Estruturas de Dados Avançadas (INF1010)

Árvores B

Árvores B

Estruturas de Dados para armazenamento **externo**

- tempo de acesso à memória secundária é significativo
- dados são lidos em **blocos**

Árvores Binárias têm nós pequenos (um único elemento)

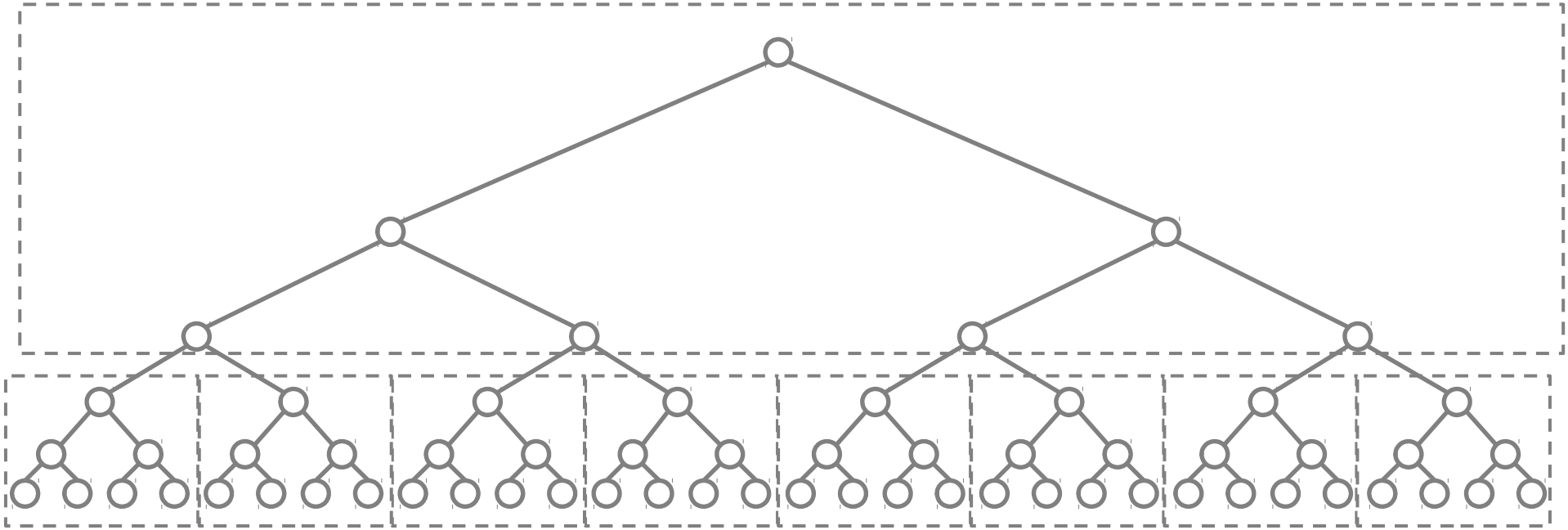
- nós num caminho de busca podem estar em blocos diferentes

A ideia de árvores B é permitir que cada acesso obtenha um **grupo de elementos**

Árvores B: aglomeração em páginas

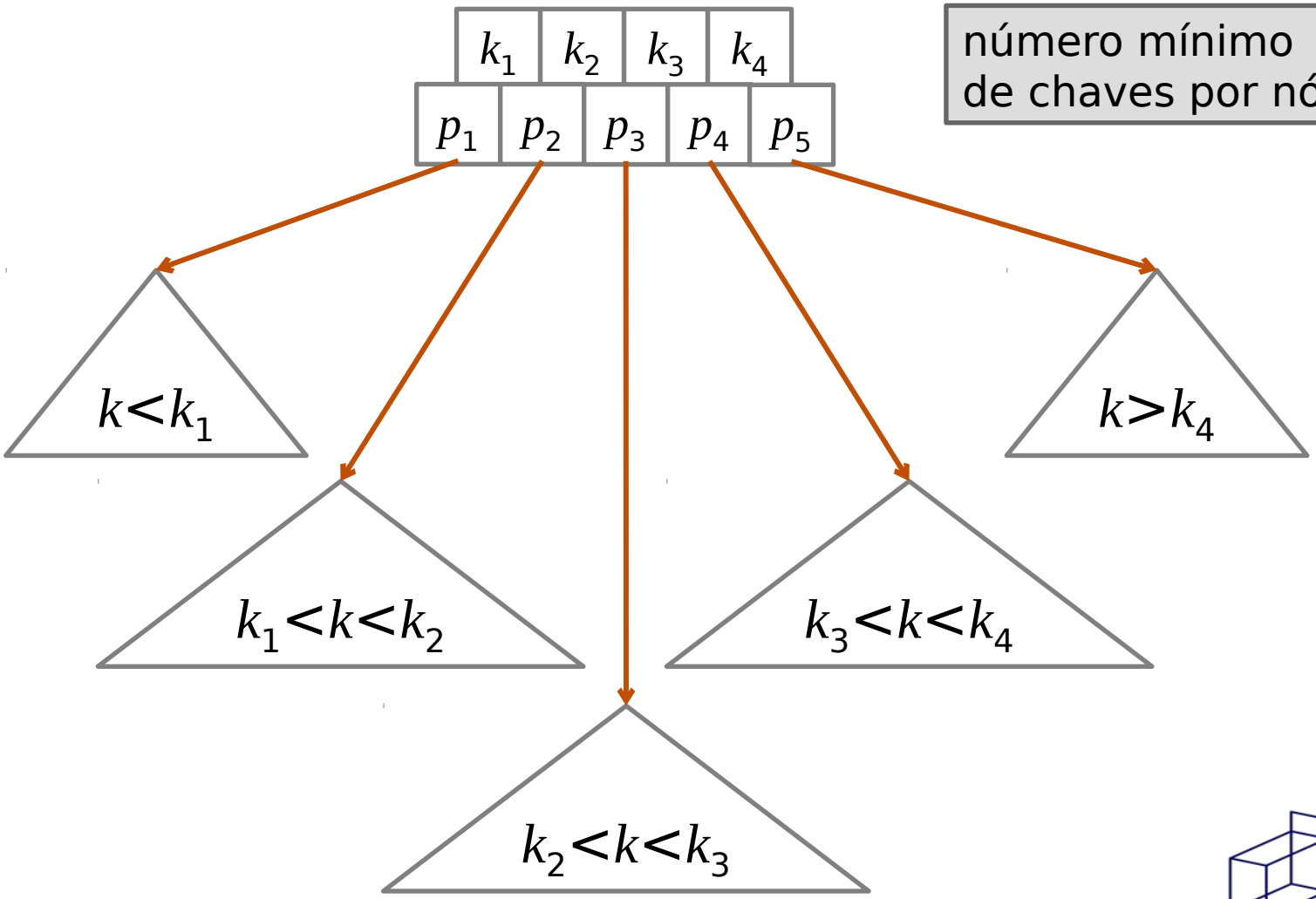
Cada acesso à memória secundária obtém um grupo de elementos

Sub-árvores são aglomeradas em **páginas**



Árvore B: idéia básica

Árvore n-ária com chaves de busca nos nós

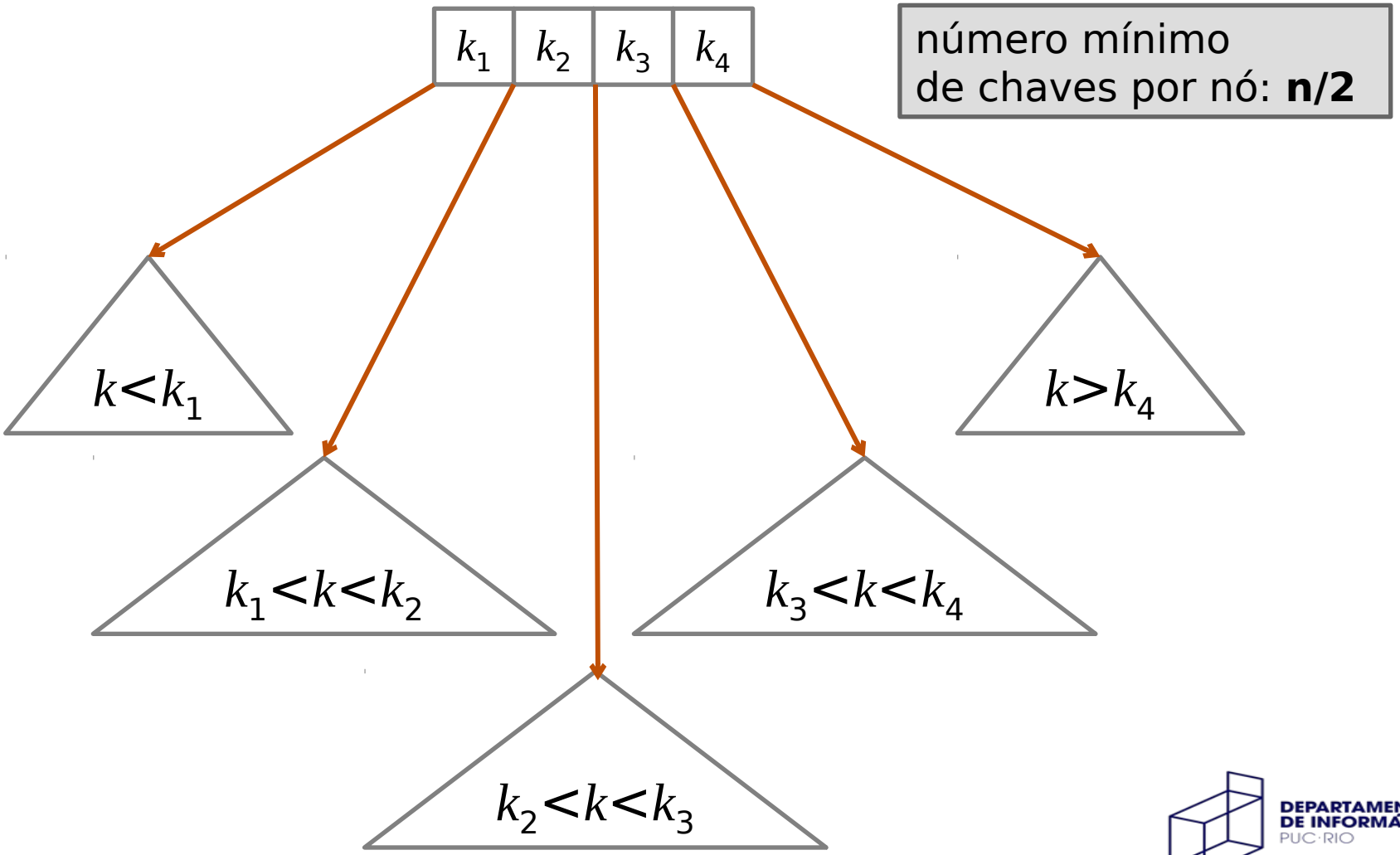


número mínimo
de chaves por nó: **n/2**

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Árvore B: idéia básica

Árvore n-ária com chaves de busca nos nós

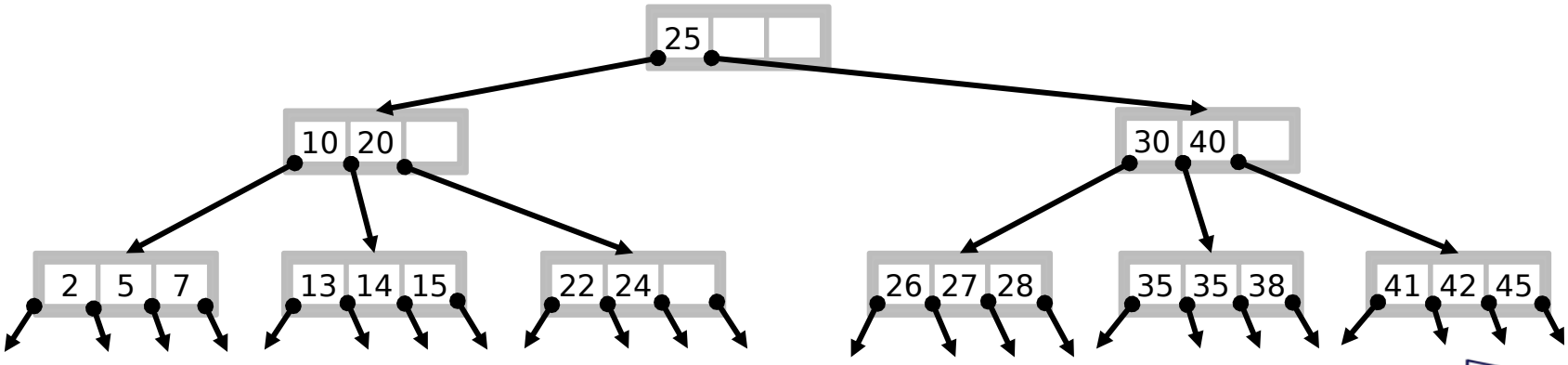


Árvore B: definição (Knuth, 1997)

Em uma árvore B de ordem **m**

- todo nó interno tem no máximo **m** filhos e no mínimo **m/2** filhos
- a raiz tem ao menos 2 filhos (exceto se for uma folha)
- todo nó não folha com **k** filhos tem **k-1** chaves
- todas as folhas estão no mesmo nível

árvore B de ordem 4 (árvore 2-3-4)



DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Árvore B: propriedades

Máximo de chaves de uma árvore B de ordem **m** e altura **h**

- a raiz tem m-1 chaves e m filhos
- cada nó tem m-1 chaves e m filhos
- cada folha tem m-1 chaves

$$n = (m-1) * (1 + m + m^2 + m^3 + \ldots + m^h)$$

$$= (m-1) * (1 - m^{h+1})/(1 - m)$$

$$= (m-1) * (m^{h+1} - 1) / (m - 1)$$

$$= m^{h+1} - 1$$

$$a + ar + ar^2 + ar^3 + \ldots + ar^{n-1} = a\,\frac{1-r^n}{1-r}$$

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Árvore B: propriedades

Mínimo de chaves de uma árvore B de ordem **m** e altura **h**

- a raiz tem no mínimo 1 chave e 2 filhos
- cada nó interno tem no mínimo m/2 filhos e (m/2) – 1 chaves
- cada folha tem no mínimo (m/2) -1 chaves

nós: $1 + 2 * (1 + m/2 + (m/2)^2 + \ldots + (m/2)^{(h-1)})$

chaves (k) $>= 1 + 2 * ((m/2 - 1) + (m/2 - 1)*(m/2) + \ldots$

$+ (m/2 - 1)*(m/2)^{(h-1)})$

$k >= 1 + (m/2 - 1) * 2 * (1 + m/2 + (m/2)^2 + \ldots + (m/2)^{h-1}$

$k >= 2 * ((m/2)^h) - 1$

Árvore B: propriedades

Mínimo de chaves de uma árvore B de ordem **m** e altura **h**

$$k >= 2 * ((m/2)^h) - 1$$

Altura:

$$2 * ((m/2)^h) - 1 <= k$$

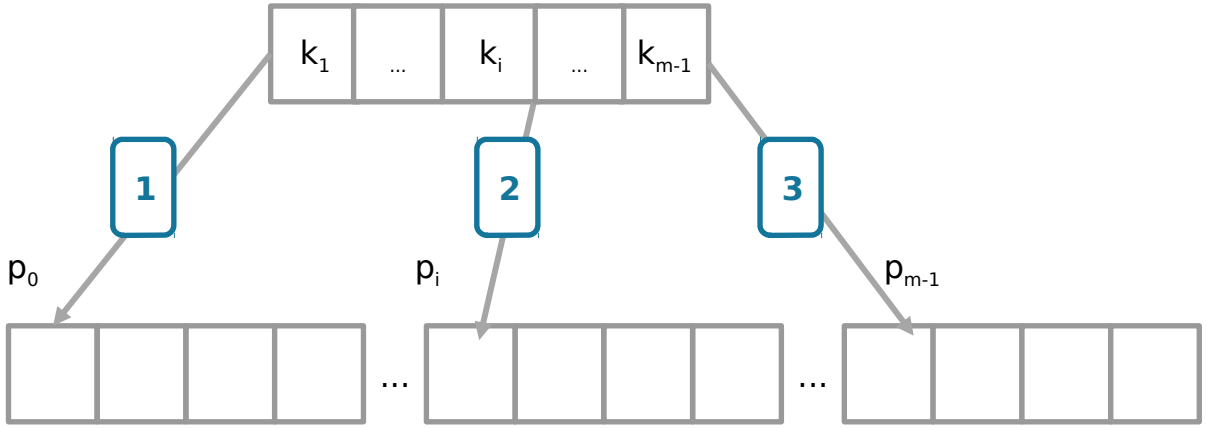
$$(m/2)^h <= (k + 1) / 2$$

$$h <= \log_{[m/2]} (k+1)/2$$

para usufruir dessa altura é necessário que o nó inteiro caiba em um bloco lido do disco
(ou, se na memória, em 1 linha de cache)

Árvore B: Busca

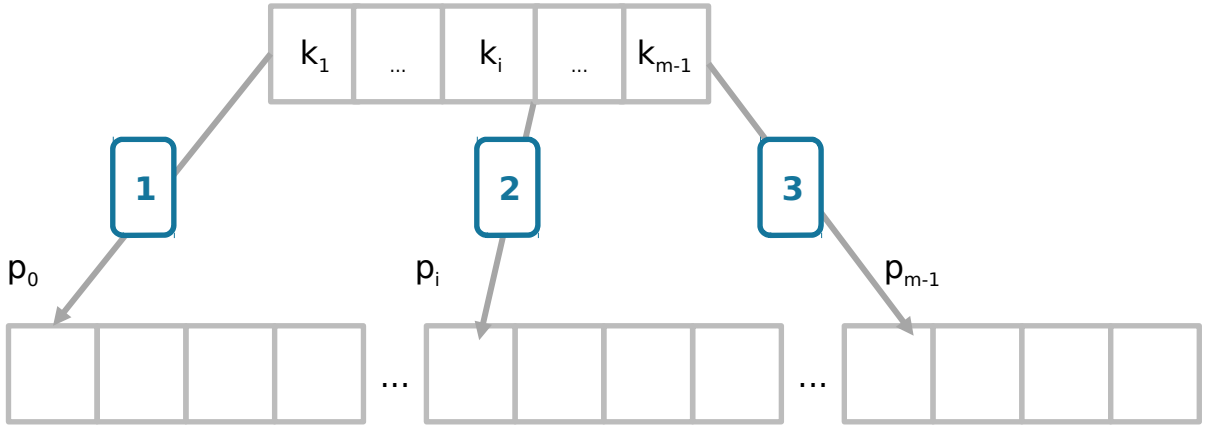
1. Busca a chave entre as chaves de um nó
2. Se não for encontrada, procura o nó para onde deve seguir



Árvore B: Busca

1. Busca a chave entre as chaves de um nó

k_1 … k_{m-1} (se m for grande: busca binária)



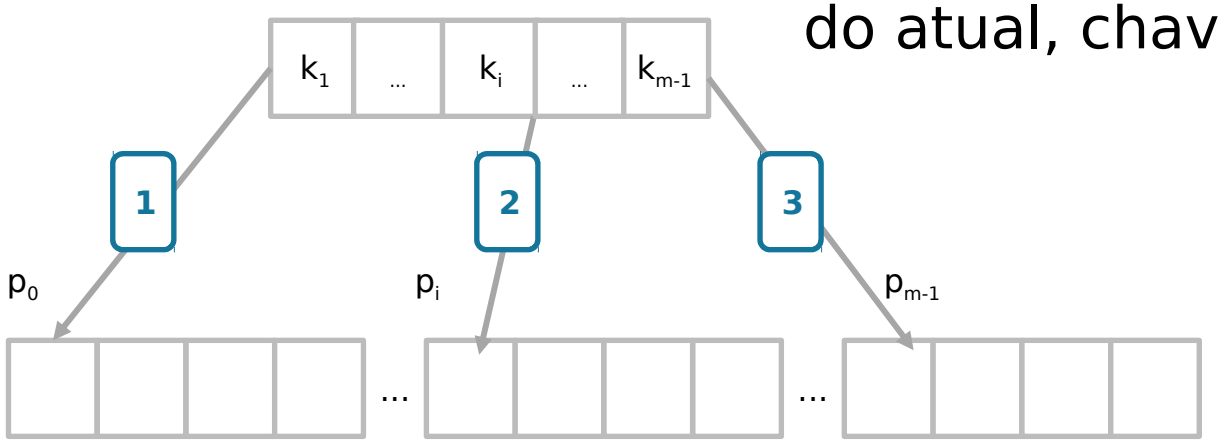
DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Árvore B: Busca

2. Se a chave **x** não for encontrada:

   1. $\mathbf{x < k_1}$ → busca continua no nó $\mathbf{p_0}$
   2. $\mathbf{k_i < x < k_{i+1}}$ para $1 <= i < m-1$ → busca continua no nó $\mathbf{p_i}$
   3. $\mathbf{k_{m-1} < x}$ → busca continua no nó $\mathbf{p_{m-1}}$

→ se não houver nós abaixo do atual, chave não existe



Árvore B: Inserção

1. Procura a chave até chegar a uma folha (se chave não existente)

2. Seja $\mathbf{p_i}$ o nó onde **x** deve ser inserido

    1. se $\mathbf{p_i}$ tiver menos de m-1 elementos ...

    2. se $\mathbf{p_i}$ estiver “lotado” ...

Exemplo: Árvore B de ordem 5

insere 10, 30, 50, 70, 90

10				

10	**30**			

10	30	**50**		

10	30	50	**70**	

10	30	50	70	**90**

1. Procura a chave até chegar a uma folha
2. Seja $\mathbf{p_i}$ o nó onde **x** deve ser inserido
    1. se $\mathbf{p_i}$ tiver menos de m-1 elementos

       → insere em **pi** na posição adequada
    2. se $\mathbf{p_i}$ estiver “lotado” ...

Árvore B: Inserção

2. Seja $\mathbf{p_i}$ o nó onde **x** deve ser inserido

   1. se $\mathbf{p_i}$ tiver menos de m-1 elementos

      → insere em **pi** na posição adequada

   2. se $\mathbf{p_i}$ estiver “lotado”:

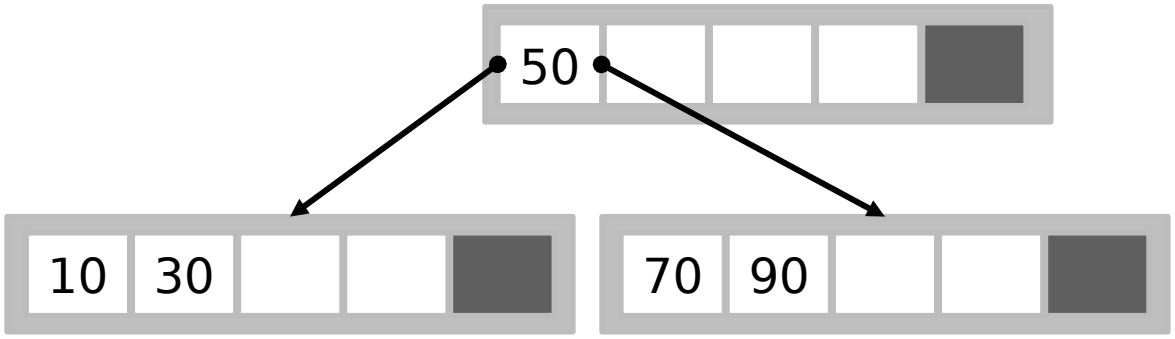
      1. aloca um novo nó $\mathbf{p_k}$
      2. distribui as **m** chaves da seguinte maneira:
         - [m/2]-1 menores chaves em $\mathbf{p_k}$
         - m-[m/2] maiores chaves em $\mathbf{p_i}$
         - insere a chave mediana (em [m/2]) no nó superior (se $\mathbf{p_i}$ for raiz, cria nova raiz)

Exemplo: Árvore B de ordem 5

insere 90

se $\mathbf{p_i}$ estiver "lotado":

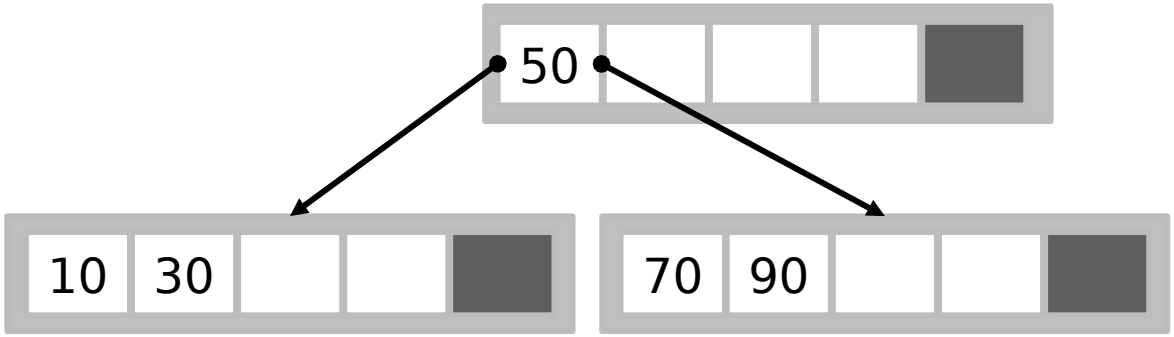
- aloca um novo nó $\mathbf{p_k}$
- distribui as **m** chaves:
  - [m/2]-1 menores chaves em $\mathbf{p_k}$
  - m-[m/2] maiores chaves em $\mathbf{p_i}$
  - insere a chave mediana no nó superior (se $\mathbf{p_i}$ for raiz, cria nova raiz)

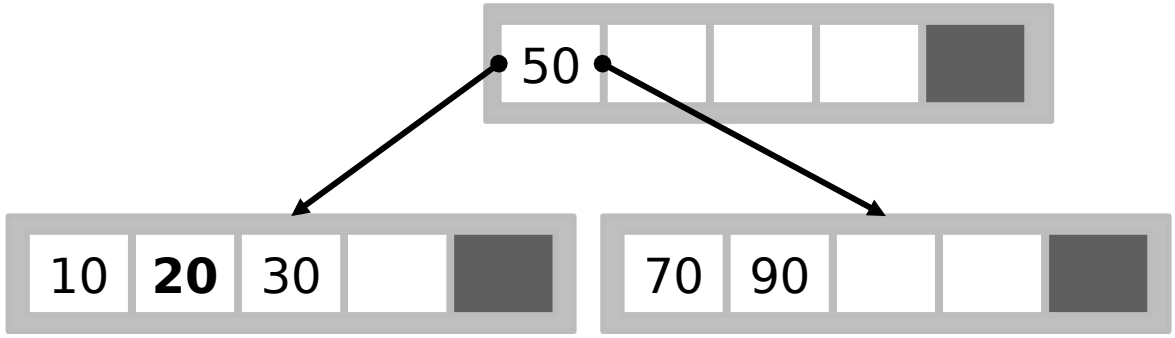


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

insere 20

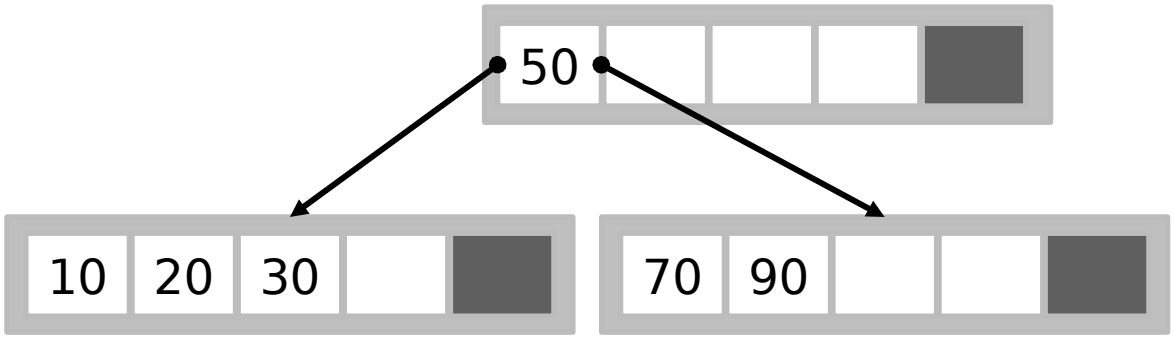


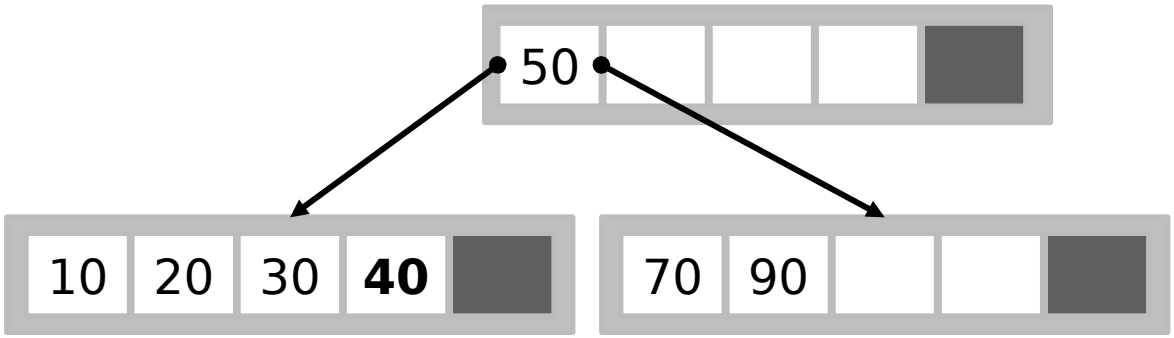


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

insere 40

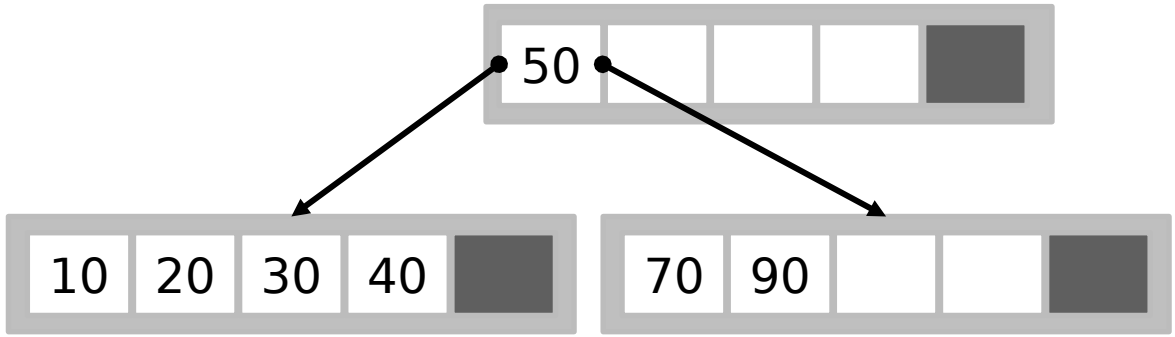


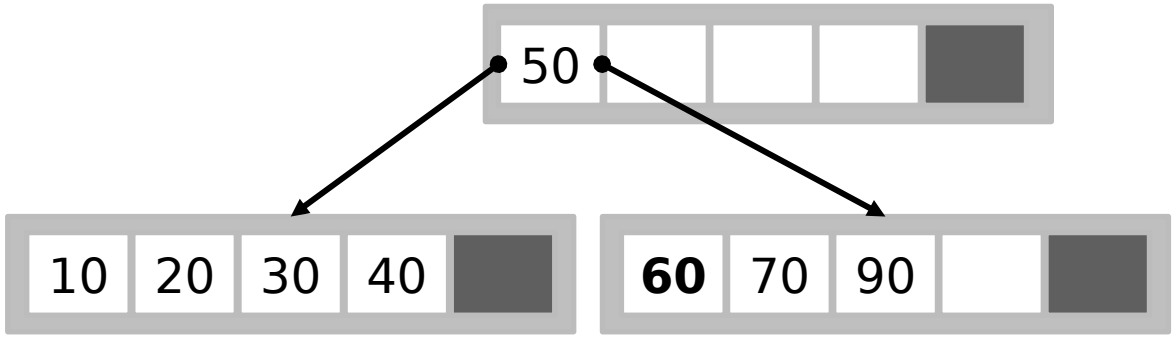


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

insere 60

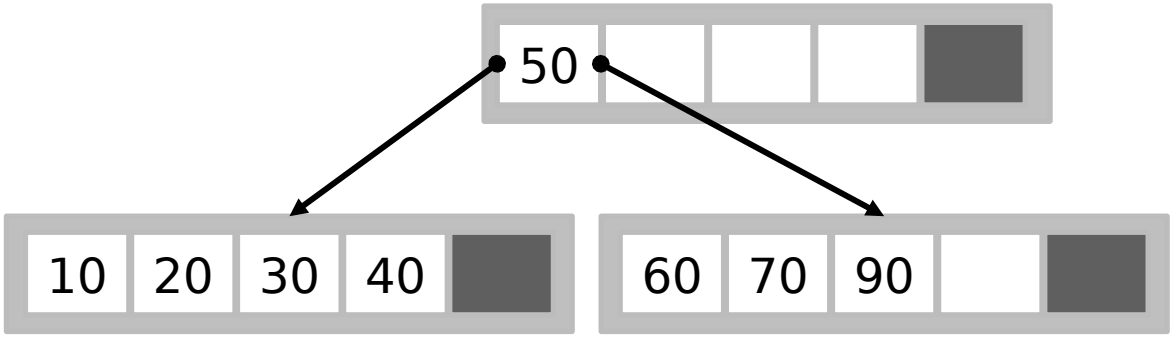


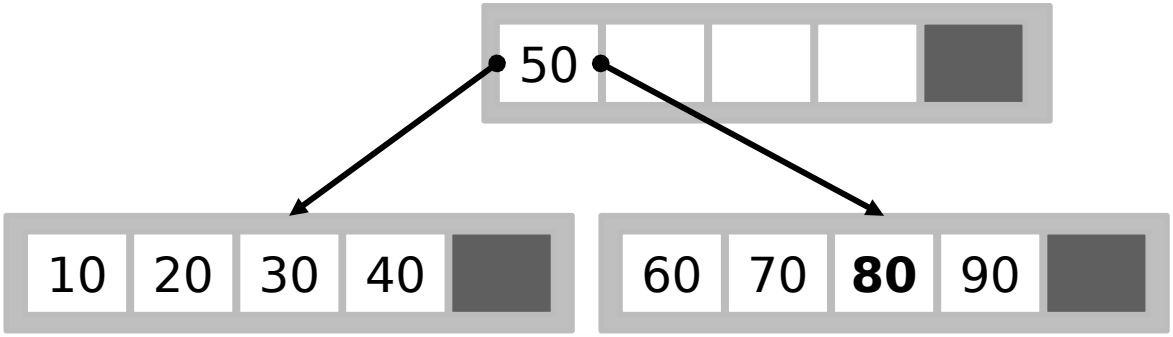


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

insere 80

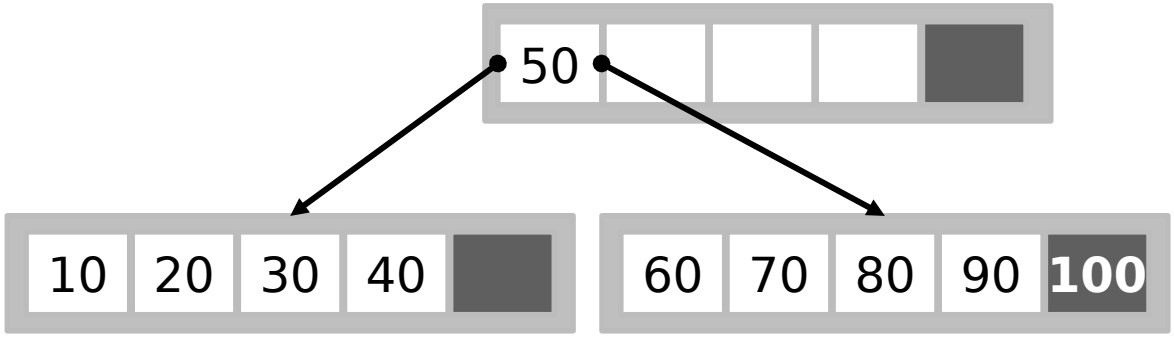


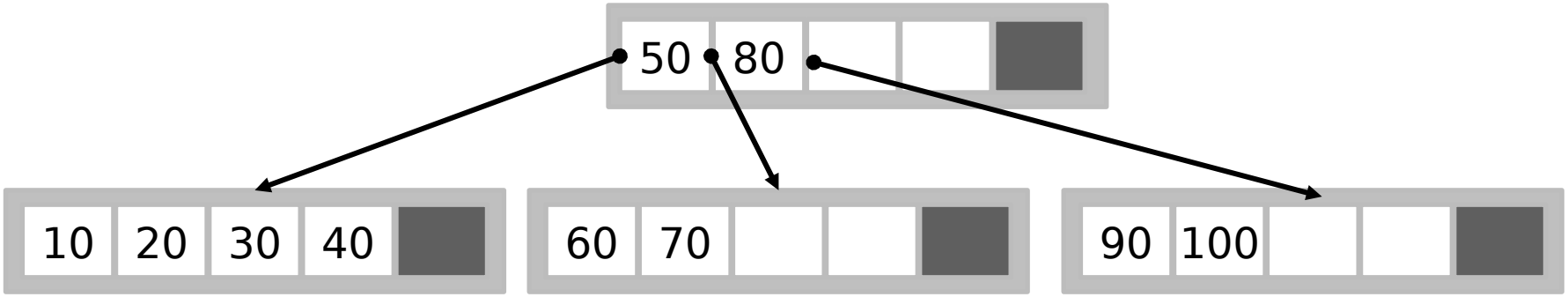


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

insere 100



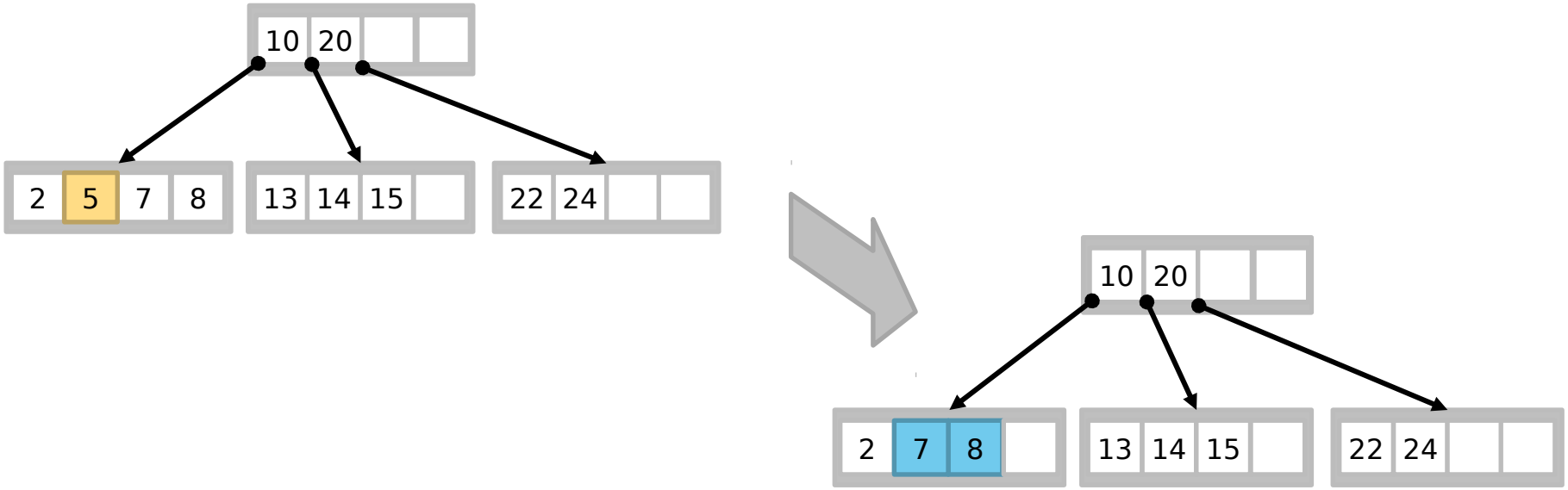


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Árvore B: Remoção

Deve ser realizada em um nó folha

caso simples:

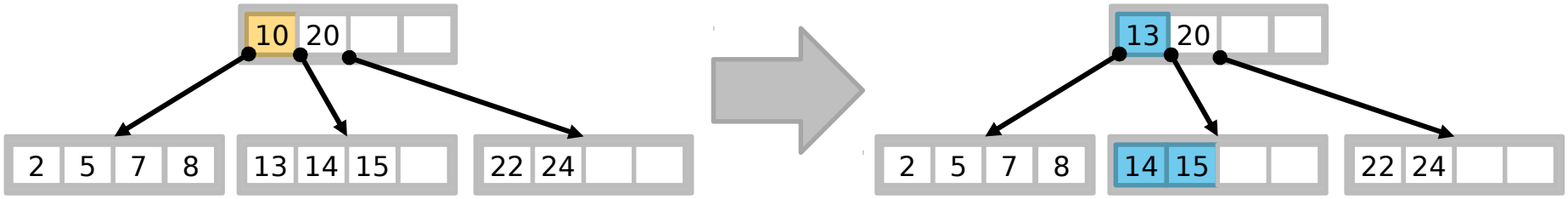


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Árvore B: Remoção

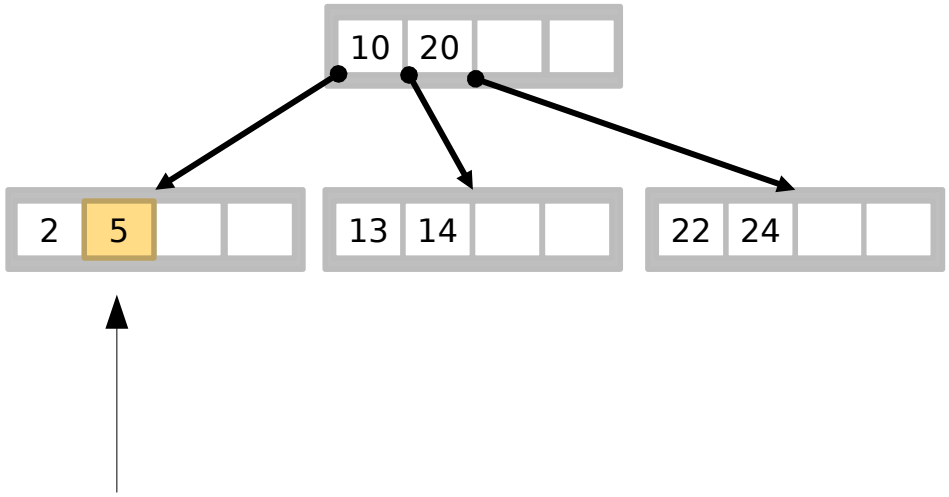
Deve ser realizada em um nó folha

→ se a chave a ser removida não estiver em um nó folha, substitua-a pela maior chave da sua subárvore à esquerda ou a menor chave da sua sub-árvore à direita



Remoção de chave em uma folha

Se a folha ficar com menos de [m/2] – 1 chaves, deve-se realizar uma **concatenação** ou uma **redistribuição**

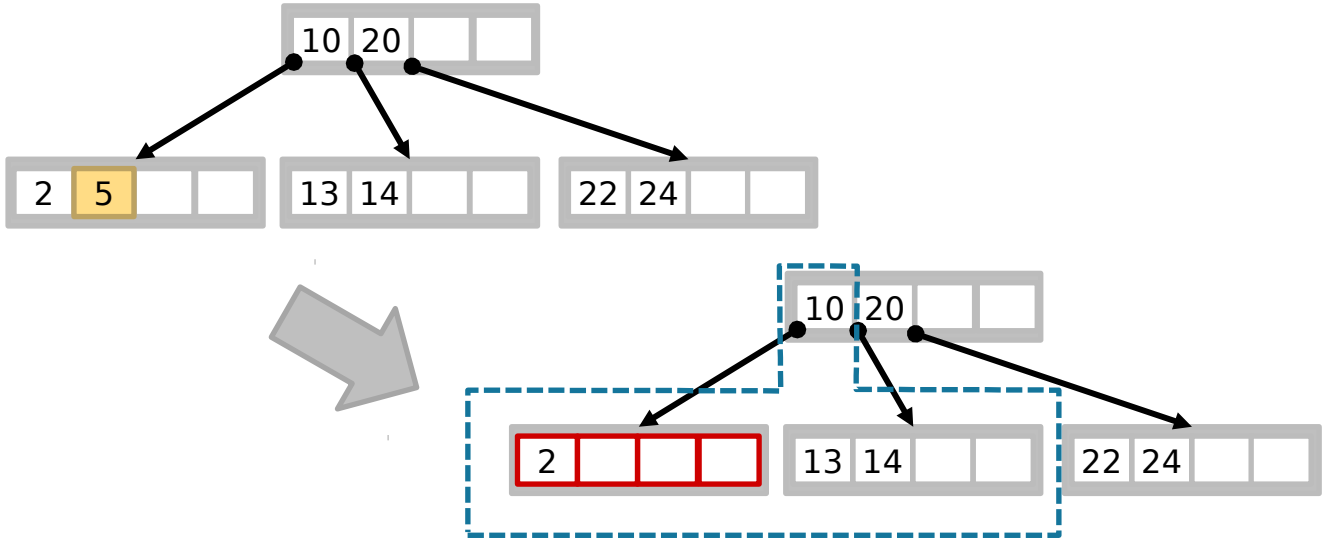


folha ficará com 1 chave → menos que 2!

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

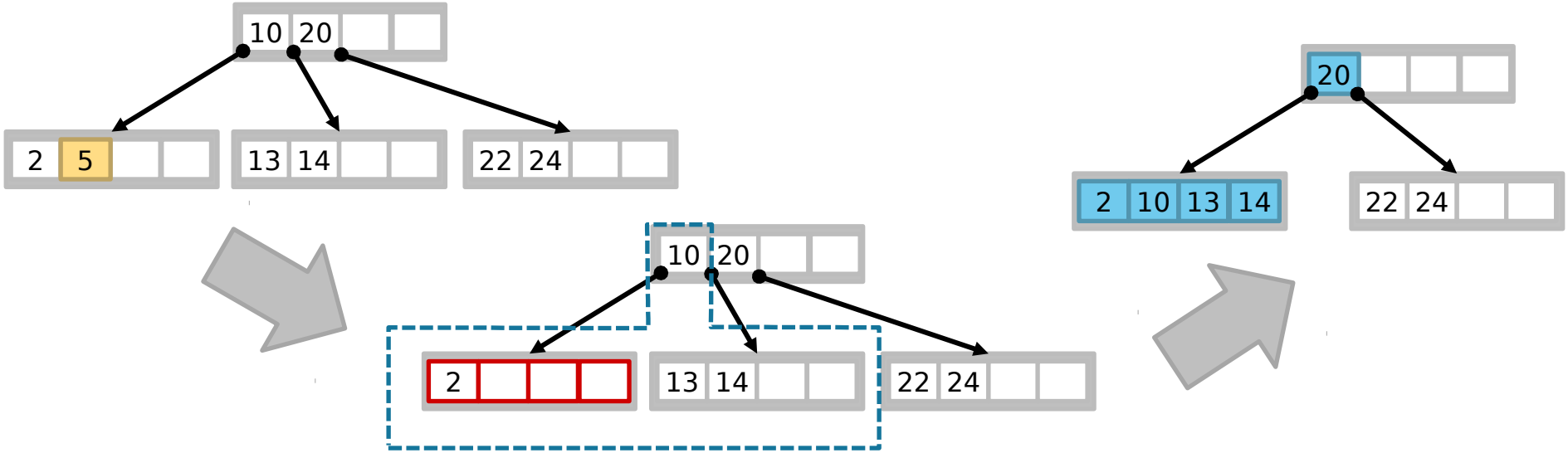
Concatenação

Ocorre quando, após a remoção, o nó de onde a chave foi removida e um nó adjacente têm, em conjunto, menos de **m-1** chaves



Concatenação

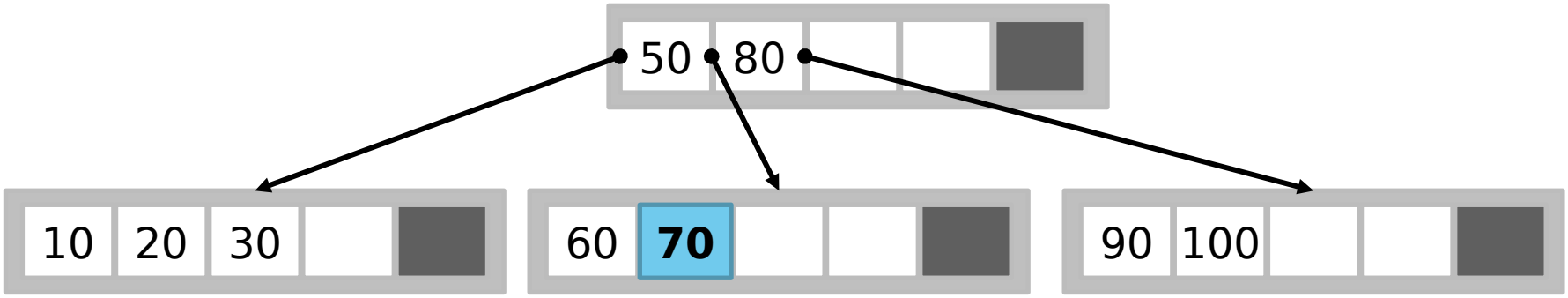
Ao concatenar os nós adjacentes, a chave do “pai” entre os nós concatenados desce para a concatenação

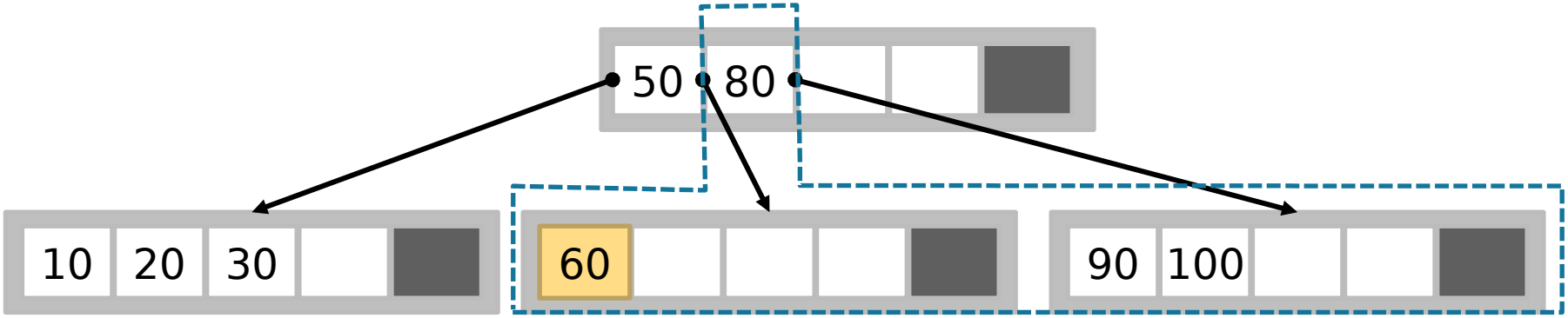


Pode-se precisar repetir o procedimento se o nó ficar com menos de m/2 chaves (chegando até a raiz, se necessário)

Exemplo: Árvore B de ordem 5

remove 70

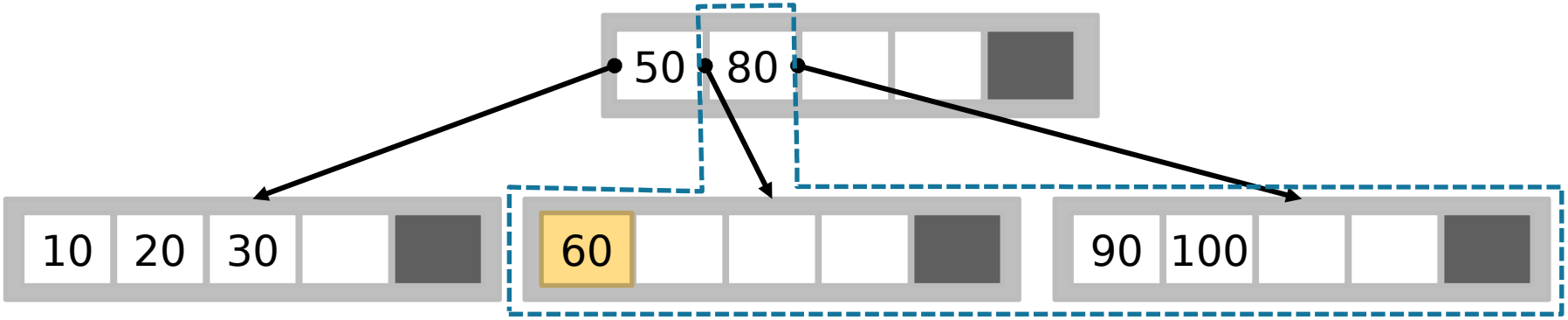


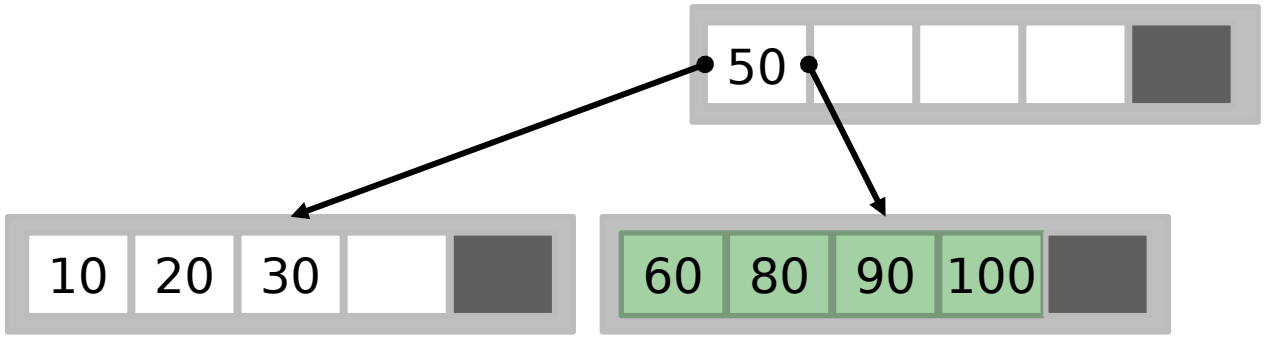


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

remove 70

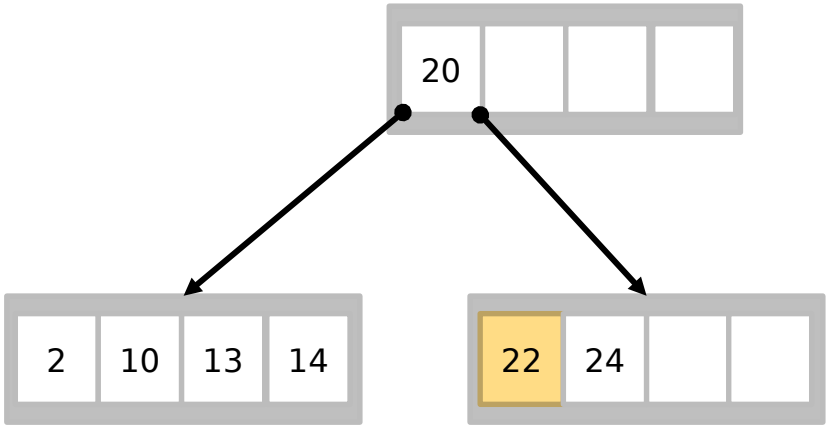




DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Redistribuição

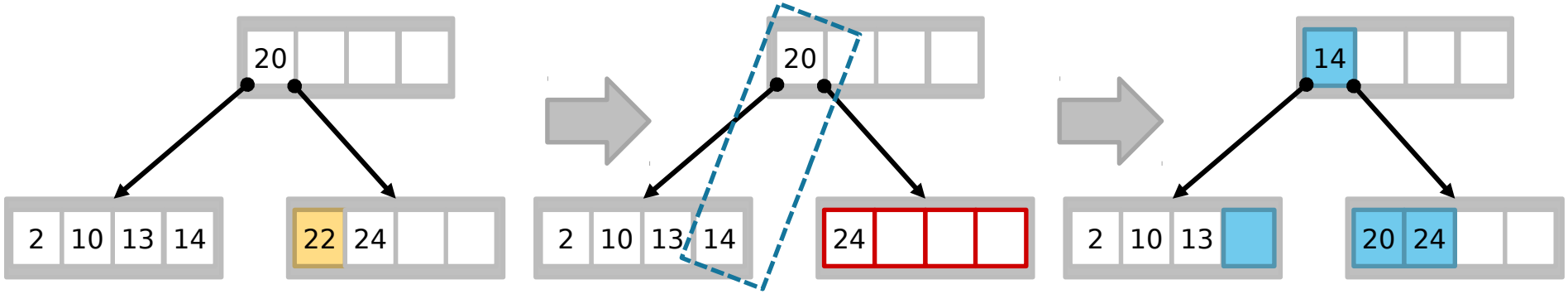
Ocorre quando, após a remoção, o nó de onde a chave foi removida e um nó adjacente têm, em conjunto, **m** chaves, ou mais



Redistribuição

Move-se a chave do nó pai para o nó “diminuído” e a chave do nó adjacente para o nó pai

Não há propagação porque o número de chaves do nó pai não muda

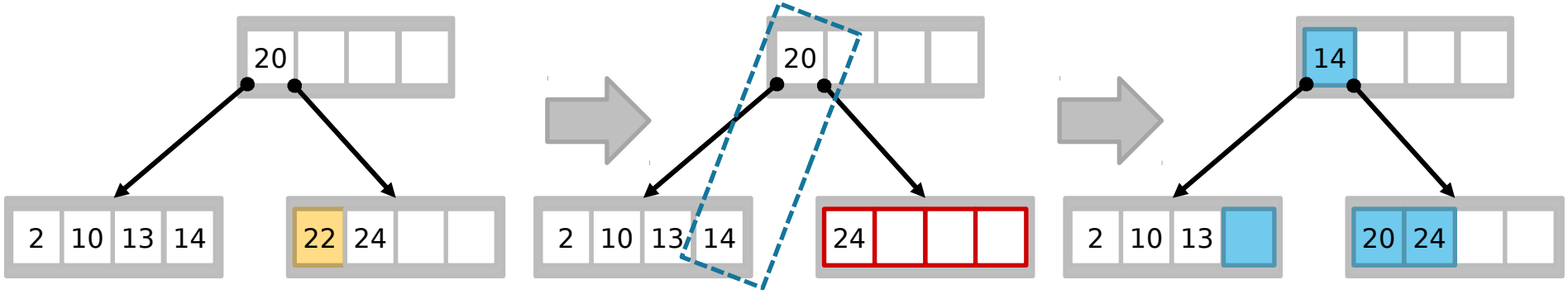


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Redistribuição

Move-se a chave do nó pai para o nó “diminuído” e a chave do nó adjacente para o nó pai

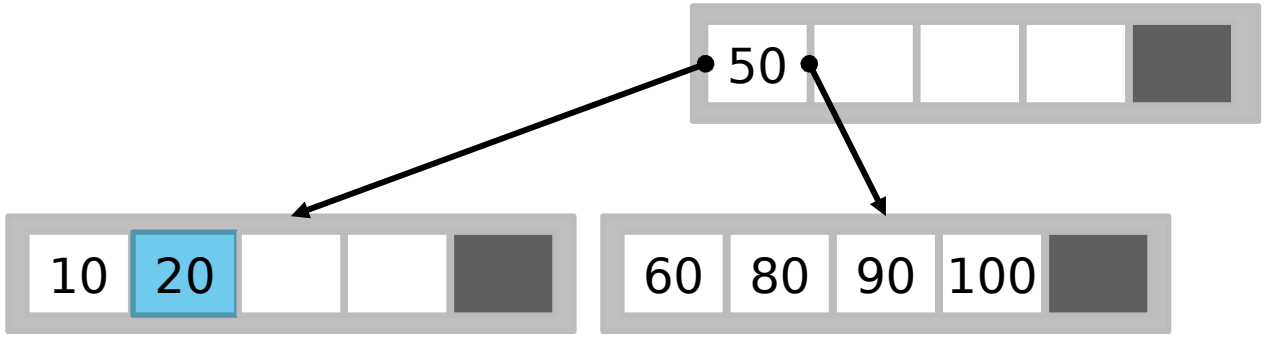
- se o nó adjacente estiver à esquerda, a chave movida é a maior daquele nó (*borrow from left*).
- se o nó adjacente estiver à direita, a chave movida é a menor daquele nó (*borrow from right*).

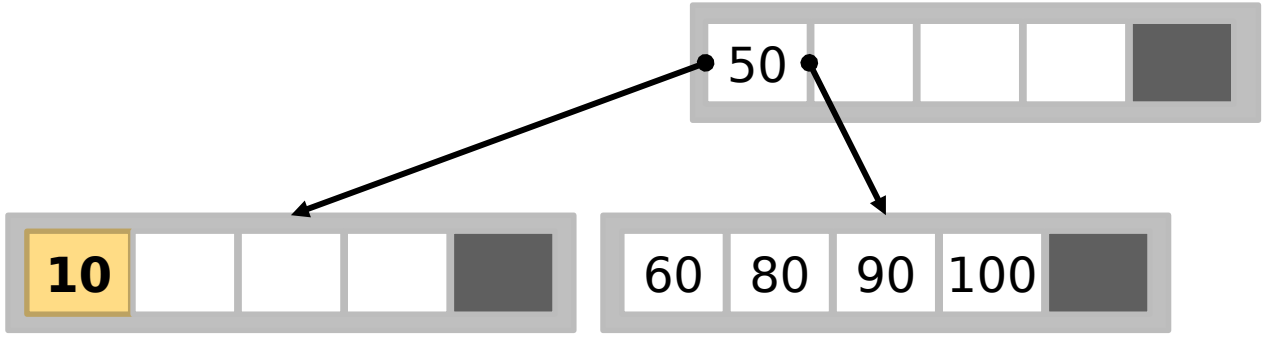


DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

remove 20

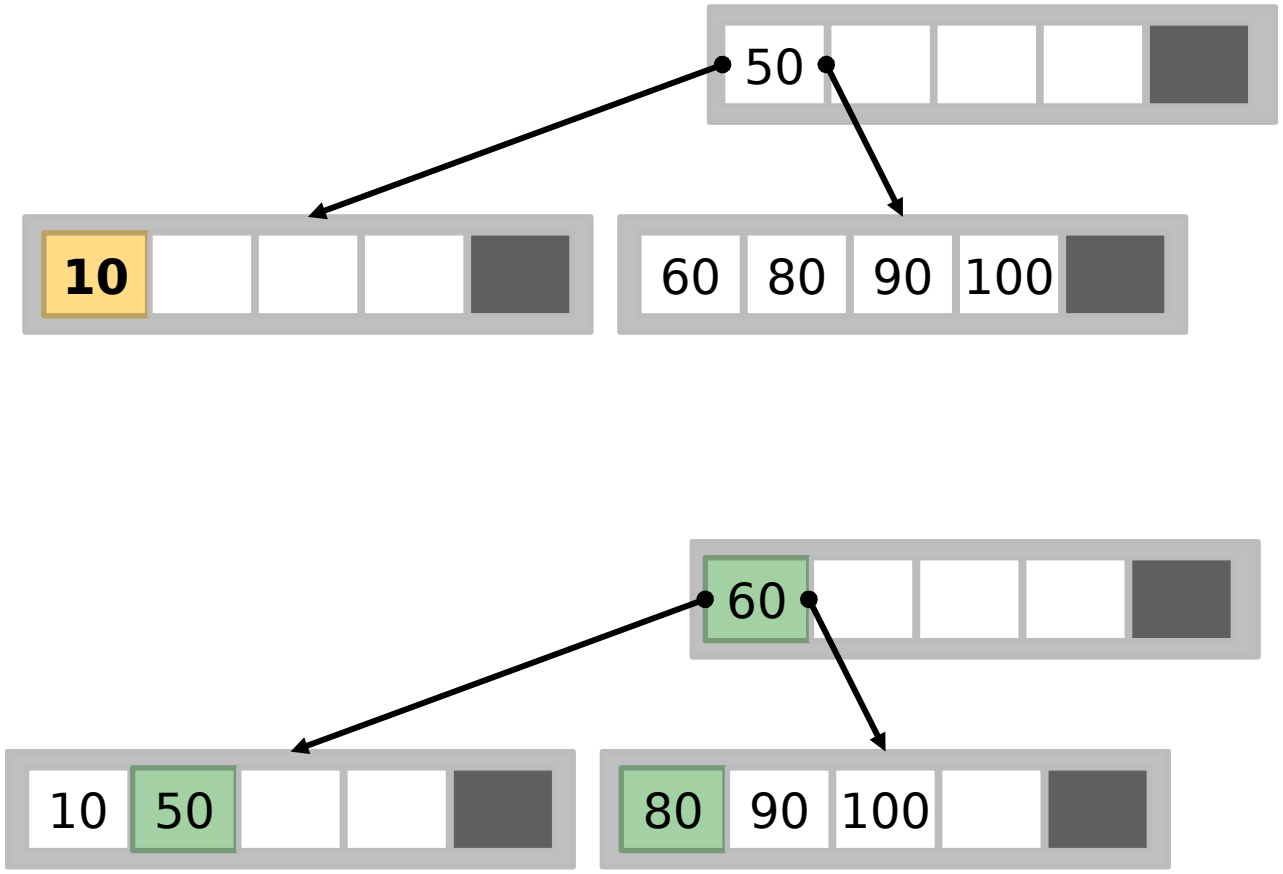




DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo: Árvore B de ordem 5

remove 20



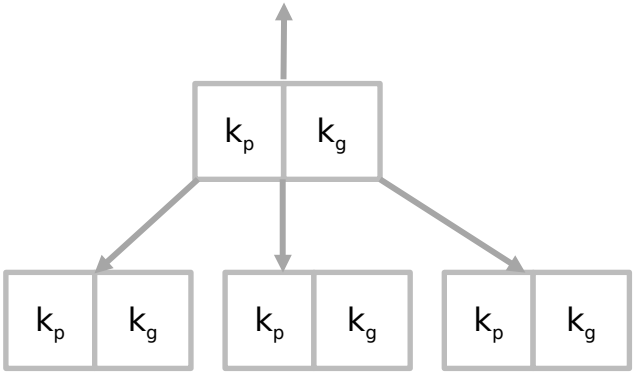
DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Implementação de árvore 2-3

Árvores B de ordem **3**: árvores **2-3**

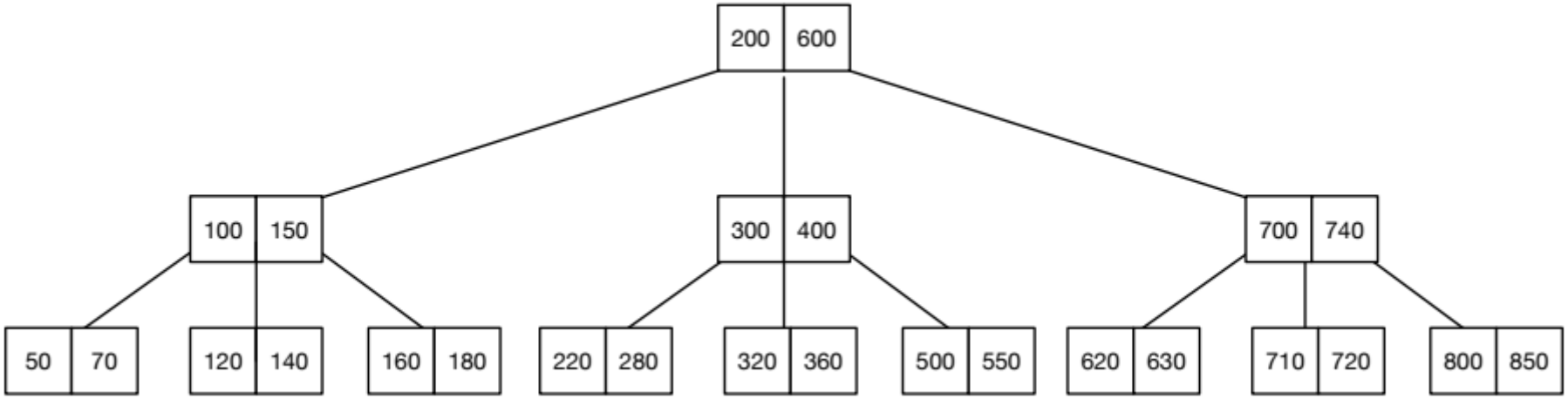
- cada nó não folha tem no mínimo uma chave e dois filhos
- cada nó não folha tem no máximo duas chaves e três filhos

```
struct smapa {

   int kp, kg;   /* chaves: kp<kg, se kg existir.
                    se kg = -1, chave não existe */
   int dado;
   struct smapa *pai;
   struct smapa *esq;
   struct smapa *meio;
   struct smapa *dir;

};
```



DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo de Árvore 2-3



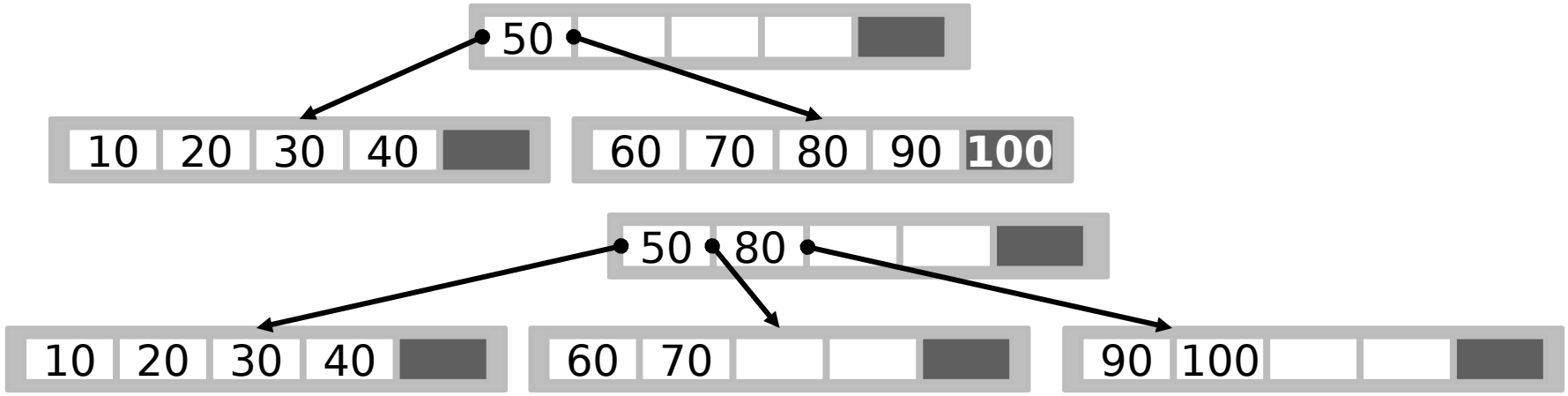
Ideia de Inserção

1. se há espaço no nó (não tem as duas chaves), coloca a chave neste nó

   → pode ser necessário “empurrar” a chave existente para a direita

2. se não há espaço, deve-se “quebrar” o nó em dois

   → neste caso o valor mediano deve “subir” (e o ponteiro para o novo nó)



DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

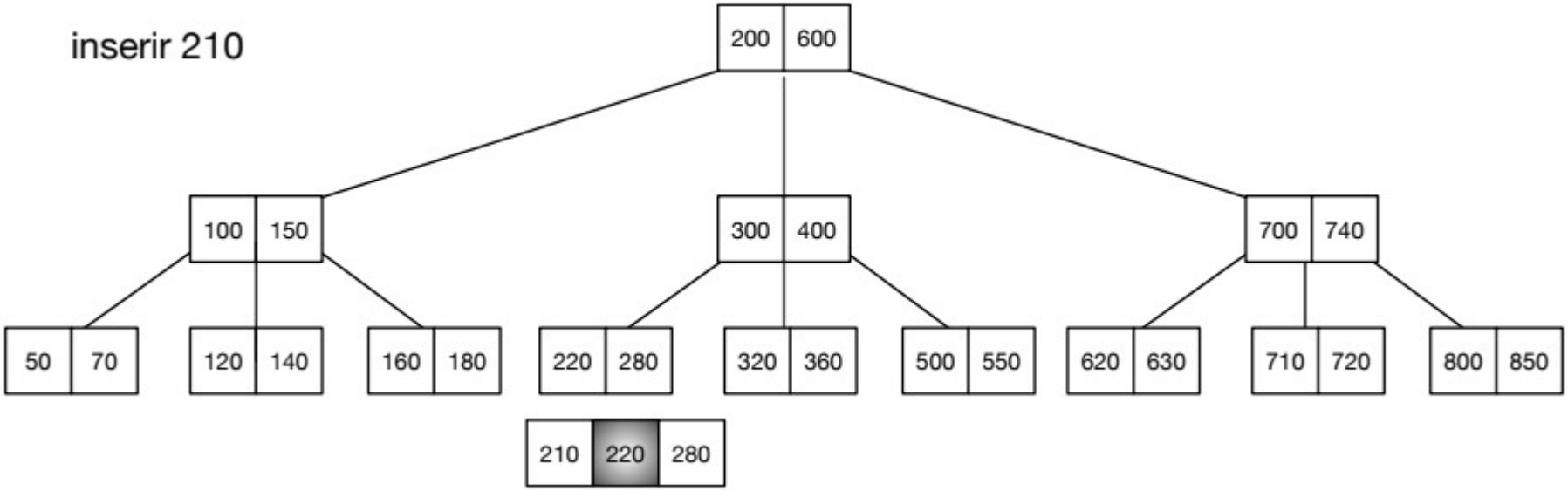
Implementação de Inserção

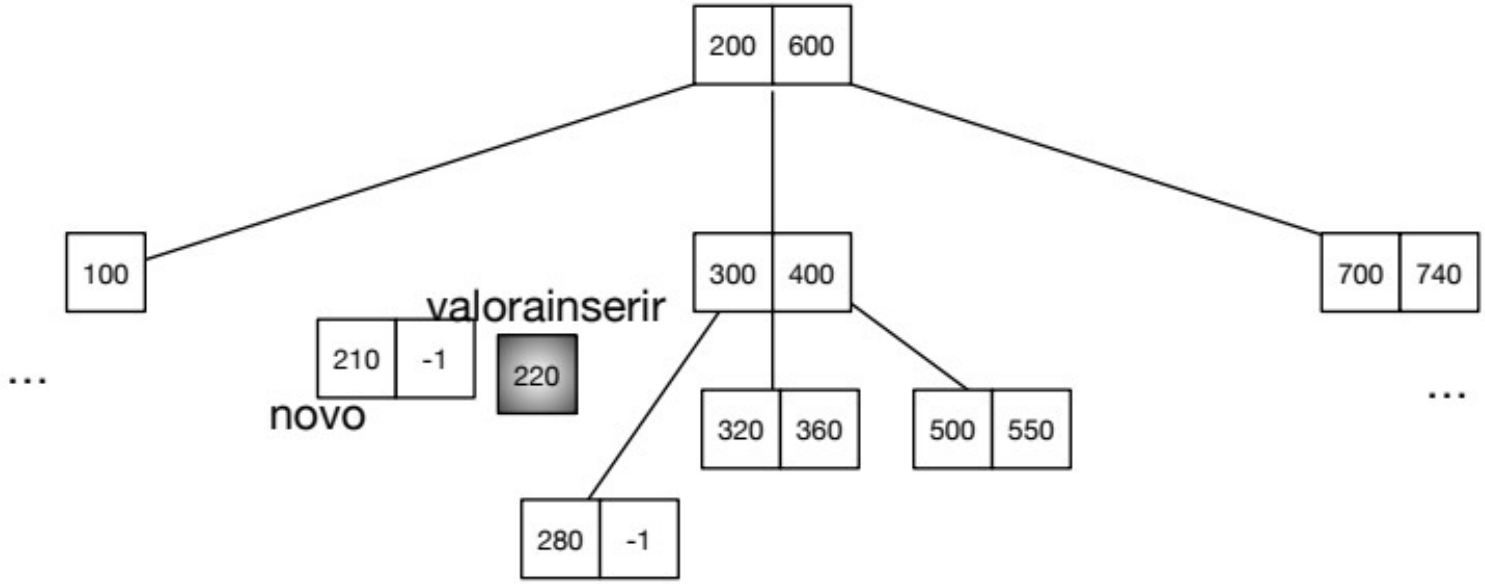
```
Mapa* insere (Mapa* m, int chave) {
  int valorquesubiu;
  Mapa* novofilho, novaraiz;

  if (m==NULL) {
    m = novoNo (chave);
    m->pai = novoNo (-1);
  }
  else {
    if (insere2 (m, chave, &valorquesubiu, &novofilho)) {
      /* cria nova raiz */
      novaraiz = novoNo (valorquesubiu);
      novaraiz->pai = m->pai;
      novaraiz->esq = novofilho;
      novaraiz->esq->pai = novaraiz;
      novaraiz->meio = m;
      novaraiz->meio->pai = novaraiz;
      m = novaraiz;
    }
  }
```

```
static Mapa* novoNo (int chave) {
  Mapa *m =
   (Mapa*)malloc(sizeof(struct smapa));
  if (m==NULL) return NULL;
  m->pai = NULL;
  m->kp = chave;
  m->kg = -1;
  m->esq = m->meio = m->dir = NULL;
  return m;
}
```

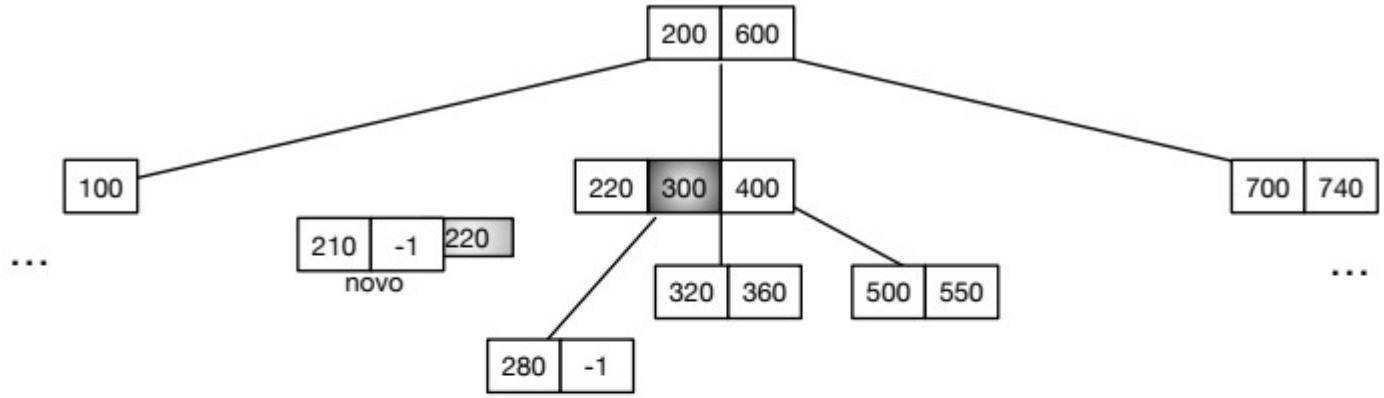
DEPARTAMENTO DE INFORMÁTICA PUC·RIO

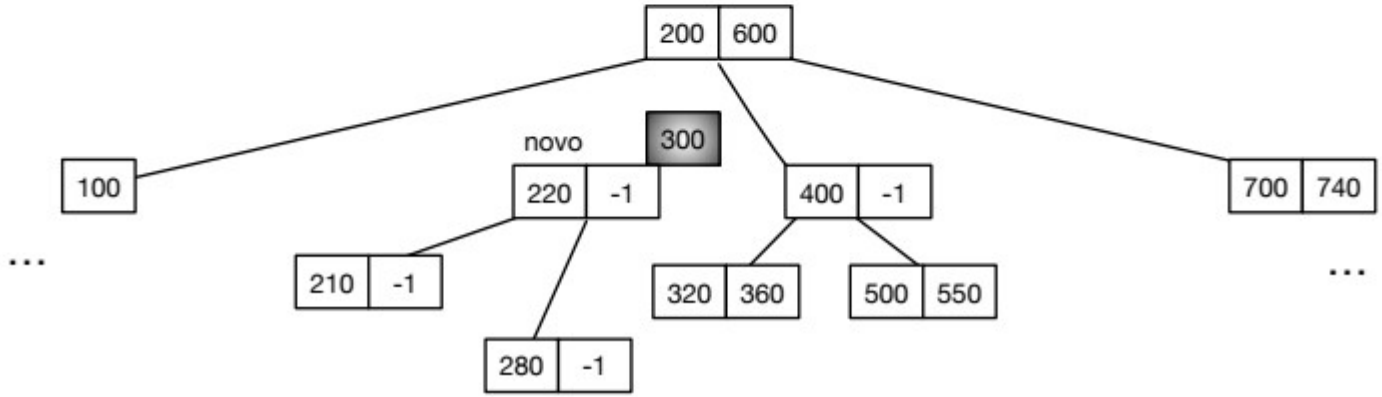
Exemplo de Árvore 2-3



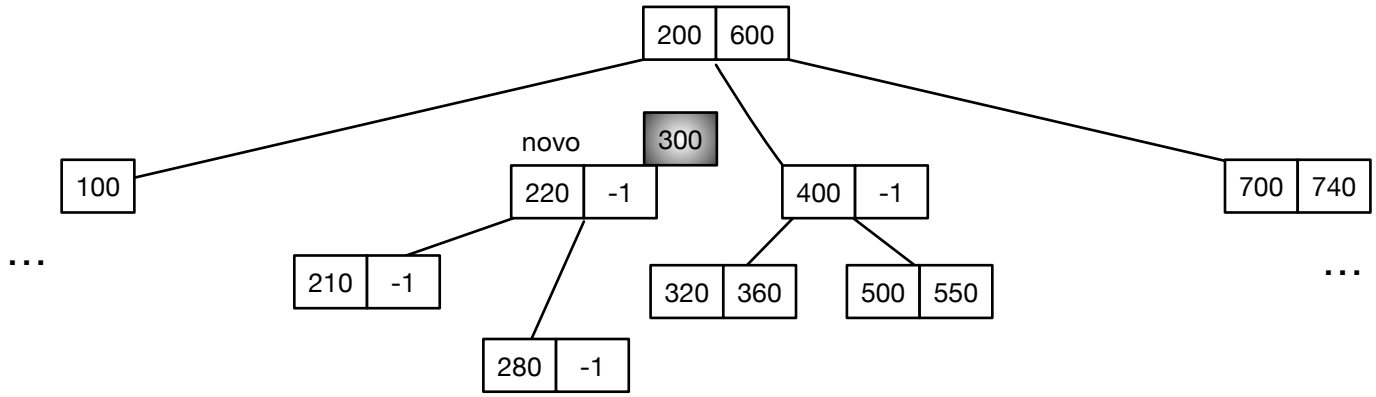


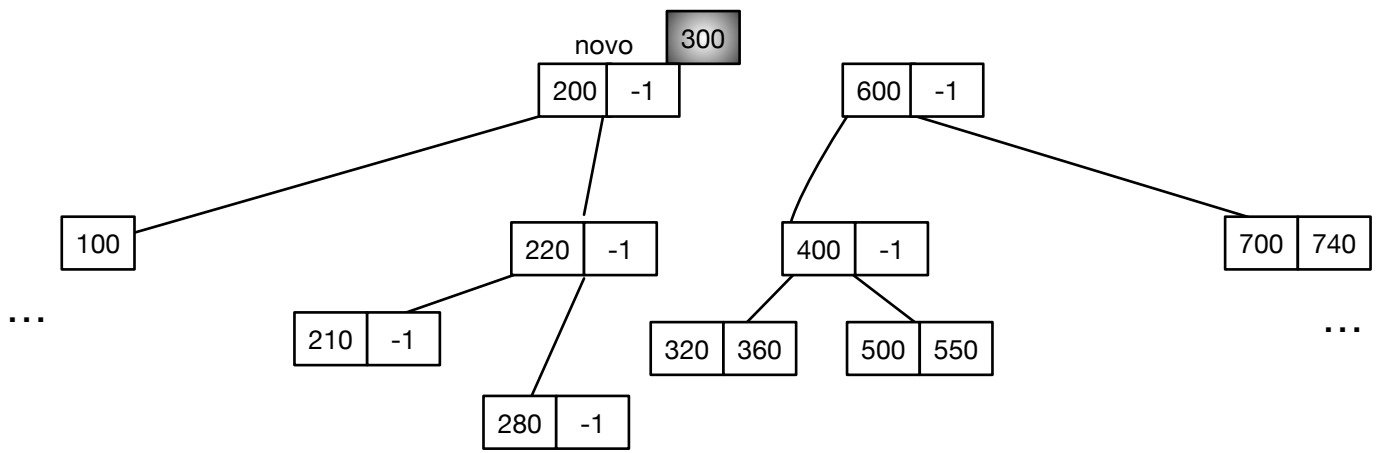
Exemplo de Árvore 2-3



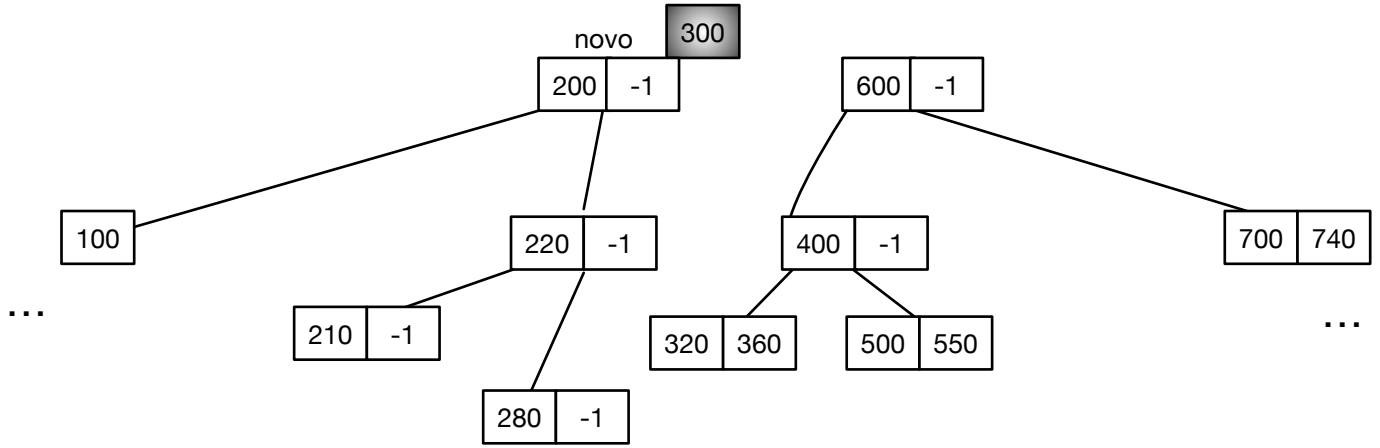


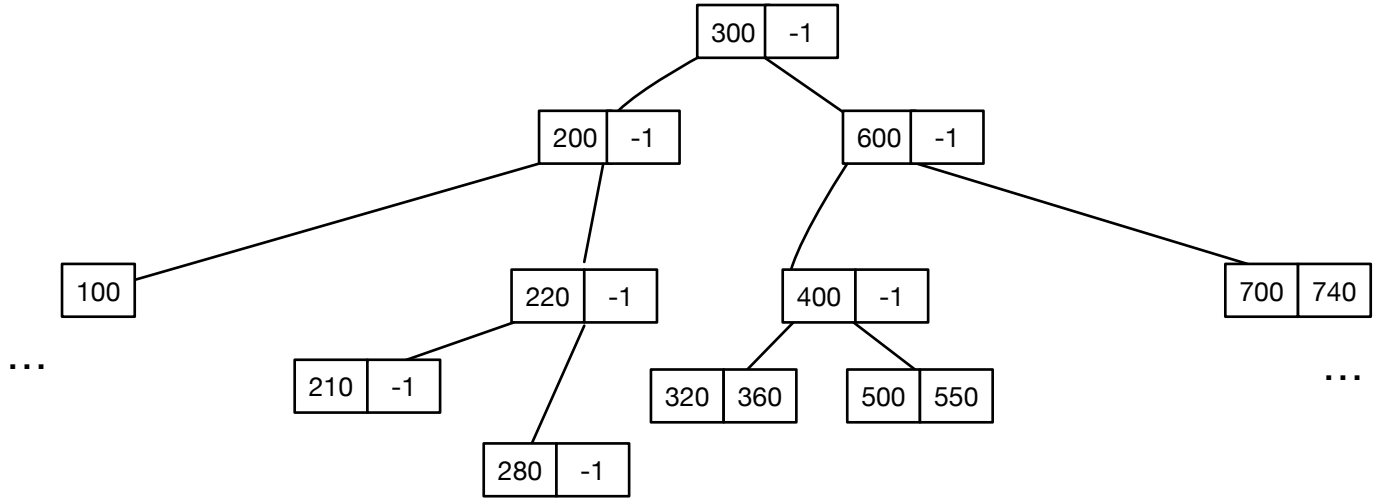
Exemplo de Árvore 2-3





Exemplo de Árvore 2-3





DEPARTAMENTO
DE INFORMÁTICA
PUC·RIO

Implementação de Inserção

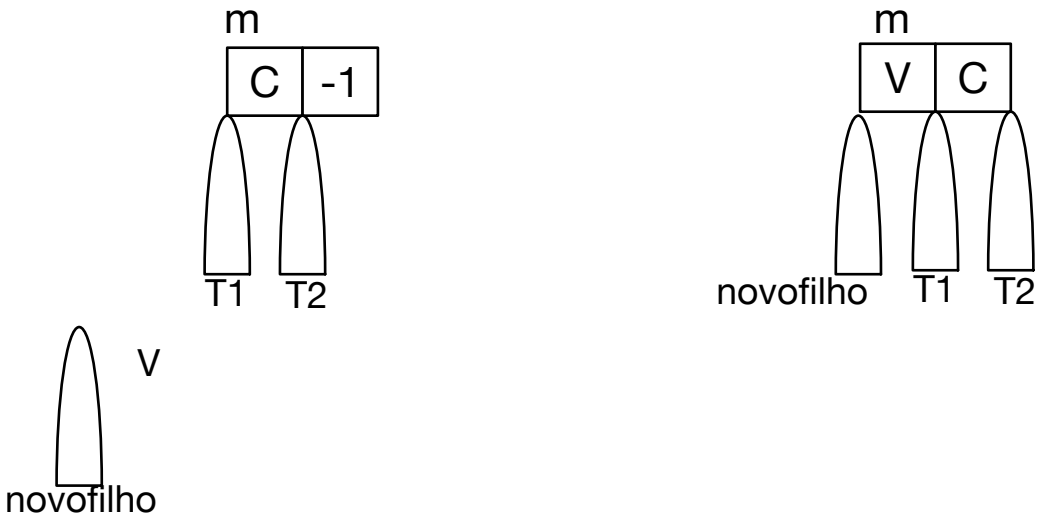
```
static int insere2 (Mapa* m, int chave, int* valorainserir, Mapa** novofilho) {

  int inseriraqui = 0; /* indica se deve inserir neste nó */

  if (m==NULL) { printf("erro! subárvore nula! \n"); exit (1);}

  if (m->esq != NULL) { /* não é folha, só insere aqui se subir um valor  */

    if (chave < m->kp) {
      inseriraqui = insere2(m->esq, chave, valorainserir, novofilho);
    }
    else if (((m->kg != -1) && (chave < m->kg)) || (m->kg == -1)) {

      /* ou está entre as duas chaves ou só tem uma chave no nó */
      inseriraqui = insere2(m->meio, chave, valorainserir, novofilho);
    }
    else { /* chave > m->kg */
      inseriraqui = insere2(m->dir, chave, valorainserir, novofilho);
    }
  }
  else { /* este nó é folha, tem que inserir nele de qq jeito */
    *valorainserir = chave;
    inseriraqui = 1;
    *novofilho = NULL;
  }

  if (!inseriraqui) return 0; /* inserção já está completa */
```

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Implementação de Inserção

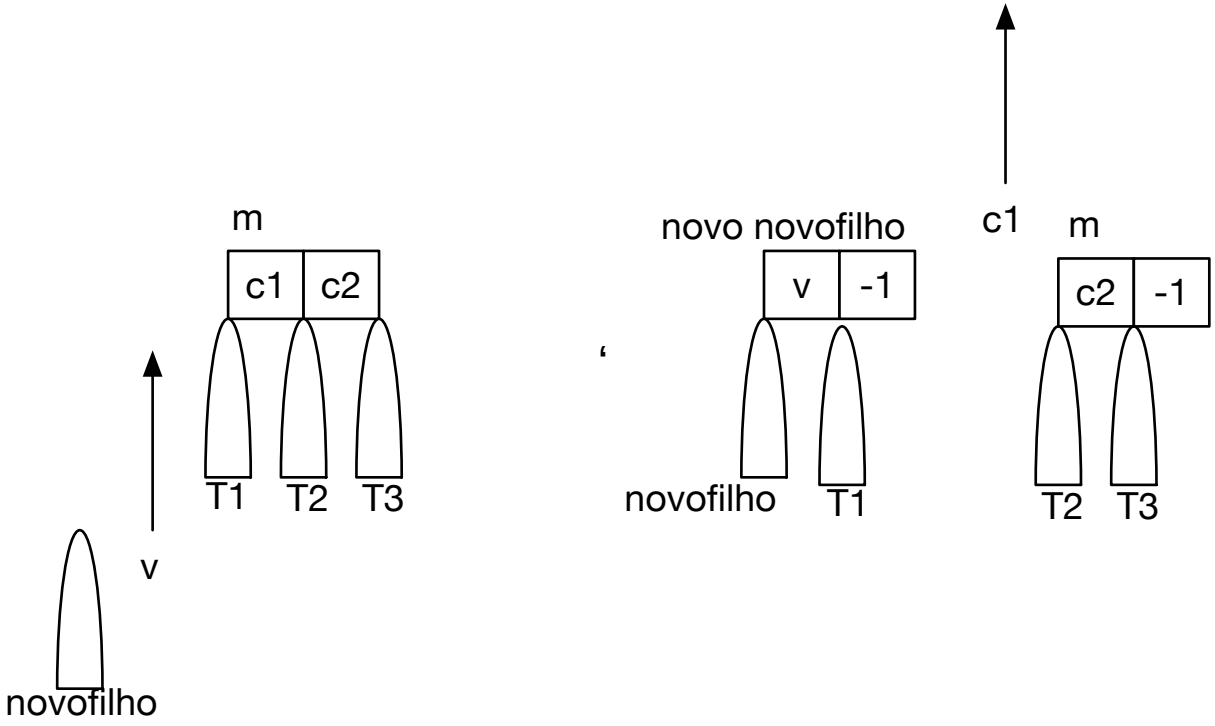
```
if (!inseriraqui) return 0; /* inserção já está completa */

  /* procura espaço no nó */

  if (m->kg==-1) {  /* tem  espaço no nó */
    if (*valorainserir < m->kp) { /* empurra chave kp */
      …
    }
    else { /* é maior que a chave que já está lá */
      …

    }
    return 0; /* como havia espaço, não sobem valores */
  }

  *novofilho = overflowQuebra (m, valorainserir, *novofilho);

  return 1; /* se há quebra, sobe mediana para nova inserção */
}
```

Há espaço no nó



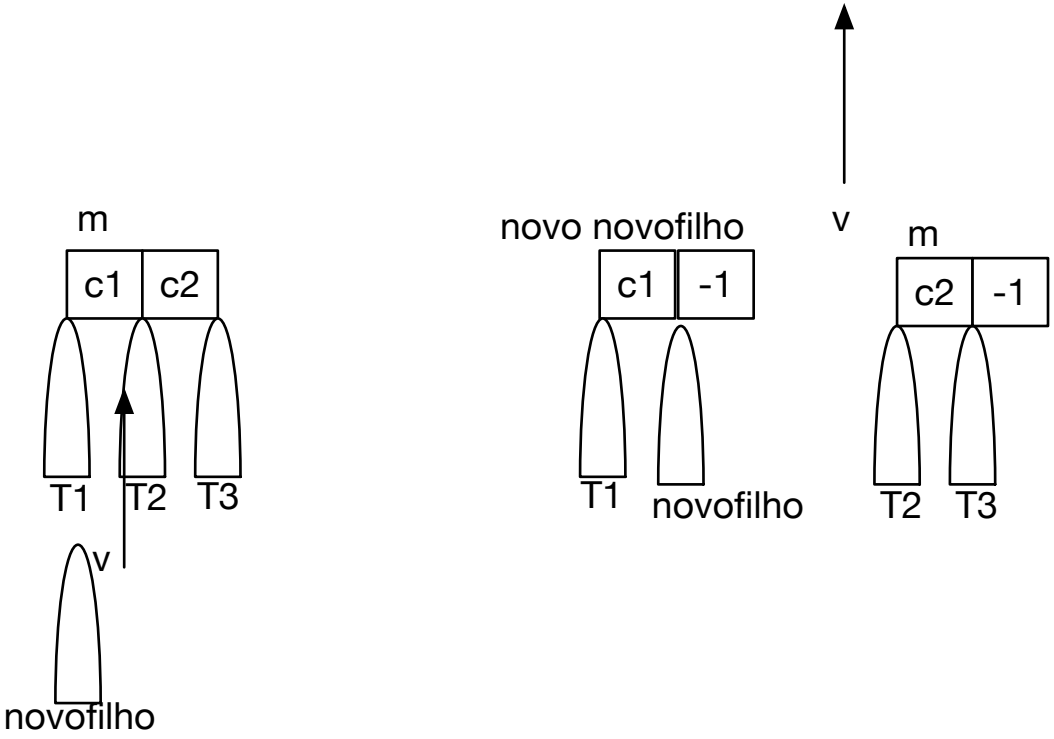
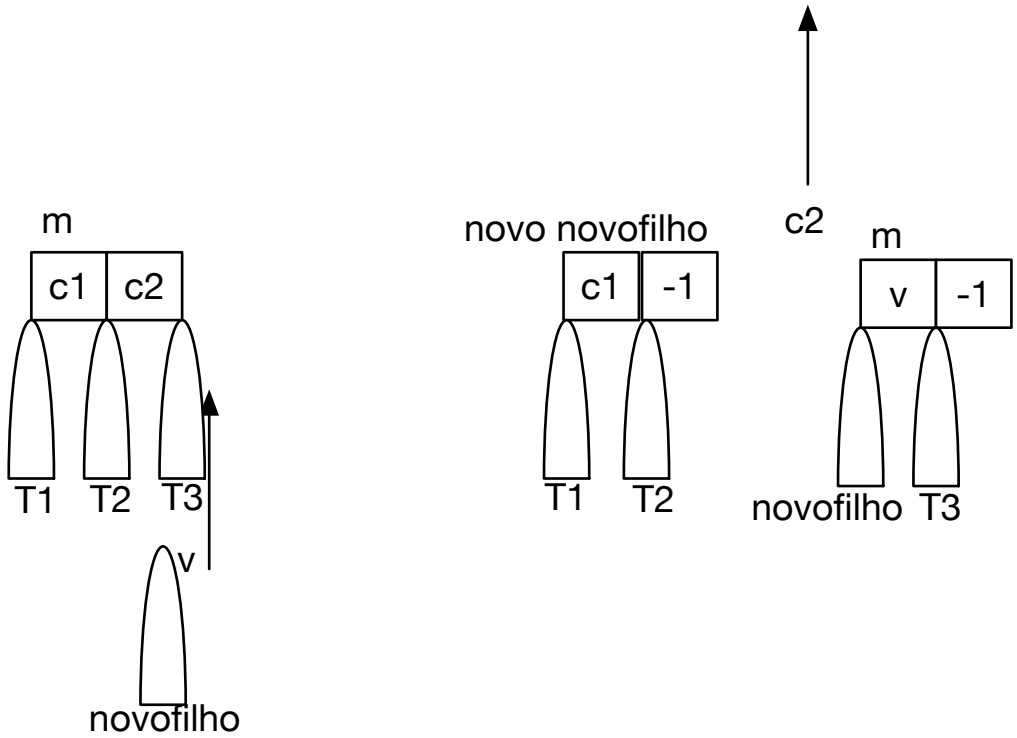
DEPARTAMENTO
DE INFORMÁTICA
PUC·RIO

Propagação da mediana (1)



DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Propagação da mediana (2)



Propagação da mediana (3)