REMOÇÃO DA ÁRVORE BINÁRIA DE BUSCA

Objetivo

- Compreender o funcionamento de um método de remoção em uma árvore binária de busca simples.

Como remover?

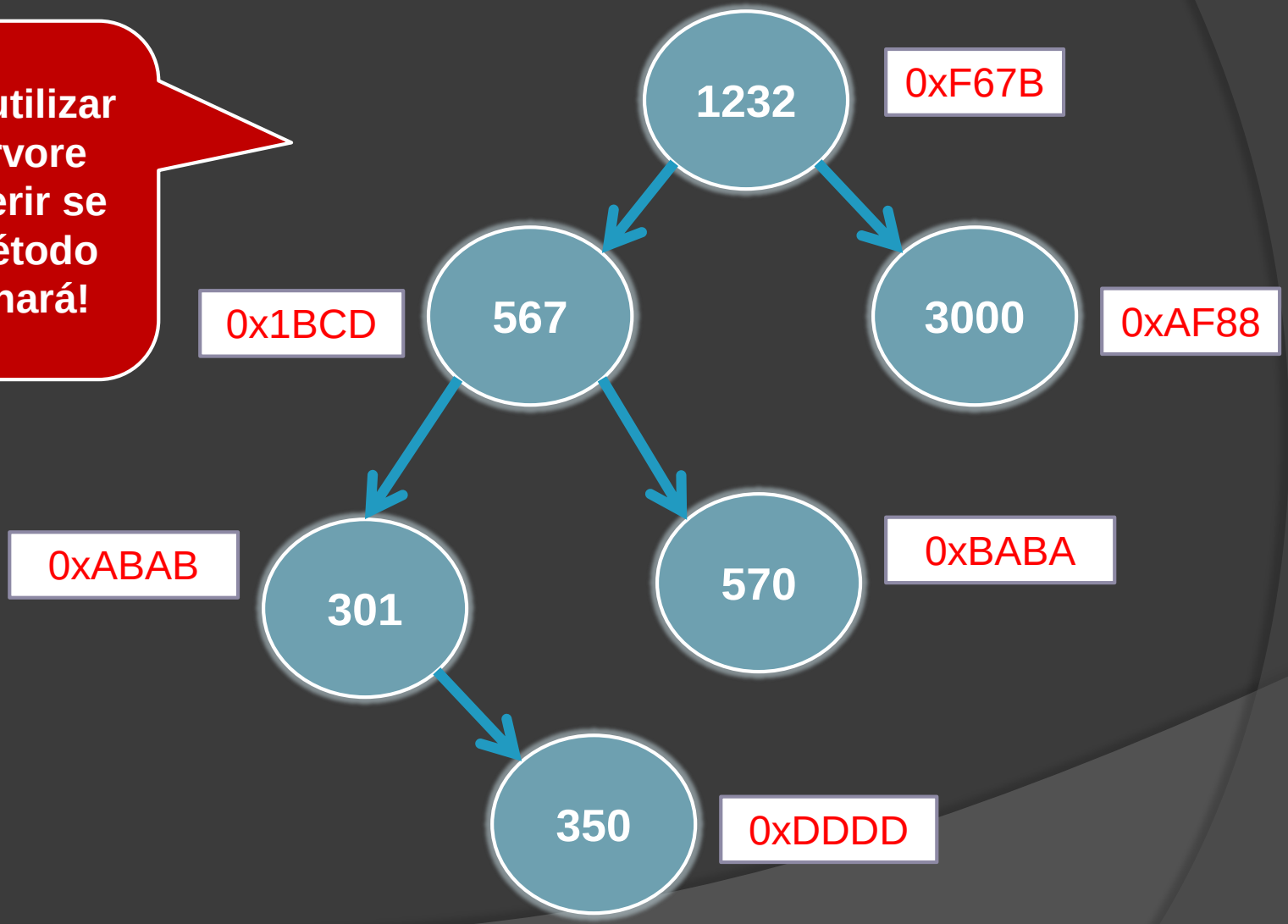
- Para remover de uma árvore binária de busca, sem se preocupar com balanceamento.
- Ao retirar o nó, devemos substituí-lo. Deste modo trabalharemos com duas perspectivas:
  - elegendo a chave de maior valor da esquerda como substituto; ou
  - elegendo a chave de menor valor da direita

Como remover?

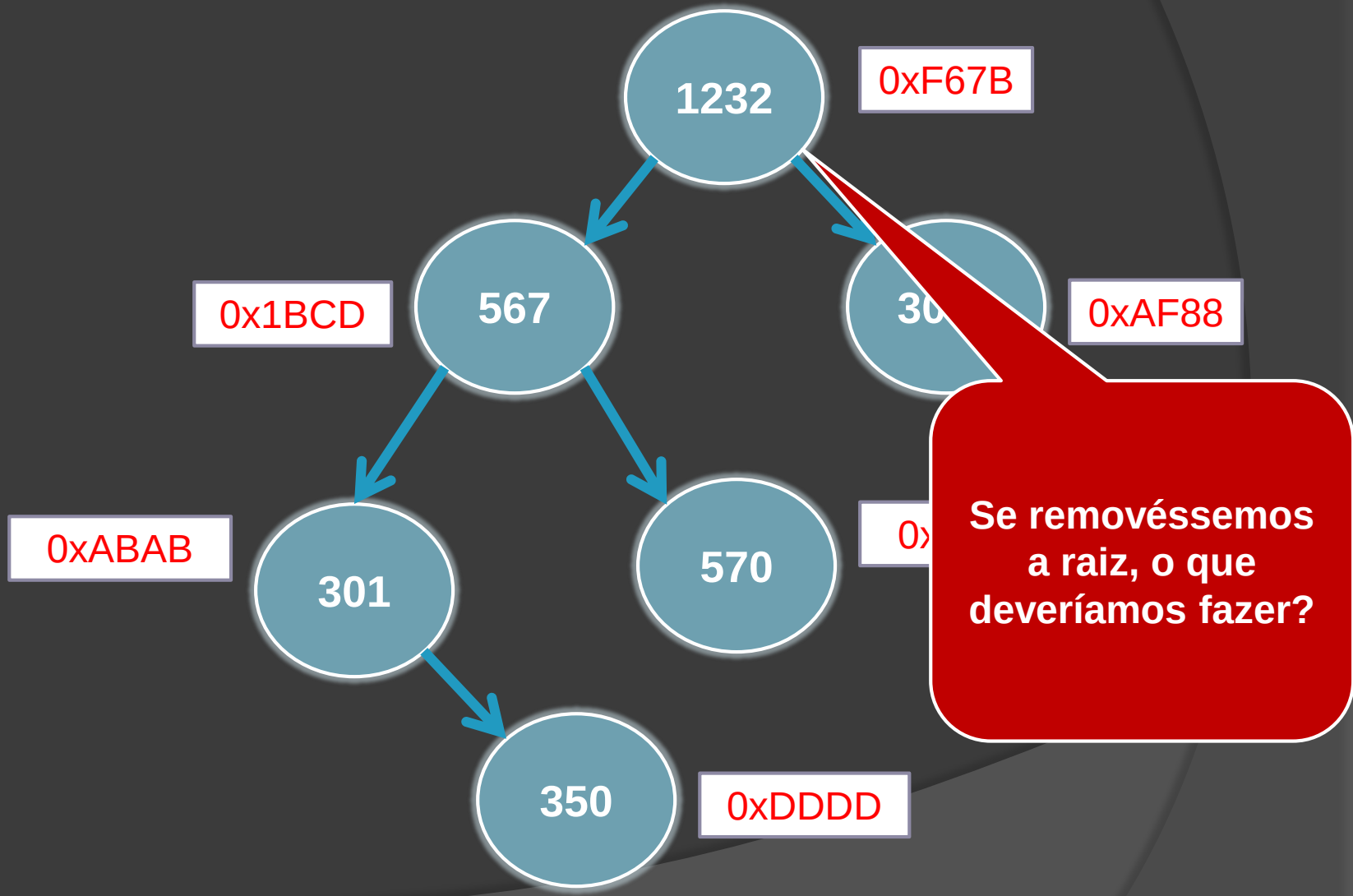
- Iremos utilizar a perspectiva de encontrar o maior chave da esquerda como substituto

Exemplo

Vamos utilizar esta árvore para aferir se este método funcionará!

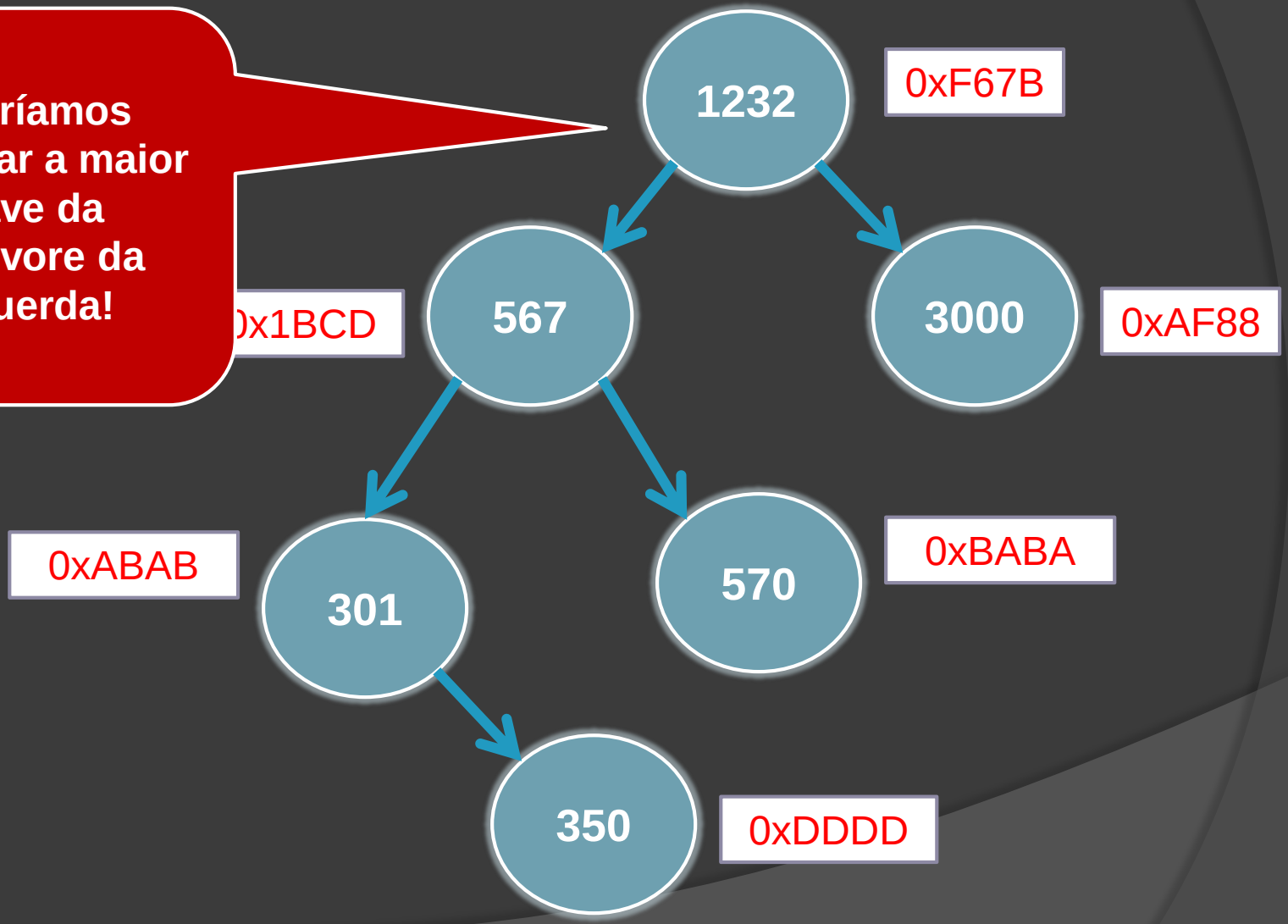


Exemplo

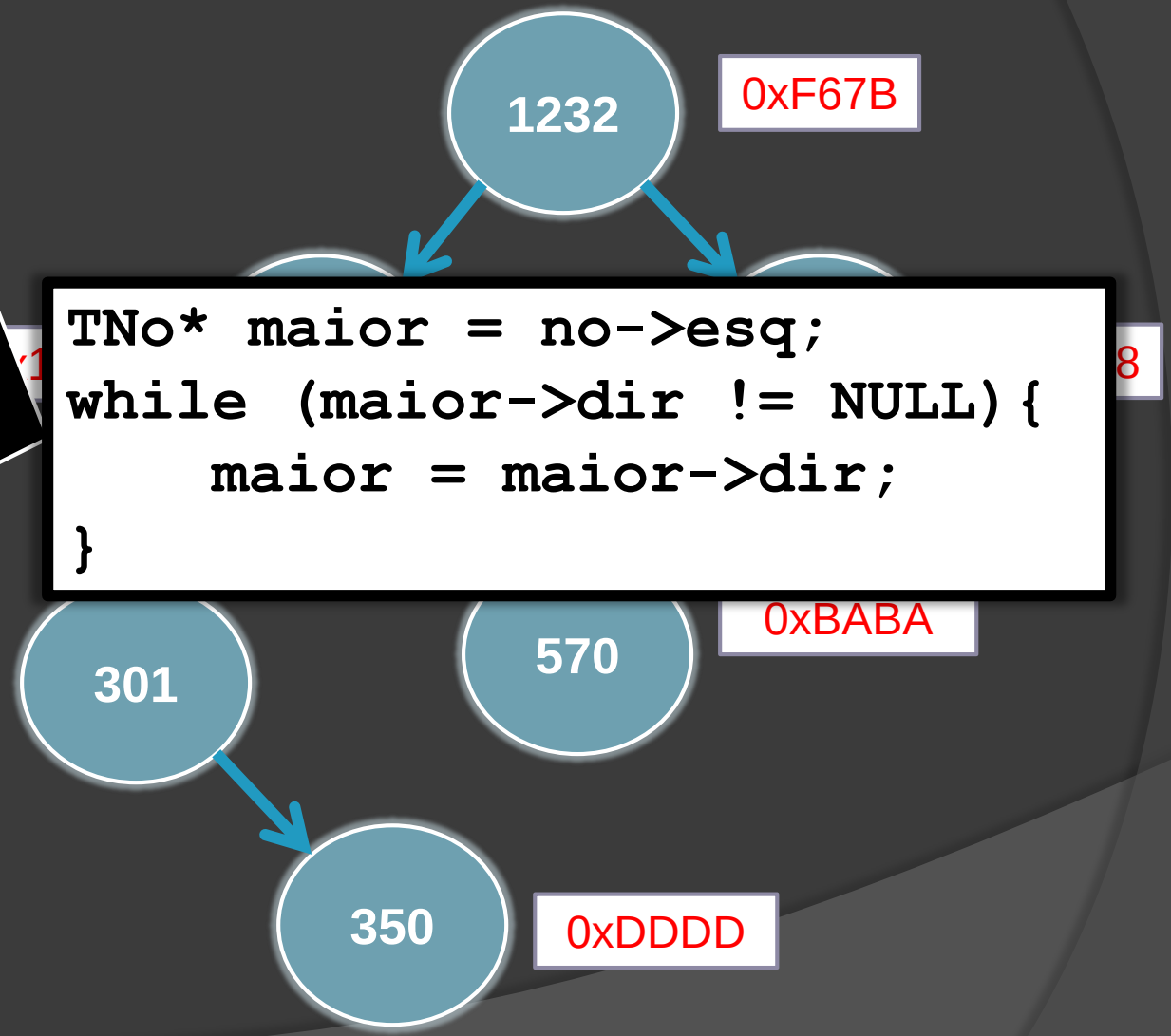

1232
0xF67B
567
0x1BCD
0xAF88
0xABAB
301
570
350
0xDDDD
Se removéssemos a raiz, o que deveríamos fazer?

Exemplo

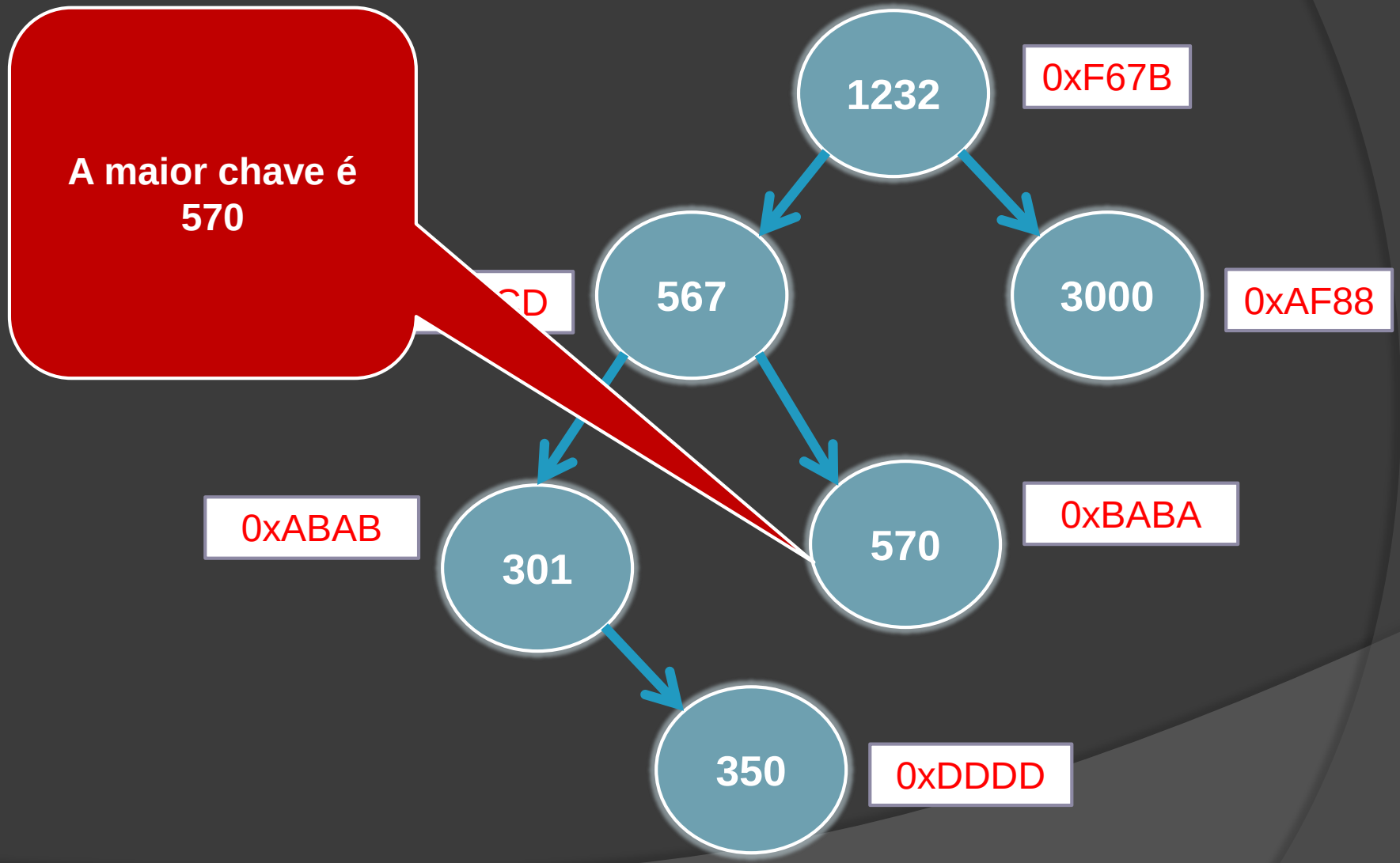
Deveríamos encontrar a maior chave da subárvore da esquerda!



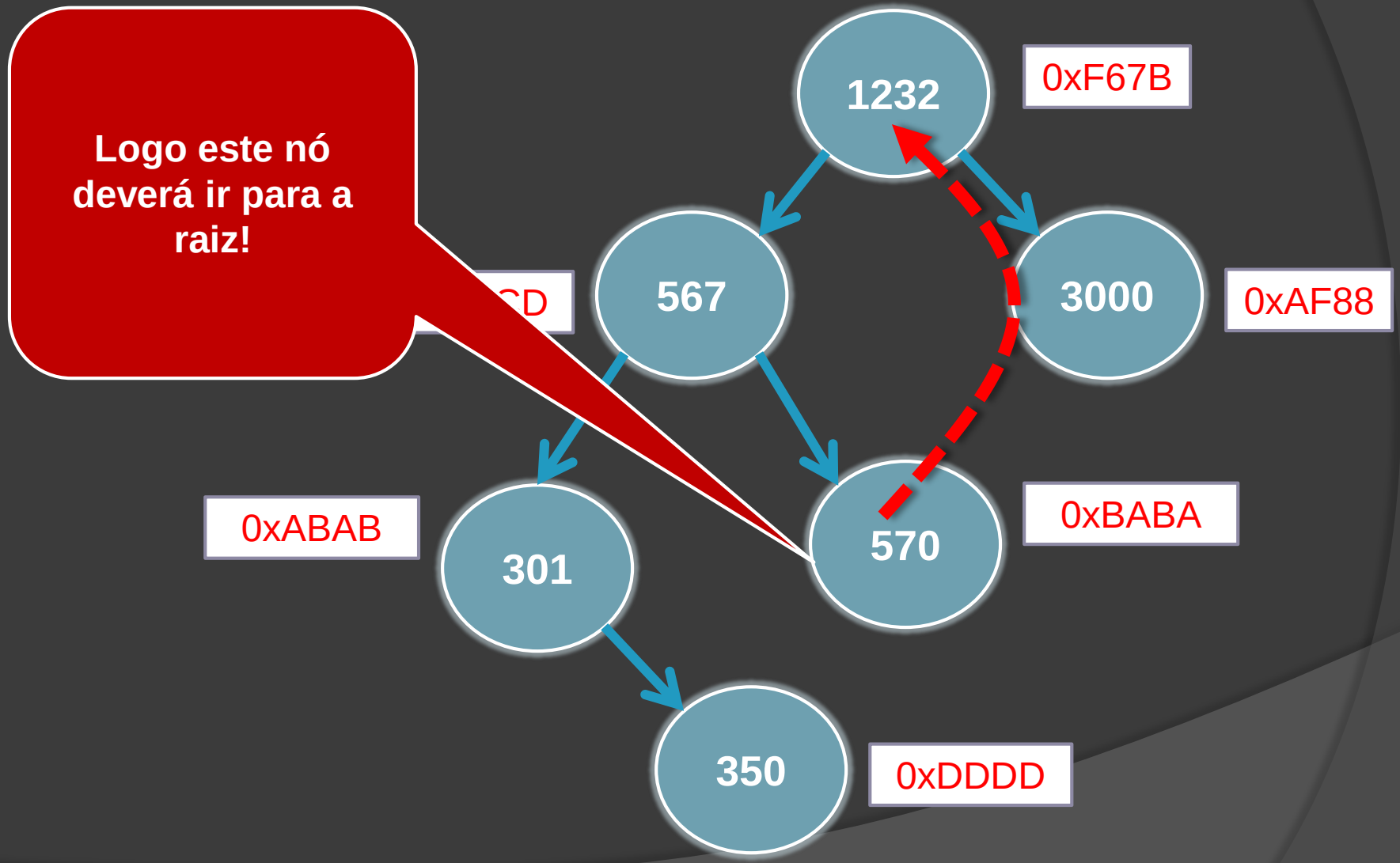
Pensando no código:
Como encontrar maior chave da esquerda (sendo “no” o endereço que desejo apagar)



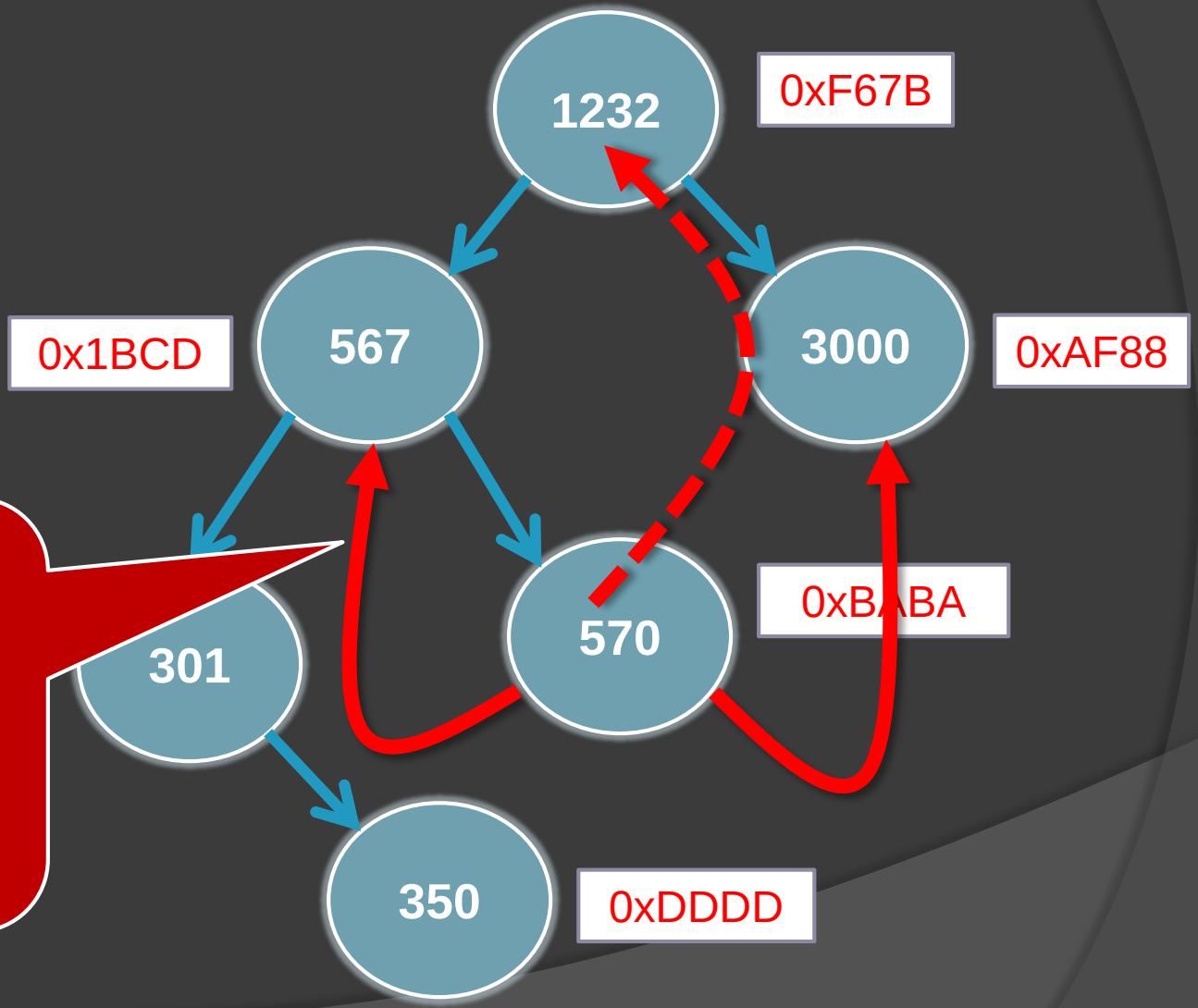
Exemplo


A maior chave é 570
1232
0xF67B
567
3000
0xAF88
CD
0xABAB
301
570
0xBABA
350
0xDDDD

Exemplo


Logo este nó deverá ir para a raiz!
1232
0xF67B
567
3000
0xAF88
0xABAB
301
570
0xBABA
350
0xDDDD

Exemplo

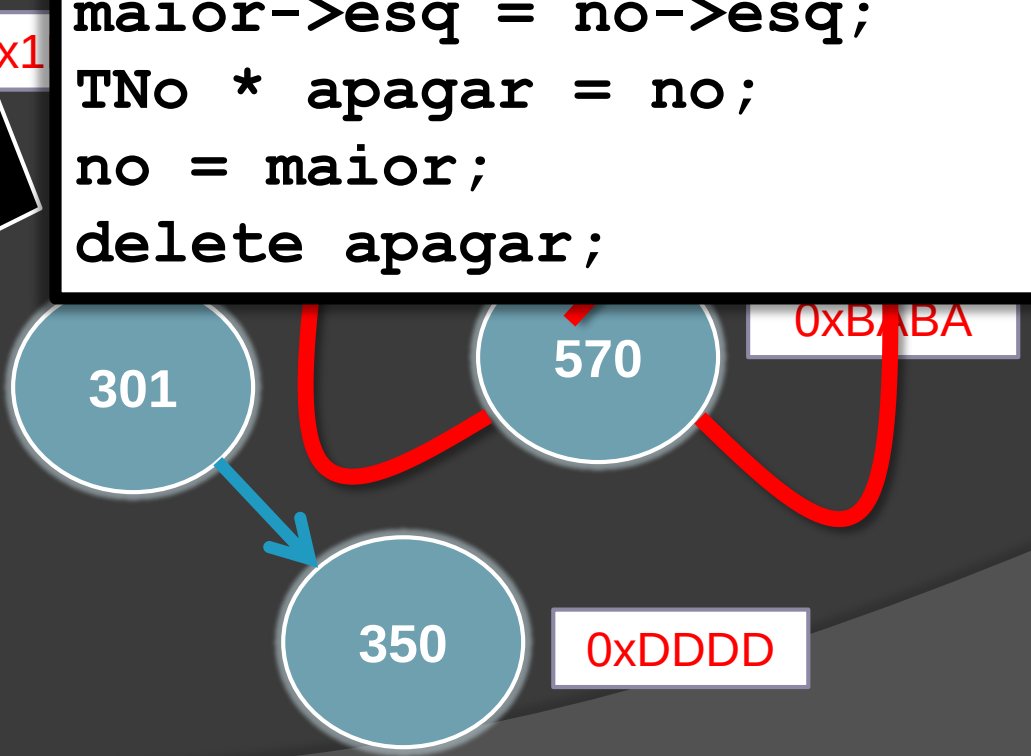


Devemos então fazer o nó de 570 ter as referências da esquerda e direita de 1232

Exemplo

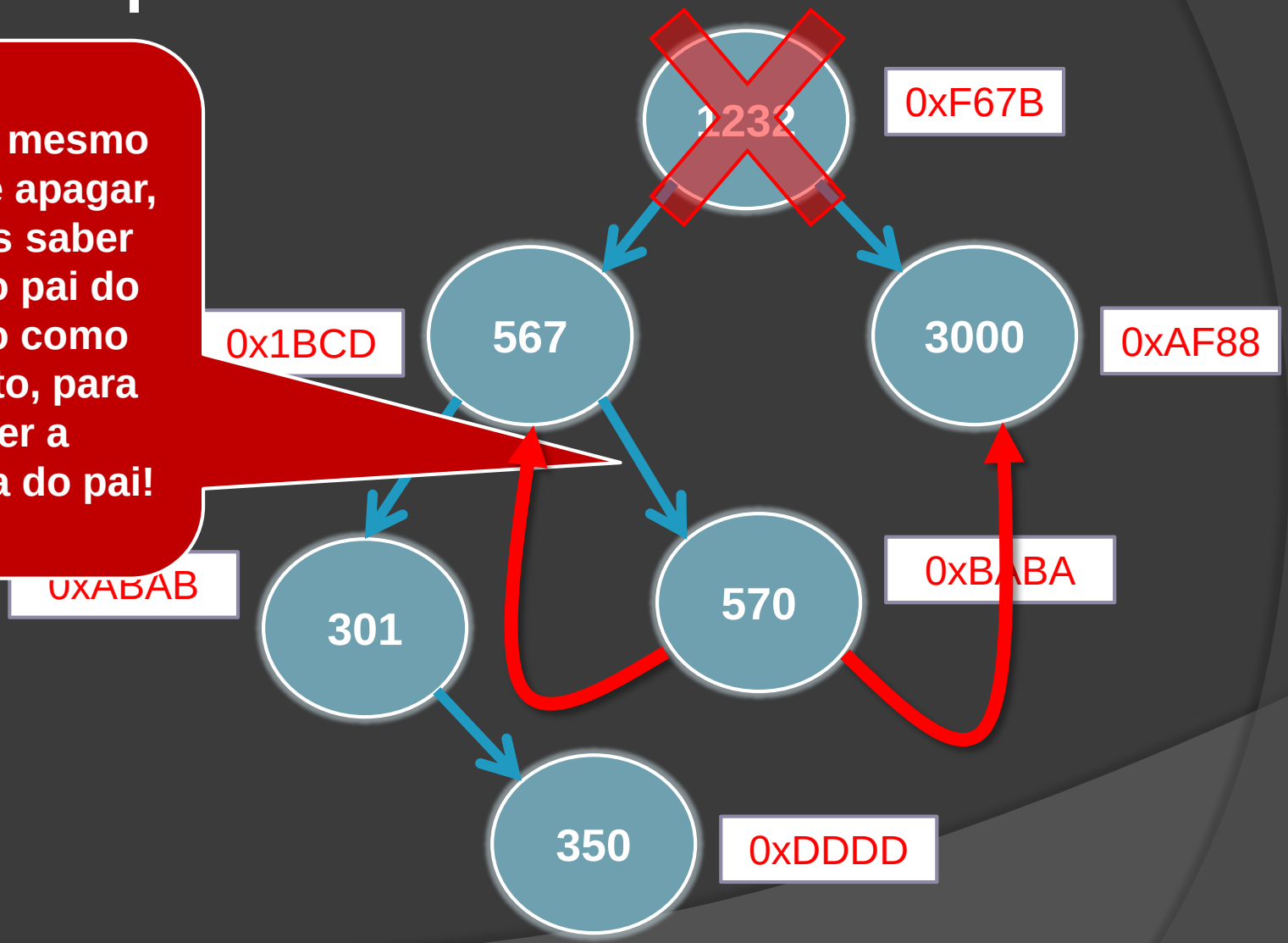
Pensando no código: realizando apontamentos e trocando item a ser apagado pelo nó eleito (maior da esquerda)! Já estamos apagando o nó!

```
TNo* maior = no->esq;
while (maior->dir != NULL){
    maior = maior->dir;
}
maior->dir = no->dir;
maior->esq = no->esq;
TNo * apagar = no;
no = maior;
delete apagar;
```



Exemplo

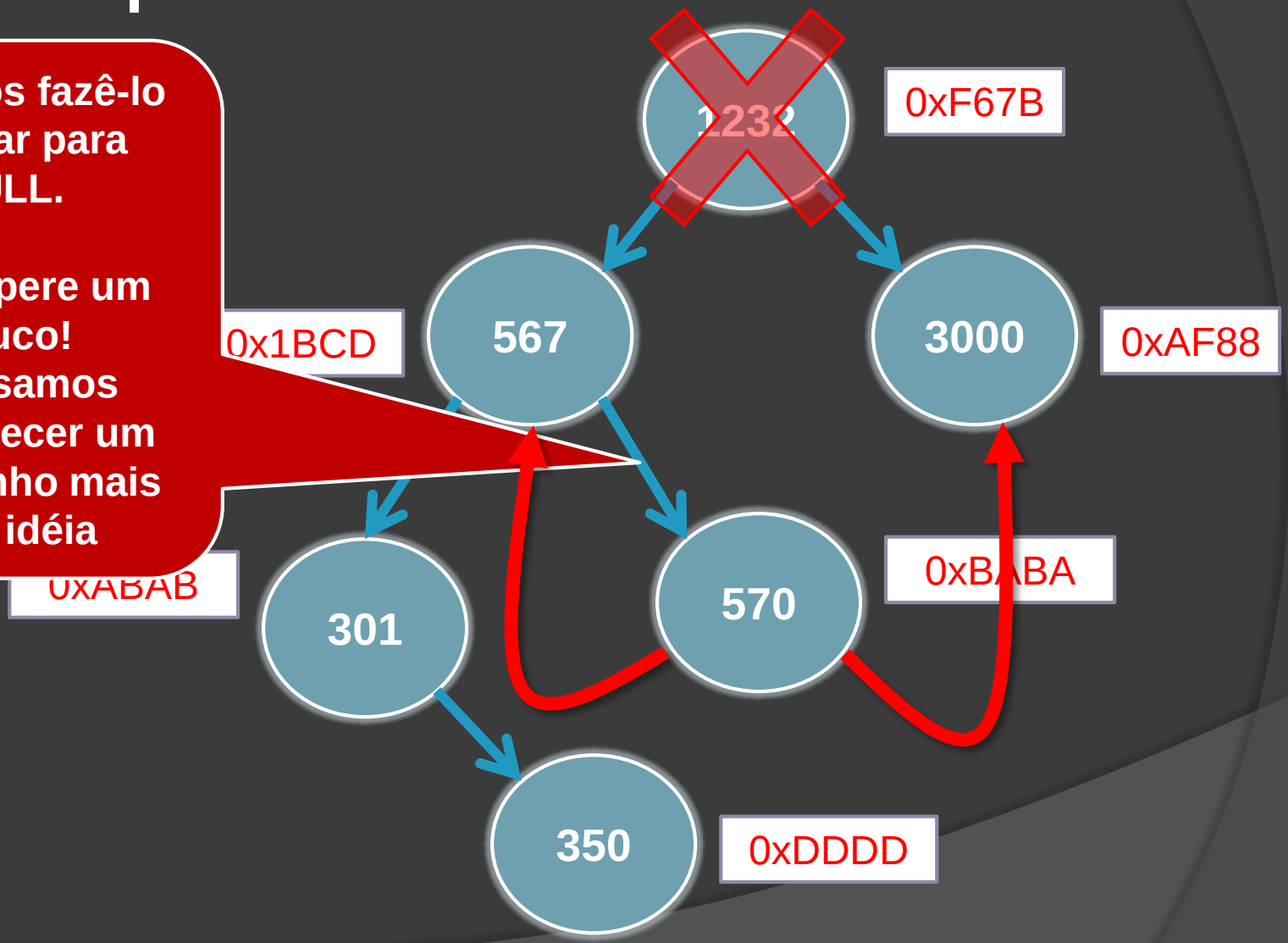
Note que mesmo depois de apagar, devemos saber quem é o pai do nó eleito como substituto, para refazer a referência do pai!



Exemplo

Podemos fazê-lo apontar para NULL.

Mas espere um pouco! Precisamos amadurecer um pouquinho mais esta idéia

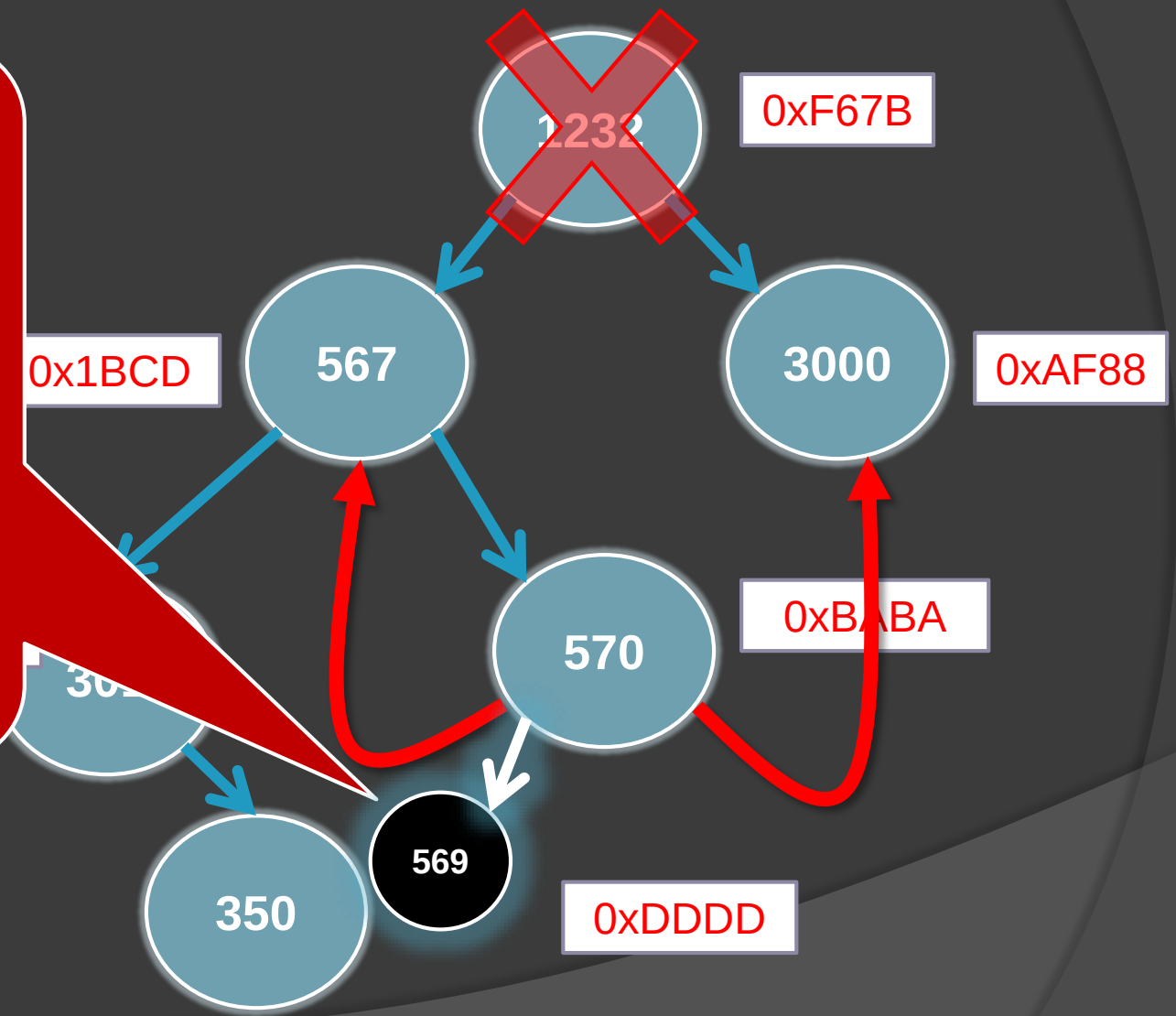


Exemplo

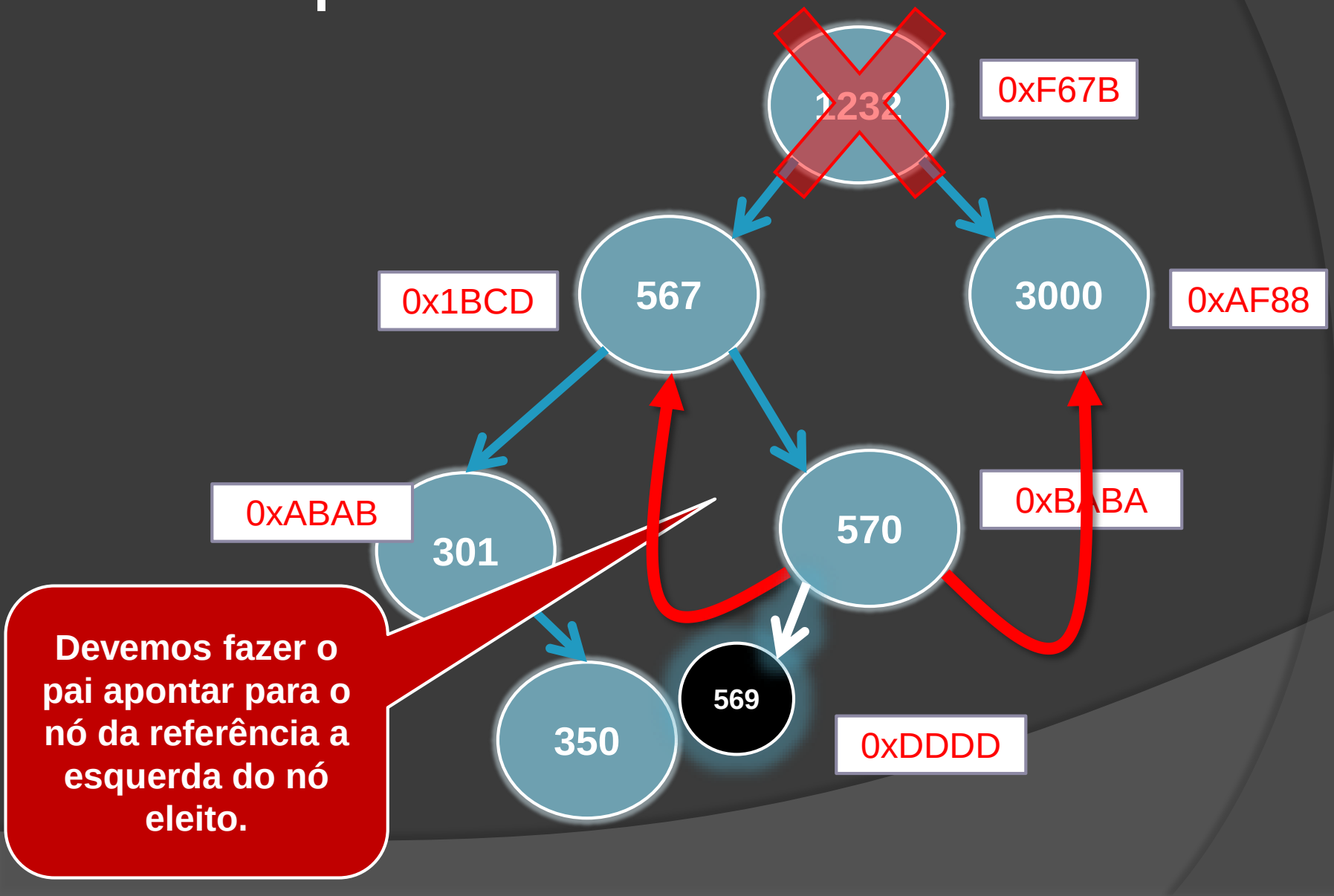
Digamos que o nó 570 possuísse um filho a esquerda.

O que faríamos?

Pelo procedimento que estamos utilizando, perderíamos a referência deste filho!



Exemplo


1232
0xF67B
0x1BCD
567
3000
0xAF88
0xABAB
301
570
0xBABA
350
569
0xDDDD
Devemos fazer o pai apontar para o nó da referência a esquerda do nó eleito.

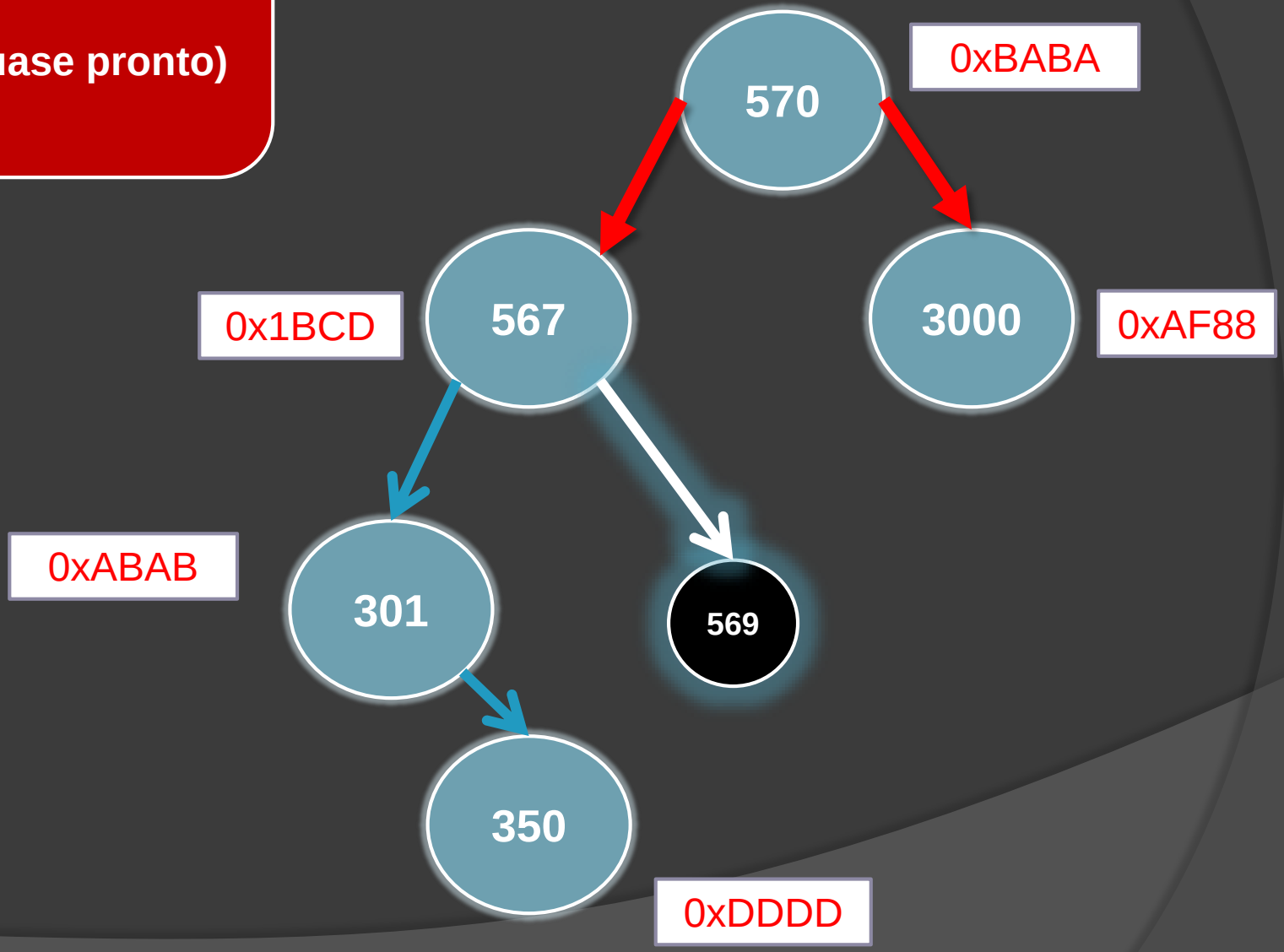
Exemplo

Pensando no código: mantendo registro do pai do maior e fazendo-o apontar para esquerda do maior. a esquerda do maior só passa a apontar para a esquerda do nó se o elemento à no->esq não for o maior (quando pai!= NULL)!

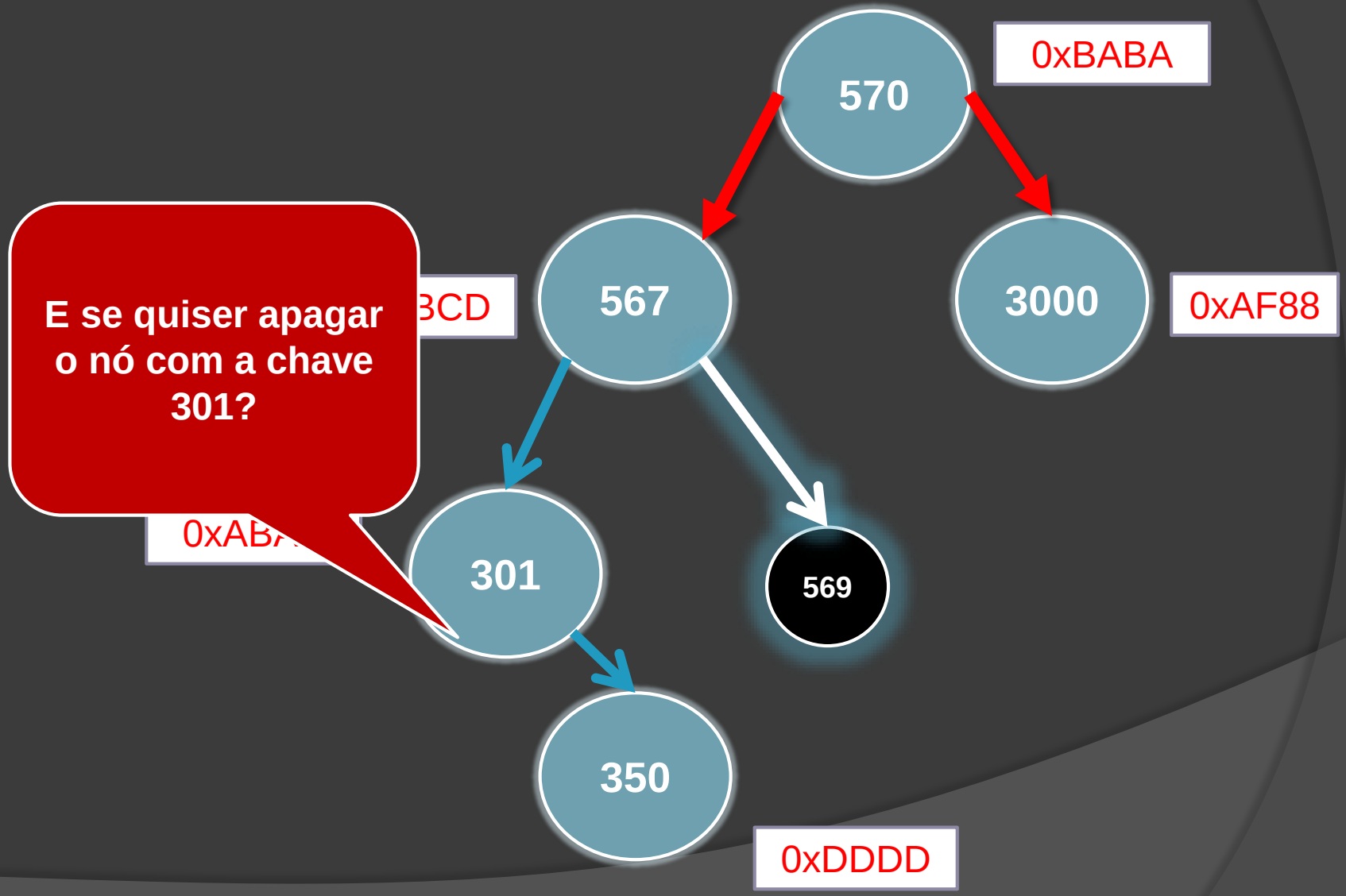
```
TNo* maior = no->esq;
TNo* pai = NULL;
while (maior->dir != NULL){
    pai = maior;
    maior = maior->dir;
}
maior->dir = no->dir;
if (pai != NULL){
    pai->dir = maior->esq;
    maior->esq = no->esq;
}
TNo * apagar = no;
no = maior;
delete apagar;
```

Pronto!

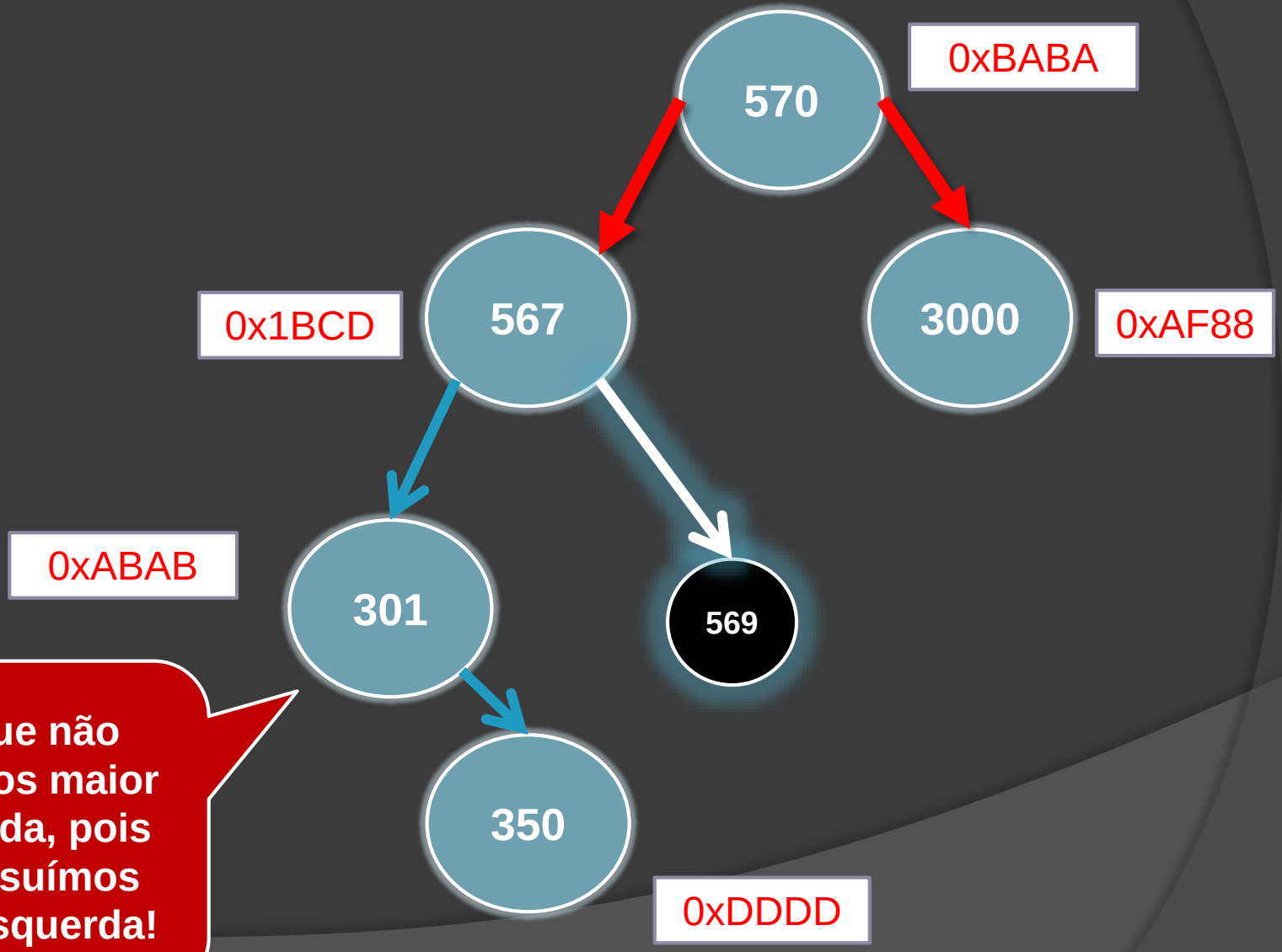
(ou quase pronto)



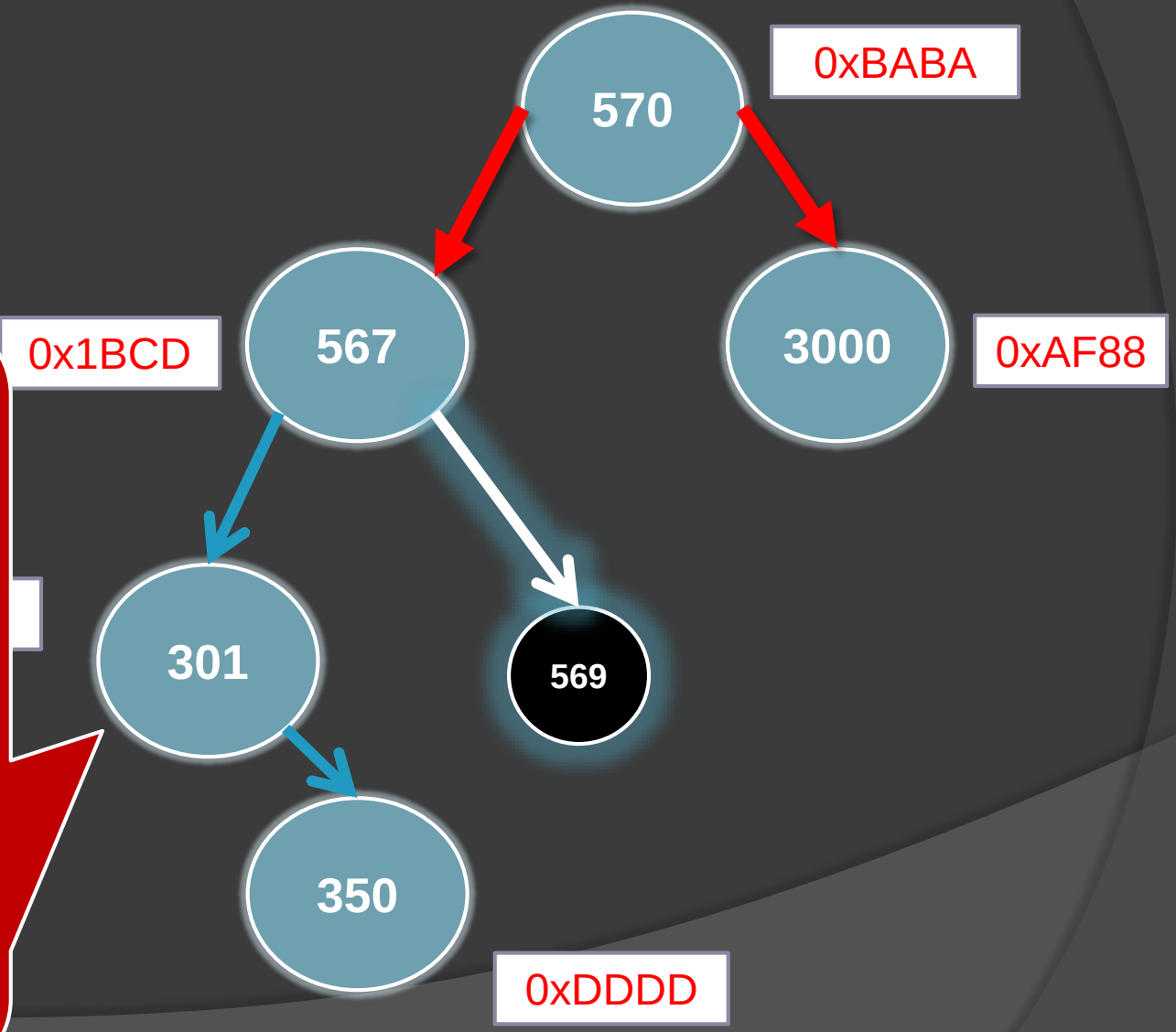
Exemplo


570
0xBABA
567
3000
0xAF88
BCD
E se quiser apagar o nó com a chave 301?
0xAB
301
569
350
0xDDDD

Exemplo

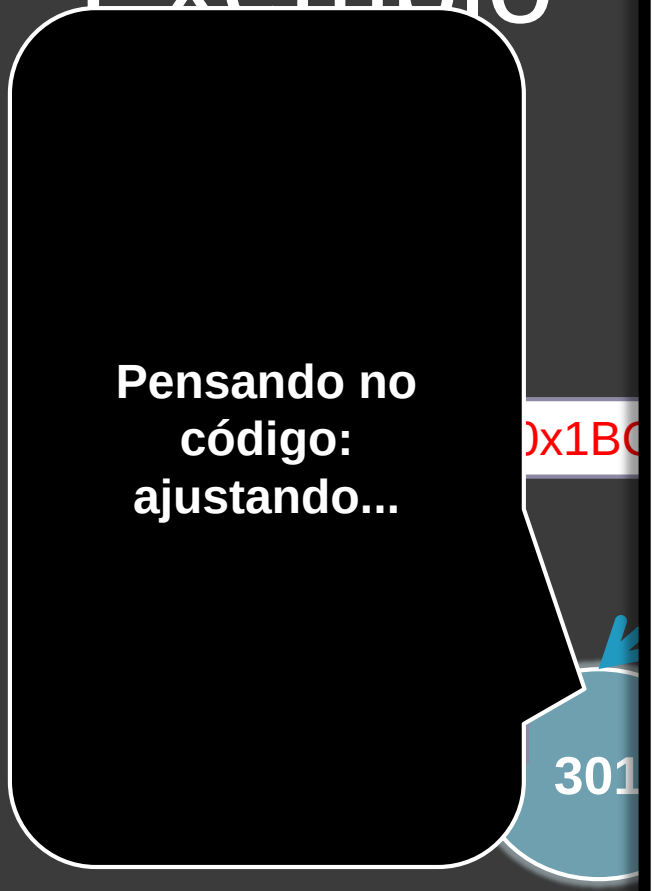

570
0xBABA
567
0x1BCD
3000
0xAF88
301
0xABAB
569
350
0xDDDD
Note que não possuímos maior à esquerda, pois não possuímos nada à esquerda!

Exemplo



Nestes casos, iremos, como o nó à esquerda é NULL, o substituto será o nó imediatamente à direita do nó a ser apagado!

Exemplo




```
TNo * apagar;
TNo* maior = no->esq;
if(maior == NULL){
    apagar = no;
    no = no->dir;
    delete apagar;
    return;
}
TNo* pai = NULL;
while (maior->dir != NULL){
    pai = maior;
    maior = maior->dir;
}
maior->dir = no->dir;
if (pai != NULL){
    pai->dir = maior->esq;
    maior->esq = no->esq;
}
apagar = no;
no = maior;
delete apagar;
```

Exemplo

```
TNo * apagar;
TNo* maior = no->esq;
if(maior == NULL){
    maior->esq = no->esq;
}
apagar = no;
no = maior;
delete apagar;
```

Note que o código está assumindo que o nó a ser apagado já foi encontrado. Precisamos de um outro procedimento!

Logo, dividiremos em duas funções:
→ procura_remove: procura e requisita remoção
→ remover: procedimentos para remoção do nó

```
void procura_remove(


                TNo *&no,
                int chave){
    if(no != NULL){
        if(no->chave == chave){
                remover(no);
        }else{
            if(chave > no->chave){
                procura_remove(
                        no->dir, chave);
            }else{
                procura_remove(
                        no->esq, chave);
            }
        }
    }
}
```

```
void remover(TNo *&no){
    TNo * apagar;
    TNo* maior = no->esq;
    if(maior == NULL){

        apagar = no;
        no = no->dir;
        delete apagar;
        return;
    }
    TNo* pai = NULL;
    while (maior->dir !=
    NULL){
        pai = maior;
        maior = maior->dir;
    }
    maior->dir = no->dir;
    if (pai != NULL){
        pai->dir = maior->esq;
        maior->esq = no->esq;
    }
    apagar = no;
    no = maior;
    delete apagar;
```