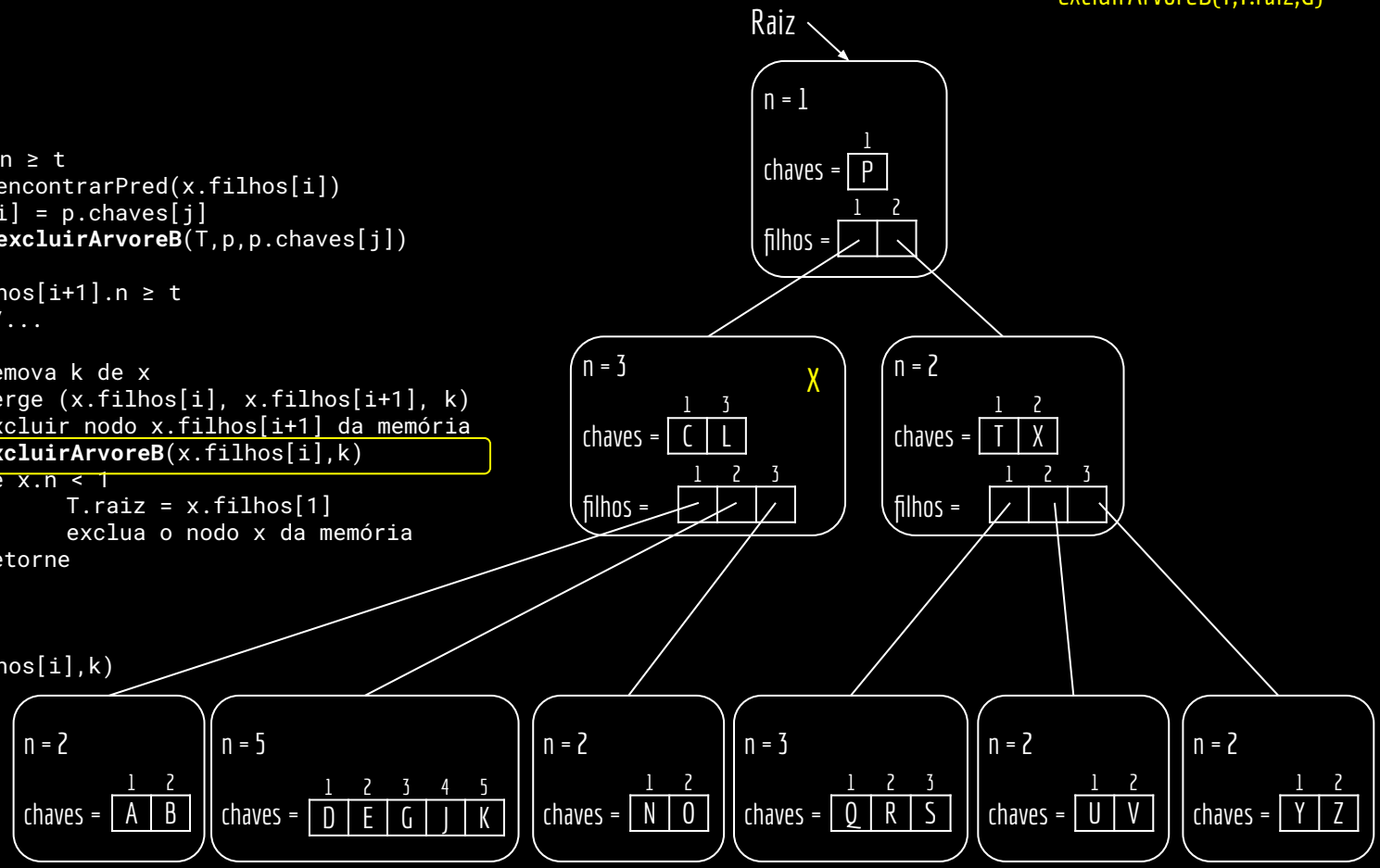
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

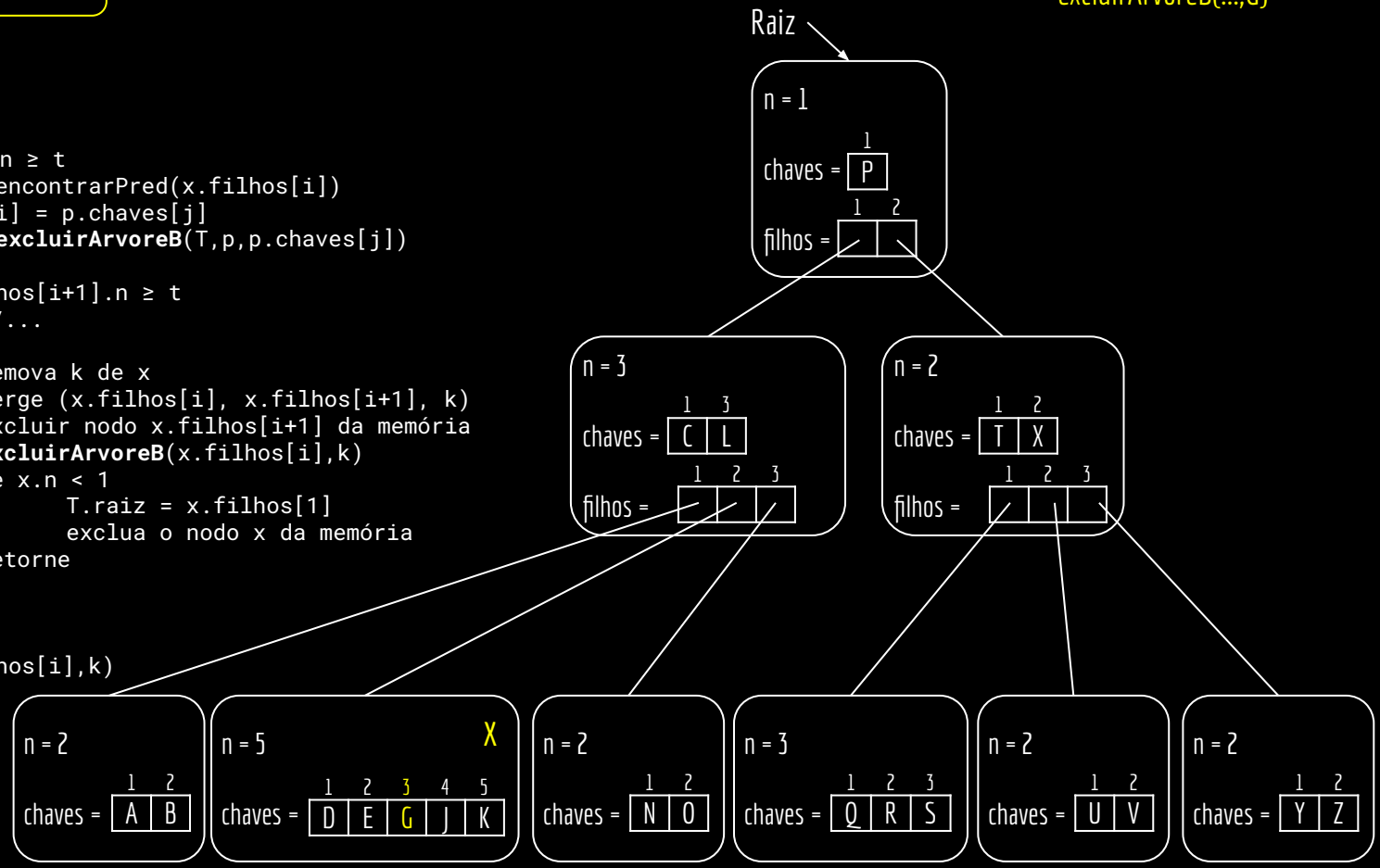
excluirArvoreB(T,T.raiz,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

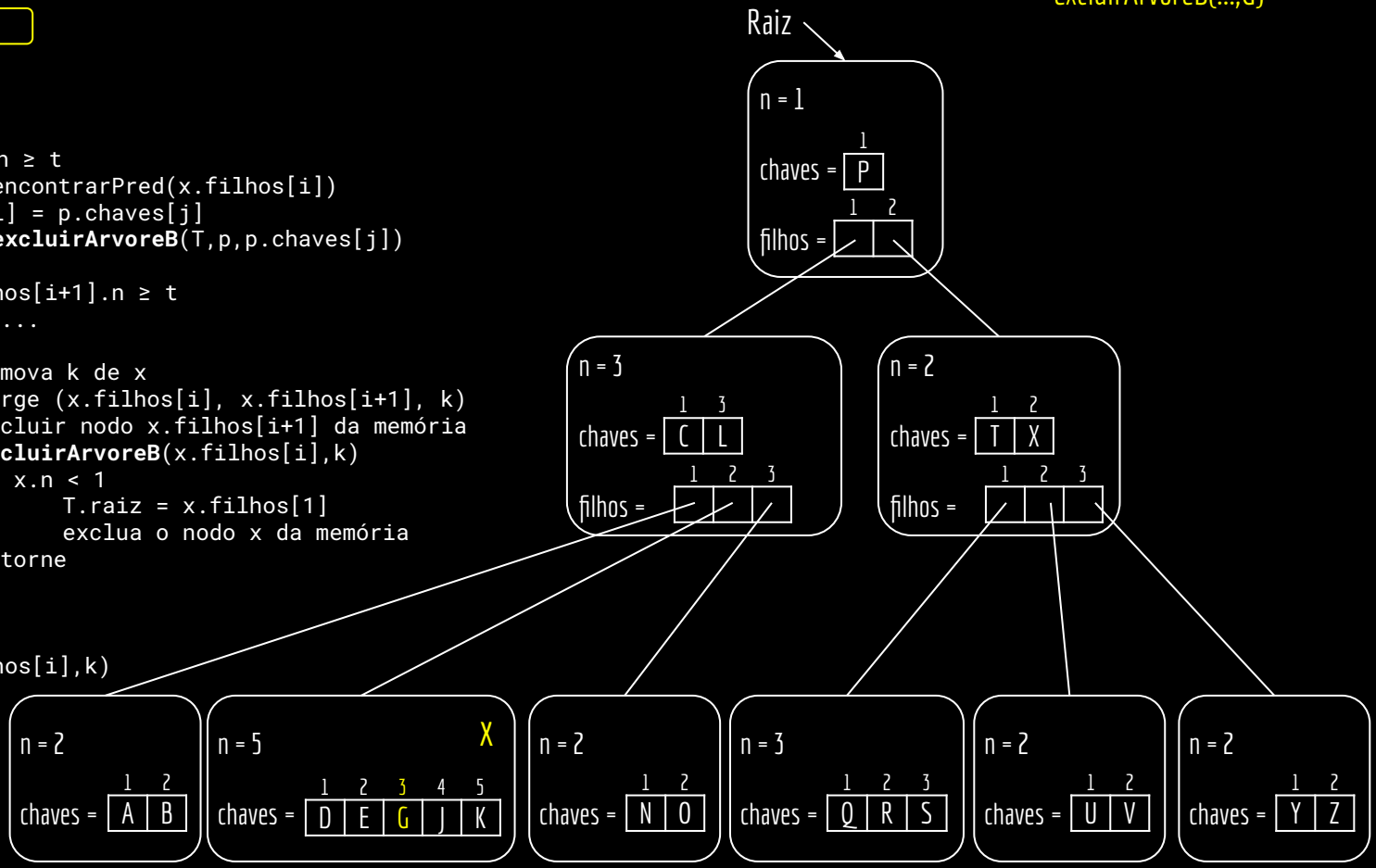
excluirArvoreB(...,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

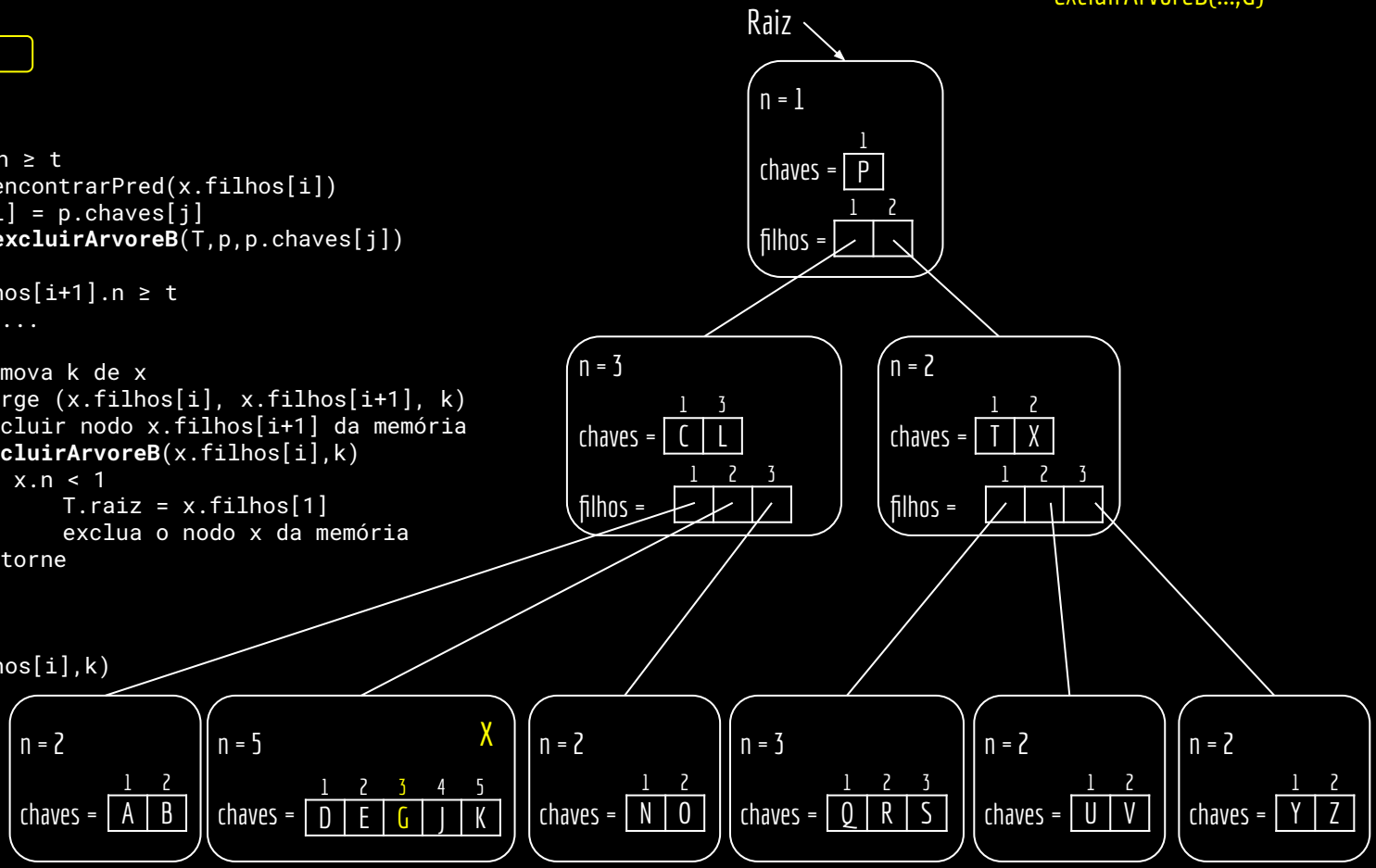
excluirArvoreB(...,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

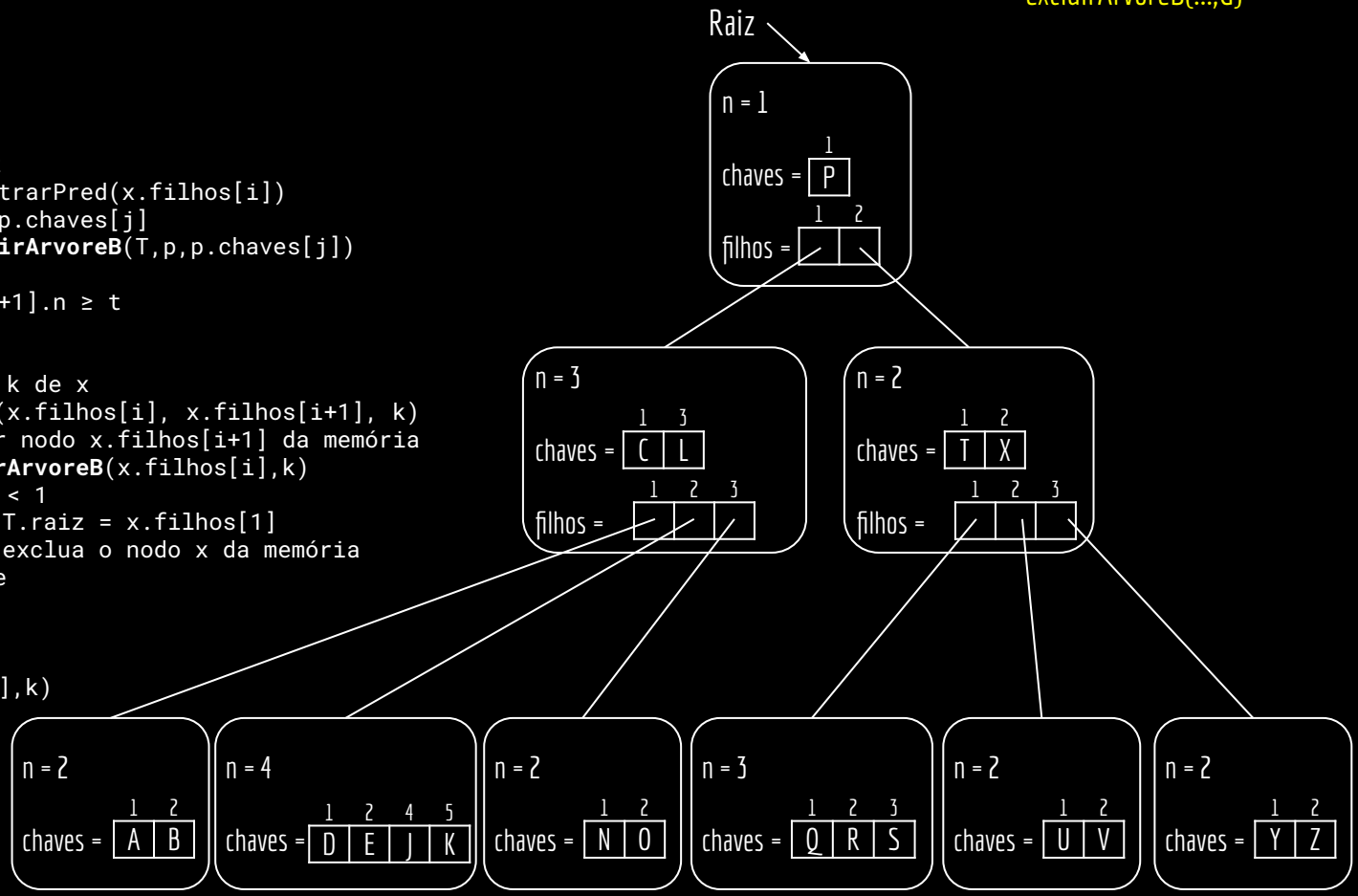
excluirArvoreB(...,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

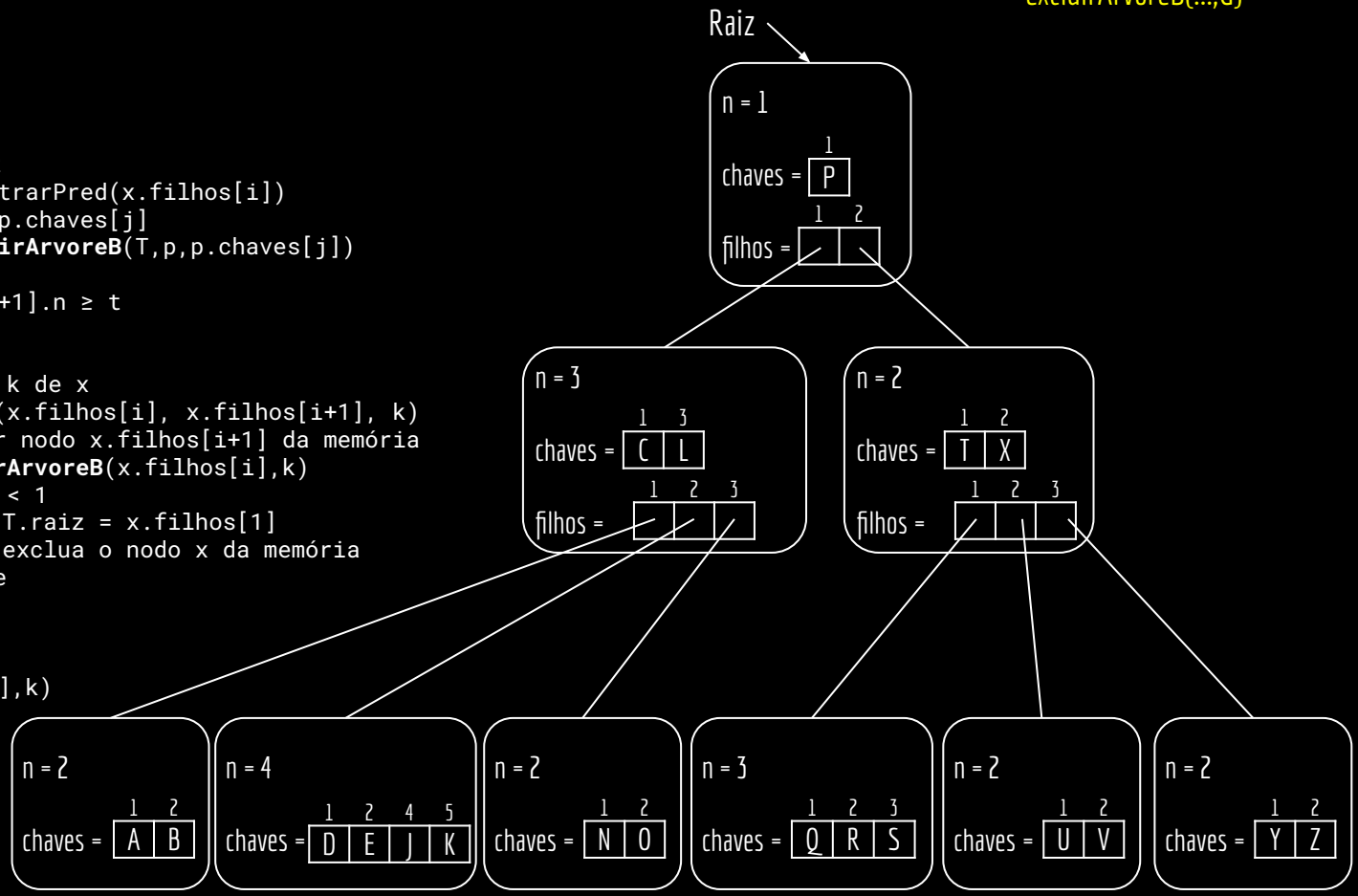
excluirArvoreB(...,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

excluirArvoreB(...,G)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i] //chave encontrada
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x

retorne excluirArvoreB(T,x.filhos[i],k)
```

Caso 3: Chegamos em um nodo interno que não possui k. Antes de continuar a busca, garantir que todo nodo visitado possui pelo menos t chaves.

Caso 3a. O próximo nodo x.filhos[i] possui t-1 chaves, mas tem um irmão imediato com pelo menos t chaves. Mova uma chave de x para x.filhos[i], e transfira uma chave de um dos irmãos para x.

```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
     i = i+1
se i ≤ x.n e k == x.chave[i] //chave encontrada
     //…
senão
     se x é folha
          retorne //chave não encontrada
     senão
          se x.filhos[i].n < t
               b = irmaoImediatoComMaisChaves(x,i)
               se b.n ≥ t
                    passar x.chaves[i] para x.filhos[i]
                    se b era irmão direito
                         passar b.chaves[1] para x
                    senão
                         passar b.chaves[b.n] para x
               senão
                    merge (x.filhos[i], b, x.chaves[i])
                    excluir nodo b da memória
                    remova x.chaves[i] de x
                    se x.n < 1 //x era raiz e pode não ter mais chaves
                         T.raiz = x.filhos[1]
                         exclua o nodo x da memória
                         retorne excluirArvoreB(T,T.raiz,k)
retorne excluirArvoreB(T,x.filhos[i],k)
```

Caso 3: Chegamos em um nodo interno que não possui k. Antes de continuar a busca, garantir que todo nodo visitado possui pelo menos t chaves.

Caso 3a. O próximo nodo x.filhos[i] possui t-1 chaves, mas tem um irmão imediato com pelo menos t chaves. Mova uma chave de x para x.filhos[i], e transfira uma chave de um dos irmãos para x.

Caso 3b. Nenhum irmão imediato possui pelo menos t chaves. Passar os dados (merge) de um irmão imediato para x.filhos[i], juntamente com a chave x.chaves[i] que será o mediano do novo nodo.

Caso3b-1: x era a raiz e ficou sem nodos.

```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
     i = i+1
se i ≤ x.n e k == x.chave[i] //chave encontrada
     //…
senão
     se x é folha
          retorne //chave não encontrada
     senão
          se x.filhos[i].n < t
               b = irmaoImediatoComMaisChaves(x,i)
               se b.n ≥ t
                    passar x.chaves[i] para x.filhos[i]
                    se b era irmão direito
                         passar b.chaves[1] para x
                    senão
                         passar b.chaves[b.n] para x
               senão
                    merge (x.filhos[i], b, x.chaves[i])
                    excluir nodo b da memória
                    remova x.chaves[i] de x
                    se x.n < 1 //x era raiz e pode não ter mais chaves
                         T.raiz = x.filhos[1]
                         exclua o nodo x da memória
                         retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

Caso 3: Chegamos em um nodo interno que não possui k. Antes de continuar a busca, garantir que todo nodo visitado possui pelo menos t chaves.

Caso 3a. O próximo nodo x.filhos[i] possui t-1 chaves, mas tem um irmão imediato com pelo menos t chaves. Mova uma chave de x para x.filhos[i], e transfira uma chave de um dos irmãos para x.

Caso 3b. Nenhum irmão imediato possui pelo menos t chaves. Passar os dados (merge) de um irmão imediato para x.filhos[i], juntamente com a chave x.chaves[i] que será o mediano do novo nodo.

Caso3b-1: x era a raiz e ficou sem nodos.

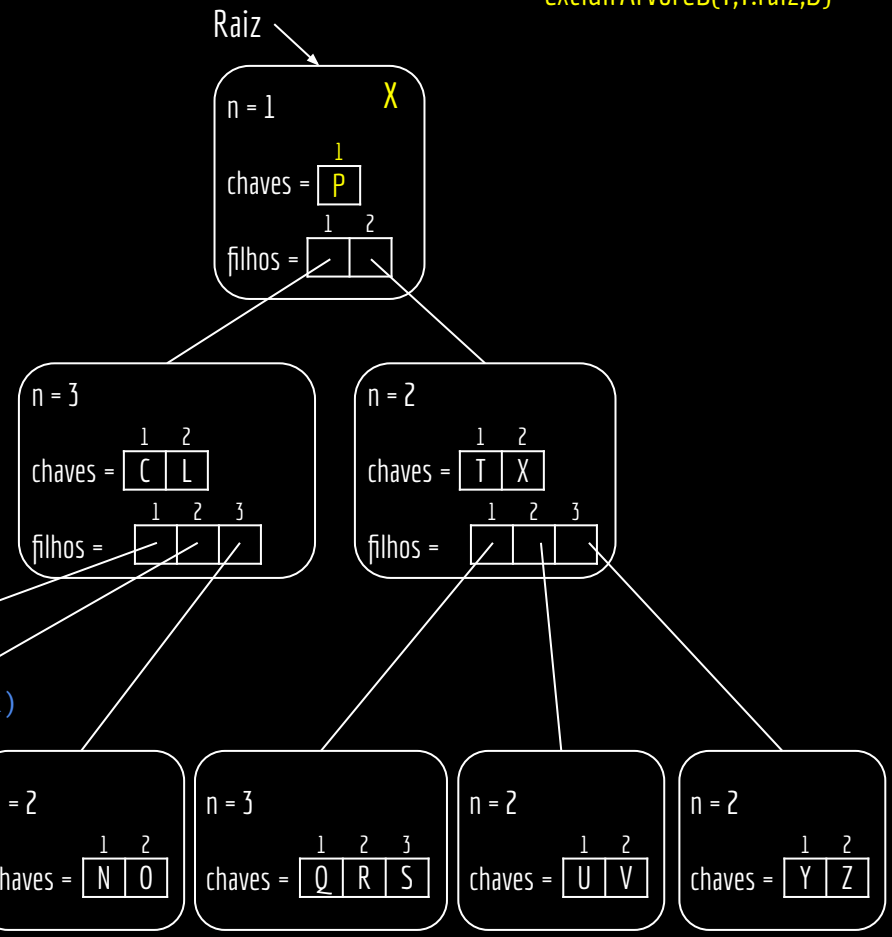
Caso 3c. O nodo x.filhos[i] possui t ou mais chaves.

```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

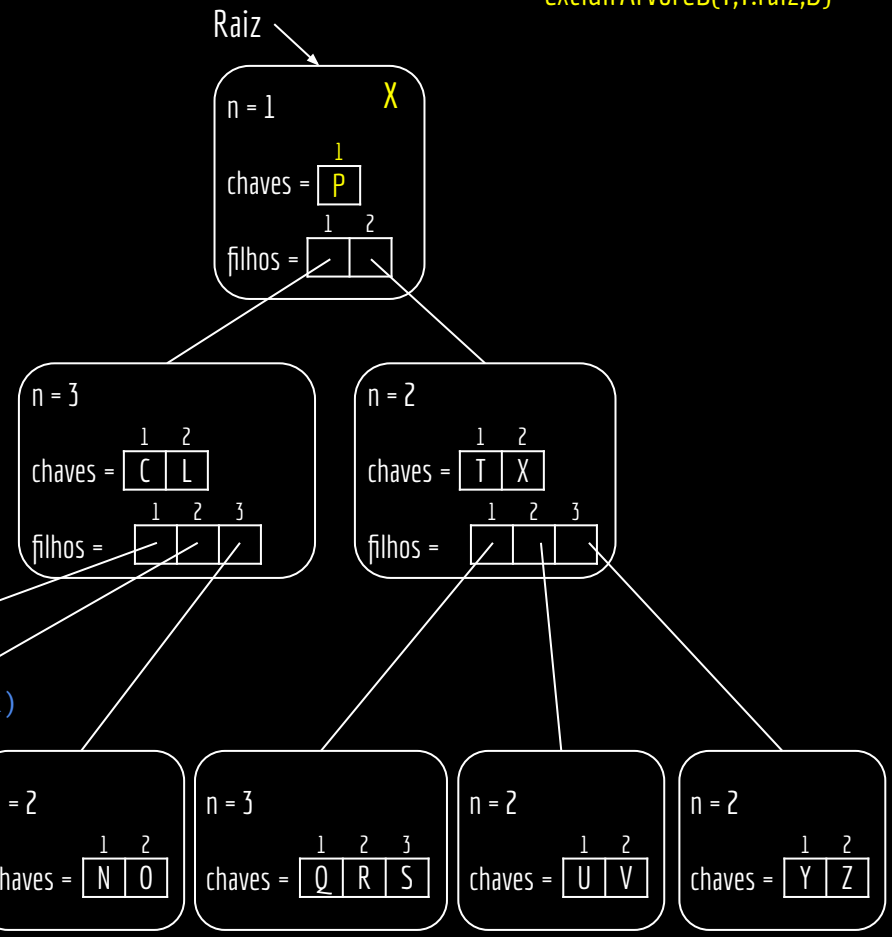


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

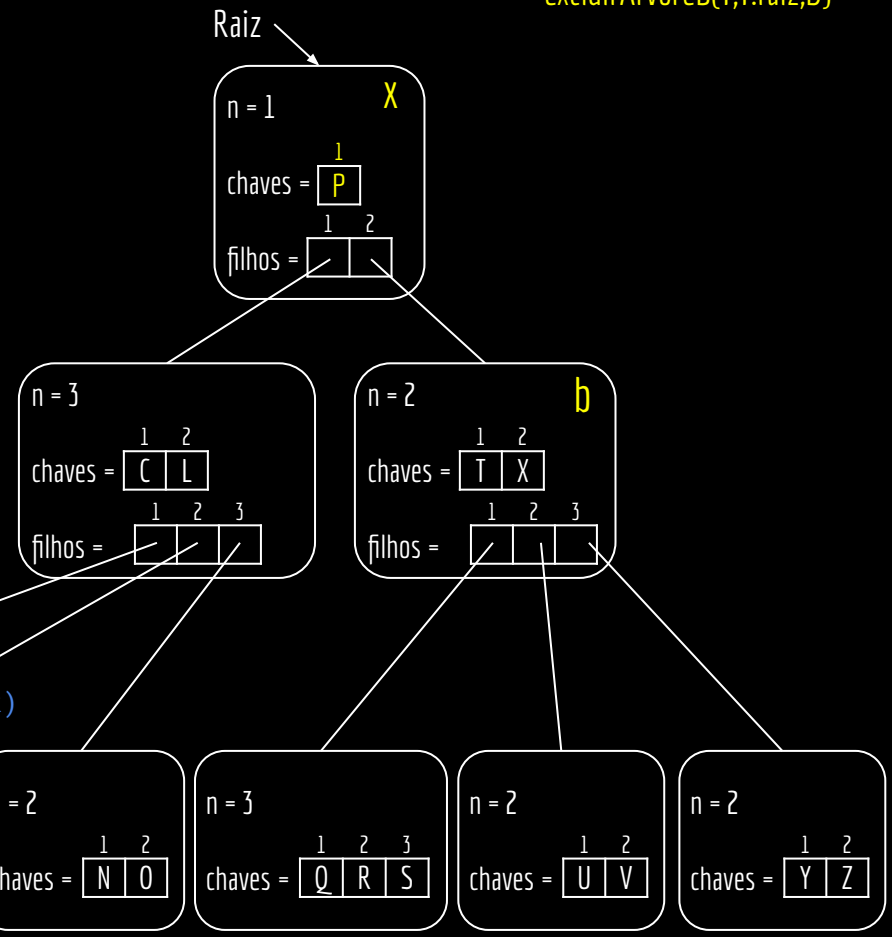


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

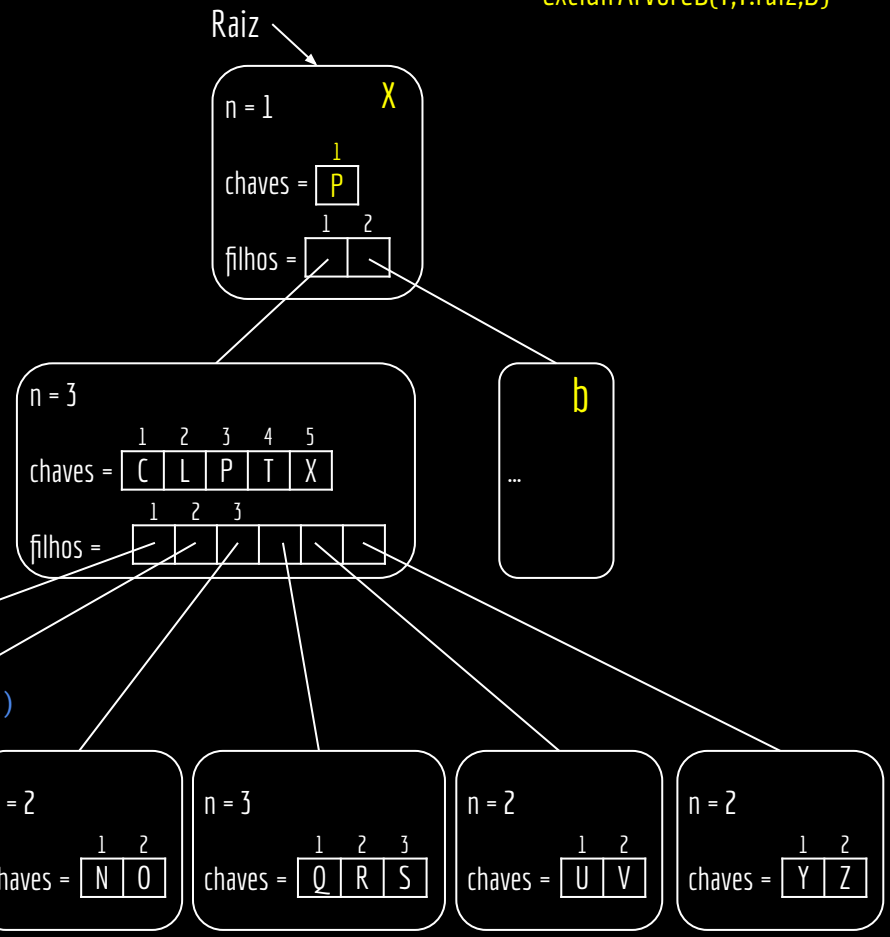


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memoria
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

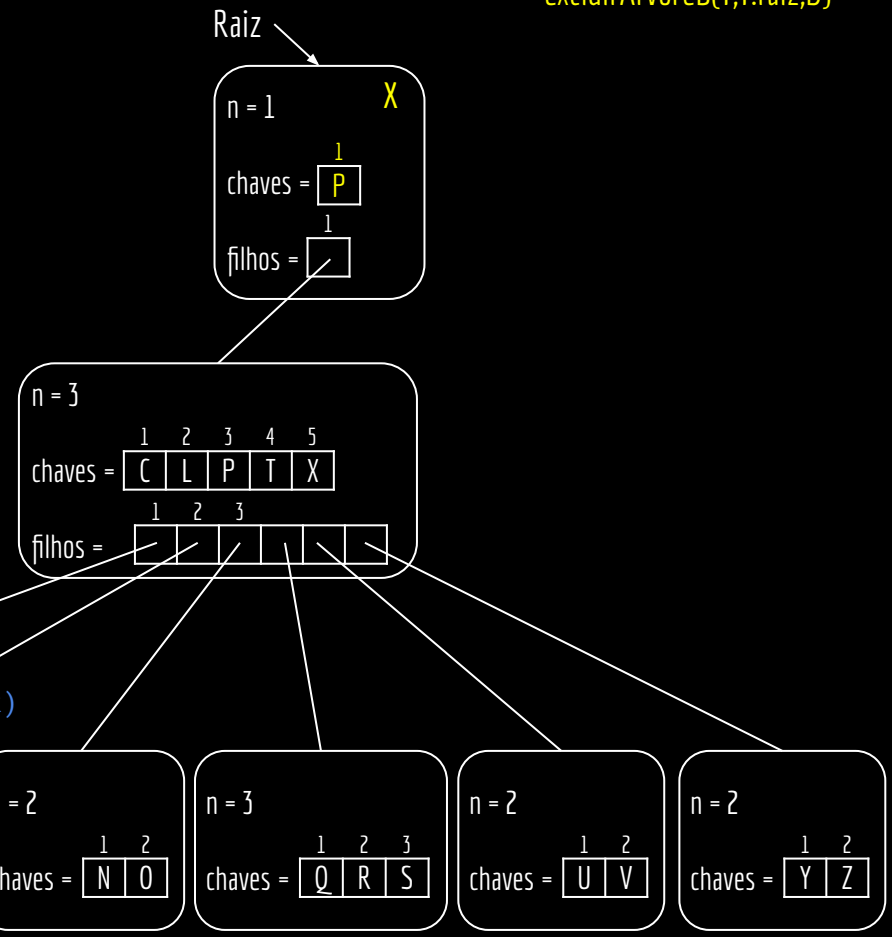


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memoria
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

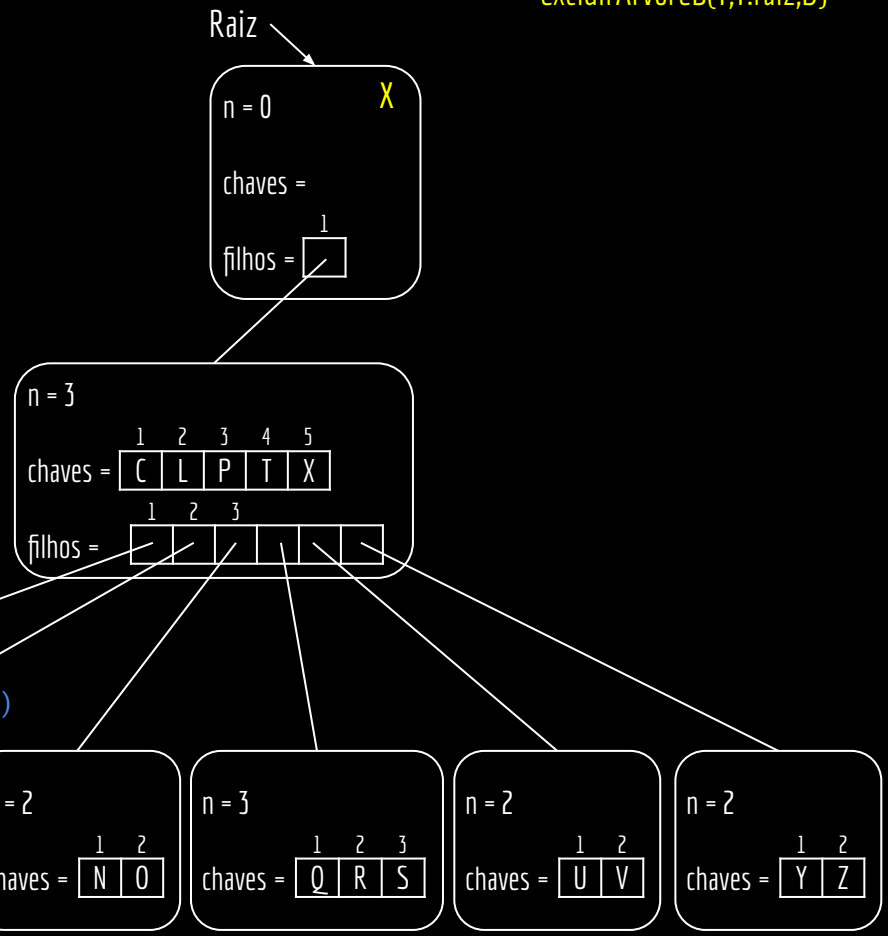


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

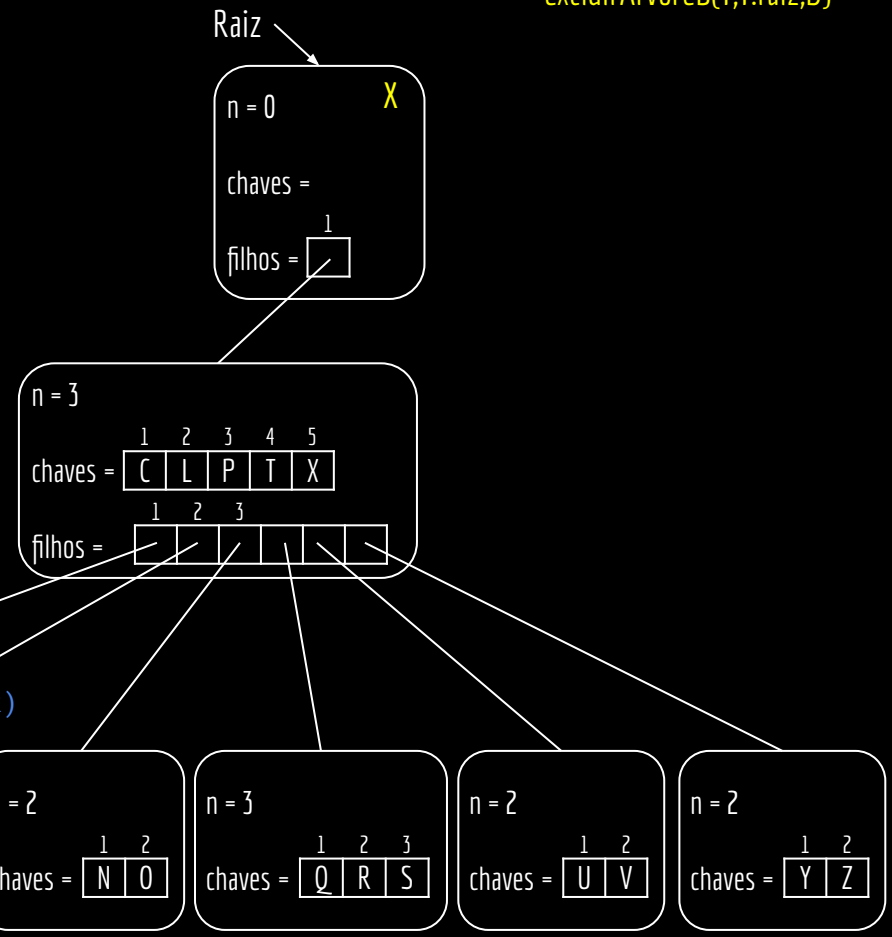


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

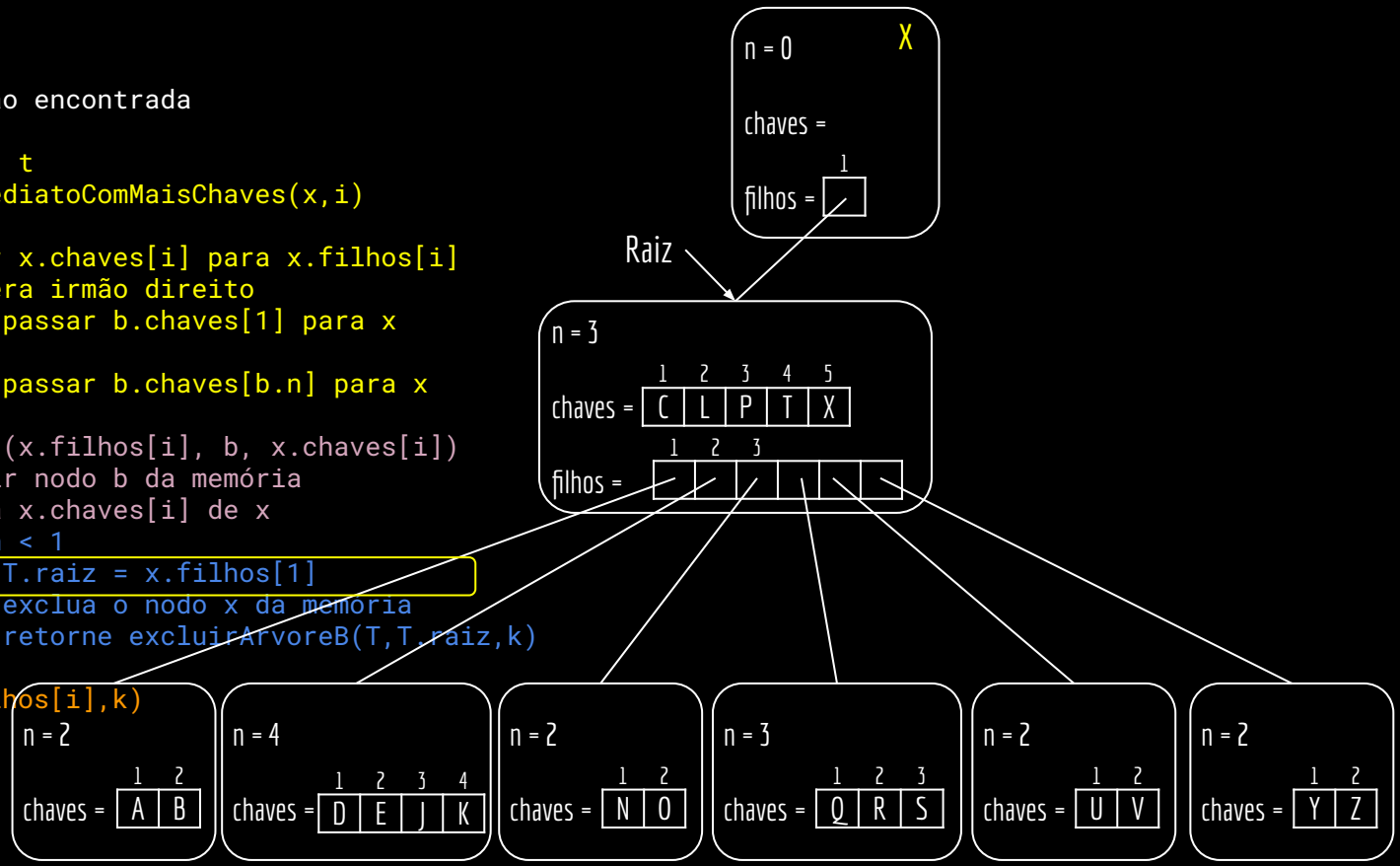


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

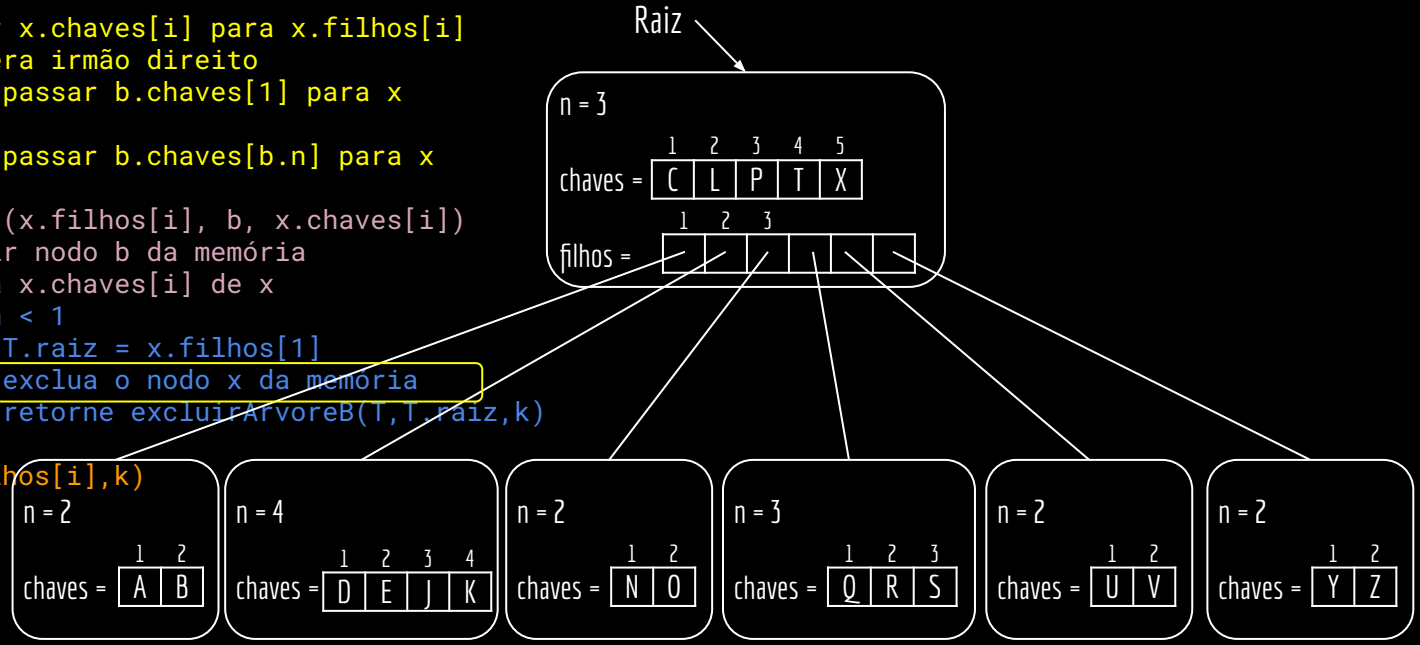


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T,T.raiz,D)

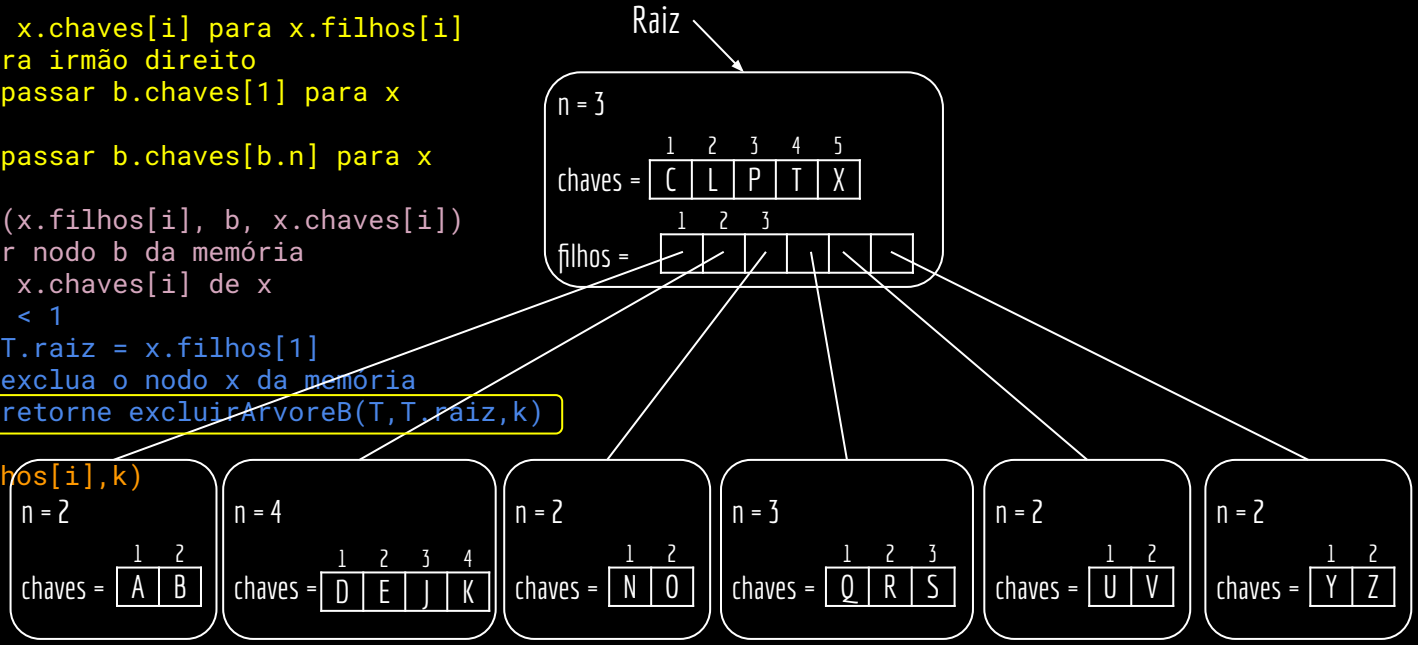


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)

retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

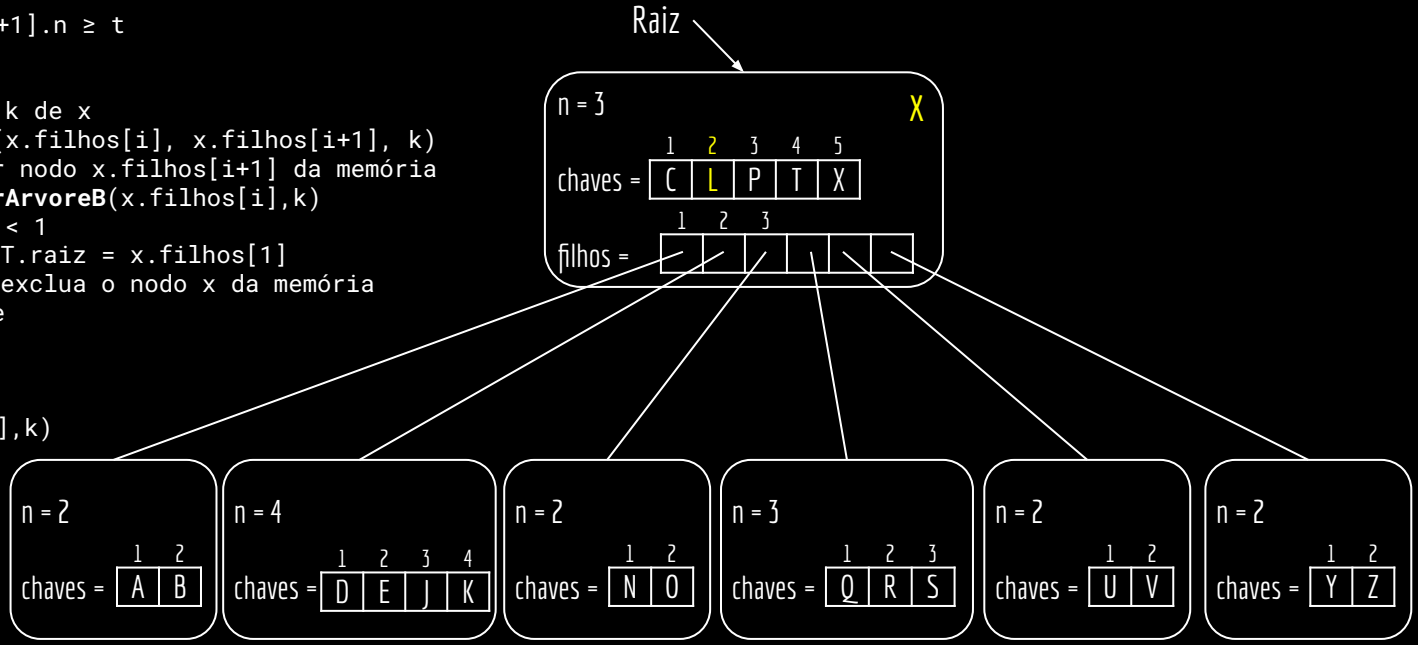
excluirArvoreB(T,T.raiz,D)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

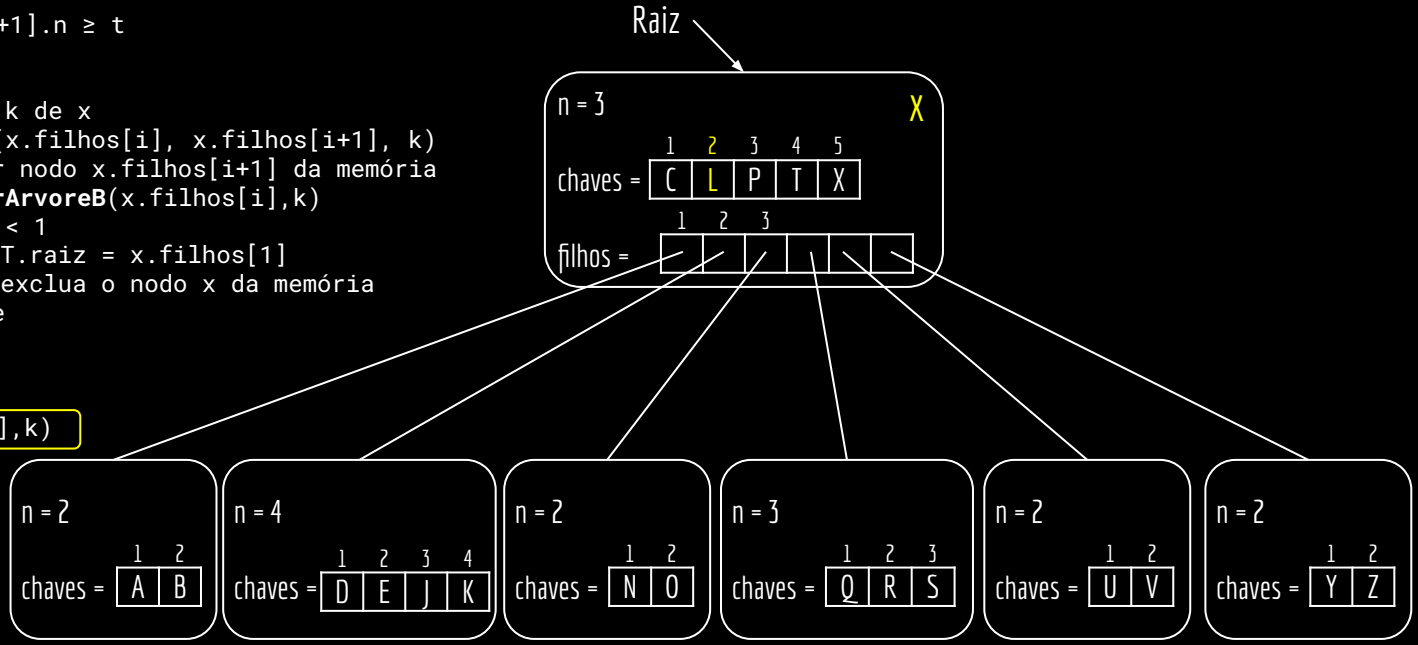
excluirArvoreB(T,T.raiz,D)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

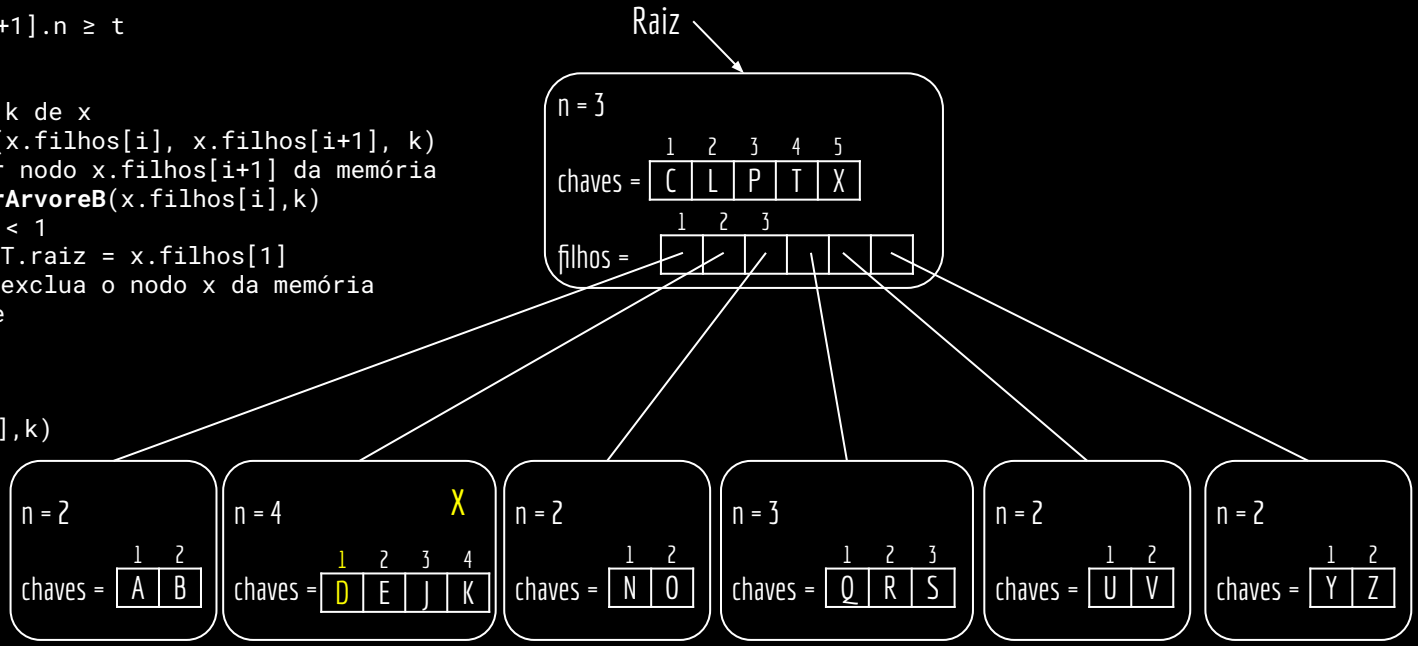
excluirArvoreB(T,T.raiz,D)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

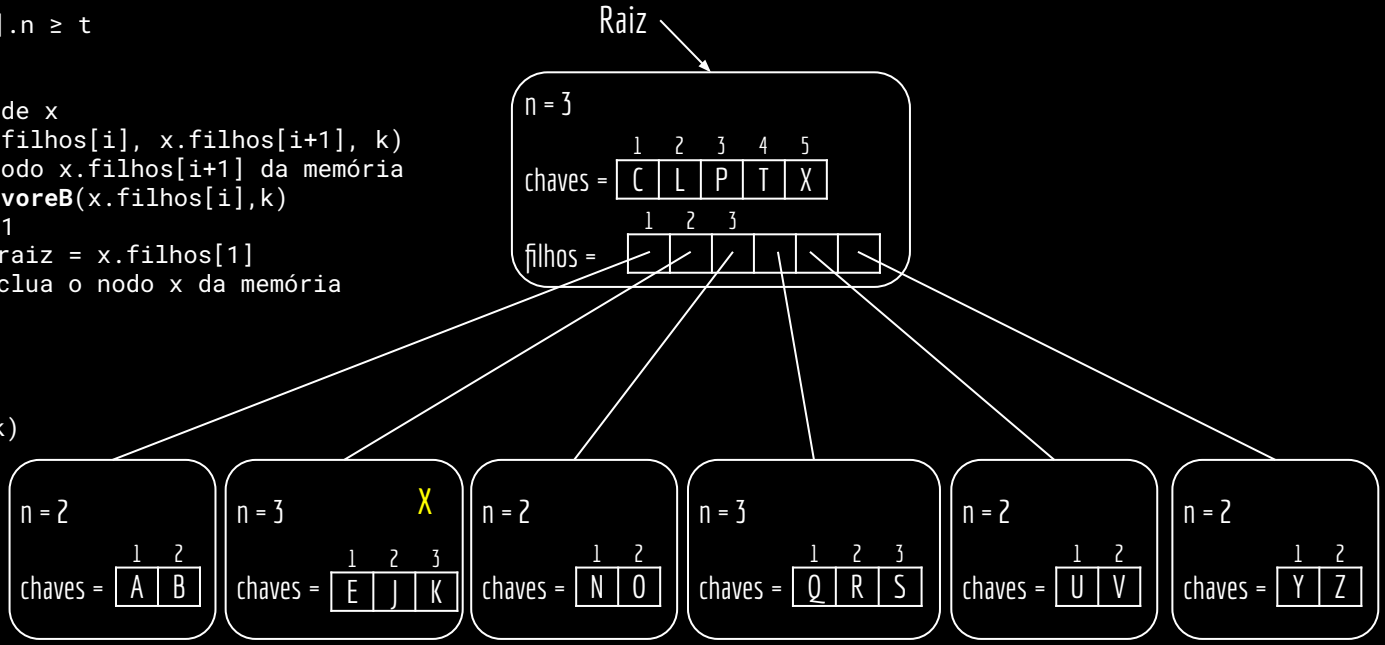
excluirArvoreB(...,D)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

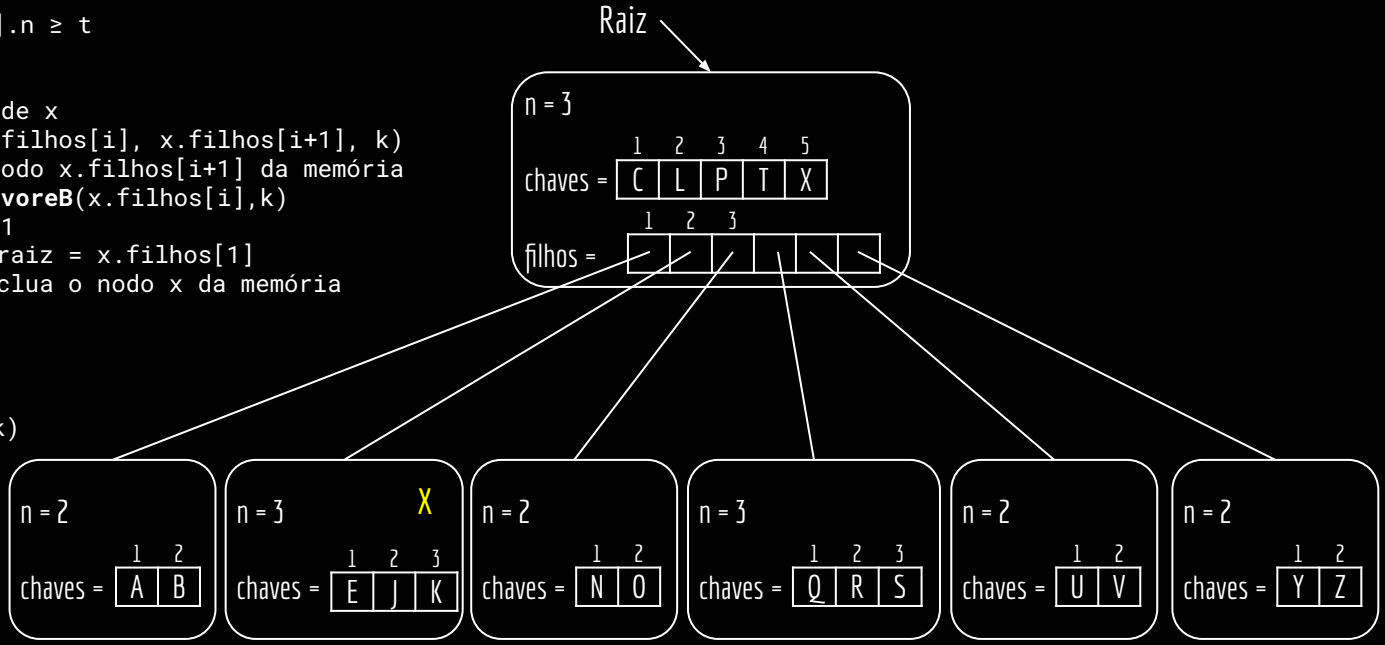
excluirArvoreB(...,D)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

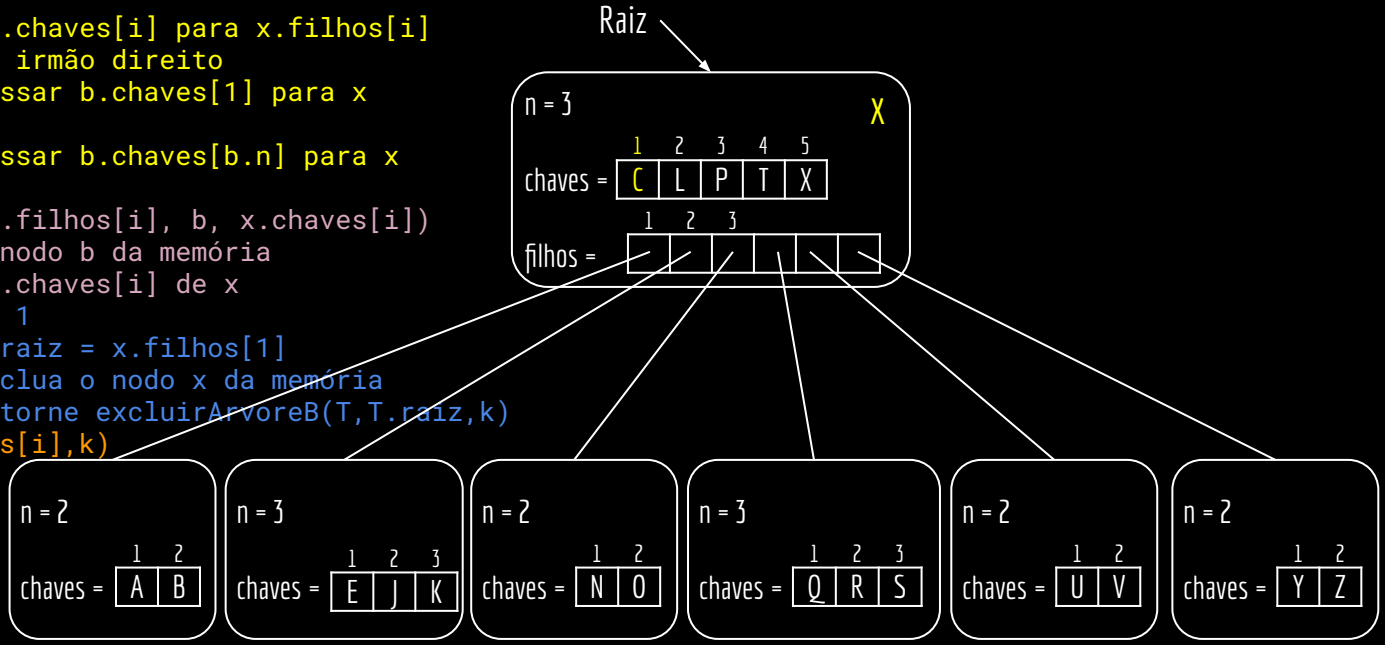
excluirArvoreB(...,D)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)
retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

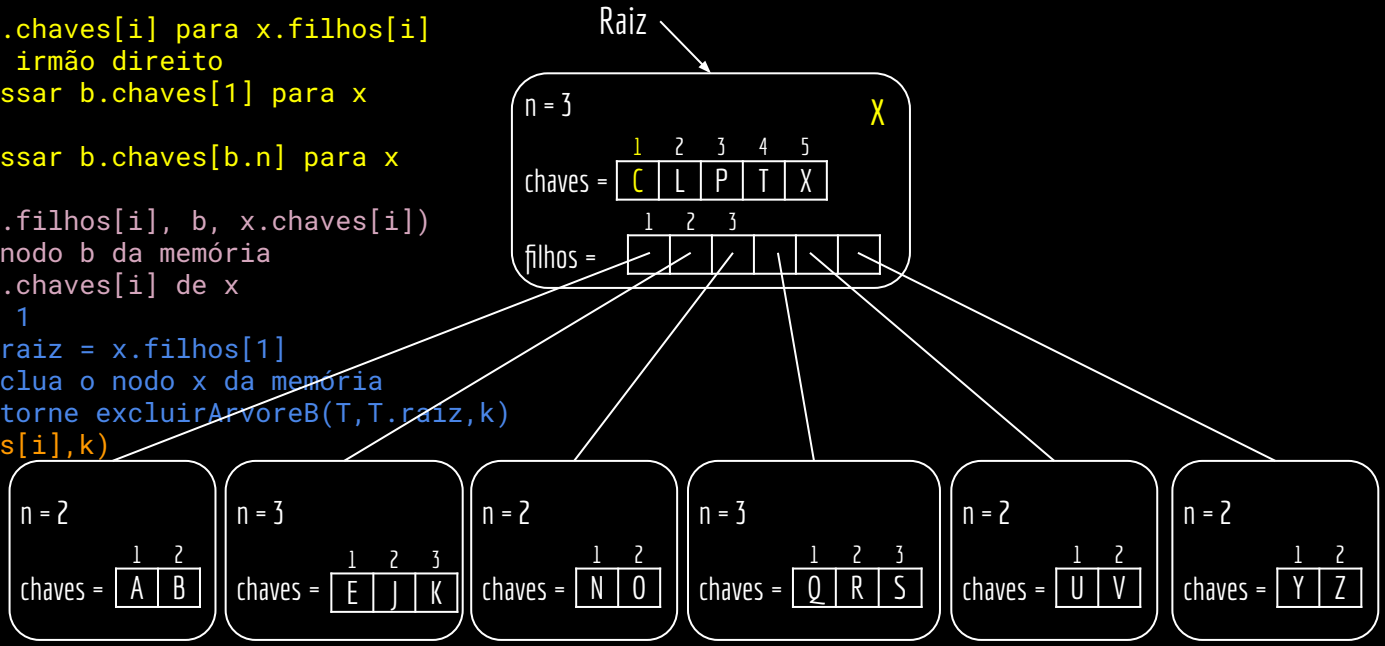
excluirArvoreB(T, T.raiz,B)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)
retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

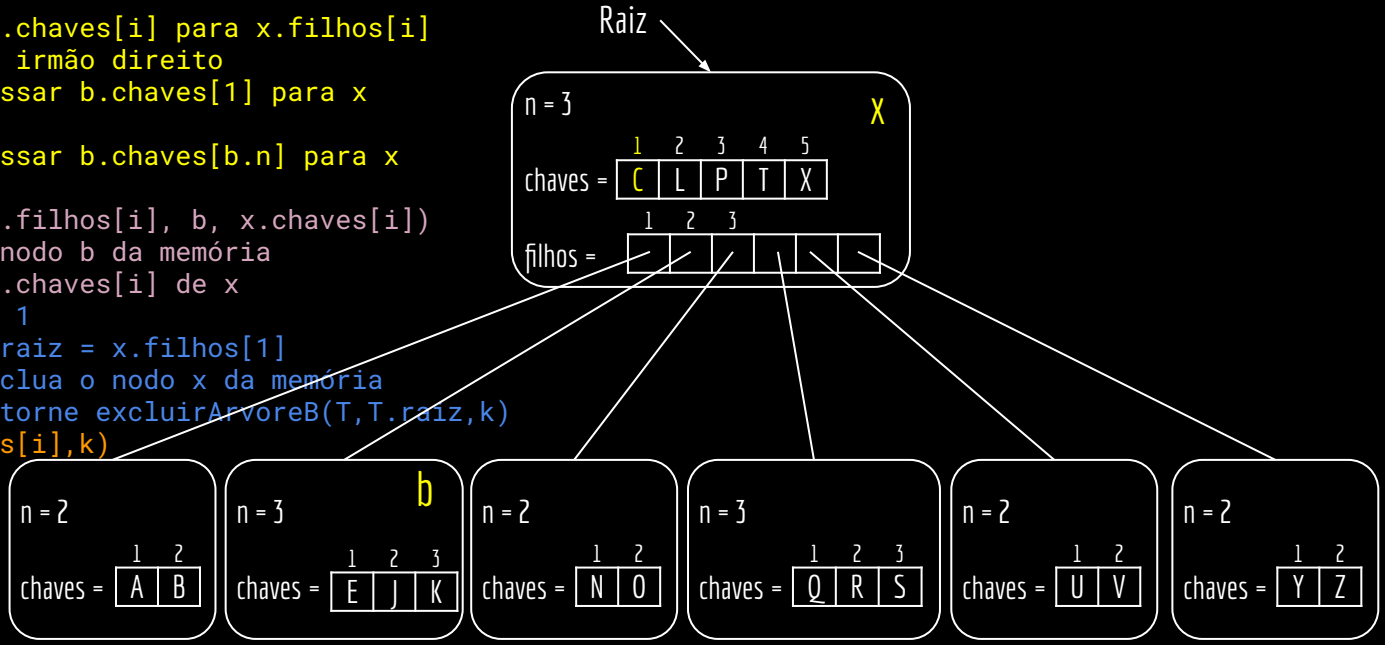
excluirArvoreB(T, T.raiz,B)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)
retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

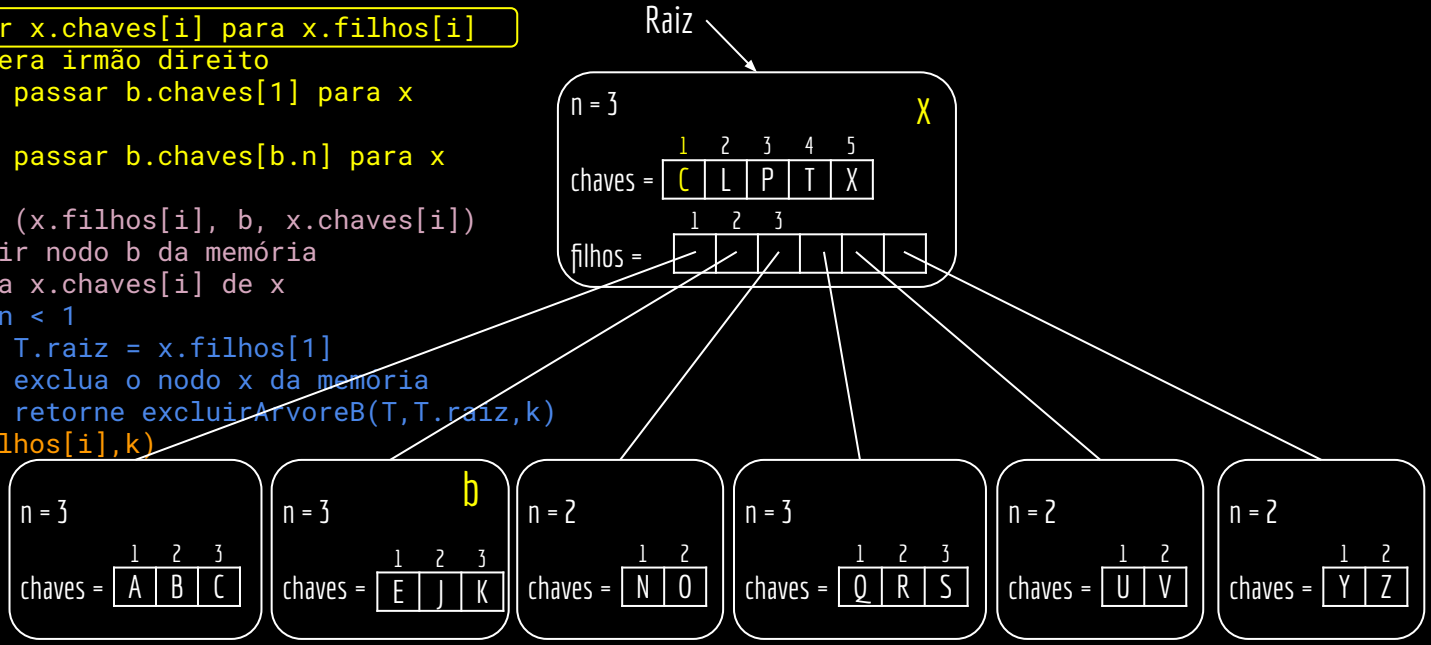
excluirArvoreB(T, T.raiz,B)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memoria
                              retorne excluirArvoreB(T,T.raiz,k)
retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T, T.raiz,B)

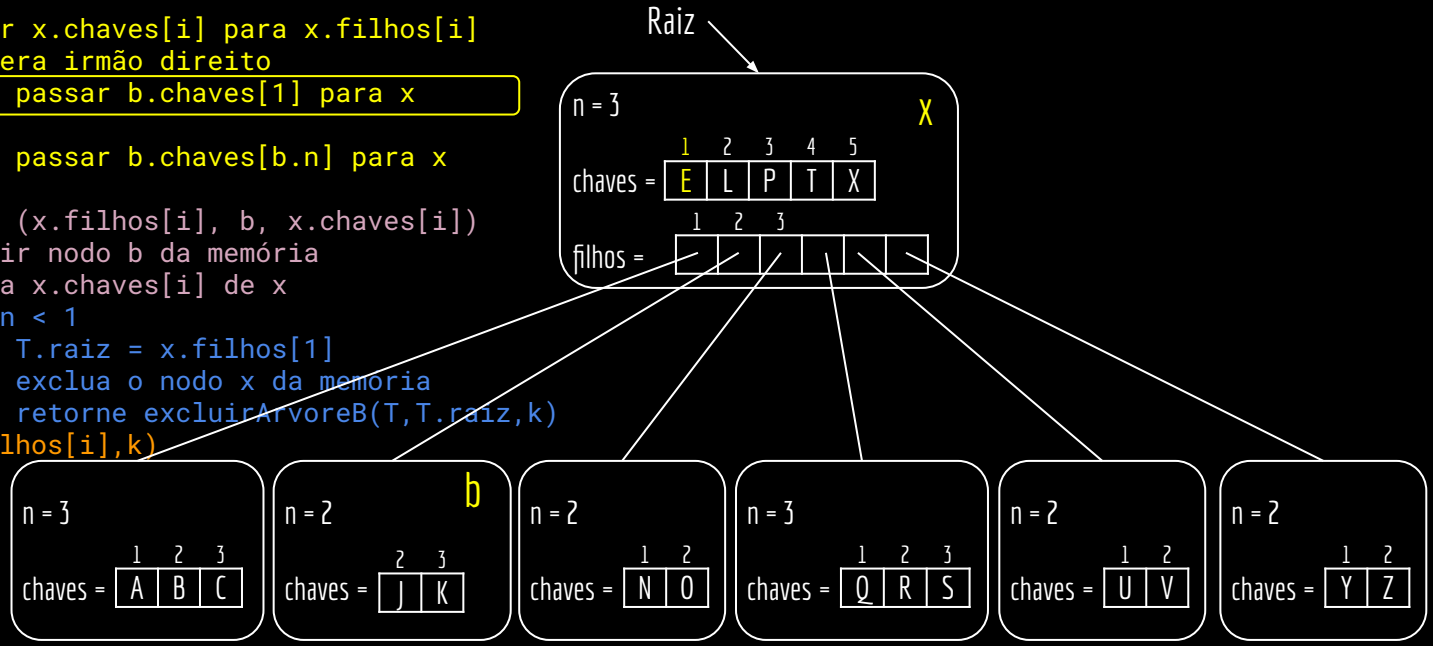


```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                              retorne excluirArvoreB(T,T.raiz,k)
retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

excluirArvoreB(T, T.raiz,B)

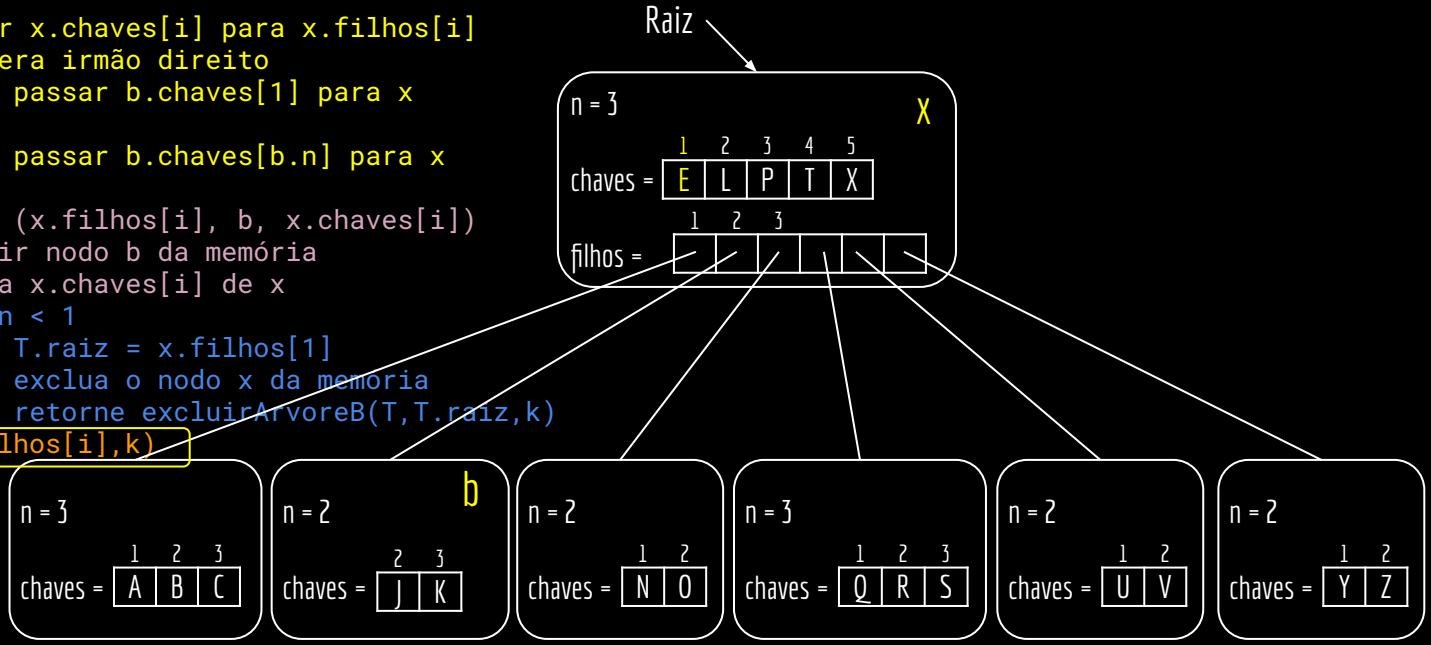
O processo é um pouco mais complicado. Note que o nodo b poderia não ser folha, e precisariamos cuidar dos filhos dele também.



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      //…
senão
      se x é folha
            retorne //chave não encontrada
      senão
            se x.filhos[i].n < t
                  b = irmaoImediatoComMaisChaves(x,i)
                  se b.n ≥ t
                        passar x.chaves[i] para x.filhos[i]
                        se b era irmão direito
                              passar b.chaves[1] para x
                        senão
                              passar b.chaves[b.n] para x
                  senão
                        merge (x.filhos[i], b, x.chaves[i])
                        excluir nodo b da memória
                        remova x.chaves[i] de x
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memoria
                              retorne excluirArvoreB(T,T.raiz,k)
retorne excluirArvoreB(T,x.filhos[i],k)
```

t=3

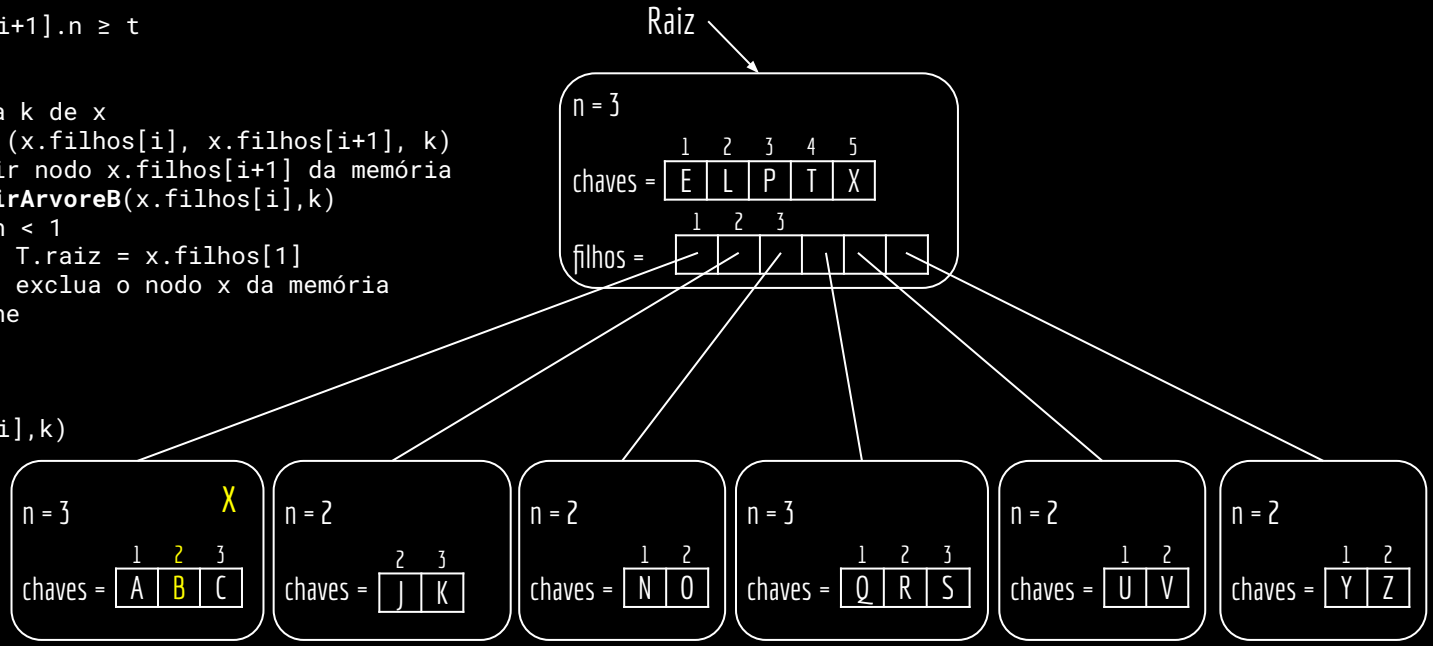
excluirArvoreB(T, T.raiz,B)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

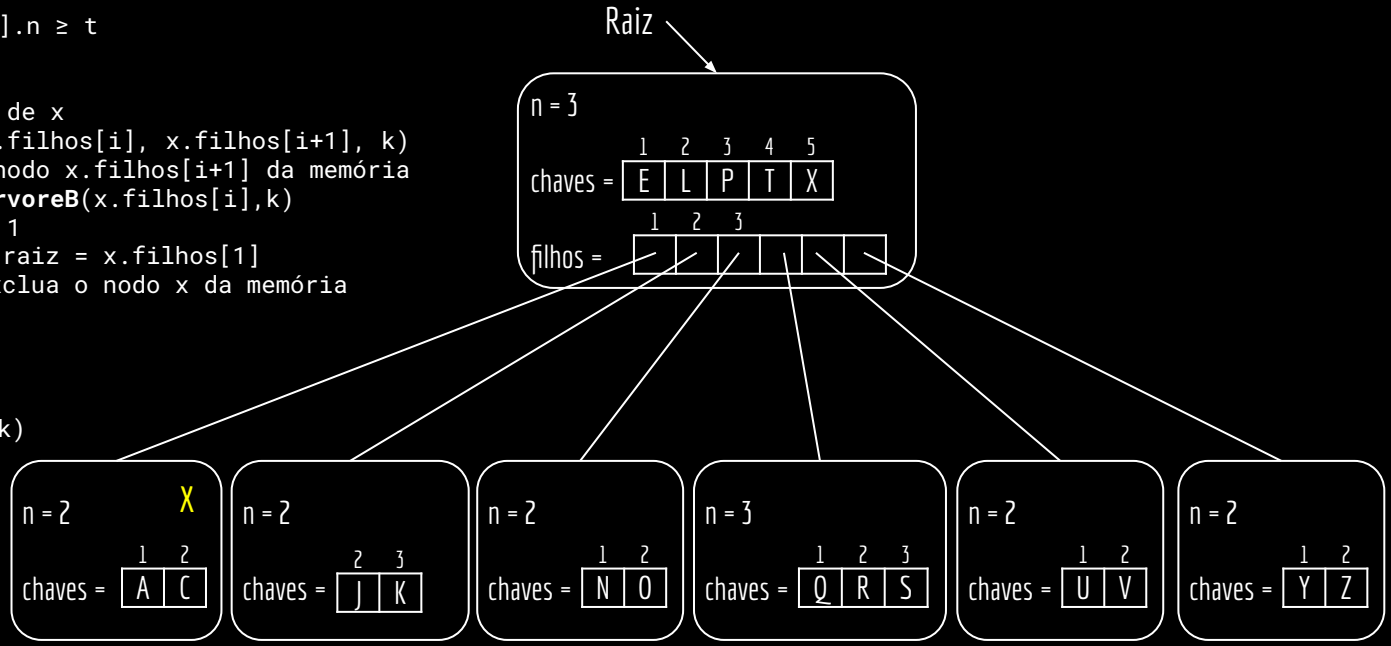
excluirArvoreB(...,B)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

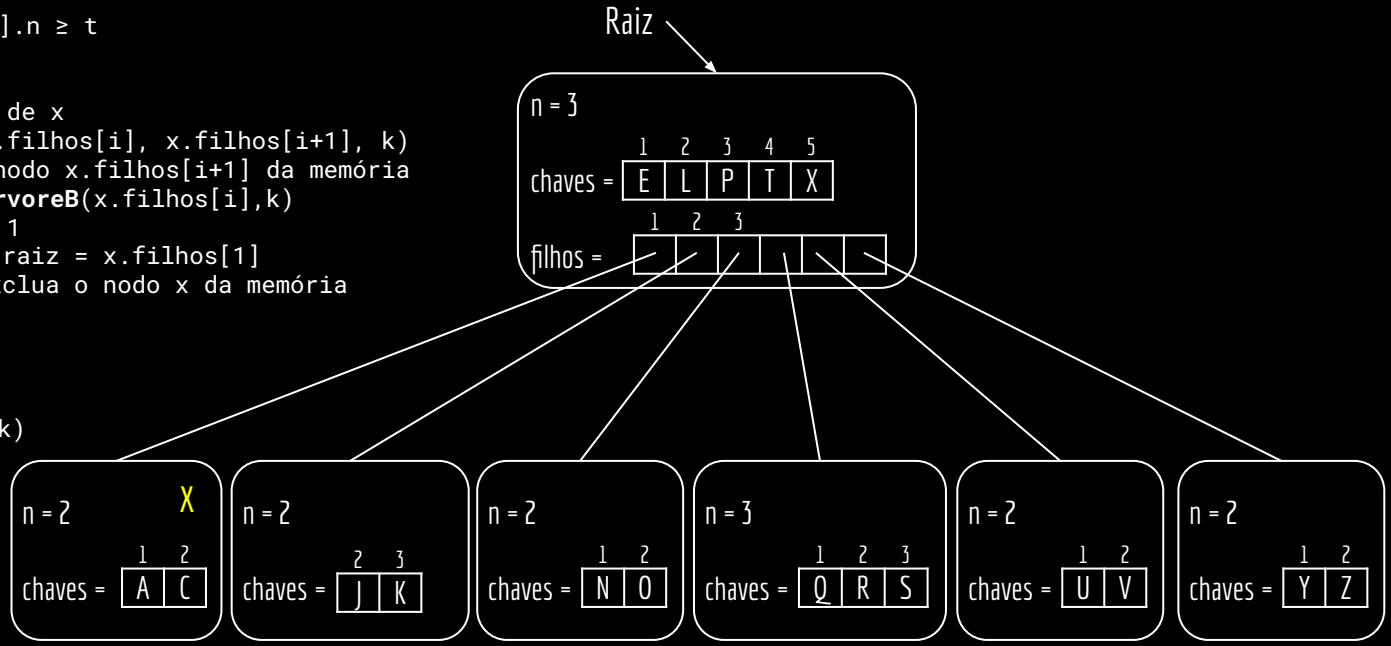
excluirArvoreB(...,B)



```
função excluirArvoreB(T,x,k)
i = 1
enquanto i ≤ x.n e k > x.chave[i]
      i = i+1
se i ≤ x.n e k == x.chave[i]
      se x é folha
            remova k de x
            retorne
      senão
            se x.filhos[i].n ≥ t
                  (p,j) = encontrarPred(x.filhos[i])
                  x.chave[i] = p.chaves[j]
                  retorne excluirArvoreB(T,p,p.chaves[j])
            senão
                  se x.filhos[i+1].n ≥ t
                        //...
                  senão
                        remova k de x
                        merge (x.filhos[i], x.filhos[i+1], k)
                        excluir nodo x.filhos[i+1] da memória
                        excluirArvoreB(x.filhos[i],k)
                        se x.n < 1
                              T.raiz = x.filhos[1]
                              exclua o nodo x da memória
                        retorne
senão
      se x é folha
            retorne
retorne excluirArvoreB(x.filhos[i],k)
```

t=3

excluirArvoreB(...,B)

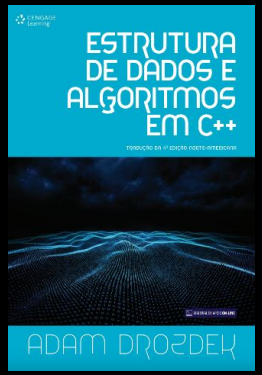


Exercícios

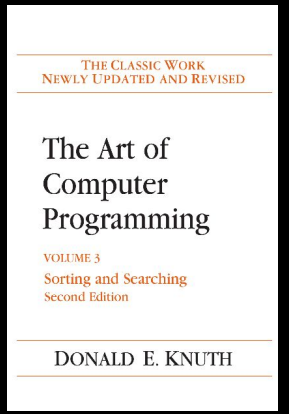
1. Leia em Cormen et al. (2022) ou (2012) sobre o procedimento de exclusão de uma chave na Árvore B.
    a. Compare a discussão apresentada com o algoritmo dado em aula.
    b. Baseado na discussão do livro, e defina sua própria versão do algoritmo de exclusão.
2. Complete o algoritmo de exclusão, de forma que os nodos sejam carregados/descarregados nos pontos necessários.
3. Implemente os algoritmos em C. Assuma que todos os nodos estão na memória principal.

Referências

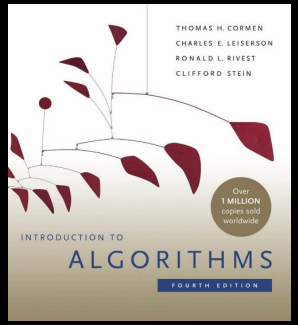
Estrutura de Dados e Algoritmos em C++. A. Drozdek. 4a ed. 2016.



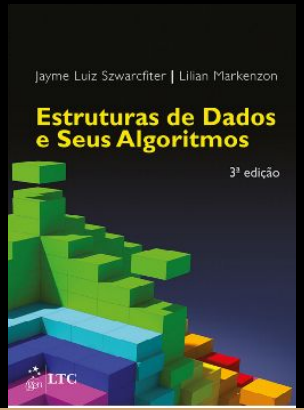
Knuth, D. The Art of Computer Programming: Volume 3: Sorting and Searching. 1998.



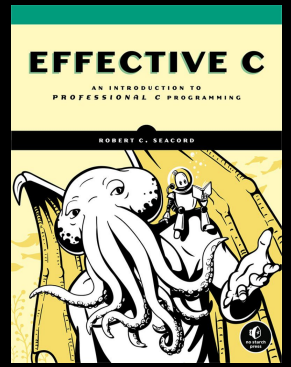
T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 4a ed. 2022.



Szwarcfiter, Markenzon, L. Estruturas de dados e seus algoritmos.2010.



Seacord, R. C. Effective C: An ntroduction to Professional C Programming. 2020.



Licença