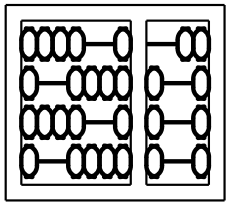


Universidade Estadual de Campinas - UNICAMP
Instituto de Computação - IC

Árvores B

Tópicos

• Árvores de busca *n-árias*, ou seja, possuem mais de dois descendentes por nó denominado **página**.

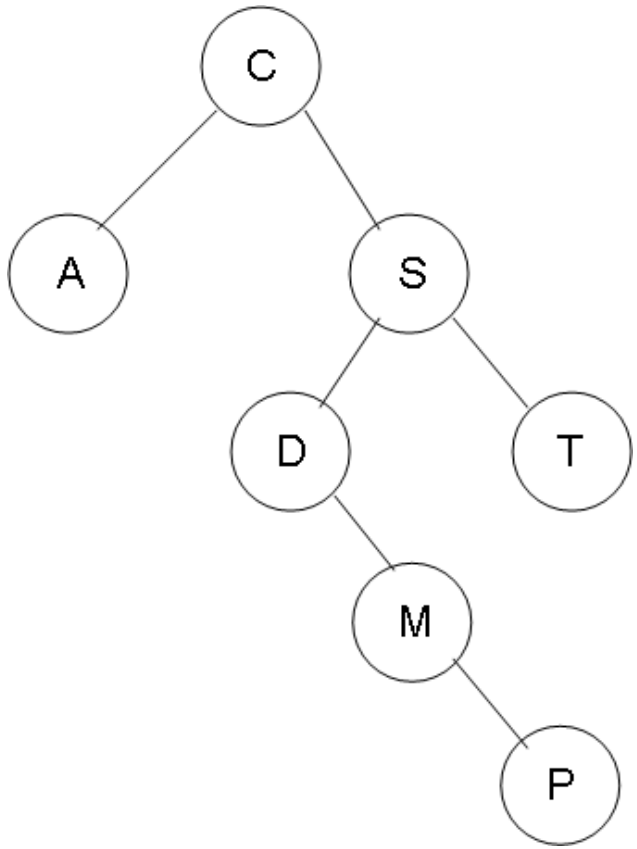
• Problema da árvore binária de busca: construção "top-down" da árvore $\Rightarrow$ desbalanceamento.

• Problema da árvore binária de busca: construção "top-down" da árvore $\Rightarrow$ desbalanceamento.

Exemplo: $C\ S\ D\ T\ A\ M\ P$

• Problema da árvore binária de busca: construção "top-down" da árvore $\Rightarrow$ desbalanceamento.

Exemplo: *C S D T A M P*



- Idéia: construção "bottom-up" da árvore: **Árvores B**

1. Construção da árvore a partir das folhas.
2. Uma página da árvore consiste de uma sequência ordenada de chaves e de um conjunto de ponteiros.
3. O número de ponteiros em uma dada página é igual ao *número de chaves + 1* $\Rightarrow$ **ordem da árvore B**.

- Idéia: construção "bottom-up" da árvore: **Árvores B**

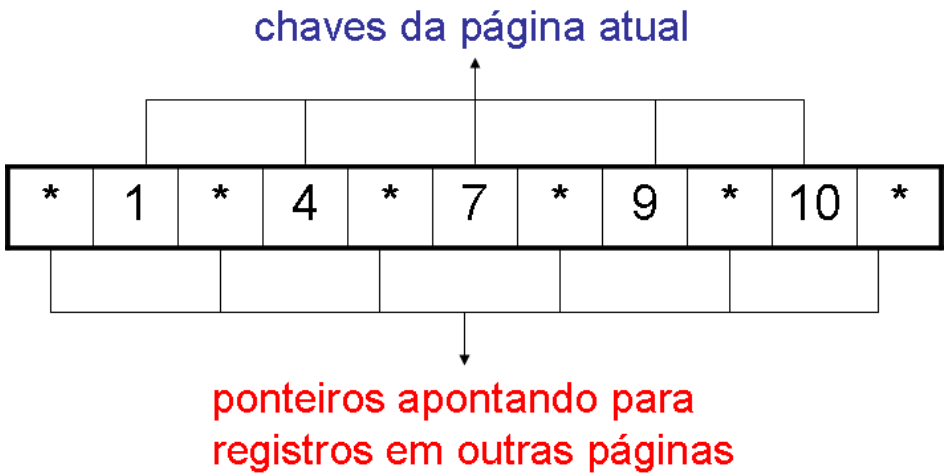
1. Construção da árvore a partir das folhas.
2. Uma página da árvore consiste de uma sequência ordenada de chaves e de um conjunto de ponteiros.
3. O número de ponteiros em uma dada página é igual ao *número de chaves + 1* $\Rightarrow$ **ordem da árvore B**.

- Idéia: construção "bottom-up" da árvore: **Árvores B**

1. Construção da árvore a partir das folhas.
2. Uma página da árvore consiste de uma sequência ordenada de chaves e de um conjunto de ponteiros.
3. O número de ponteiros em uma dada página é igual ao *número de chaves + 1* $\Rightarrow$ **ordem da árvore B**.

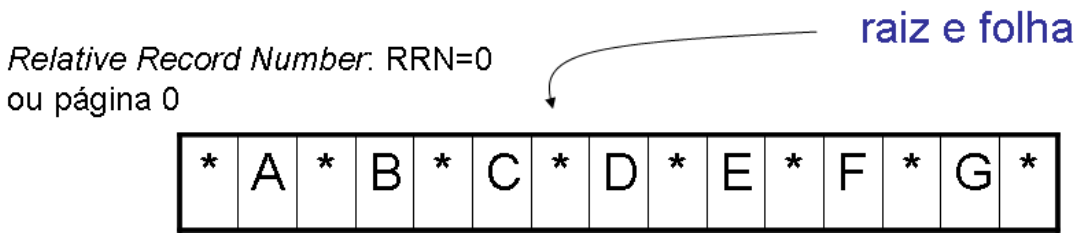
- Exemplo:

– Páginas com 5 chaves:



• Exemplo: construção da primeira página de 7 chaves ⇒ árvore de ordem 8

– Conjunto de chaves: *B C G E F D A*



- Introduzir chave J:

– Cria-se uma nova página de $RRN = 1$ e distribui-se equilibradamente as chaves entre as duas páginas, inserindo J na sua devida posição ordenada.

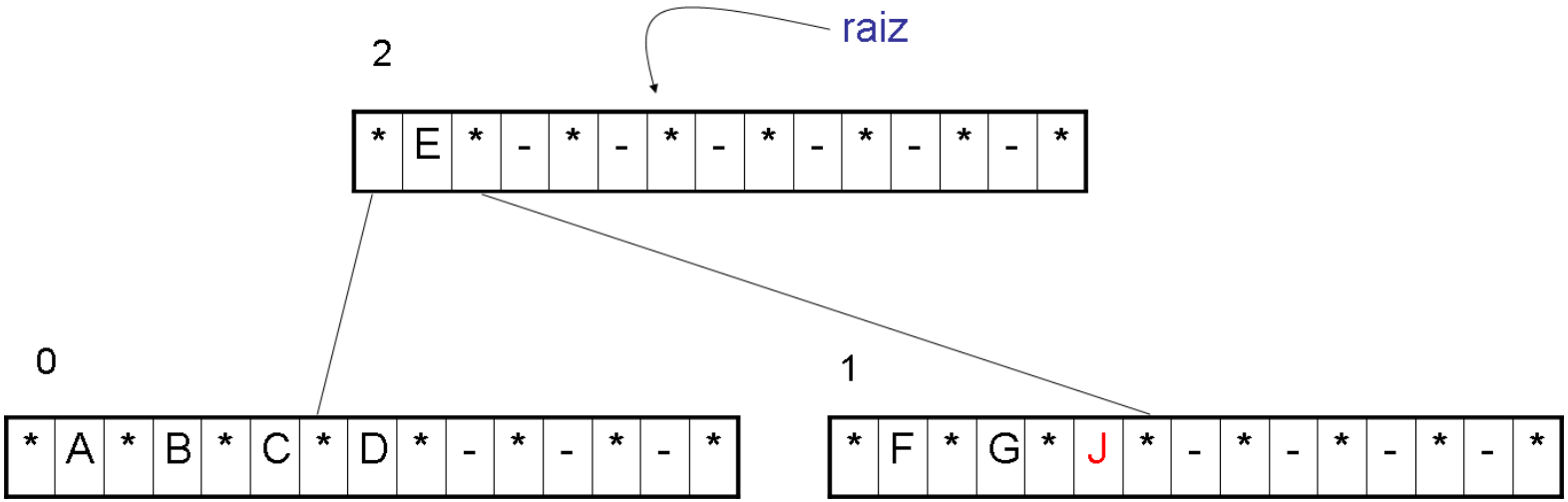
RRN=0

*	A	*	B	*	C	*	D	*	-	*	-	*	-	*

RRN=1

*	E	*	F	*	G	*	J	*	-	*	-	*	-	*

• Promovo à condição de *raiz* uma chave que separa as *folhas* subdivididas.



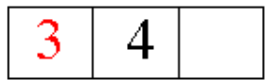
– Exercício: Definir a árvore B considerando-se a leitura sequencial das seguintes chaves:

4 3 1 9 6 8 5 10 13 7 14 2 18 19 11 20

Chaves: 4

Chaves: 4 3

Chaves: 4 3



Chaves: 4 3 1

3	4	

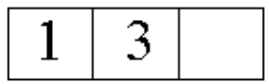
Chaves: 4 3 1

1	3	4

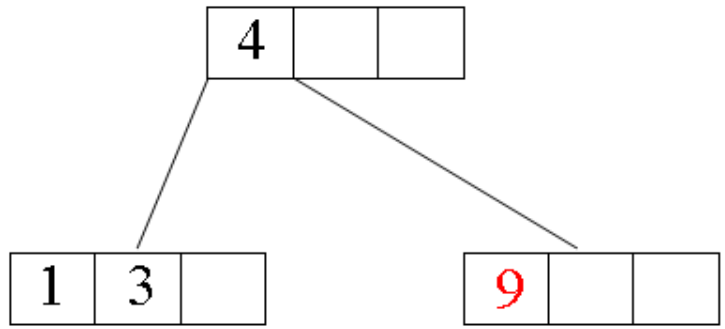
Chaves: 4 3 1 9

1	3	4

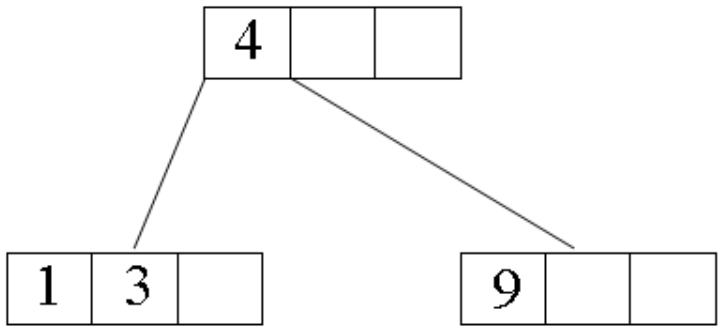
Chaves: 4 3 1 9



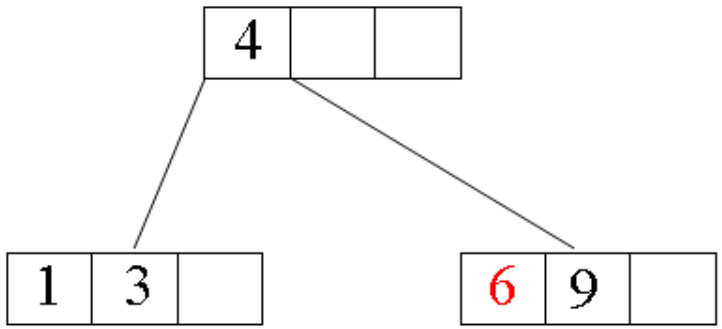
Chaves: 4 3 1 9



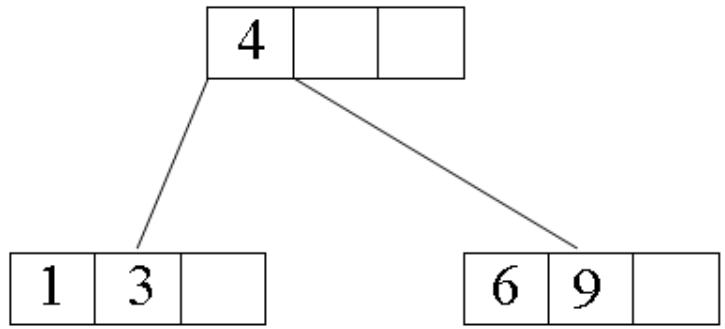
Chaves: 4 3 1 9 6



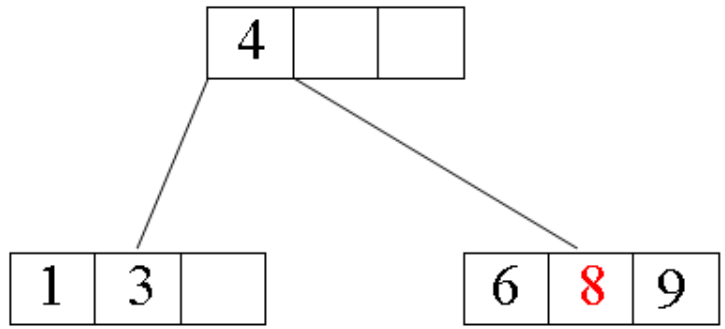
Chaves: 4 3 1 9 6



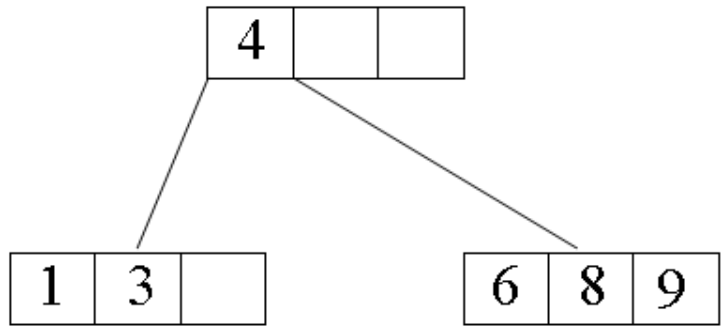
Chaves: 4 3 1 9 6 8



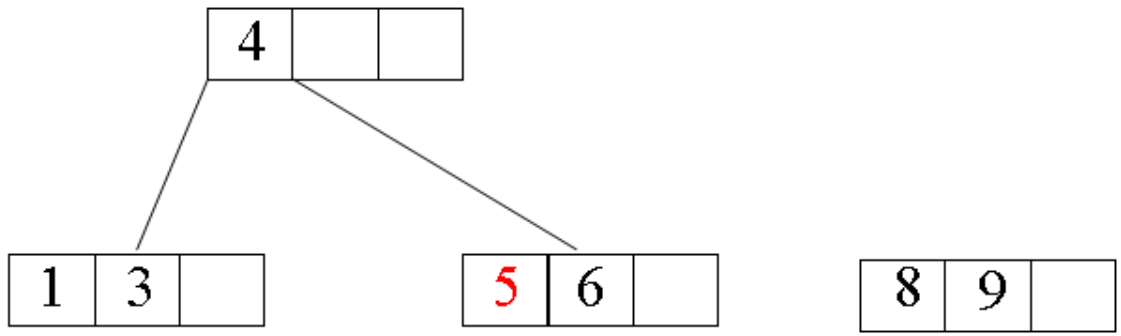
Chaves: 4 3 1 9 6 8



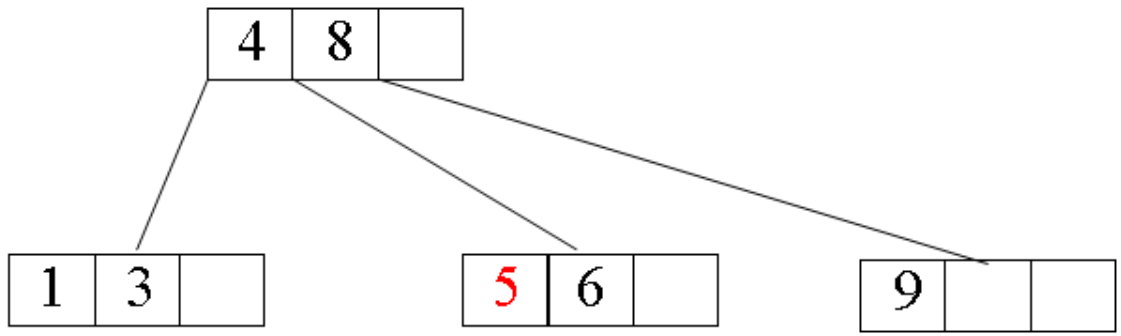
Chaves: 4 3 1 9 6 8 5



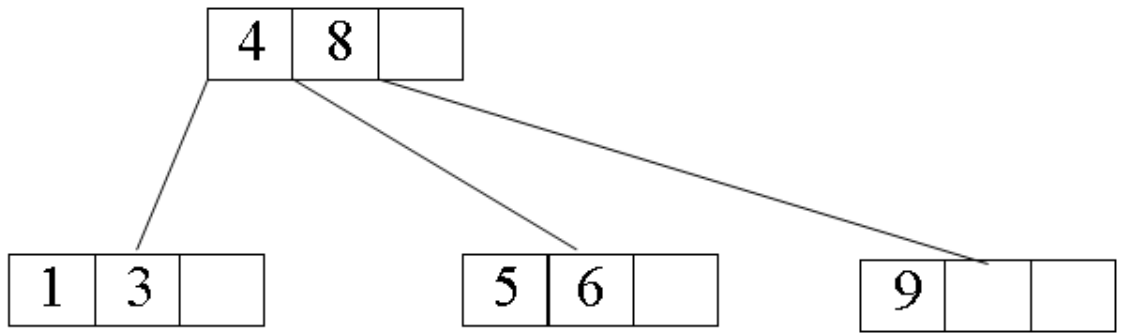
Chaves: 4 3 1 9 6 8 5



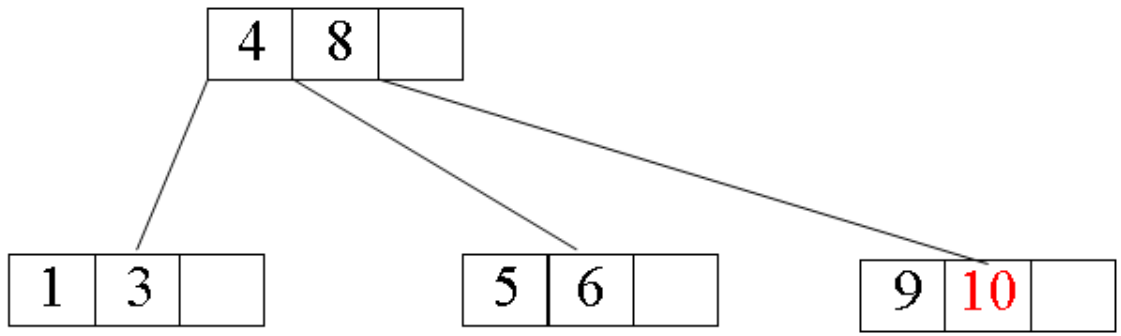
Chaves: 4 3 1 9 6 8 5



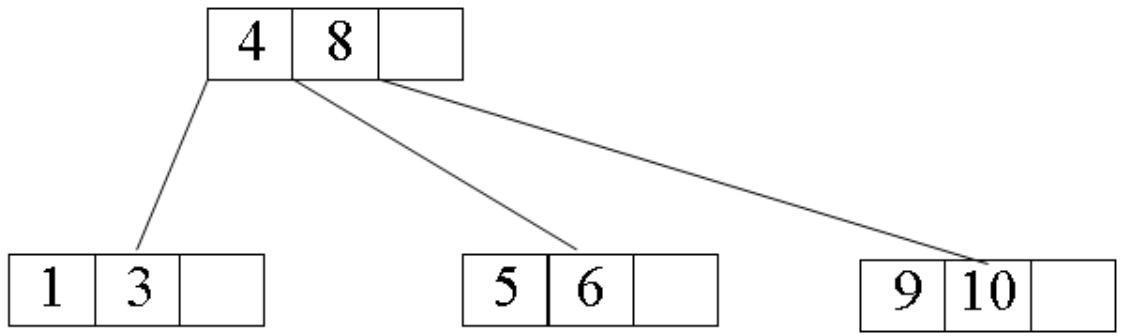
Chaves: 4 3 1 9 6 8 5 10



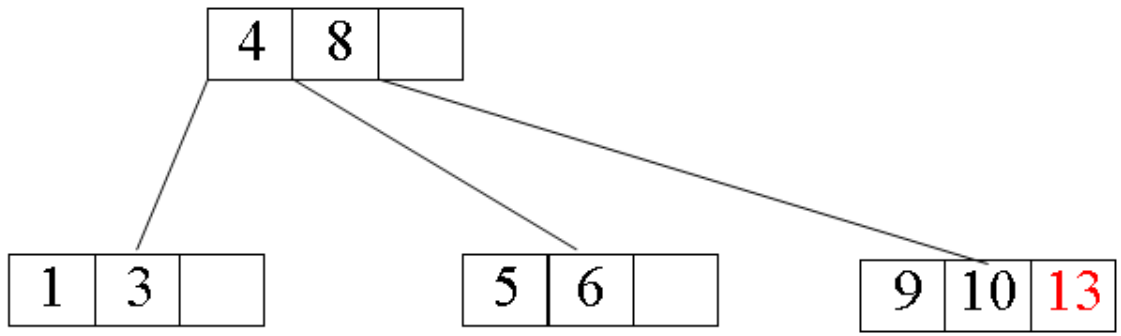
Chaves: 4 3 1 9 6 8 5 10



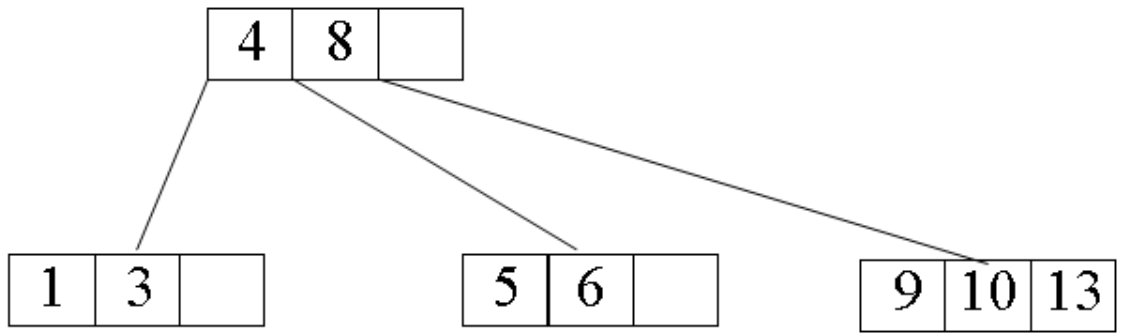
Chaves: 4 3 1 9 6 8 5 10 13



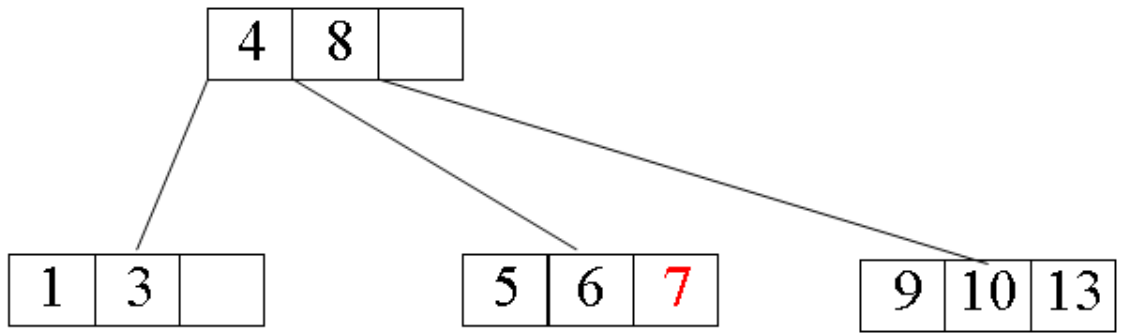
Chaves: 4 3 1 9 6 8 5 10 13



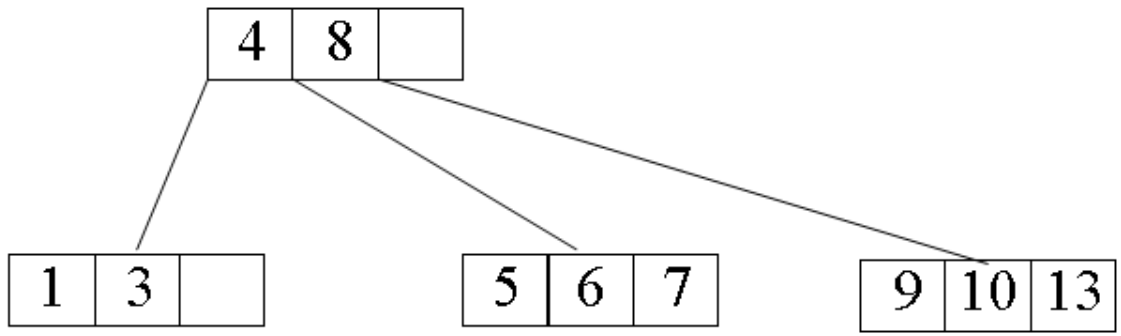
Chaves: 4 3 1 9 6 8 5 10 13 7



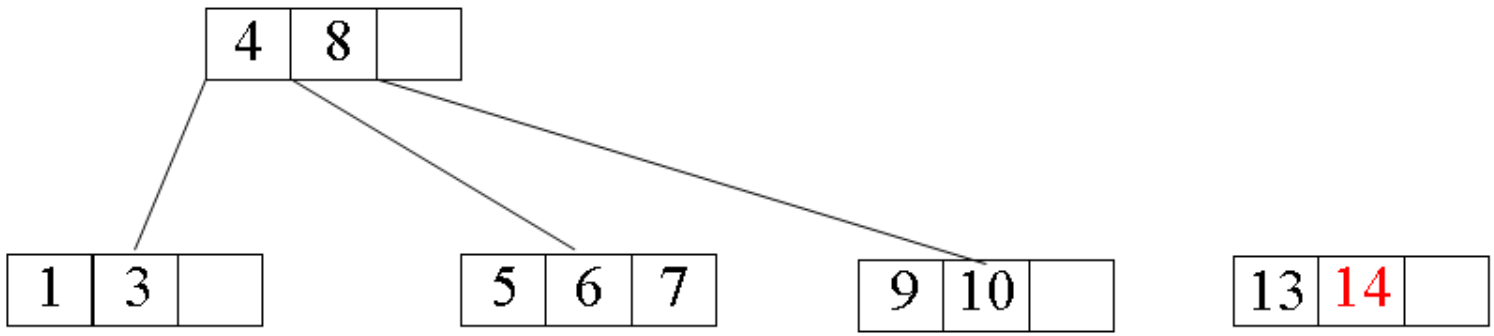
Chaves: 4 3 1 9 6 8 5 10 13 7



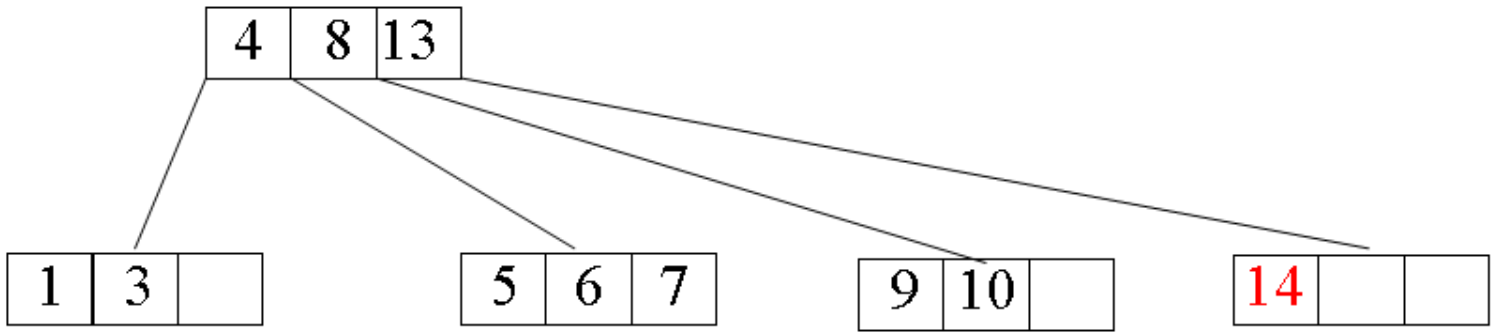
Chaves: 4 3 1 9 6 8 5 10 13 7 14



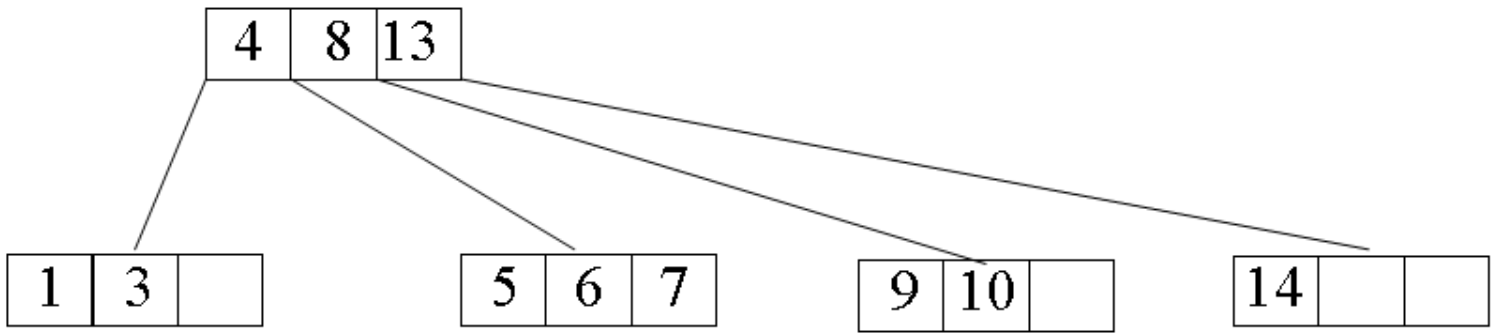
Chaves: 4 3 1 9 6 8 5 10 13 7 14



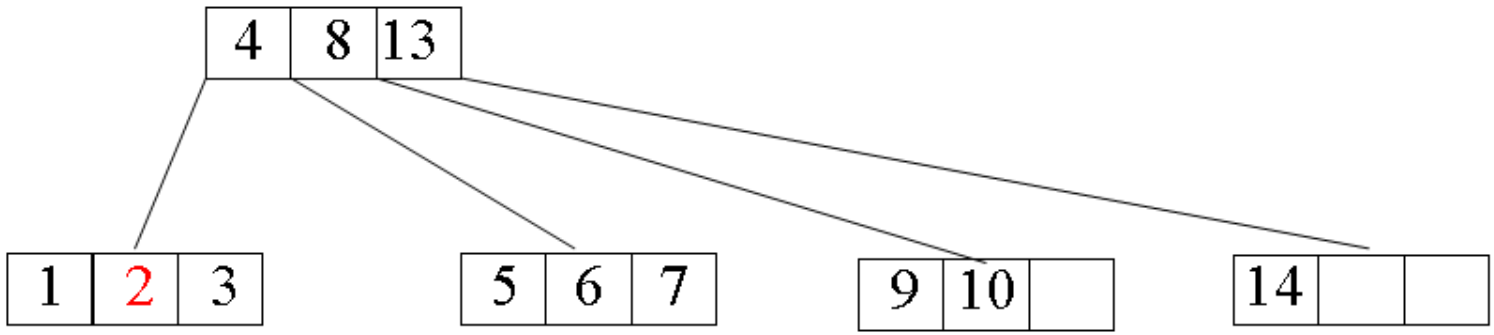
Chaves: 4 3 1 9 6 8 5 10 13 7 14



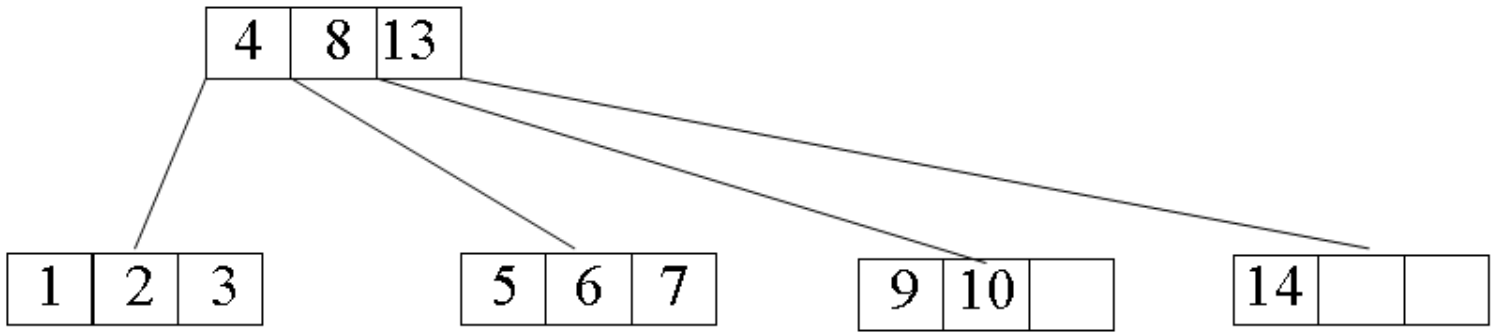
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2



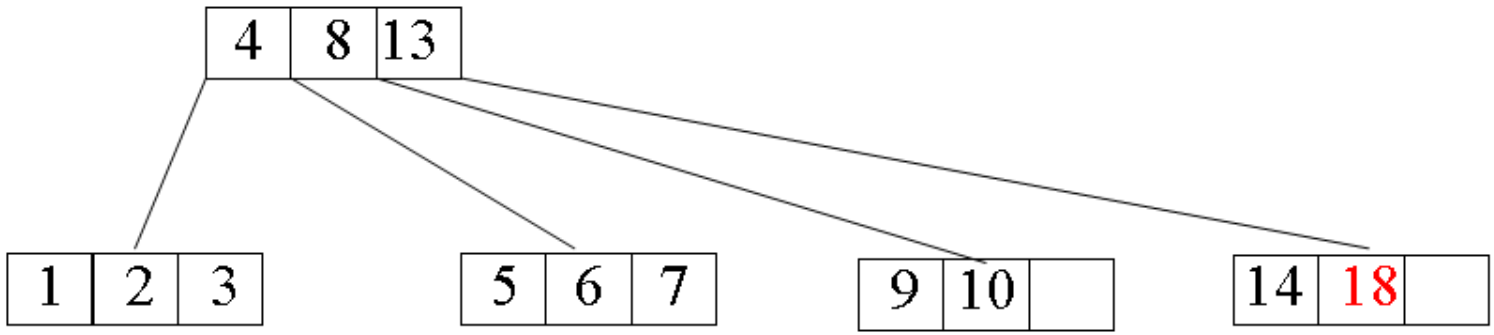
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2



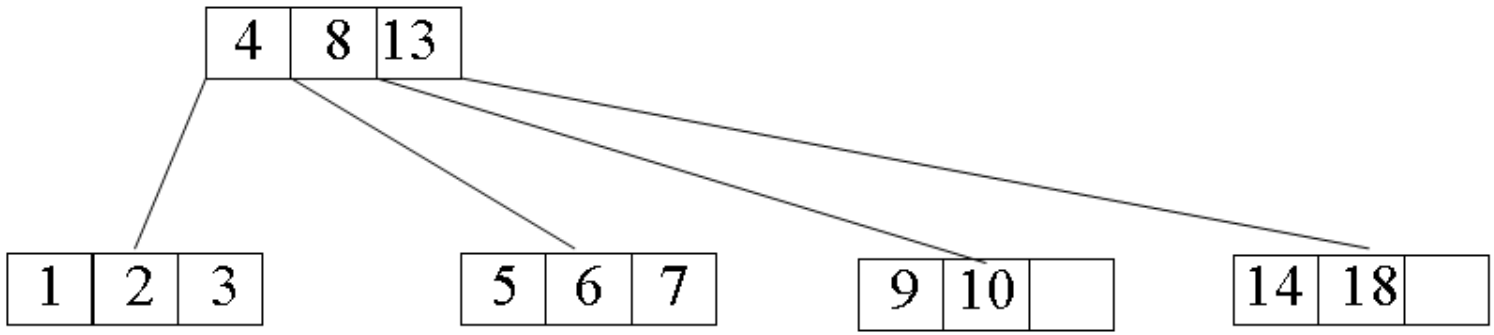
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18



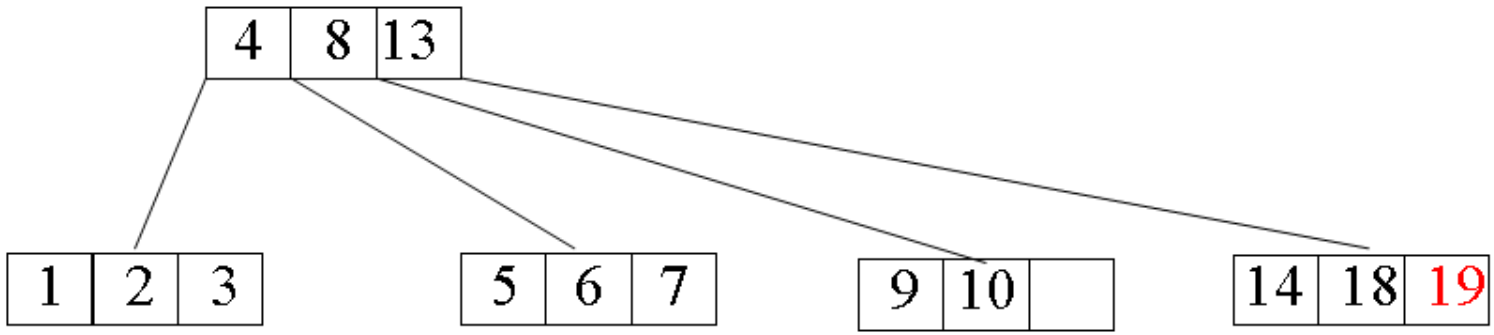
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18



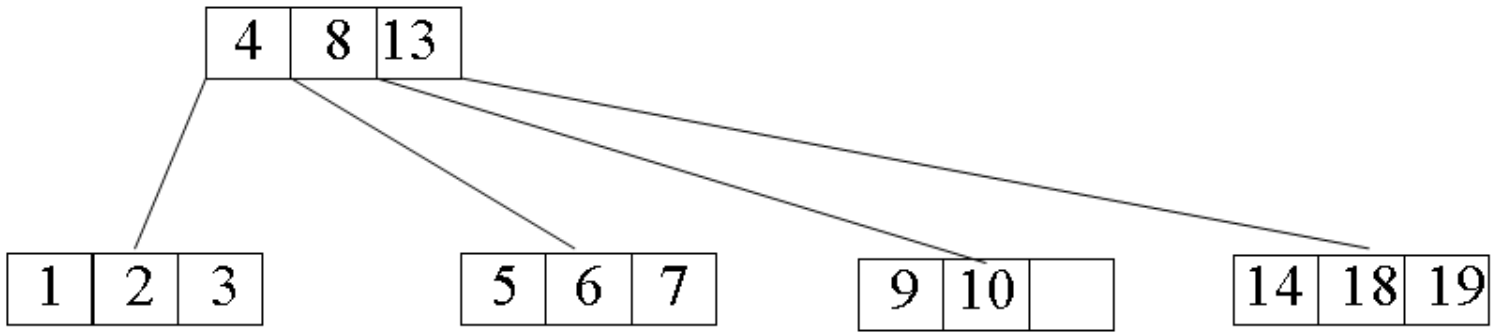
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19



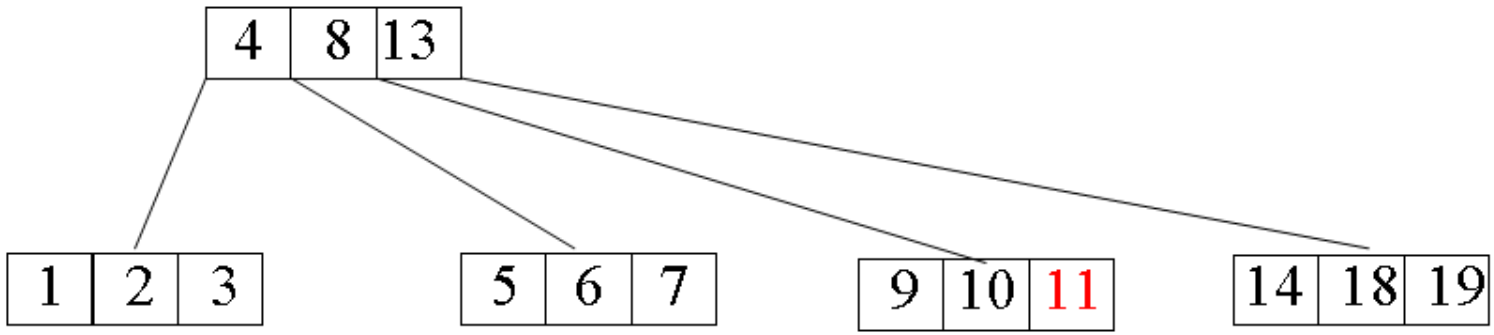
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19



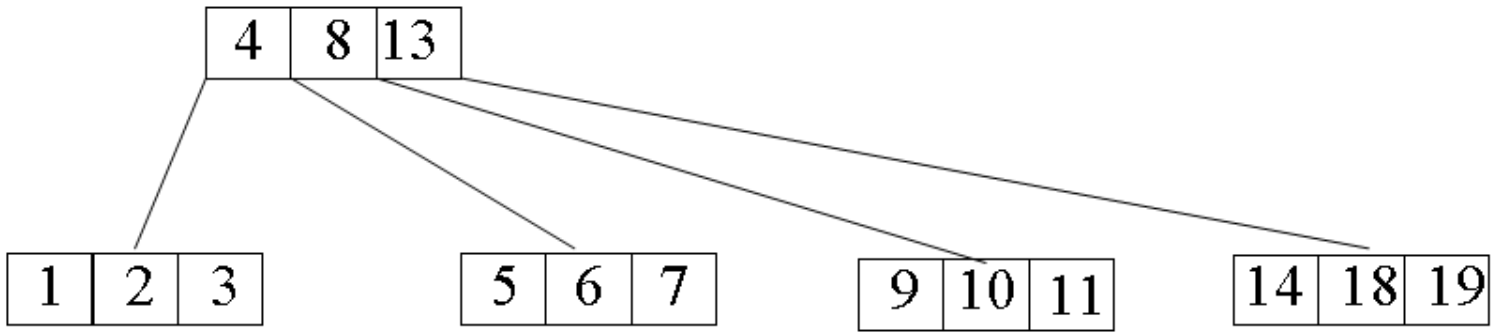
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19 11



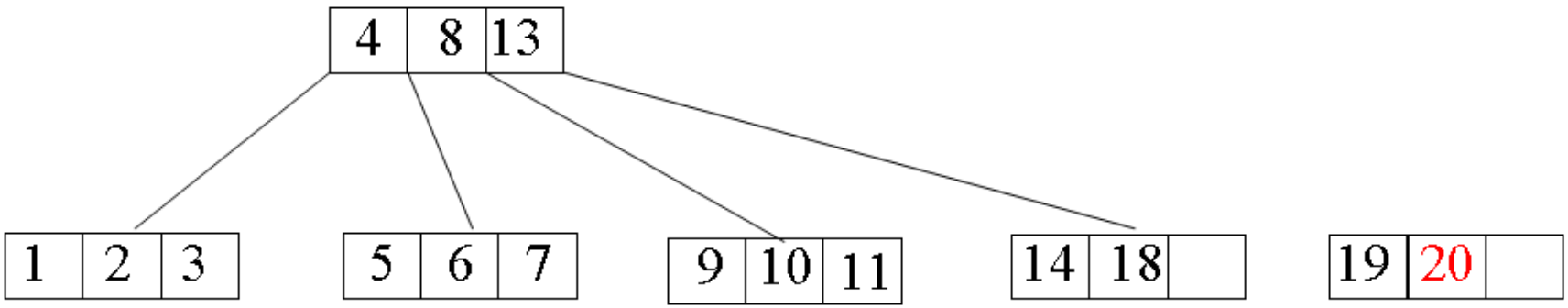
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19 11



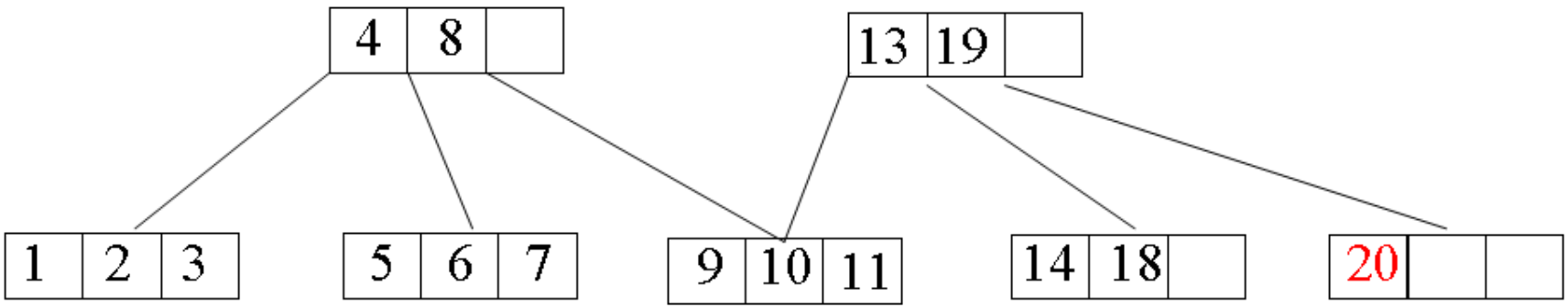
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19 11 20



Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19 11 20

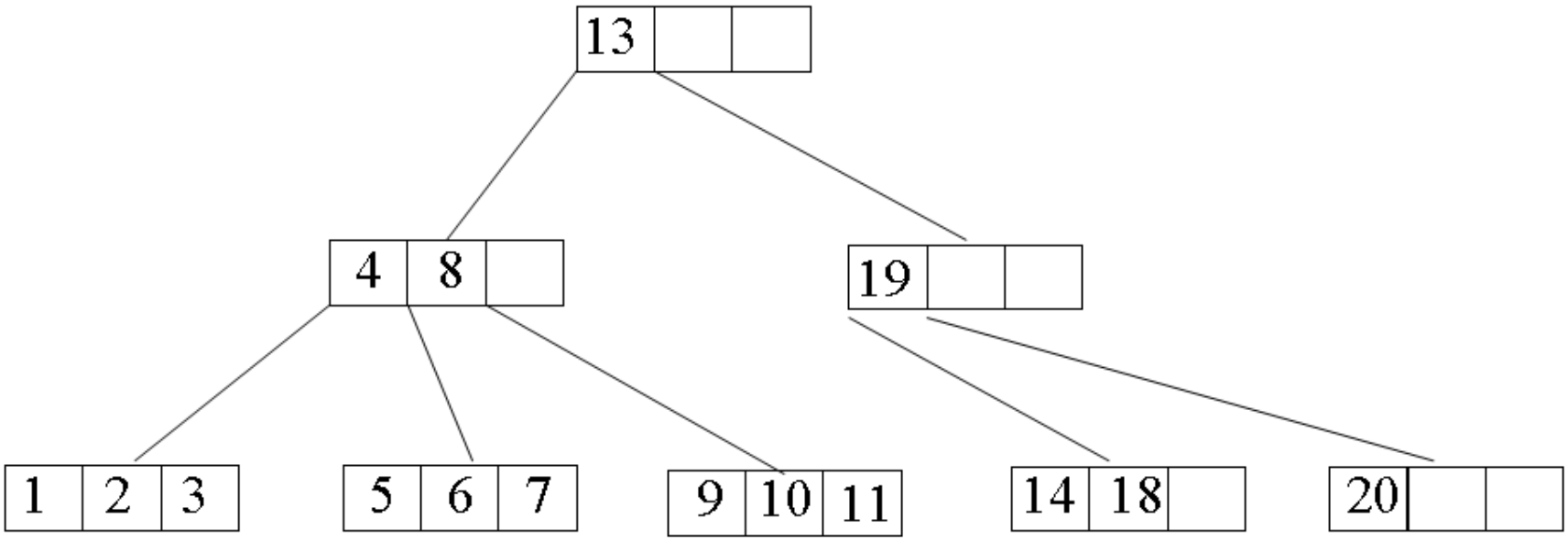


Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19 11 20

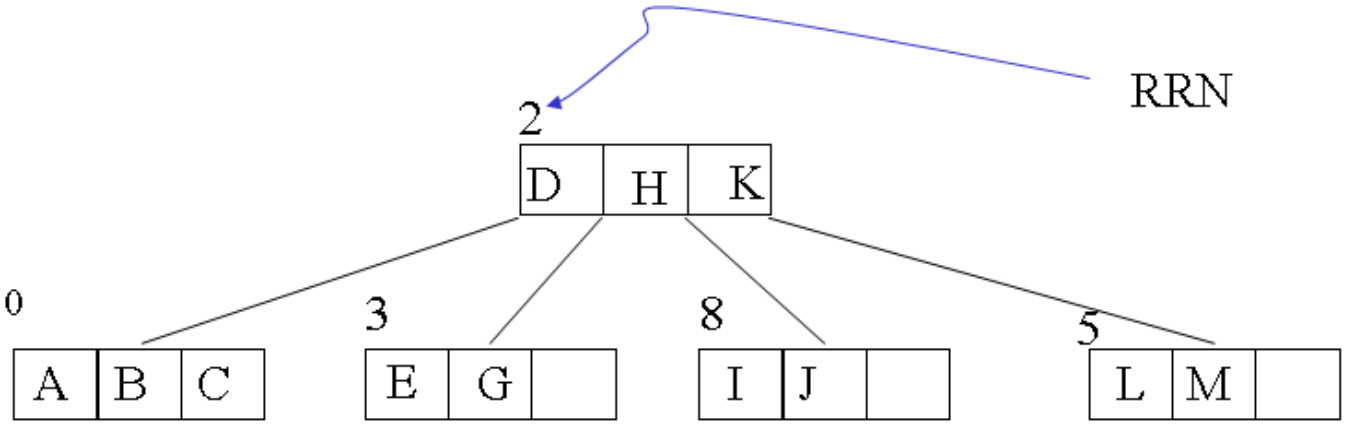


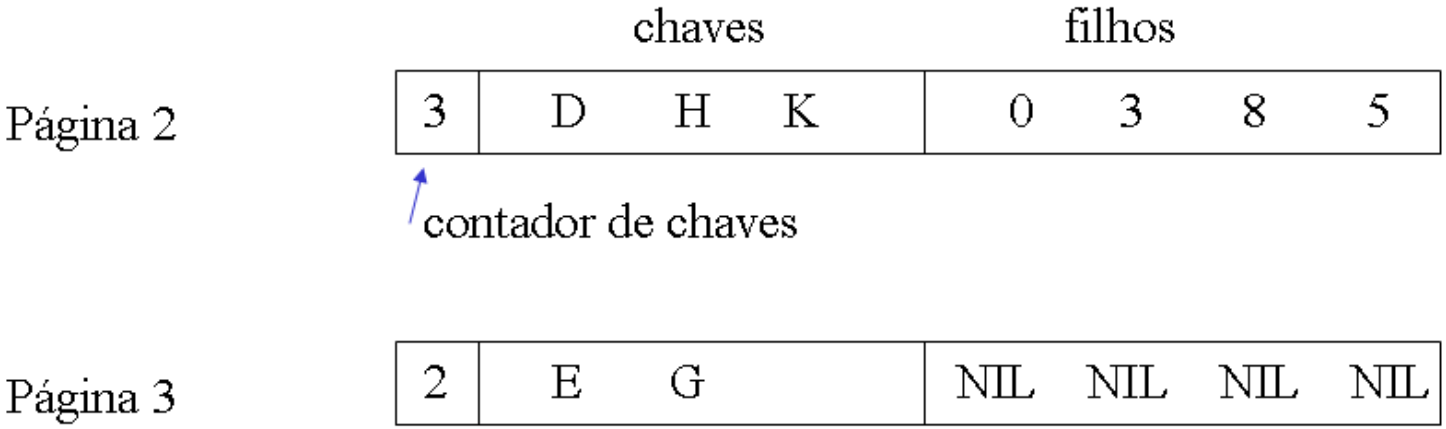
Chaves: 4 3 1 9 6 8 5 10 13 7 14 2 18 19 11 20

FIM!



- Estrutura das páginas:





- Análise da profundidade:

– Uma árvore de N chaves, organizada em k chaves por página, tem profundidade de

$$p = log_{k+1}(N + 1)$$

.

- Análise da profundidade:

– Uma árvore de N chaves, organizada em k chaves por página, tem profundidade de

$$p = log_{k+1}(N + 1)$$

.

– Lema: O número de descendentes em qualquer nível da árvore é igual ao número de chaves+1 contidas até este nível.

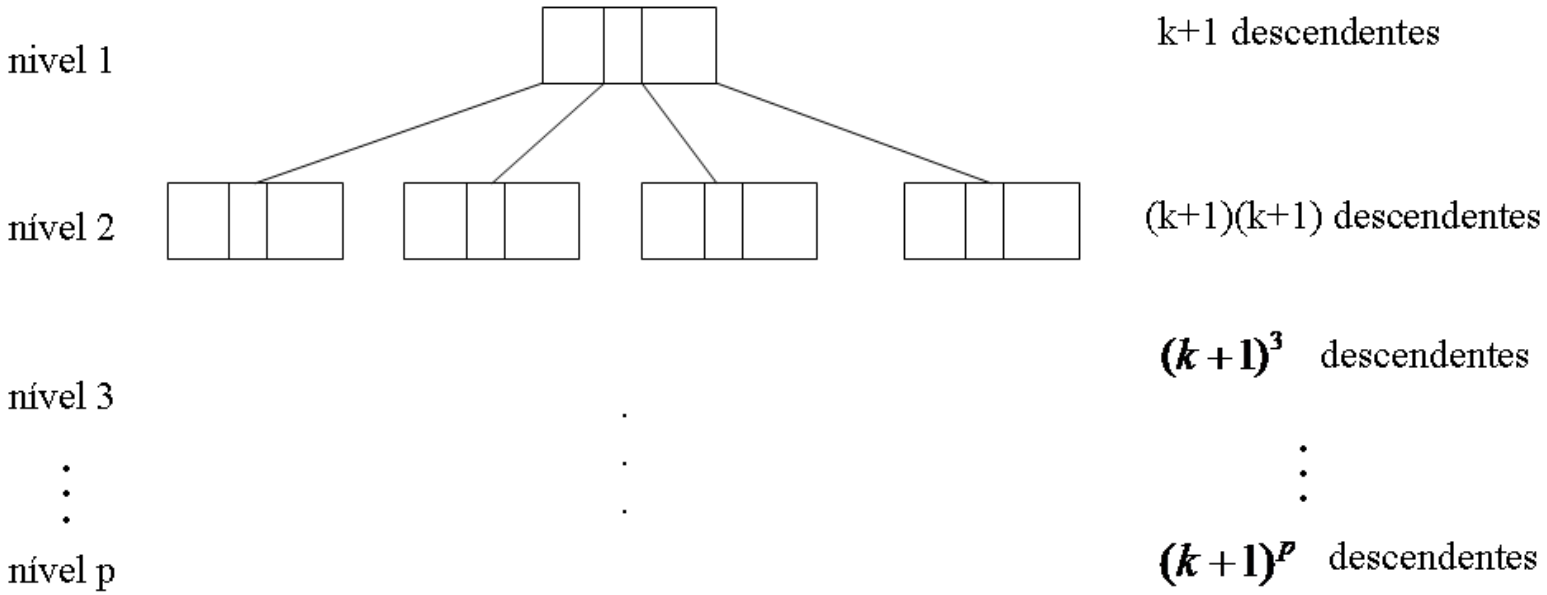
- Análise da profundidade:

– Uma árvore de N chaves, organizada em k chaves por página, tem profundidade de

$$p = log_{k+1}(N + 1)$$

.

– <u>Lema:</u> O número de descendentes em qualquer nível da árvore é igual ao número de chaves+1 contidas até este nível.



... ou em termos das chaves:

nível 1:	(k+1)-1	chaves
nível 2:	(k+1)(k+1)-1	chaves
nível 3	$(k+1)^3 - 1$	chaves
	.	
	.	
	.	
nível p:	$(k+1)^p - 1$	= N chaves

$$\Rightarrow p = \log_{(k+1)}(N+1)$$

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k-1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m-1$.

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k-1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m-1$.

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k-1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m-1$.

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k-1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m-1$.

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k-1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m-1$.

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k-1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m-1$.

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k-1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m-1$.

• Algumas propriedades da árvore B (devem ser preservadas em todas as operações na árvore):

– Para uma árvore de ordem m:

1. Cada página da árvore tem no máximo m descendentes.
2. Cada página, exceto raiz e folhas, tem no mínimo $\lceil \frac{m}{2} \rceil$ descendentes.
3. Uma raiz não-folha tem no mínimo 2 descendentes.
4. **Todas as folhas aparecem no mesmo nível**.
5. Uma página não-folha com k descendentes contém $k - 1$ chaves.
6. Uma página folha contém no mínimo $\lceil \frac{m}{2} \rceil - 1$ chaves e no máximo $m - 1$.

- Implementação:

– Exemplo de estrutura de uma página considerando-se MAXCHAVES por página:

```
typedef struct ArvBPag {
  short n;    /* numero de chaves na pagina */
  TipoChaves CHAVE[MAXCHAVES];   /* as chaves da pagina */
  TipoApontador FILHOS[MAXCHAVES+1]; /* os RRN dos filhos */
} ArvBPag ;
```

Exemplo de função

- Buscar uma chave na árvore B.

```
FUNCTION busca (RRN, CHAVE, ACHEI-RRN, ACHEI-POS)
  if RRN = NULL then   /* para a recursão */
     return NAO-ACHEI
  else
     carregue a página indicada por RRN em PAGE
     procure a CHAVE em PAGE
     POS = posição onde CHAVE ocorre ou devia ocorrer

     if CHAVE encontrada then
        ACHEI-RRN = RRN
        ACHEI-POS = POS
        return ACHEI

     else              /* desçe um nível na direção do FILHO */
        return(busca(PAGE.FILHO[POS], CHAVE, ACHEI-RRN, ACHEI-POS))
     endif
  endif
end FUNCTION
```

- Inserir uma chave na árvore B.

```
FUNCTION insere(ATUAL_RRN, CHAVE, PROMOVE_D_FILHO, PROMOVE_CHAVE)
  if ATUAL_RRN == NULL then        /*  base da árvore */
    PROMOVE_CHAVE = CHAVE
    PROMOVE_D_FILHO = NULL
    return PROMOVE                  /* promove CHAVE e NULL */
  else
    carregue página em ATUAL-RNN em PAGE
    procure CHAVE em PAGE
    POS = posição onde CHAVE ocorre ou deveria ocorrer

  if CHAVE foi encontrada then
    escreva mensagem de chave duplicada
    return ERRO

  RETURN_VALUE = insere(PAGE.FILHO[POS].CHAVE, P_B_RRN, P_B_CHAVE)

  if RETURN_VALUE == NAO-PROMOVE or ERRO then
    return RETURN_VALUE

  else if há espaço em PAGE para P_B_CHAVE then
    insere P_B_CHAVE e P_B_RRN, vindos da chamada anterior, em PAGE
    return NAO-PROMOVE
  else
    split(P_B_CHAVE, P_B_RRN, PAGE, PROMOVE_CHAVE, PROMOVE_D_FILHO, NEWPAGE)
    return PROMOVE  /* promove PROMOVE_CHAVE e PROMOVE_D_FILHO */
  endif
```

fase de busca

chamada recursiva

fase de inserção, splitting e promoção

- A função *split*:

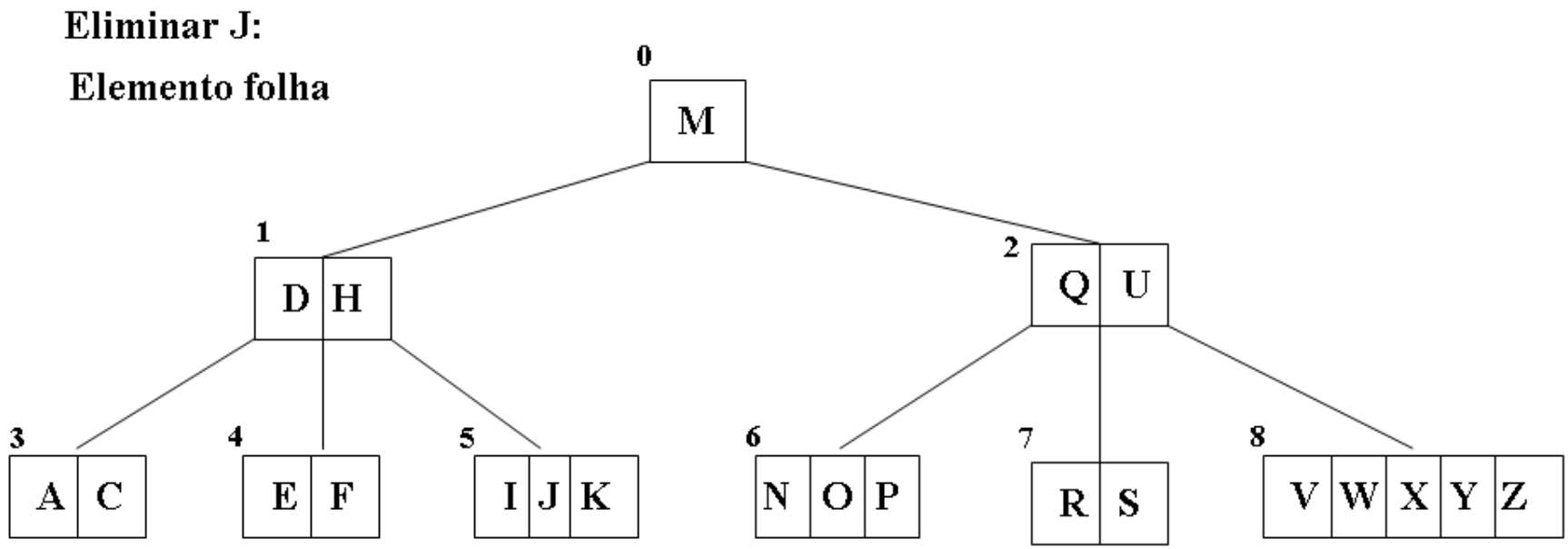
```
FUNCTION split(I_CHAVE, I_RRN, PAGE, PROMOVE_CHAVE, PROMOVE_D_FILHO, NEWPAGE)
    copie PAGE com suas chaves e ponteiros para uma área TEMP que possa conter uma chave e filho a mais.
    insira I_CHAVE e I_RRN na devida posição em TEMP
    aloque e inicialize uma nova página que conterá NEWPAGE
    defina PROMOVE_CHAVE = chave do meio de PAGE a ser promovida após o split
    defina PROMOVE_D_FILHO = RRN de NEWPAGE
    copie chaves e ponteiros anteriores a PROMOVE_CHAVE de TEMP para PAGE
    copie chaves e ponteiros posteriores a PROMOVE_CHAVE de TEMP para NEWPAGE
endFunction
```

- O programa principal:

```
Main Program: Árvore B

 if árvore B existe then
   acesse árvore B
 else
   crie árvore B e insira a primeira chave na raiz

 leia RRN da página raiz e guarde em RAIZ
 obtenha uma nova chave e guarde em CHAVE

 while existir CHAVE
   if(insere(RAIZ, CHAVE, PROMOVE_D_FILHO, PROMOVE_CHAVE) ==PROMOVE) then
     crie uma nova página raiz com a chave = PROMOVE_CHAVE
     filho esquerdo = RAIZ
     filho direito = PROMOVE_D_FILHO
     RAIZ = RRN da nova página raiz
   obtenha nova chave e guarde em CHAVE
 endwhile

endMain
```

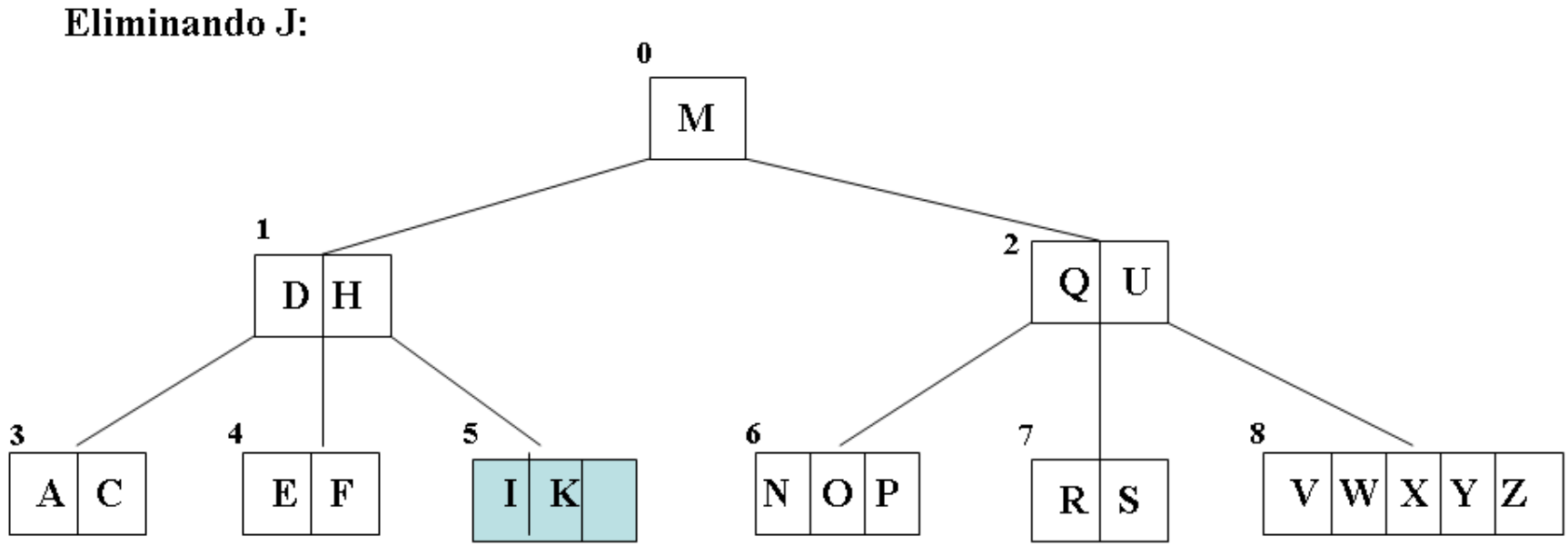
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



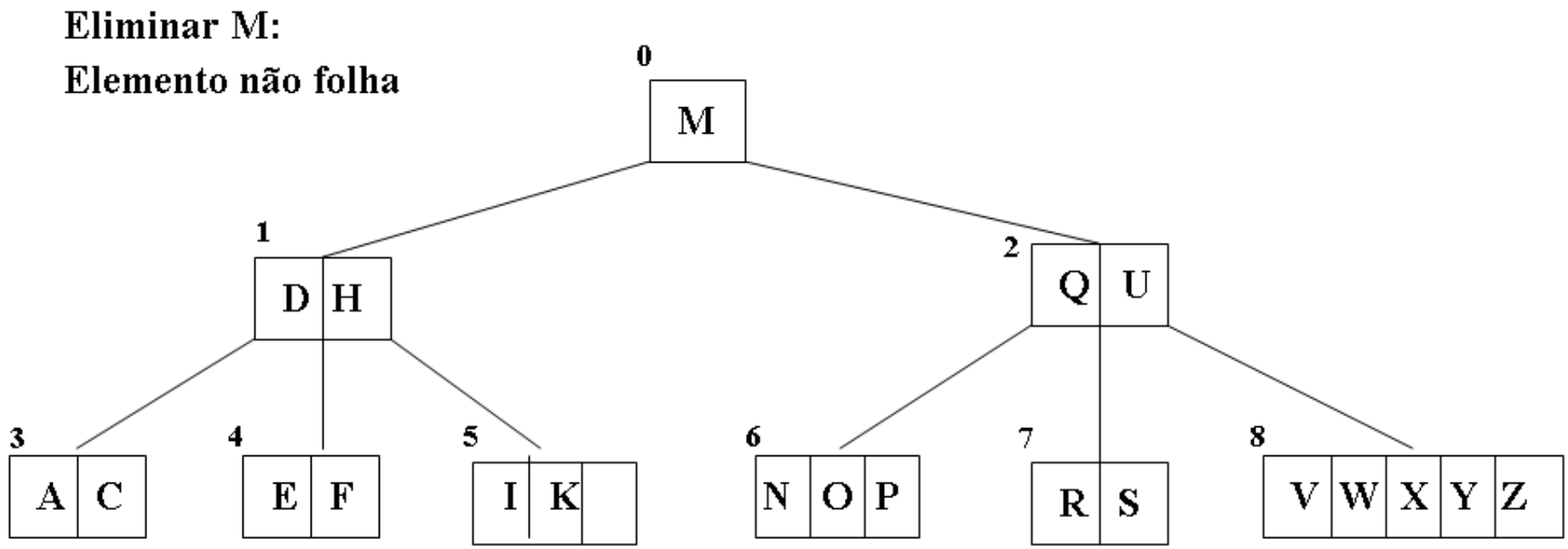
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



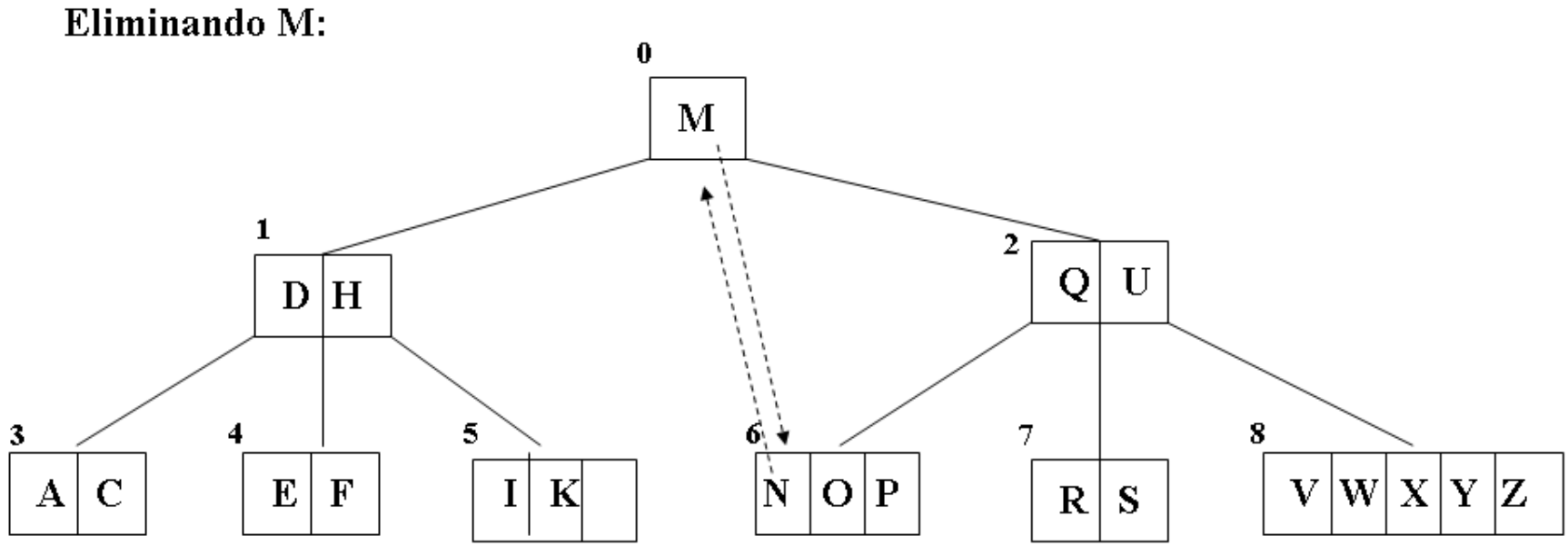
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



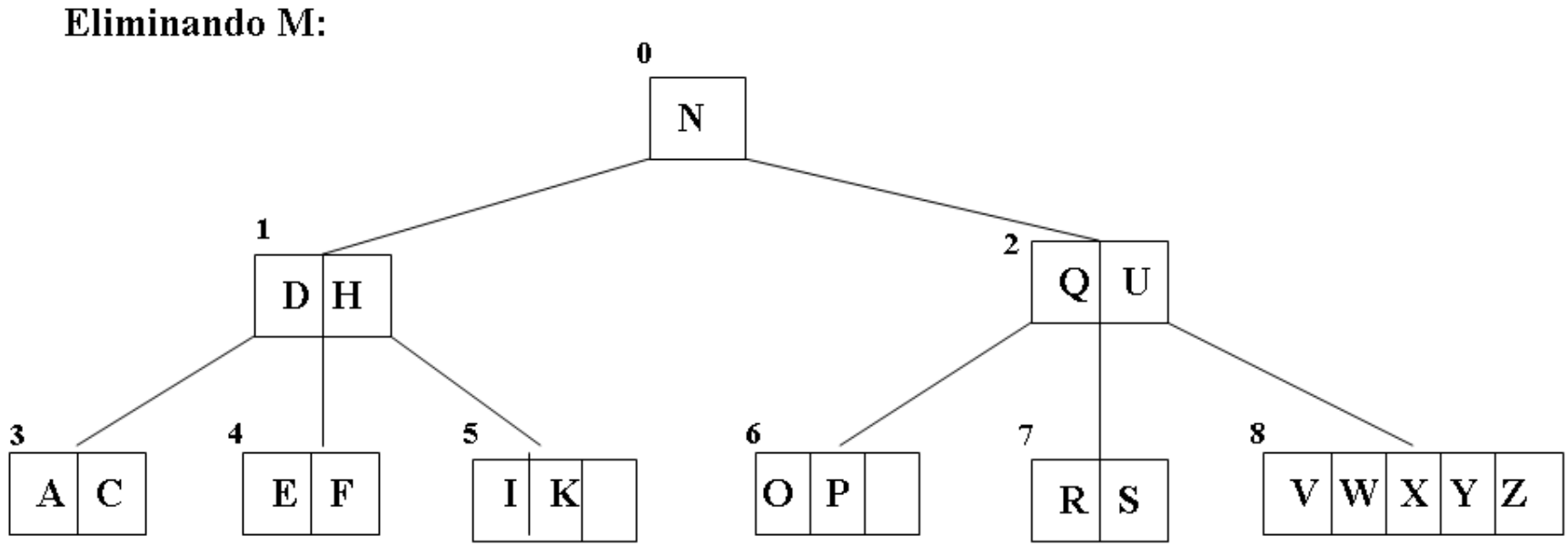
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



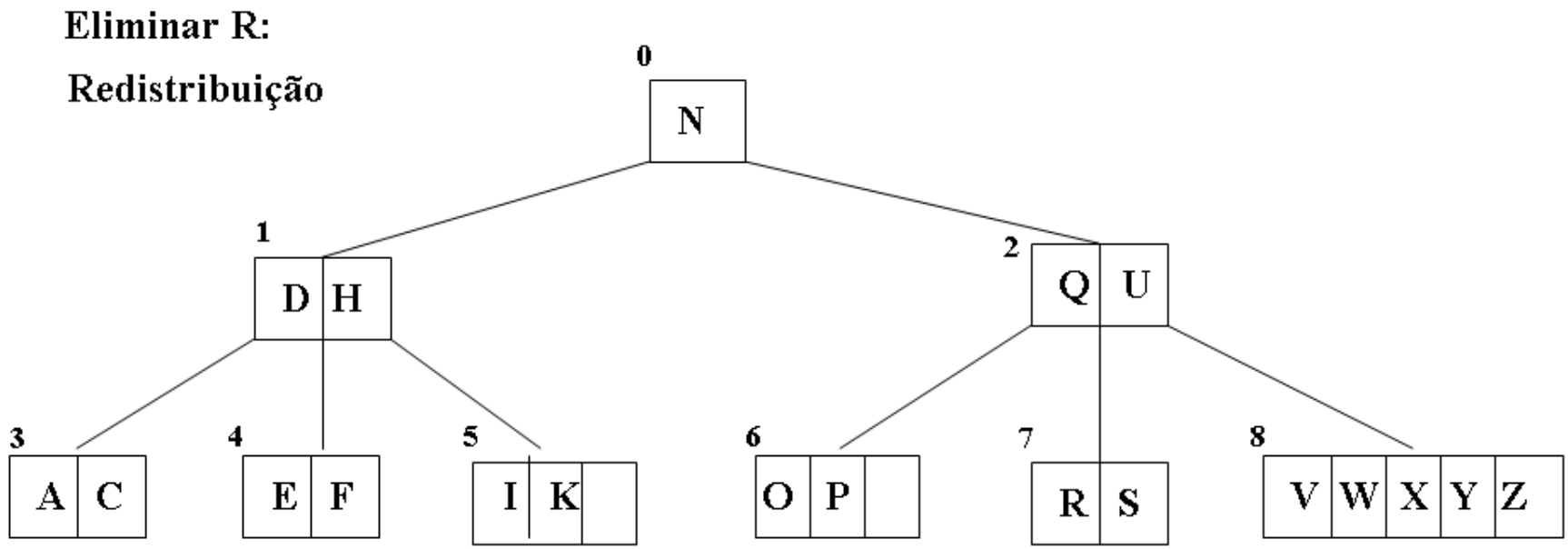
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



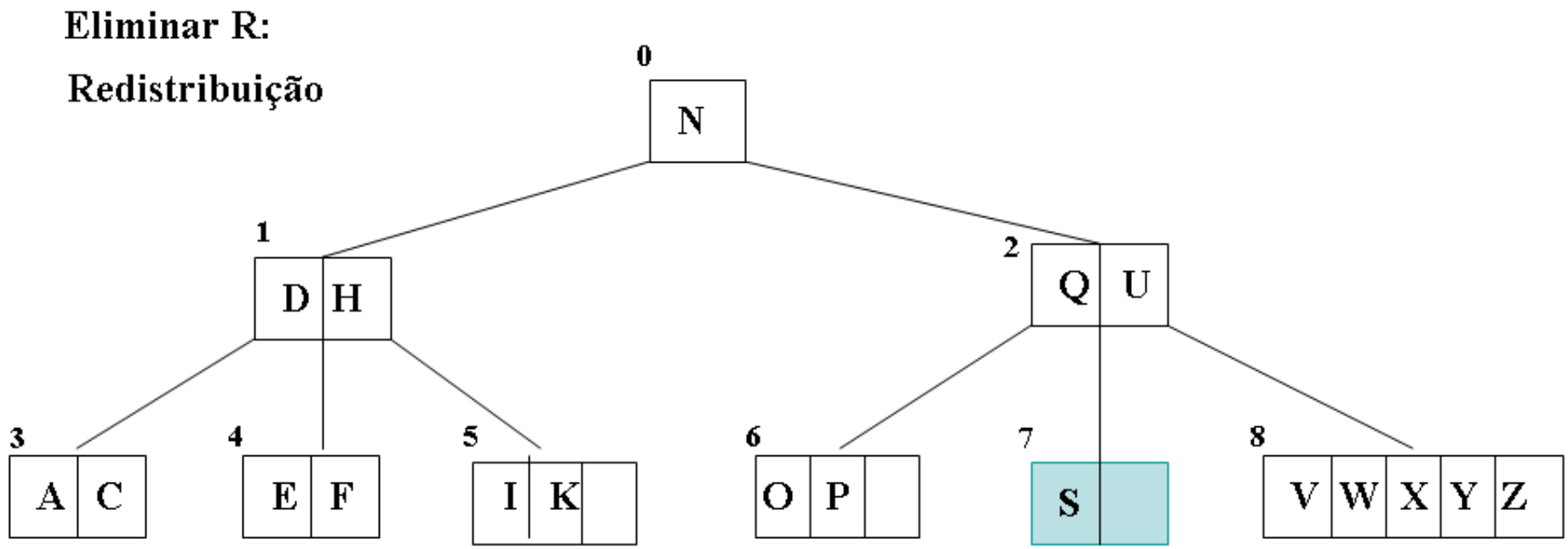
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



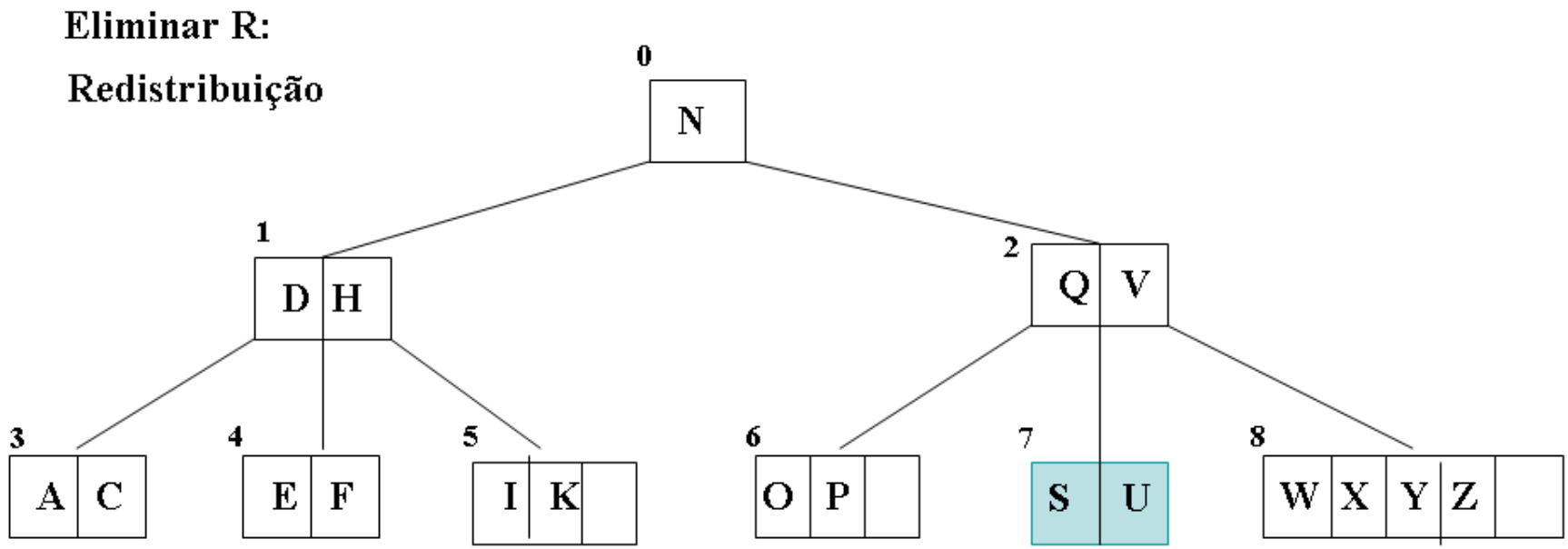
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



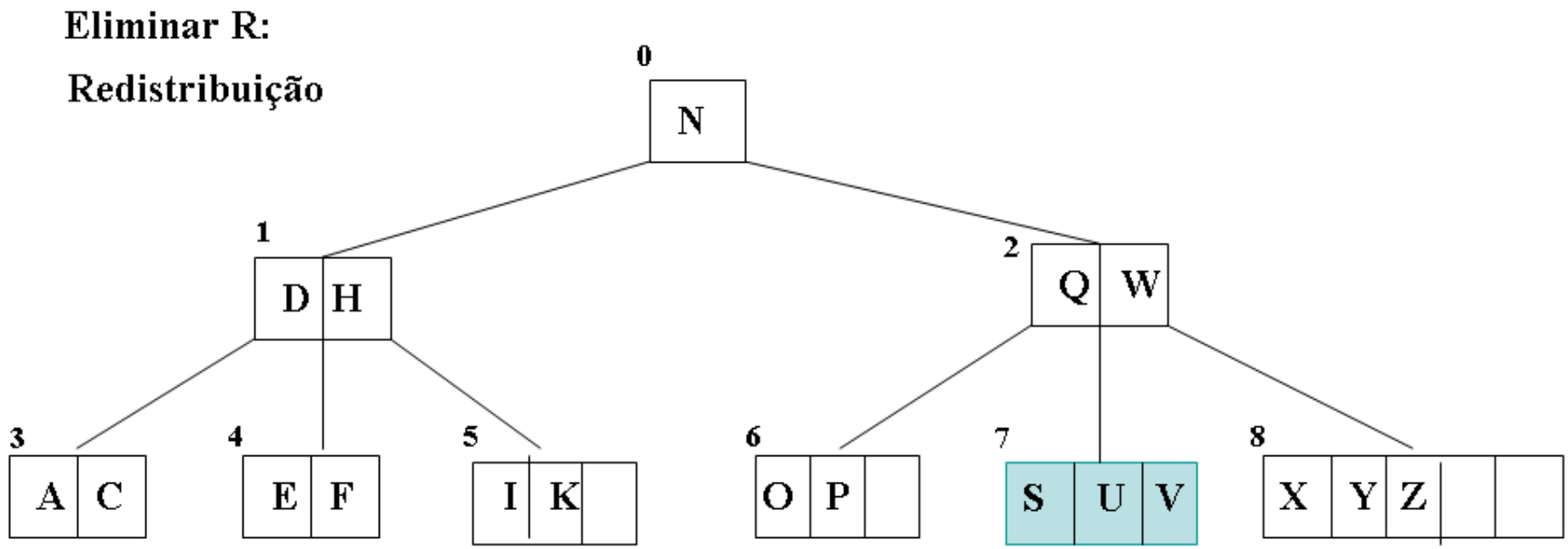
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



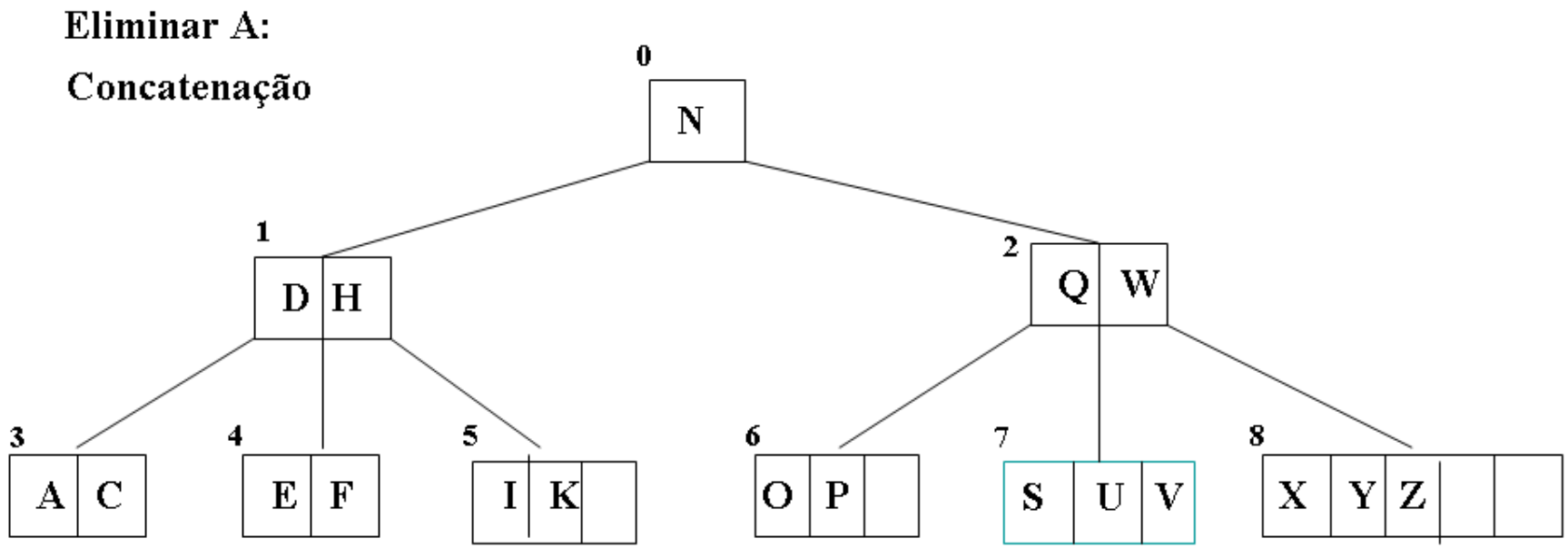
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



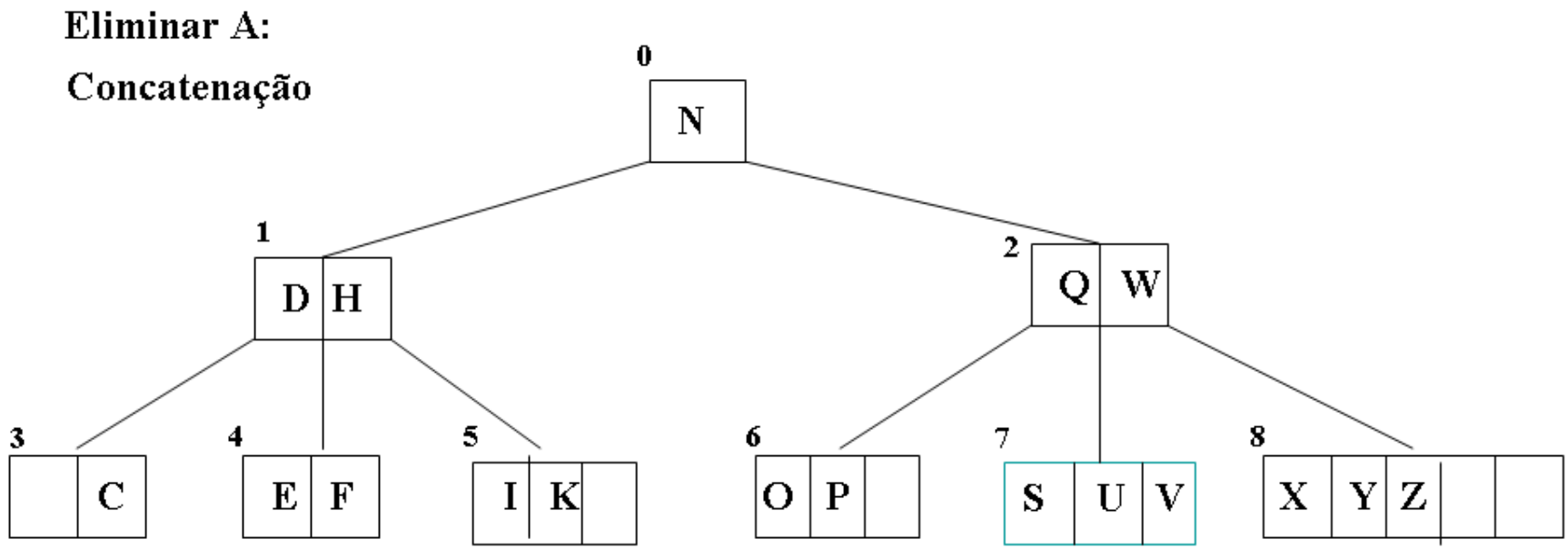
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



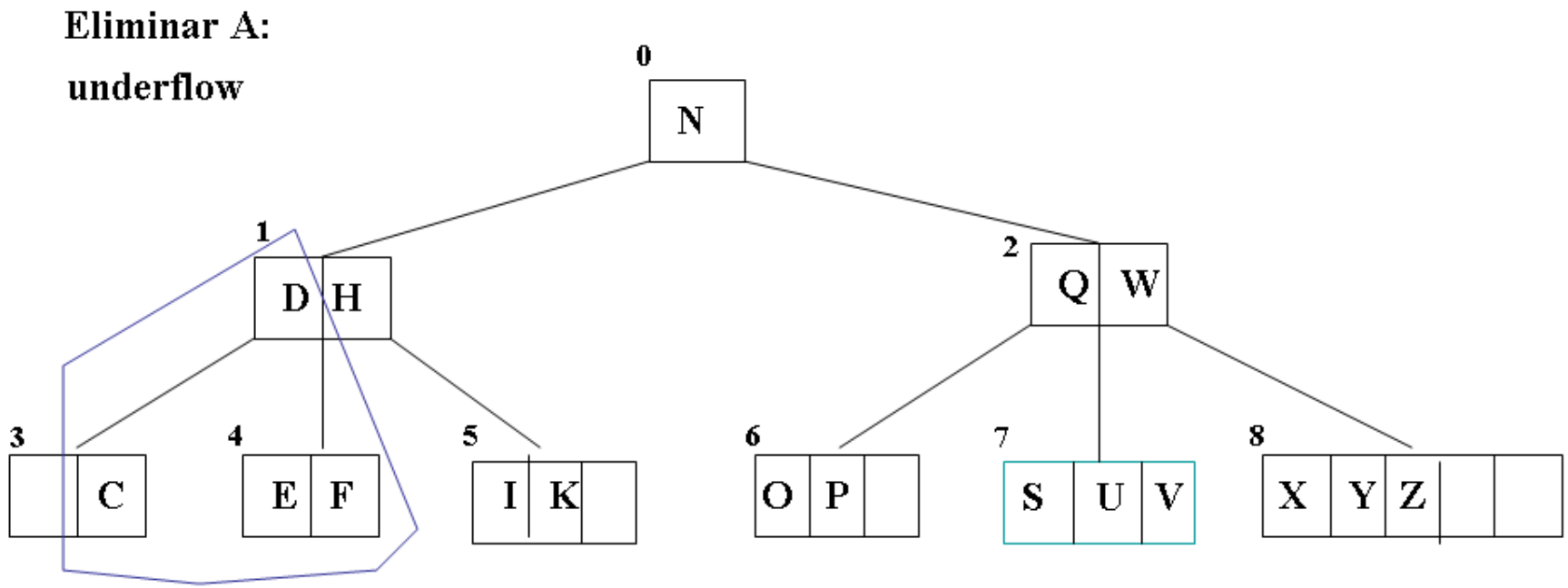
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



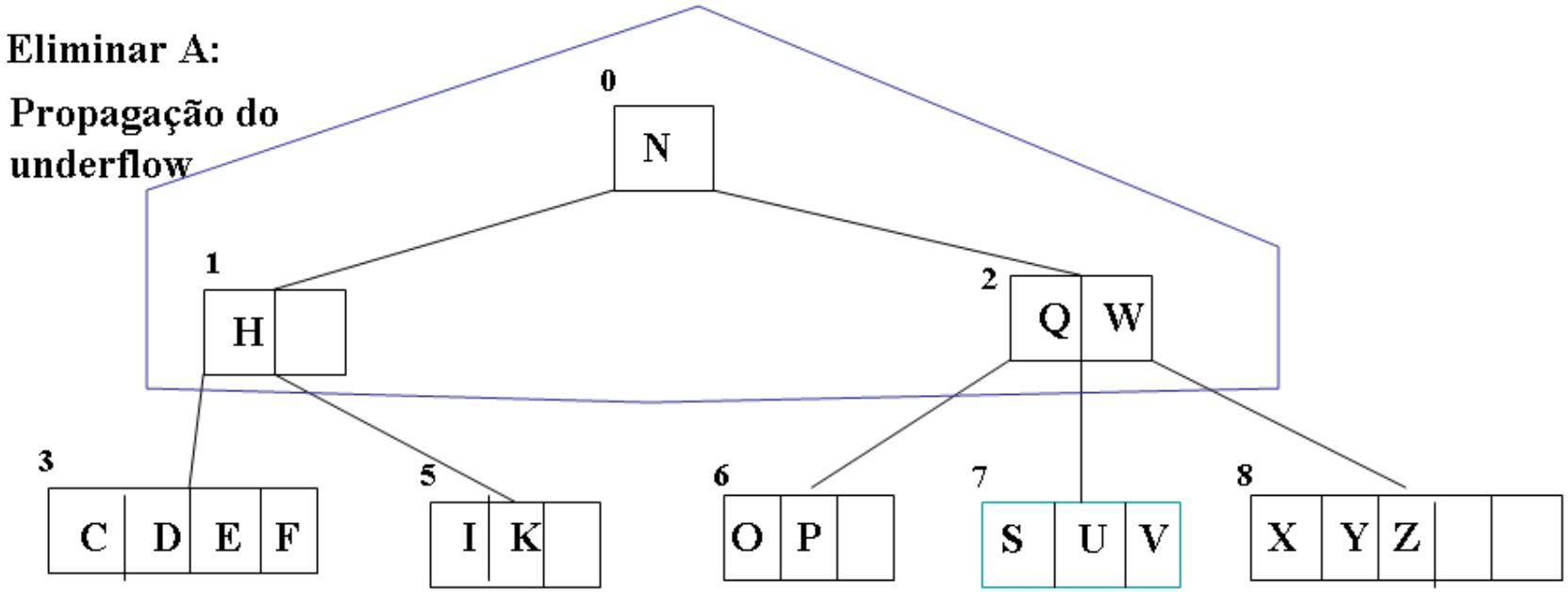
Remoção em árvores B

– Exemplo 1: Árvore de ordem 6



Remoção em árvores B

– Exemplo 1: Árvore de ordem 6

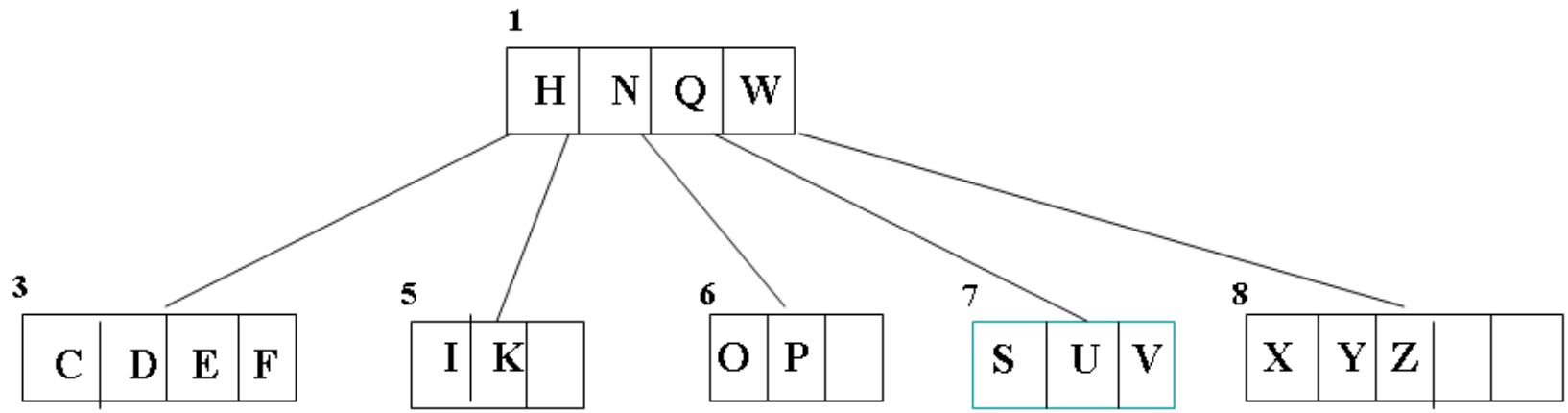


Remoção em árvores B

– Exemplo 1: Árvore de ordem 6

Eliminar A:

redução da altura



• Resumidamente, os passos envolvidos na remoção de um chave da árvore B são:

1. Se a chave a ser removida não é uma folha, troque-a por sua sucessora (ou antecessora) imediata de uma página folha.
2. Remova a chave em questão.
3. Se a respectiva página folha contém o mínimo possível de chaves, então nenhuma outra ação é requerida.
4. Se ao invés disto a respectiva página contém menos chaves do que o permitido, procure nas páginas irmãs, à esquerda e à direita:
    1. Se uma das irmãs possui mais do que o mínimo possível de chaves, **redistribua**-as.
    2. Se isto não acontece com nenhuma das irmãs, **concatene** numa só folha estas duas páginas irmãs, juntamente com a chave mediana na página *pai*.
5. Se folhas são concatenadas, então aplique os passos $3 - 6$ à respectiva pagina *pai*.

– Obs.: A altura da árvore diminui se a última chave da raiz for removida.

• Resumidamente, os passos envolvidos na remoção de um chave da árvore B são:

1. Se a chave a ser removida não é uma folha, troque-a por sua sucessora (ou antecessora) imediata de uma página folha.
2. Remova a chave em questão.
3. Se a respectiva página folha contém o mínimo possível de chaves, então nenhuma outra ação é requerida.
4. Se ao invés disto a respectiva página contém menos chaves do que o permitido, procure nas páginas irmãs, à esquerda e à direita:
    1. Se uma das irmãs possui mais do que o mínimo possível de chaves, **redistribua**-as.
    2. Se isto não acontece com nenhuma das irmãs, **concatene** numa só folha estas duas páginas irmãs, juntamente com a chave mediana na página *pai*.
5. Se folhas são concatenadas, então aplique os passos $3 - 6$ à respectiva pagina *pai*.

– Obs.: A altura da árvore diminui se a última chave da raiz for removida.

• Resumidamente, os passos envolvidos na remoção de um chave da árvore B são:

1. Se a chave a ser removida não é uma folha, troque-a por sua sucessora (ou antecessora) imediata de uma página folha.
2. Remova a chave em questão.
3. Se a respectiva página folha contém o mínimo possível de chaves, então nenhuma outra ação é requerida.
4. Se ao invés disto a respectiva página contém menos chaves do que o permitido, procure nas páginas irmãs, à esquerda e à direita:
    1. Se uma das irmãs possui mais do que o mínimo possível de chaves, **redistribua**-as.
    2. Se isto não acontece com nenhuma das irmãs, **concatene** numa só folha estas duas páginas irmãs, juntamente com a chave mediana na página *pai*.
5. Se folhas são concatenadas, então aplique os passos $3 - 6$ à respectiva pagina *pai*.

– Obs.: A altura da árvore diminui se a última chave da raiz for removida.

• Resumidamente, os passos envolvidos na remoção de um chave da árvore B são:

1. Se a chave a ser removida não é uma folha, troque-a por sua sucessora (ou antecessora) imediata de uma página folha.
2. Remova a chave em questão.
3. Se a respectiva página folha contém o mínimo possível de chaves, então nenhuma outra ação é requerida.
4. Se ao invés disto a respectiva página contém menos chaves do que o permitido, procure nas páginas irmãs, à esquerda e à direita:
    1. Se uma das irmãs possui mais do que o mínimo possível de chaves, **redistribua**-as.
    2. Se isto não acontece com nenhuma das irmãs, **concatene** numa só folha estas duas páginas irmãs, juntamente com a chave mediana na página *pai*.
5. Se folhas são concatenadas, então aplique os passos $3 - 6$ à respectiva pagina *pai*.

– Obs.: A altura da árvore diminui se a última chave da raiz for removida.

• Resumidamente, os passos envolvidos na remoção de um chave da árvore B são:

1. Se a chave a ser removida não é uma folha, troque-a por sua sucessora (ou antecessora) imediata de uma página folha.
2. Remova a chave em questão.
3. Se a respectiva página folha contém o mínimo possível de chaves, então nenhuma outra ação é requerida.
4. Se ao invés disto a respectiva página contém menos chaves do que o permitido, procure nas páginas irmãs, à esquerda e à direita:
    1. Se uma das irmãs possui mais do que o mínimo possível de chaves, **redistribua**-as.
    2. Se isto não acontece com nenhuma das irmãs, **concatene** numa só folha estas duas páginas irmãs, juntamente com a chave mediana na página *pai*.
5. Se folhas são concatenadas, então aplique os passos $3 - 6$ à respectiva pagina *pai*.

– Obs.: A altura da árvore diminui se a última chave da raiz for removida.

• Resumidamente, os passos envolvidos na remoção de um chave da árvore B são:

1. Se a chave a ser removida não é uma folha, troque-a por sua sucessora (ou antecessora) imediata de uma página folha.
2. Remova a chave em questão.
3. Se a respectiva página folha contém o mínimo possível de chaves, então nenhuma outra ação é requerida.
4. Se ao invés disto a respectiva página contém menos chaves do que o permitido, procure nas páginas irmãs, à esquerda e à direita:
    1. Se uma das irmãs possui mais do que o mínimo possível de chaves, **redistribua**-as.
    2. Se isto não acontece com nenhuma das irmãs, **concatene** numa só folha estas duas páginas irmãs, juntamente com a chave mediana na página *pai*.
5. Se folhas são concatenadas, então aplique os passos $3 - 6$ à respectiva pagina *pai*.

– Obs.: A altura da árvore diminui se a última chave da raiz for removida.

• Resumidamente, os passos envolvidos na remoção de um chave da árvore B são:

1. Se a chave a ser removida não é uma folha, troque-a por sua sucessora (ou antecessora) imediata de uma página folha.
2. Remova a chave em questão.
3. Se a respectiva página folha contém o mínimo possível de chaves, então nenhuma outra ação é requerida.
4. Se ao invés disto a respectiva página contém menos chaves do que o permitido, procure nas páginas irmãs, à esquerda e à direita:
    1. Se uma das irmãs possui mais do que o mínimo possível de chaves, **redistribua**-as.
    2. Se isto não acontece com nenhuma das irmãs, **concatene** numa só folha estas duas páginas irmãs, juntamente com a chave mediana na página *pai*.
5. Se folhas são concatenadas, então aplique os passos $3 - 6$ à respectiva pagina *pai*.

– Obs.: A altura da árvore diminui se a última chave da raiz for removida.

Referências:

– Michael J. Folk e Bill Zoellick. File Structures, Addison-Wesley, 1992.

– Nívio Ziviani. Projeto de Algoritmos com Implementações em Pascal e C, Thomson Learning, 2011.