Árvore B (Parte II)

Estruturas de Dados

Prof. Jarbas Joaci de Mesquita Sá Junior

Engenharia da Computação

UFC - Sobral

2012.1

Baseado nos slides da Prof. Gustavo Nonato, do ICMC-USP

Árvore B

- As árvores B são árvores balanceadas projetadas para trabalhar com dispositivos de armazenamento secundário como discos magnéticos.
- Elas visam otimizar as operações de entrada e saída nos dispositivos. O tempo de acesso às informações em um disco é prejudicado principalmente pelo tempo de posicionamento do braço de leitura.
- Uma vez que o braço esteja posicionado no local correto, a leitura pode ser feita de forma bastante rápida. Desta forma, devemos minimizar o número de acessos ao disco.

Árvore B

- Diferente das árvores binárias, cada nó em uma árvore B pode ter muitos filhos, isto é, o grau de um nó pode ser muito grande.
- Definição: Uma *árvore B* possui as seguintes propriedades:
  - Todo o nó ***X*** possui os seguintes campos:
    - ***n***, o número de chaves armazenadas em ***X***;
    - as ***n*** chaves $k_1, k_2 \ldots k_n$ são armazenadas em ordem crescente;
    - ***folha***, que indica se ***X*** é uma folha ou um nó interno.

Árvore B

- Definição: Uma *árvore B* possui as seguintes propriedades:
  - Se ***X*** é um nó interno então ele possui ***n+1*** ponteiros $f_1, f_2 \ldots f_{n+1}$ para seus filhos
  - Todas as folhas da árvore estão na mesma altura (que é a altura da árvore).
  - Existe um número máximo e mínimo de filhos em um nó. Este número pode ser descrito em termos de um inteiro fixo ***t*** maior ou igual a 2 chamado **grau mínimo**.
  - ***t*** é a ordem da árvore, embora a definição de ordem não seja única entre os autores.

Árvore B

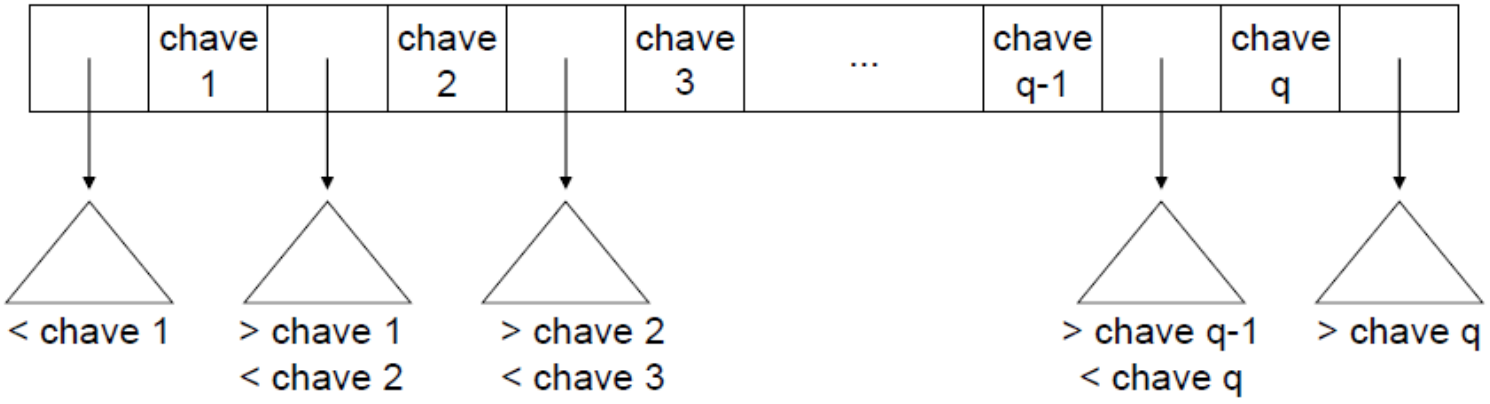
- Definição: Uma *árvore B* possui as seguintes propriedades:
  - Todo o nó diferente da raiz deve possuir pelo menos ***t-1*** chaves. Todo o nó interno diferente da raiz deve possuir pelo menos ***t*** filhos. Se a árvore não é vazia, então a raiz possui pelo menos uma chave.
  - Todo o nó pode conter no máximo ***2t - 1*** chaves. Logo um nó interno pode ter no máximo ***2t*** filhos. Dizemos que um nó é cheio se ele contém ***2t - 1*** chaves.

Árvore B

```
const t = 2;
typedef struct no_arvoreB arvoreB;

struct no_arvoreB {
   int num_chaves;
   char chaves[2*t-1];
   arvoreB *filhos[2*t];
   bool folha;
};
```

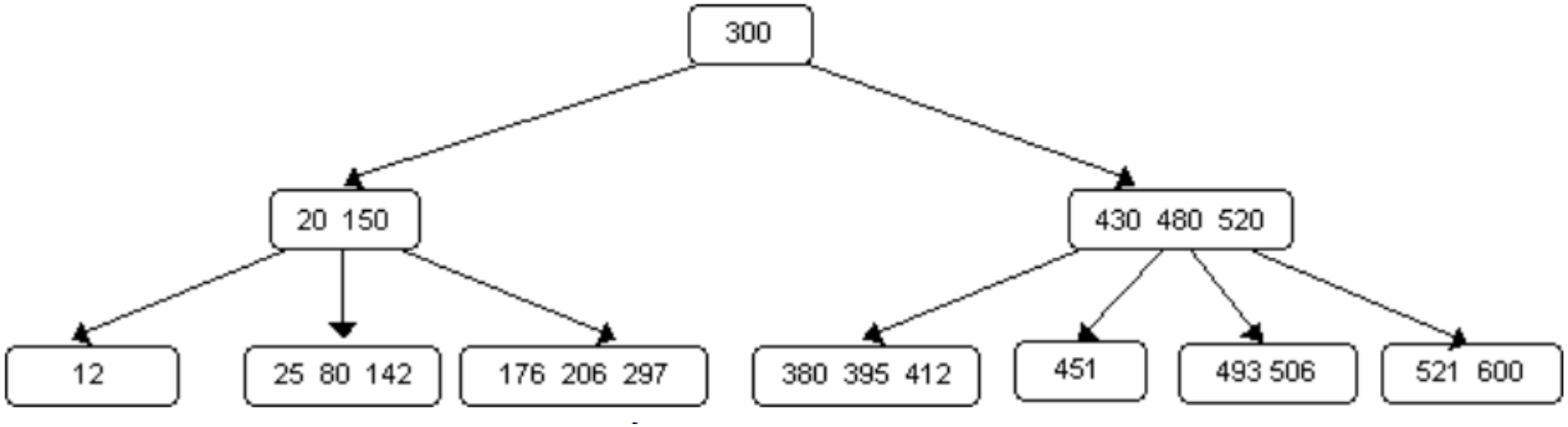
Árvore B

- Estrutura de um nó



Árvore B

Árvore $t = 2$



Busca em Árvore B

- A busca em uma árvore B é uma função parecida com a de busca em uma árvore de busca binária, exceto o fato de que se deve decidir entre vários caminhos.
- Se a chave não for encontrada no nó em questão, continua-se a busca nos filhos deste nó, realizando-se novamente a busca binária.
- Caso o nó não esteja contido na árvore a busca terminará ao encontrar um ponteiro igual a ***NULL***, ou de forma equivalente, verificando-se que o nó é uma folha.
- A busca completa pode ser realizada em tempo $O(lg(t)\log_t(n))$.

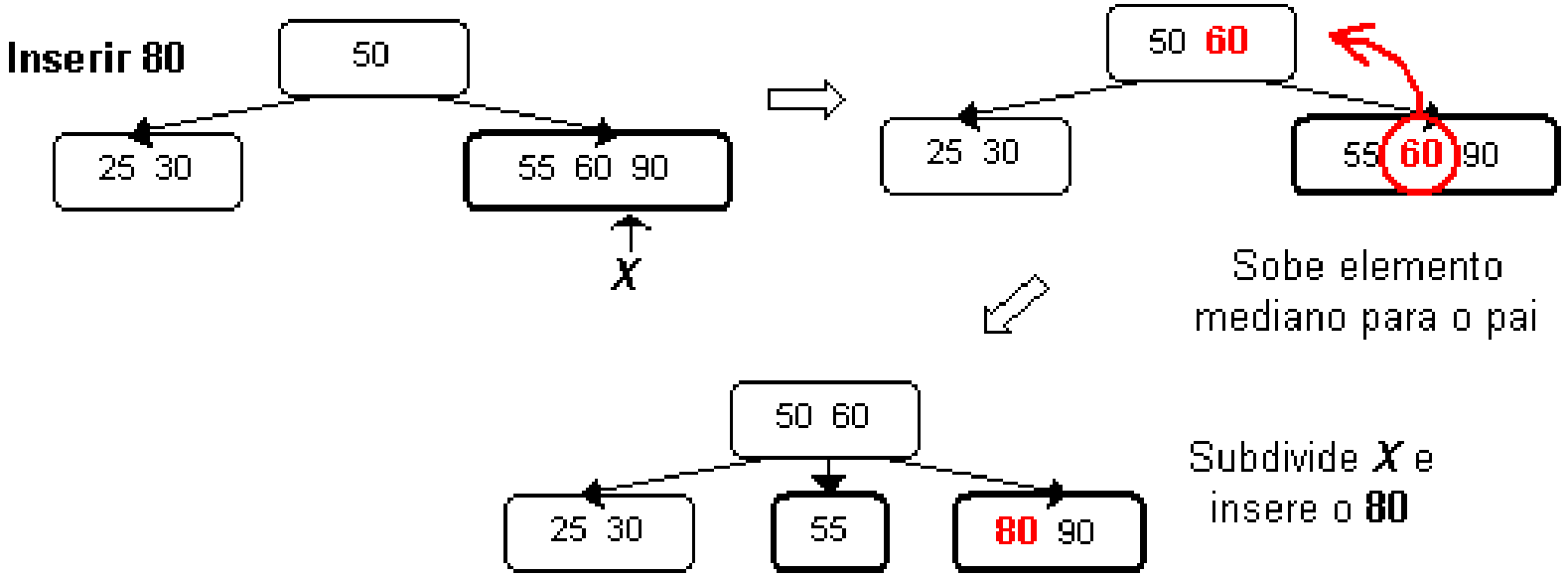
Busca em Árvore B

```
int busca_binaria(arvoreB *no, int info)
{
  int meio, i, f;

  i = 0;
  f = no->num_chaves-1;

  while (i <= f)
  {
    meio = (i + f)/2;
    if (no->chaves[meio] == info)
       return(meio); //Encontrou. Retorna a posição em que a chave está.
    else if (no->chave[meio] > info
            f = meio - 1;
         else i = meio + 1;
  }
  return(i); //Não encontrou. Retorna a posição do ponteiro para o filho.
}

bool busca(arvoreB *raiz, int info)
{
  arvoreB *no;
  int pos; //posição retornada pelo busca binária.

  no = raiz;
  while (no != NULL)
   {
     pos = busca_binaria(no, info);
     if (pos < no->num_chaves && no->chaves[pos] == info)
         return(true);
     else no = no->filhos[pos];
   }
  return(false);
}
```

Inserção em Árvore B

- Para inserir um novo elemento em uma árvore B, basta localizar o nó folha ***X*** onde o novo elemento deva ser inserido.
- Se o nó ***X*** estiver cheio, será necessário realizar uma subdivisão de nós que consiste em passar o elemento mediano de ***X*** para seu pai e subdividir ***X*** em dois novos nós com ***t - 1*** elementos e depois inserir a nova chave.

Inserção em Árvore B

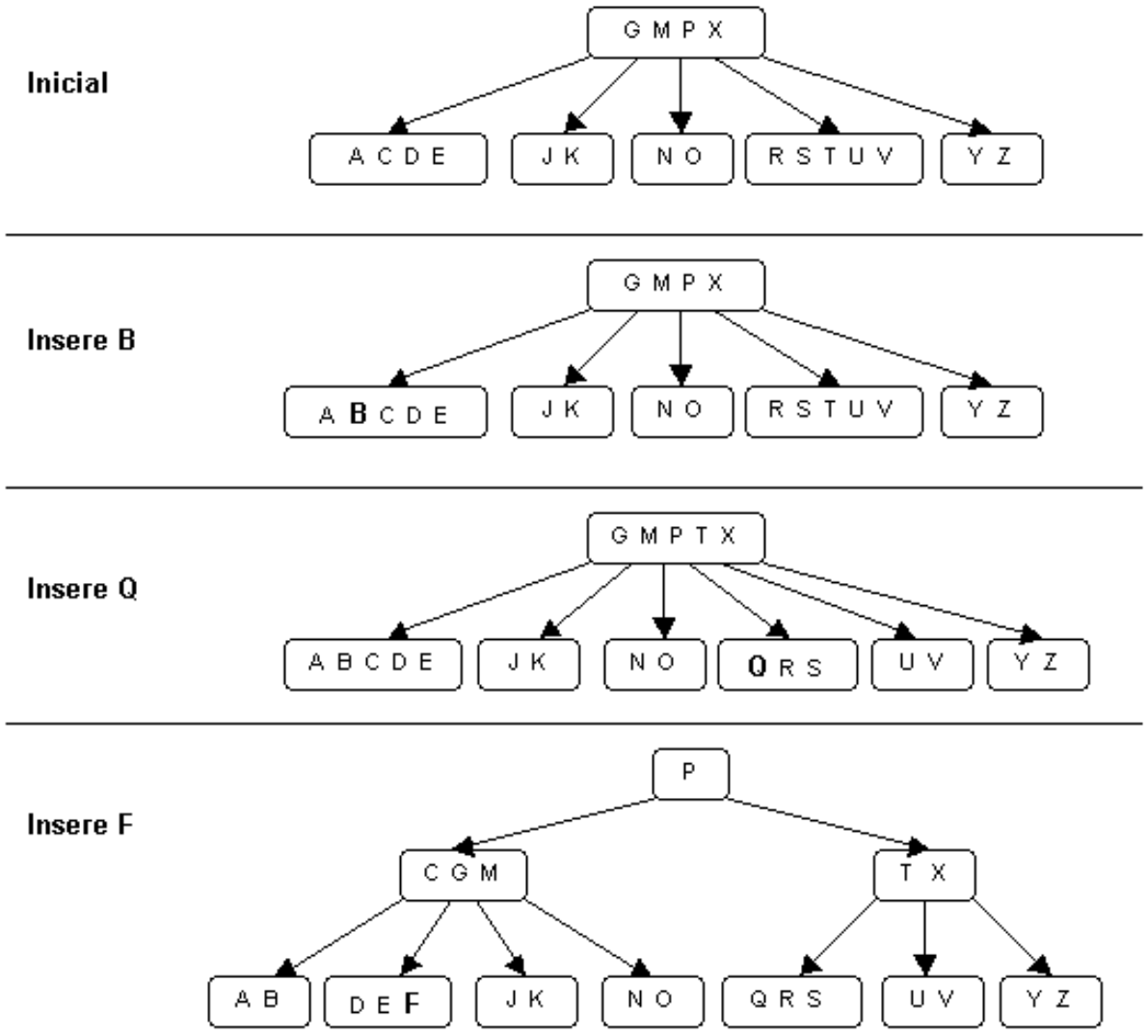


Passos para a inserção da chave 80 em uma árvore B com $t = 2$

Inserção em Árvore B

- Se o pai de ***X*** também estiver cheio, repete-se recursivamente a subdivisão acima para o pai de ***X***. No pior caso terá que aumentar a altura da árvore B para poder inserir o novo elemento.
- Note que diferentemente das árvores binárias, as árvores B crescem para cima

Inserção em Árvore B



Árvore B com *t=3*.

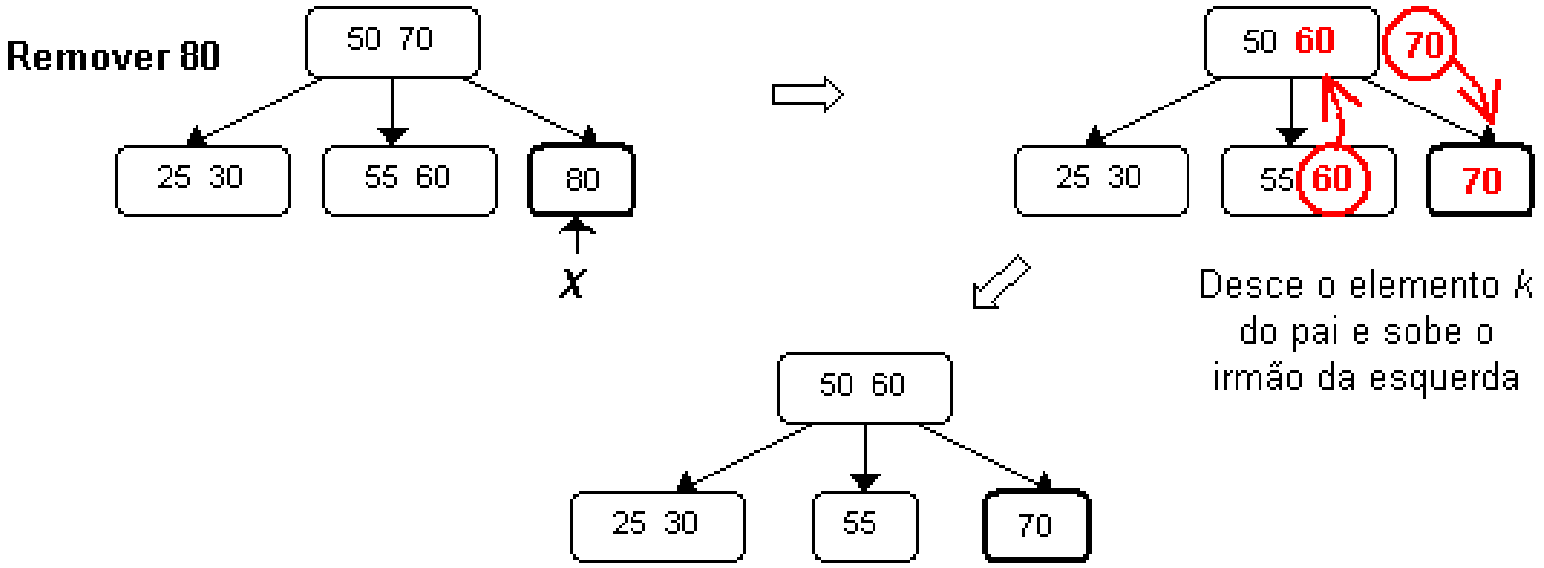
Remoção em Árvore B

- A remoção de um elemento de uma árvore B pode ser dividida em dois casos:
  1. O elemento que será removido está em uma folha
  2. O elemento que será removido está em um nó interno.
- Se o elemento estiver sendo removido de um nó não-folha, seu sucessor, que deve estar em uma folha, será movido para a posição eliminada e o processo de eliminação procede como se o elemento sucessor fosse removido do nó-folha.
- Quando um elemento for removido de uma folha ***X*** e o número de elementos no nó folha diminui para menos que ***t – 1,*** deve-se reorganizar a árvore B

Remoção em Árvore B

- Quando um elemento for removido de uma folha ***X*** e o número de elementos no nó folha diminui para menos que ***t - 1***, deve-se reorganizar a árvore B. A solução mais simples é analisarmos os irmãos da direita ou esquerda de ***X***.
- Se um dos irmãos (da direita ou esquerda) de ***X*** possui mais de ***t - 1*** elementos, a chave ***k*** do pai que separa os irmãos pode ser incluída no nó ***X*** e a última ou primeira chave do irmão (última se o irmão for da esquerda e primeira se o irmão for da direita) pode ser inserida no pai no lugar de ***k***.

Remoção em Árvore B

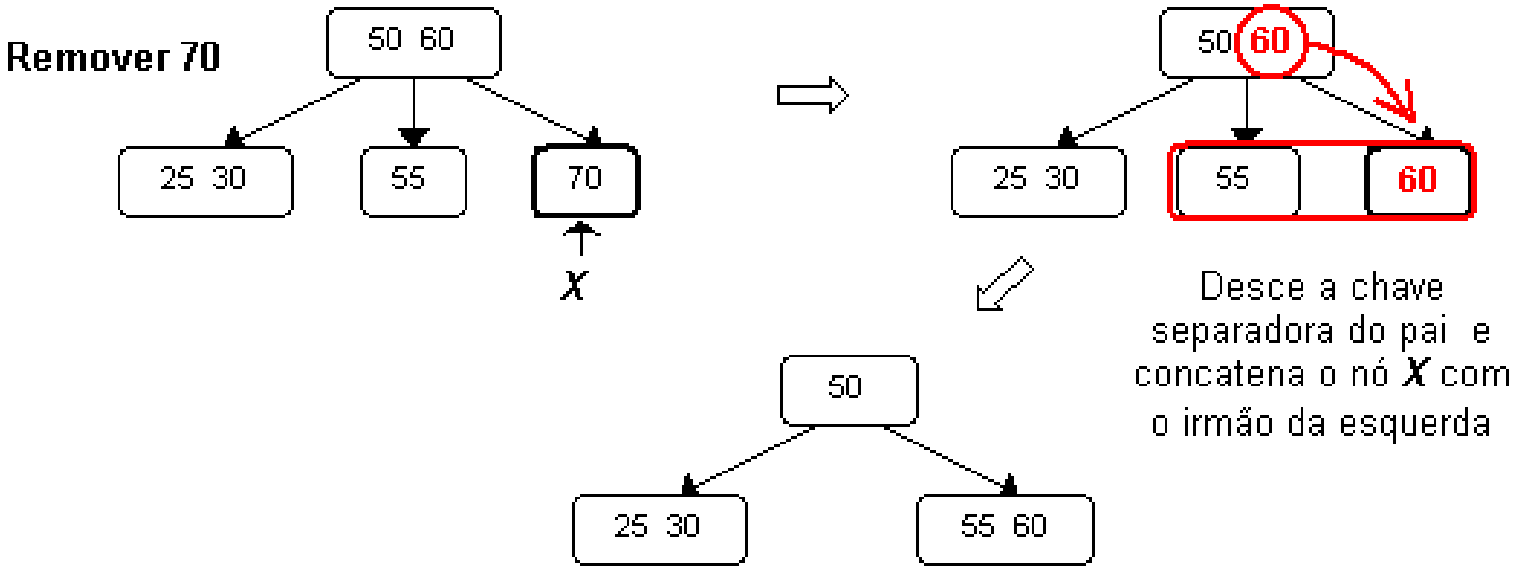


Passos para a remoção da chave 80 em uma árvore B com $t = 2$

Remoção em Árvore B

- Se os dois irmãos de ***X*** contiverem exatamente ***t - 1*** elementos (ocupação mínima), nenhum elemento poderá ser emprestado. Neste caso, o nó ***X*** e um de seus irmãos são concatenados em um único nó que também contém a chave separadora do pai.

Remoção em Árvore B

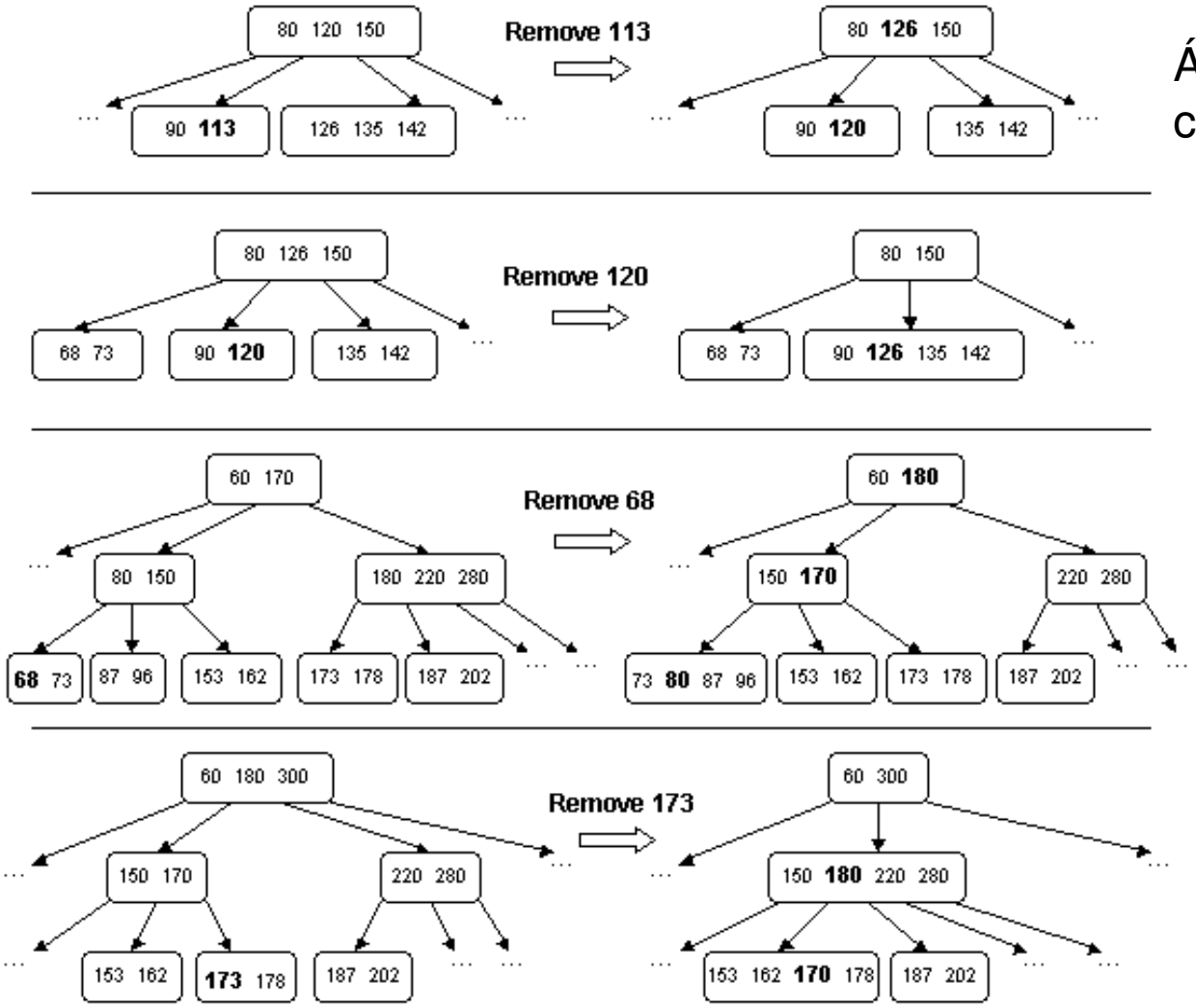


Passos para a remoção da chave 70 em uma árvore B com $t = 2$

Remoção em Árvore B

- Se o pai também contiver apenas ***t - 1*** elementos, deve-se considerar os irmãos do pai como no caso anterior e proceder recursivamente.
- No pior caso, quando todos os ancestrais de um nó e seus irmãos contiverem exatamente ***t - 1*** elementos, uma chave será tomada da raiz e no caso da raiz possuir apenas um elemento a árvore B sofrerá uma redução de altura.

Remoção em Árvore B



Árvore B com *t=3*.