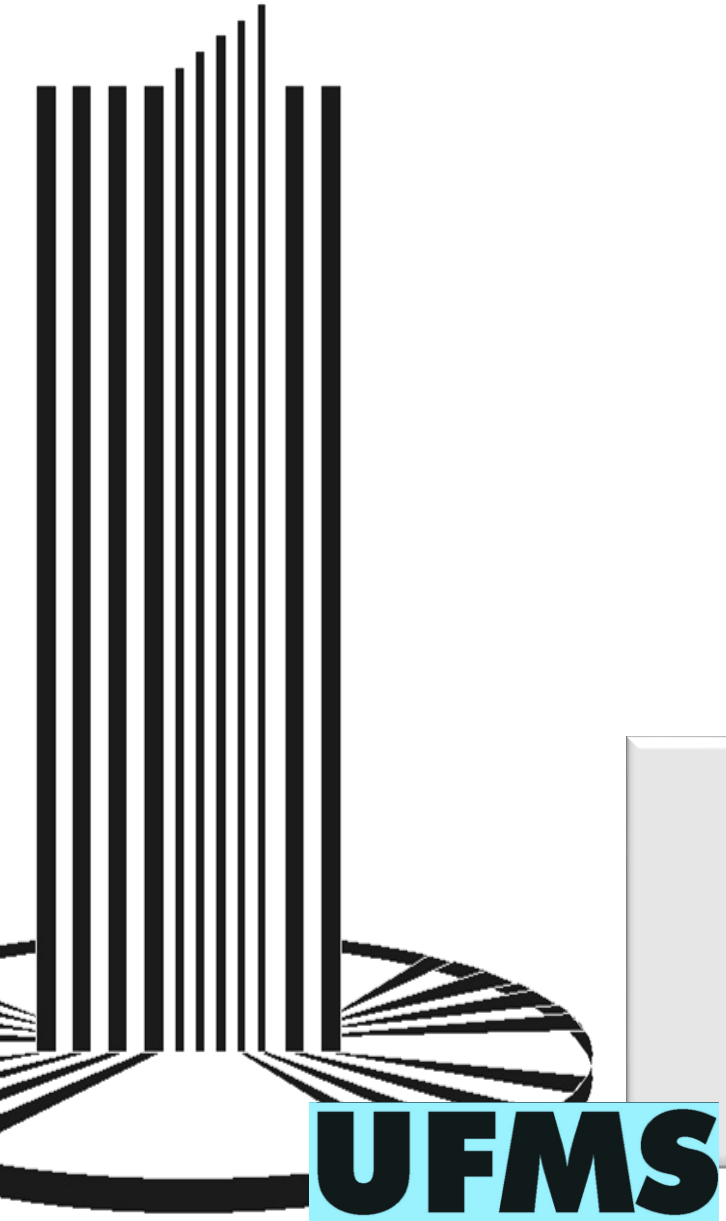


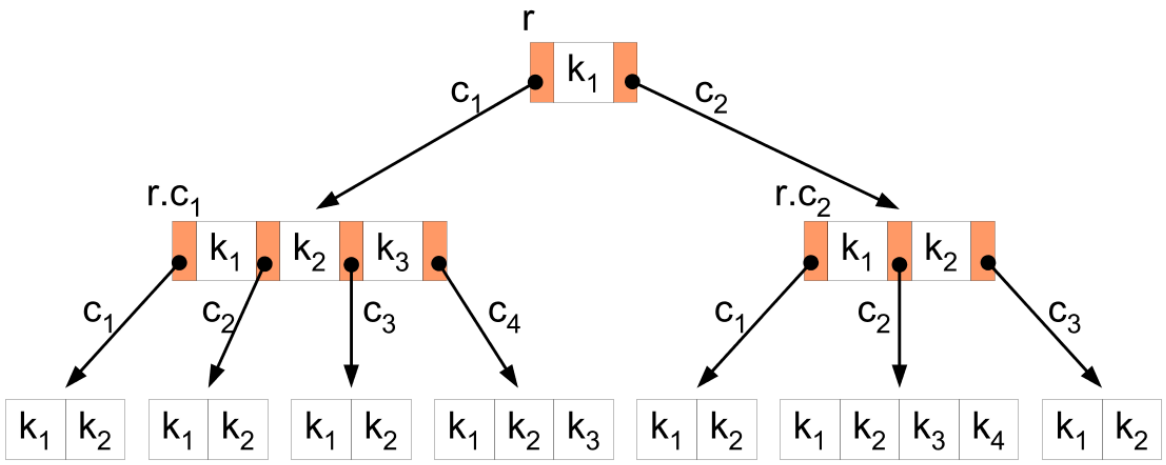
Árvore B

Profª Yorah Bosse

yorah.bosse@gmail.com
yorah.bosse@ufms.br

- Segundo DROSDEK (2002):

“ Nos programas de banco de dados, onde a maioria da informação está guardada em discos e fitas, a penalidade do tempo para acessar a armazenagem secundária pode ser significativamente reduzida pela escolha apropriada das estruturas de dados. As árvores B são uma de tais abordagens.”



Propriedades

- Uma árvore B de ordem m é uma árvore de busca múltipla com as seguintes propriedades:
  1. Se a raiz não for folha, ela possuirá pelo menos duas sub árvores
  2. Cada nó não raiz e não folha contém, k-1 chaves e k ponteiros para as sub árvores onde $(m/2) \leq k \leq m$
  3. Cada nó folha contém k-1 chaves, onde $(m/2) \leq k \leq m$
  4. Todas as folhas estão no mesmo nível
  5. Todas as chaves de um nó são armazenadas em ordem crescente

Observações

- Com as propriedades apresentadas, uma árvores B:
  - Sempre estará, pelo menos, cheia até a metade
  - Terá poucos níveis
  - Estará perfeitamente balanceada
- Usualmente, m é grande (50 a 500)

Observações

- Um nó não raiz de uma árvore B de ordem m, poderá ter:

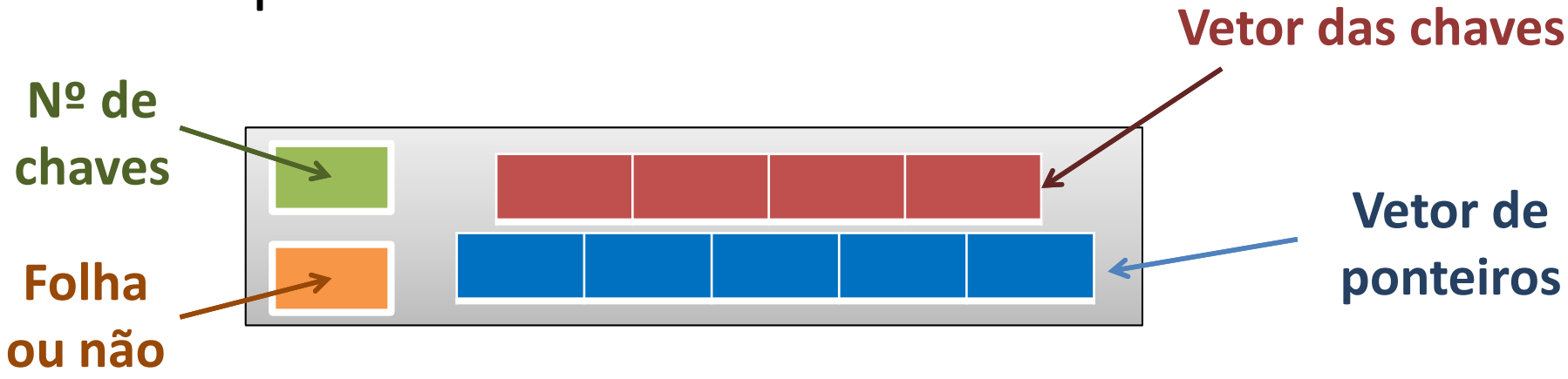
Ordem (m)	Nº máximo de filhos (m)	Nº máximo de chaves (m-1)	Nº mínimo de chaves (m/2)-1
4	4	3	1
5	5	4	2
6	6	5	2
7	7	6	3
8	8	7	3
9	9	8	4
10	10	9	4

  - O arredondamento de (m/2) é para cima

Nó da árvore

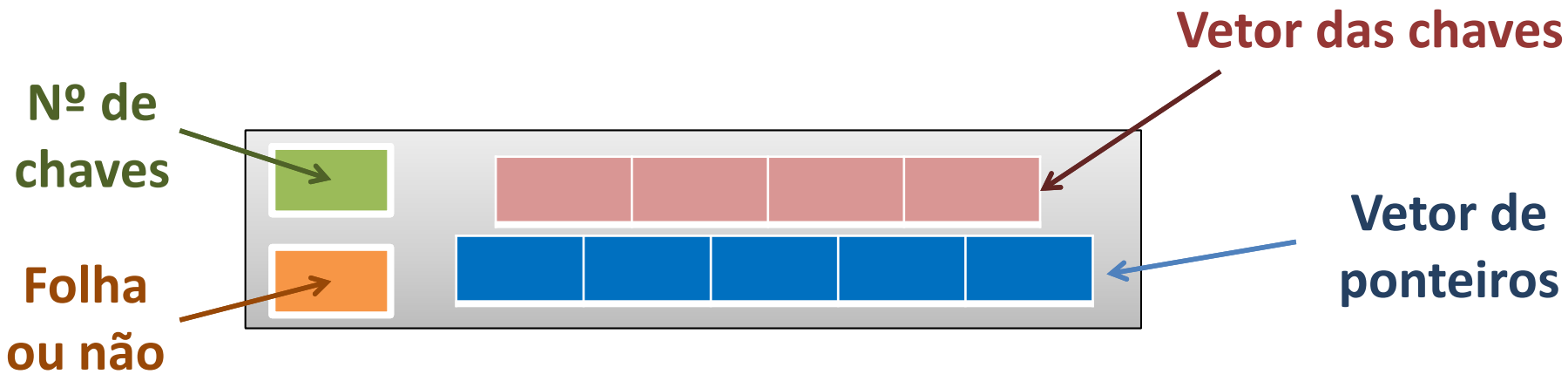
- Um nó da árvore de ordem m é uma estrutura que possui:
  - um vetor com m-1 chaves
  - um vetor com m ponteiros para outros nós, e,
  - possivelmente, outras informações que possam ajudar na manutenção, como por exemplo: números de chaves existentes, bandeira que informa se é ou não um nó folha

- Exemplo de nó de ordem 5:

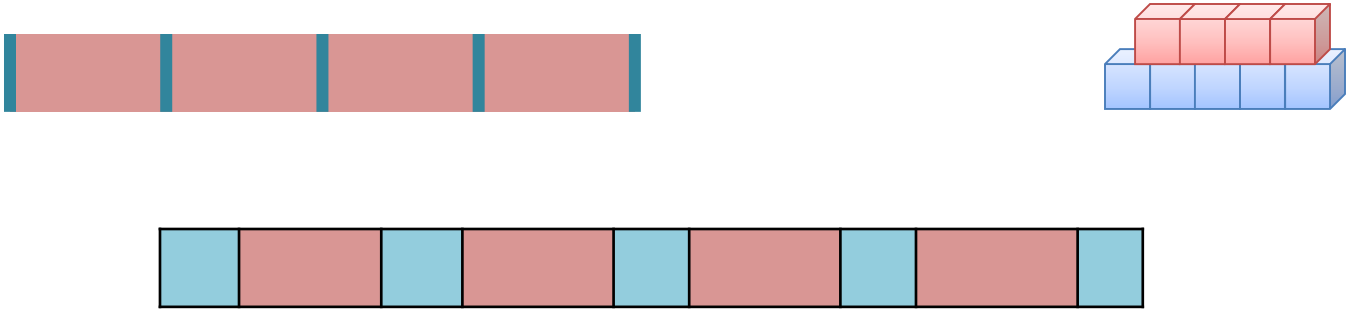


Nó da árvore

- Exemplo de nó de ordem 5:

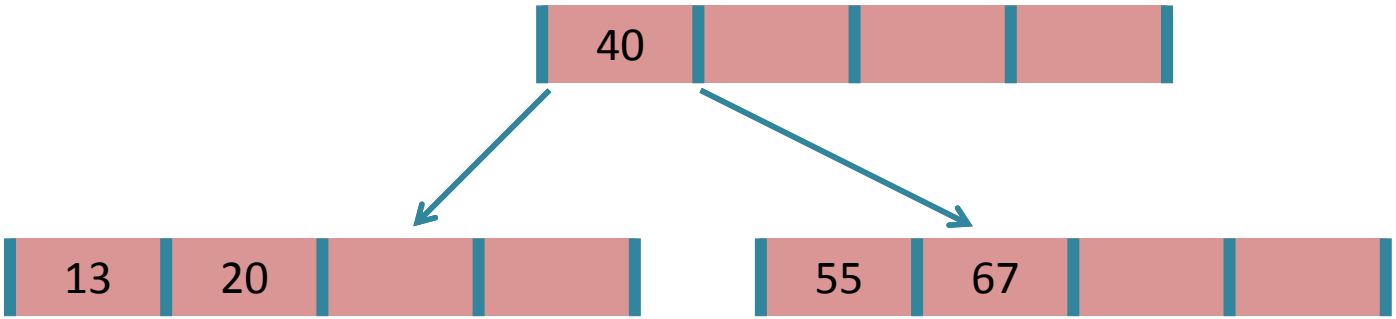


- Outras formas de representações dos nós:

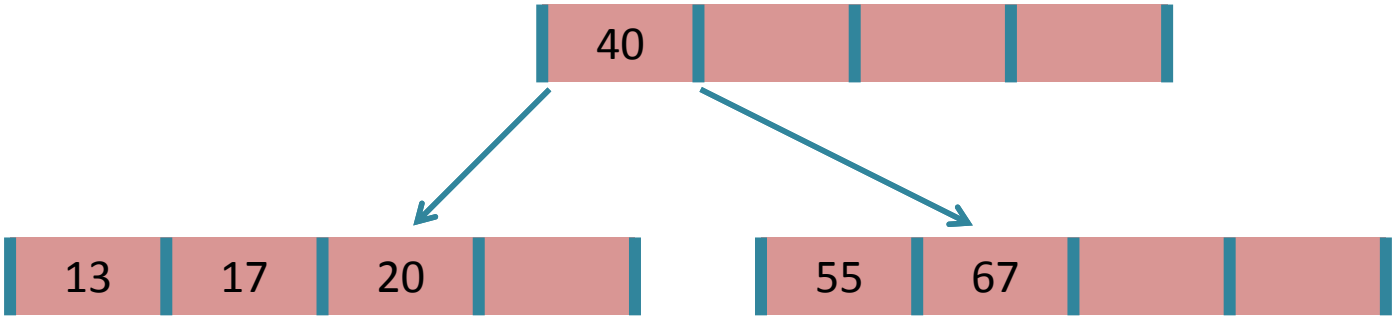


Exemplos

- Árvore B de ordem 5 :



- A mesma árvore após inserir a chave 17:



Estruturas de um nó

- Estrutura de um nó:

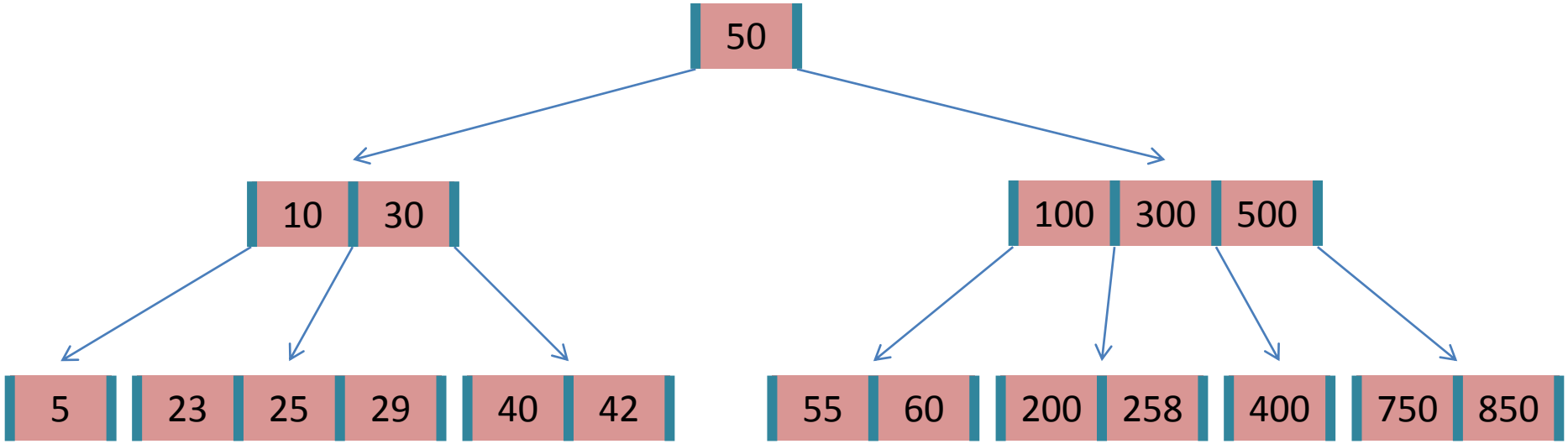
```
const m = 4;
typedef struct no_arvoreB arvoreB;

struct no_arvoreB {
   int num_chaves;
   char chaves[m-1];
   arvoreB *filhos[m];
   int folha; //1=true e 0=false
};
```

  - De acordo com a definição, a árvore B mais simples ocorre quando ***m=4***. Neste caso todo nó não folha possui 2, 3 ou 4 filhos. Esta árvore é também conhecida por **árvore 2-3-4**.

Exemplos

- Exemplo de árvore 2-3-4:



- Para inserção existem três situações comuns:
  1. Chave é colocada em uma folha que ainda tem espaço livre
     - A chave é inserida de forma ordenada dentro do nó folha correspondente

Exemplos para uma árvore de ordem 5:

a) Inserir na árvore o valor 5

b) Inserir na árvore o valor 45

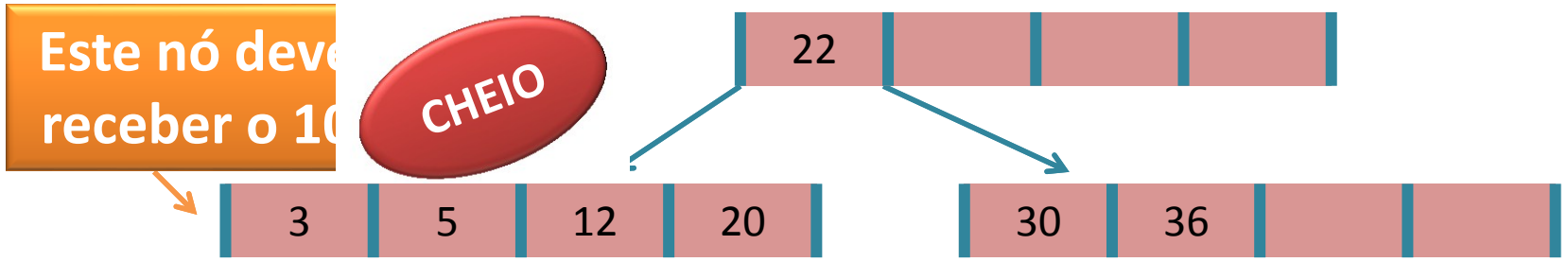
c) Inserir na árvore o valor 25

d) Inserir na árvore o valor 3

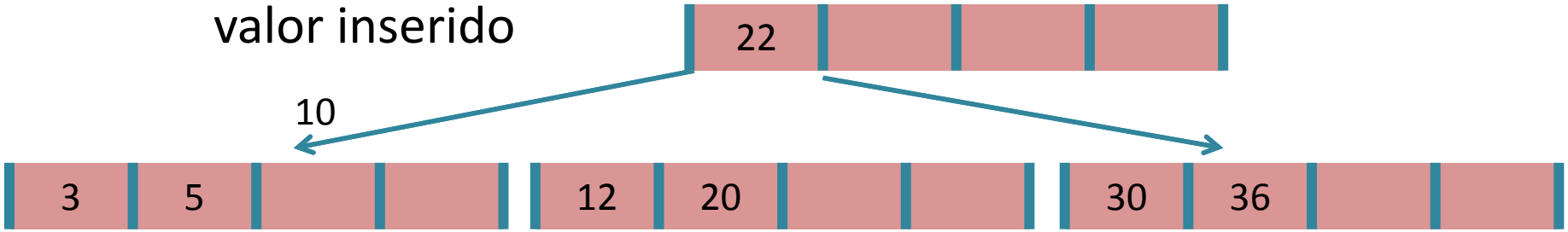
- Para inserção existem três situações comuns:
  2. Chave é colocada em uma folha cheia.

Exemplos para uma árvore de ordem 5:

Inserir na árvore abaixo o valor 10



Primeiro: O nó que receberá a nova chave é dividido em dois e o valor inserido



Inserção

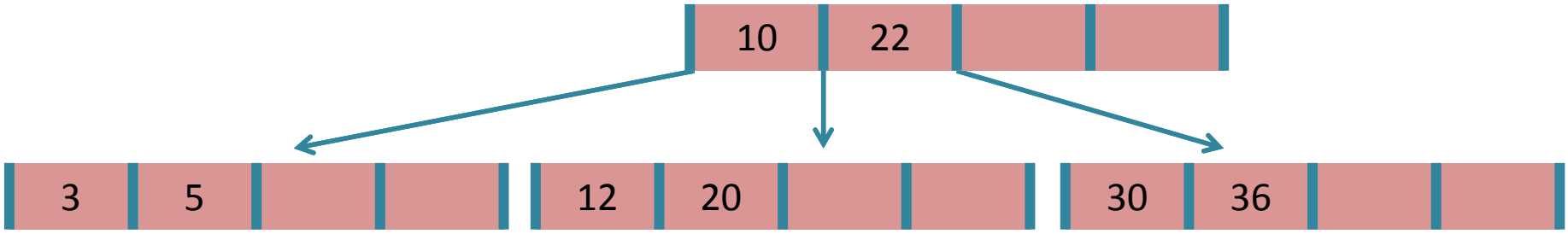
- Para inserção existem três situações comuns:

2. Chave é colocada em uma folha cheia.

Exemplos para uma árvore de ordem 5:

Inserir na árvore abaixo o valor 10

Segundo: A última chave do nó que já existia sobe e arrumam-se as chaves do nó ascendente.

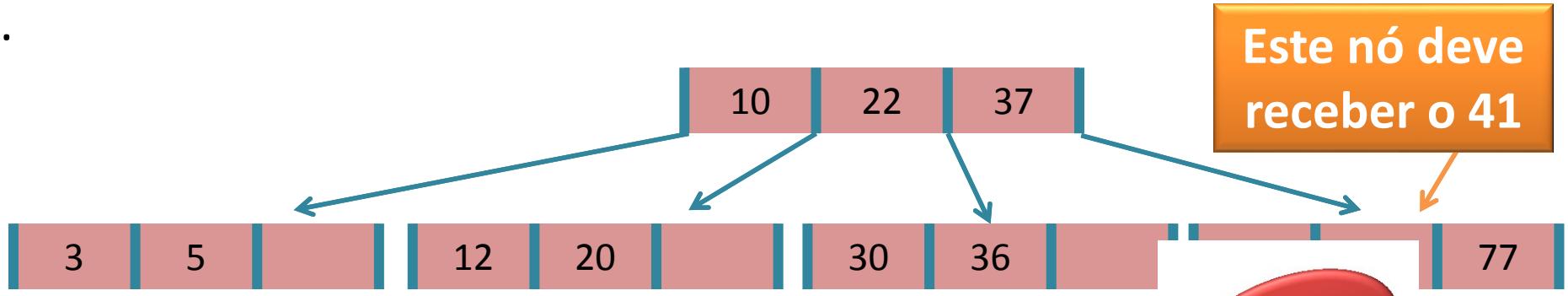


- Para inserção existem três situações comuns:
  3. Chave é colocada em uma raiz que está cheia:
     - Os passos da segunda situação comum nas inserções são realizados também na raiz, recursivamente, criando-se uma nova raiz.

Inserir na árvore abaixo o valor 41

Primeiro: Após encontrar e verificar que o nó está cheio, divide-o em dois e a nova chave é inserida

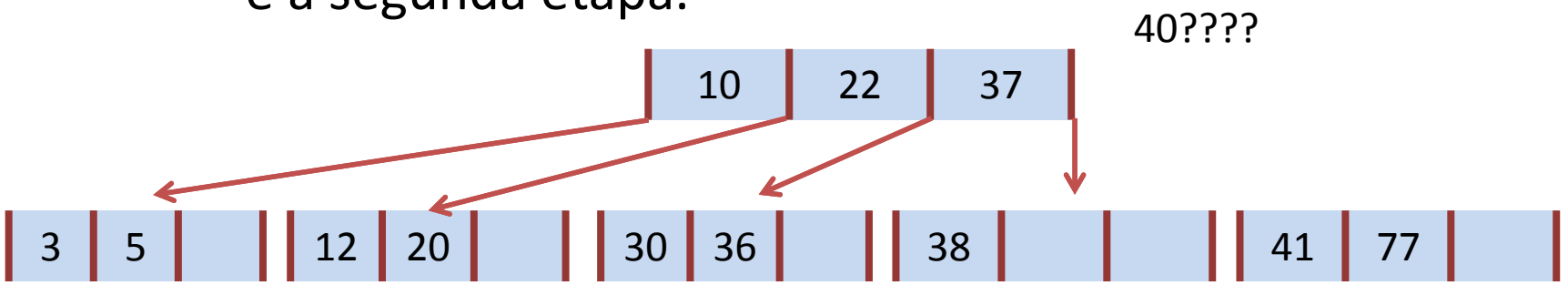
.



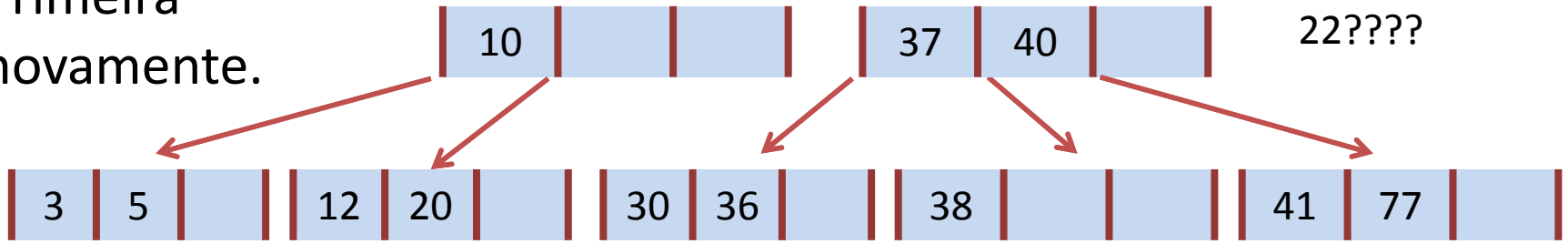
- Para inserção existem três situações comuns:
  3. Chave é colocada em uma raiz que está cheia:

Inserir na árvore abaixo o valor 41

Segundo: A última chave do nó que já existia sobe e, como o nó ascendente também encontra-se cheio, repete-se a primeira e a segunda etapa.



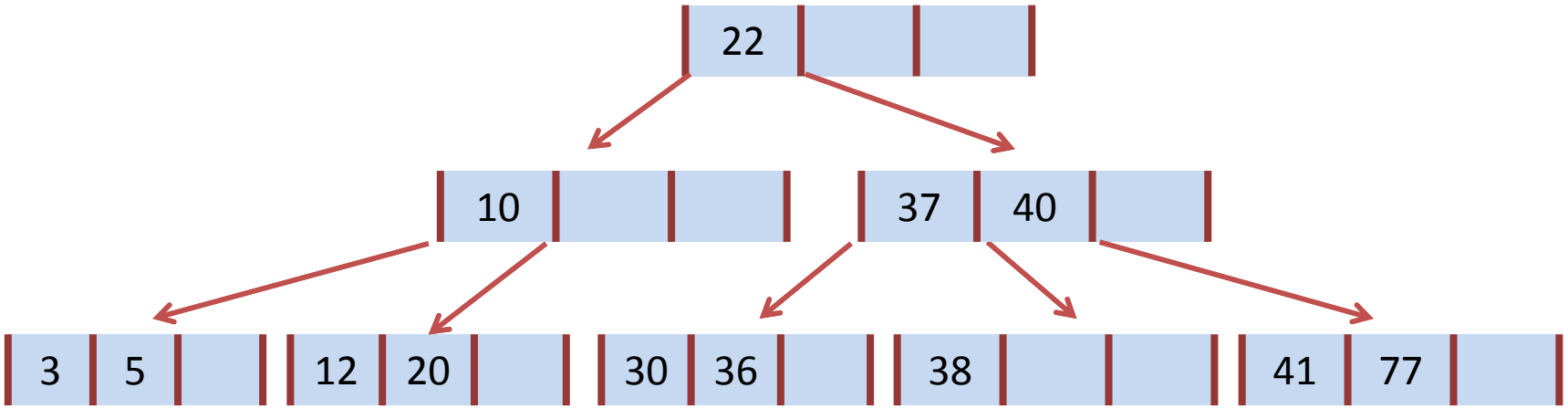
Primeira novamente.



- Para inserção existem três situações comuns:
  3. Chave é colocada em uma raiz que está cheia:

Inserir na árvore abaixo o valor 41

Segunda novamente: A última chave do nó que já existia sobe e, como o nó ascendente não existe, um novo nó é criado como raiz e a nova chave é inserida.



Exercício

- Insira em uma árvore B, de ordem 5, os valores (chaves) abaixo na sequência apresentada, um de cada vez.

m,	f,	q,	w,
k,	c,	s,	z,
t,	o,	g,	v,
d,	b,	j,	r,
u,	e,	i,	x,
a,	y,	h,	p,
l,	n		

Resposta

Inovação e Consequências

- Inovação
  - não é necessário construir a árvore a partir da raiz, como é feito para ABBs e AVLs
- Consequências
  - não mais se aloca chaves inadequadas na raiz
    - chaves na raiz da árvore “emergem” naturalmente
  - não é mais necessário tratar o problema de desbalanceamento após este ocorrer
    - como rebalancear uma árvore AVL não é mais uma questão para se resolver a posteriori
      - balanceamento ocorre naturalmente

Analisando

- Na inserção de uma nova chave, a árvore B, diferente da árvore binária, cresce para cima

- Uma árvore de p nós e altura h, terá realizado **p-h** divisões.
  - Exemplo: para uma árvore com 3 nós e altura 2:
    - Total de divisões = p – h = 3 – 2 = 1 divisão

- Em uma árvore de p nós existem, pelo menos **1+[(m/2)-1]*(p-1)** chaves
  - Exemplo: para uma árvore de ordem 6 com 3 nós:
    - Mínimo de chaves = 1+[(6/2)-1]*(3-1) = 1+2*2 = 5 chaves

Analisando

- A probabilidade média de divisões, com relação ao número de chaves é

$$\frac{p - h}{1 + \left[\left(\frac{m}{2}\right) - 1\right] * (p - 1)} = \frac{\mathbf{1}}{\left(\frac{\boldsymbol{m}}{\mathbf{2}}\right) - \mathbf{1}}$$

  - Exemplo: para uma árvore de ordem 6 a quantidade de fracionamentos é igual à 0,5 por chave inserida; de ordem 100 é igual à 0,0204; de ordem 1000 é 0,0020.

- Ou seja, quanto mais chaves cabem em cada nó, menos divisões são realizadas

Altura no Pior Caso

- Qual a altura máxima que uma árvore com **N** chaves e ordem **m** pode atingir?

- Pior caso ocorre quando cada página tem apenas o número mínimo de descendentes, e a árvore possui, portanto, altura máxima e largura mínima
  - Note que a altura é igual ao nível mais profundo
  - Pode-se provar que o nível mais profundo h de uma árvore B com n chaves e ordem **m** é tal que:

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

Altura no Pior Caso

- Exemplo (**m** = 512 e **n** = 1.000.000):
  - **h <=** $1 + \log_{256} (500.000,5)$ **=** 3,37
  - Logo, a árvore terá no máximo altura de 3 níveis
  - No pior caso, 3 acessos serão necessários para localizar uma chave dentre 1.000.000

Analise as duas situações

1. Chaves dos nós com a chave primária e a referência da sua localização dentro do arquivo

ou

2. Arquivo com estruturas de árvores binárias, servindo como arquivo de índices para acessar o arquivo de dados

Pesquisa

- A pesquisa 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
- Como as chaves estão ordenadas, basta realizar uma busca binária nos elementos de cada nó
- Isso levará tempo ***O(lg(t))***
- 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 $O(lg(t)log_t(n))$.

Pesquisa

- Abaixo, o código de busca em uma árvore B:

```
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);
}
```

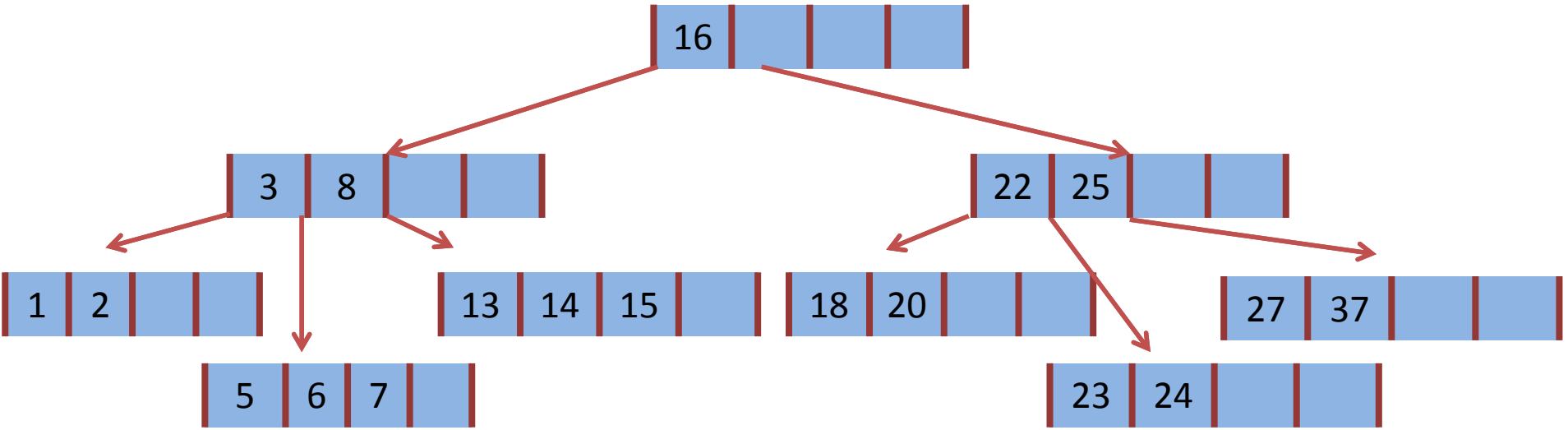
Pesquisa

```
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 posíção 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.
}
```

Exclusão

- Na exclusão existem dois casos principais:
  - Eliminação de uma chave de um nó folha
  - Eliminação de uma chave de um nó não folha

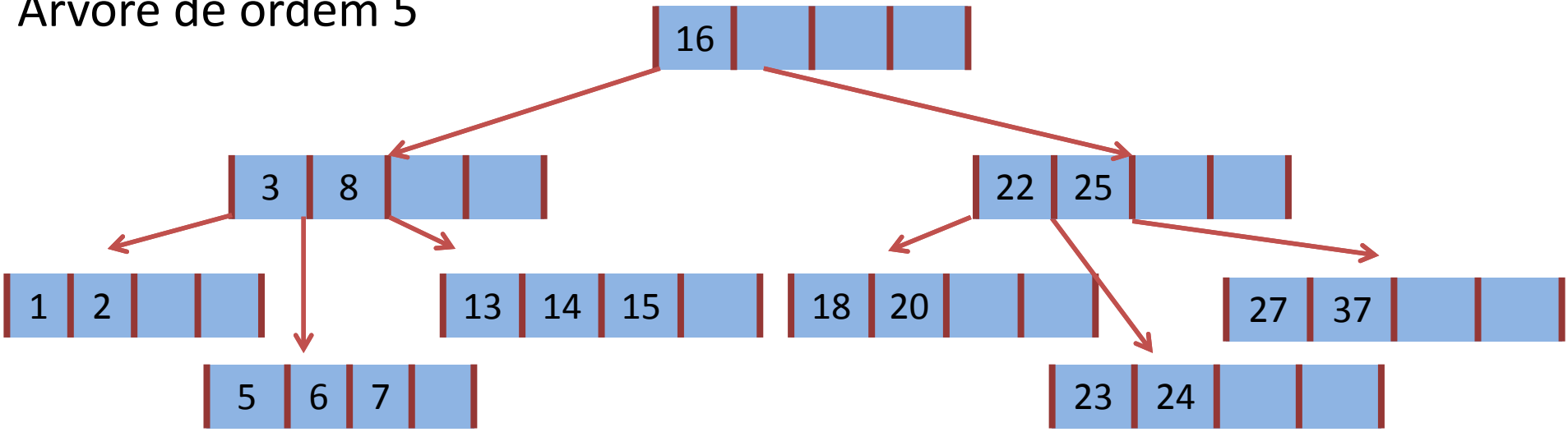


Exclusão

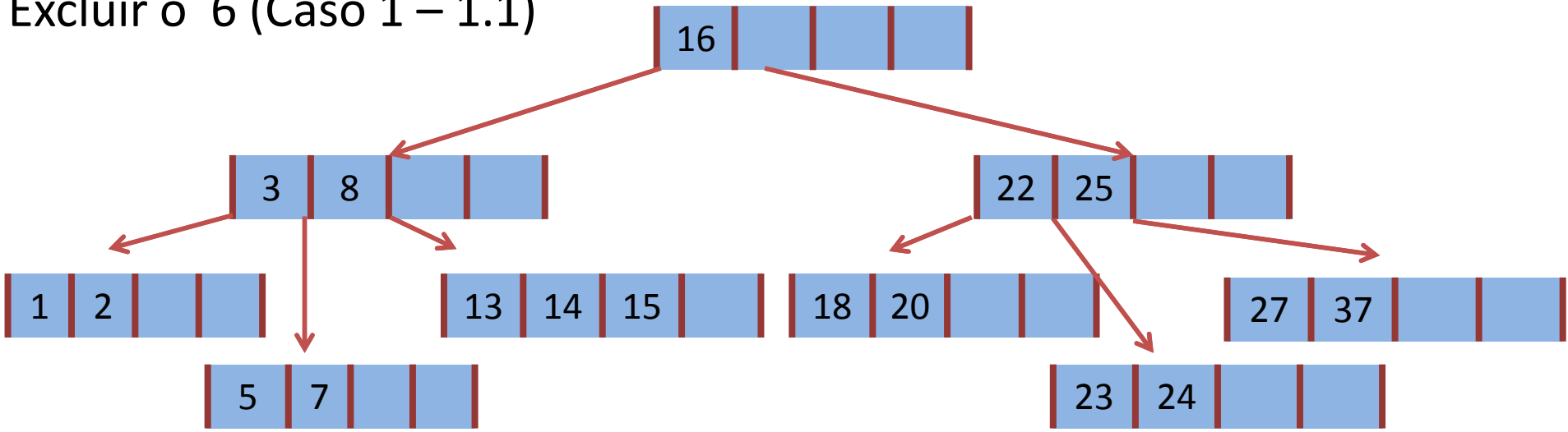
- **Caso 1 – Exclusão de uma chave de um nó folha**
  - 1.1. Após a exclusão o nó permanece com (m/2)-1 nós ou mais
  - 1.2. Após a exclusão o nó permanece com menos de (m/2)-1 nós
    - 1.2.1. Se o nó possui irmãos à esquerda ou a direita, com o número excedendo o mínimo (m/2)-1 de chaves
    - 1.2.2. Se o nó possui irmãos à esquerda ou direita, com o número mínimo (m/2)-1 de chaves
      - ** Um caso particular quando funde-se um nó folha ou não folha com seu irmão e o nó ascendente é a raiz com apenas um nó. Este é o único caso que dois nós desaparecem.
- **Caso 2 – Exclusão de uma chave de um nó não folha**
  - A chave a ser excluída é substituída pelo seu antecessor ou sucessor imediato, devendo este estar em uma folha. Para retirá-lo da folha, retorna-se ao Caso 1

Exclusão

Árvore de ordem 5

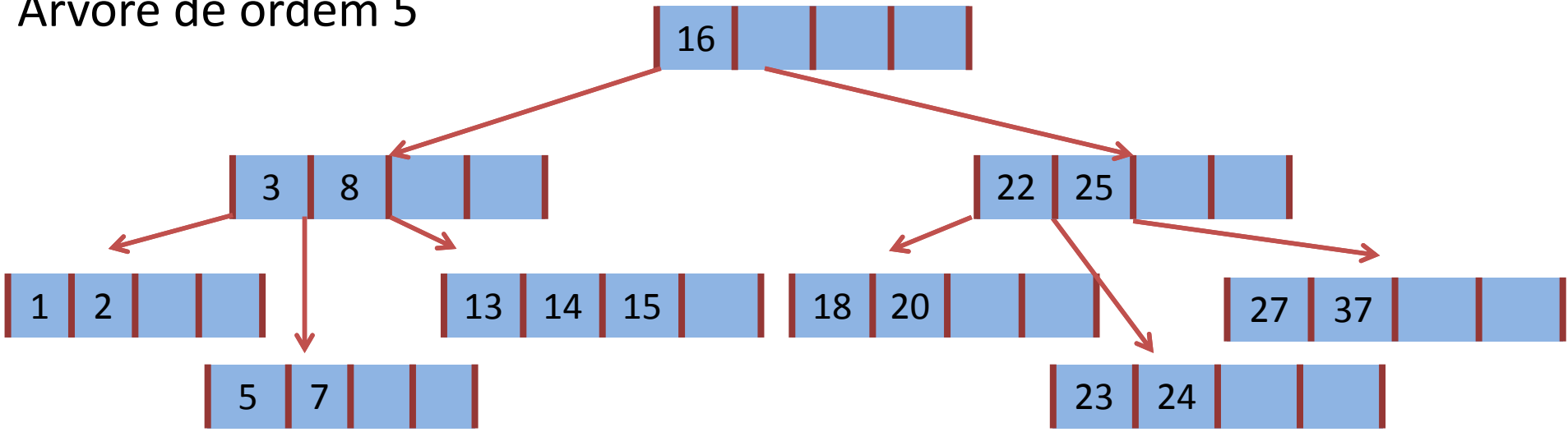


Excluir o 6 (Caso 1 – 1.1)

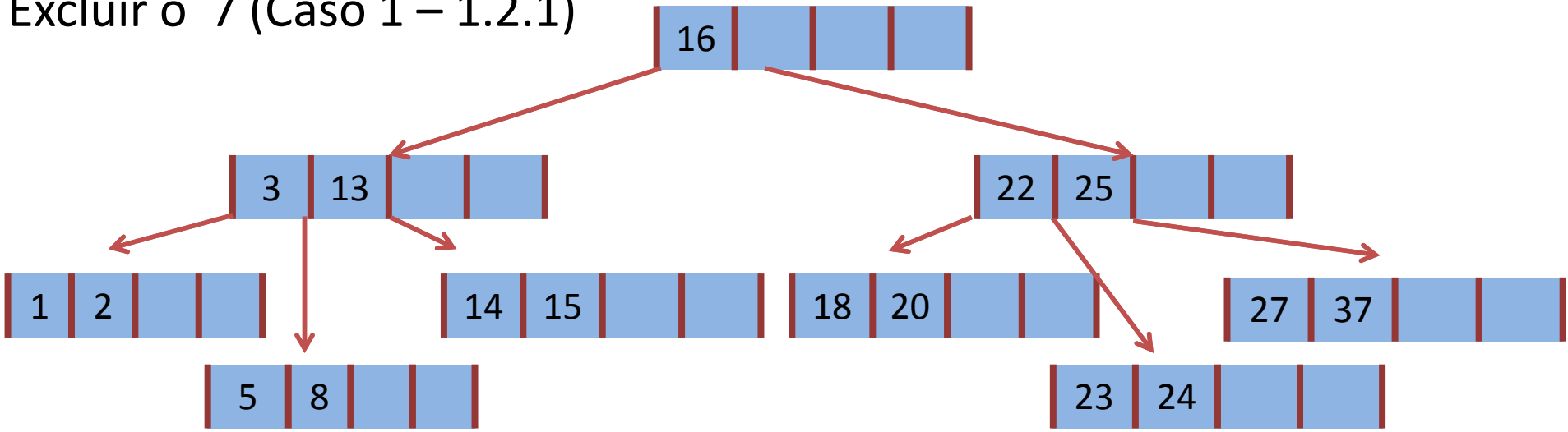


Exclusão

Árvore de ordem 5

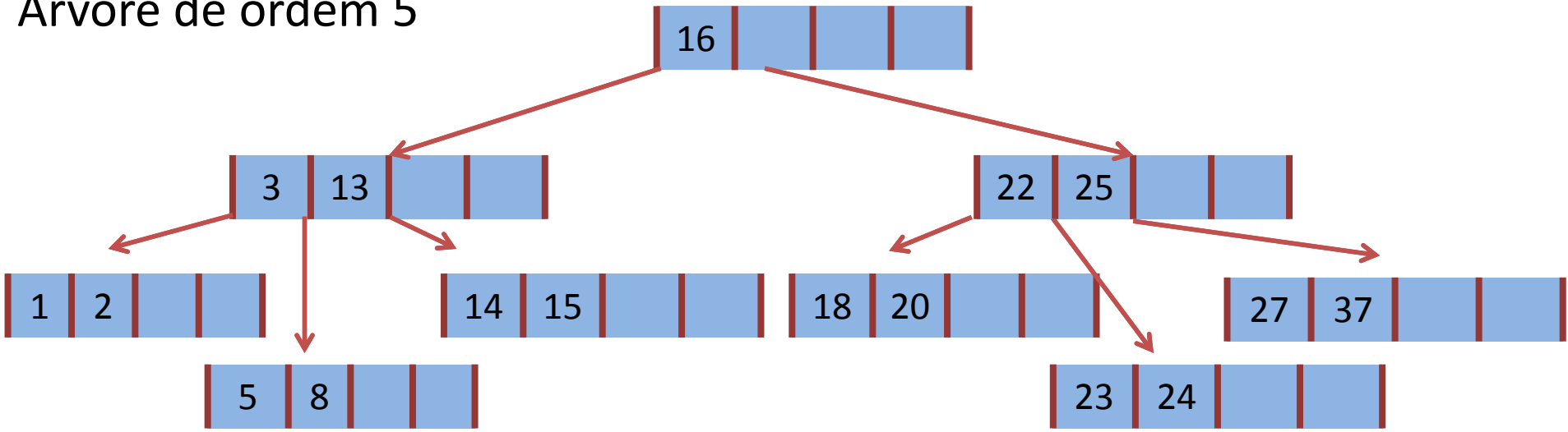


Excluir o 7 (Caso 1 – 1.2.1)

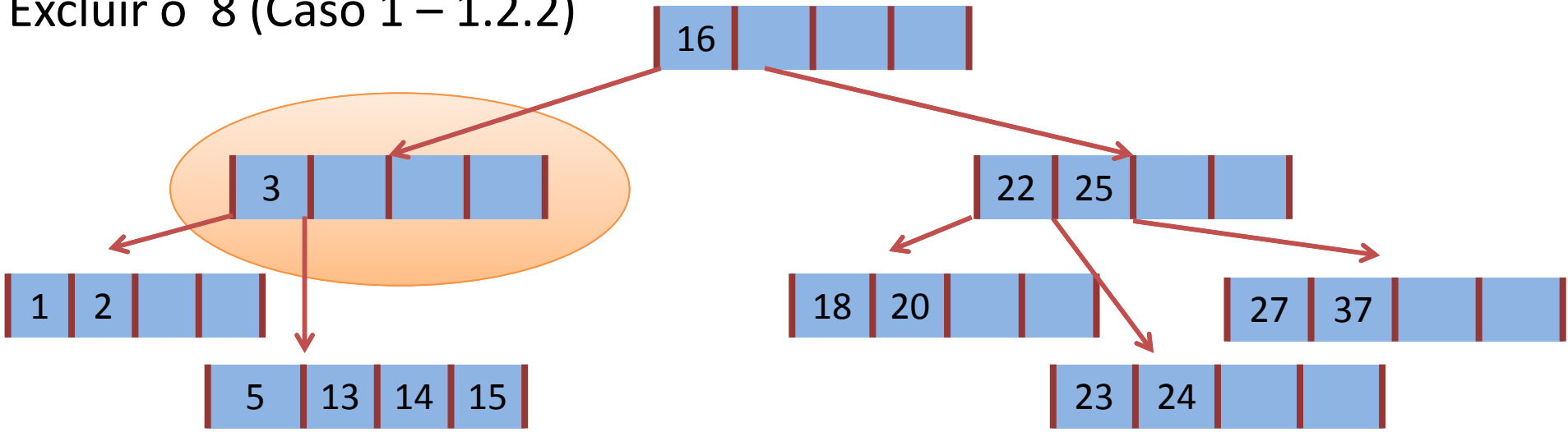


Exclusão

Árvore de ordem 5

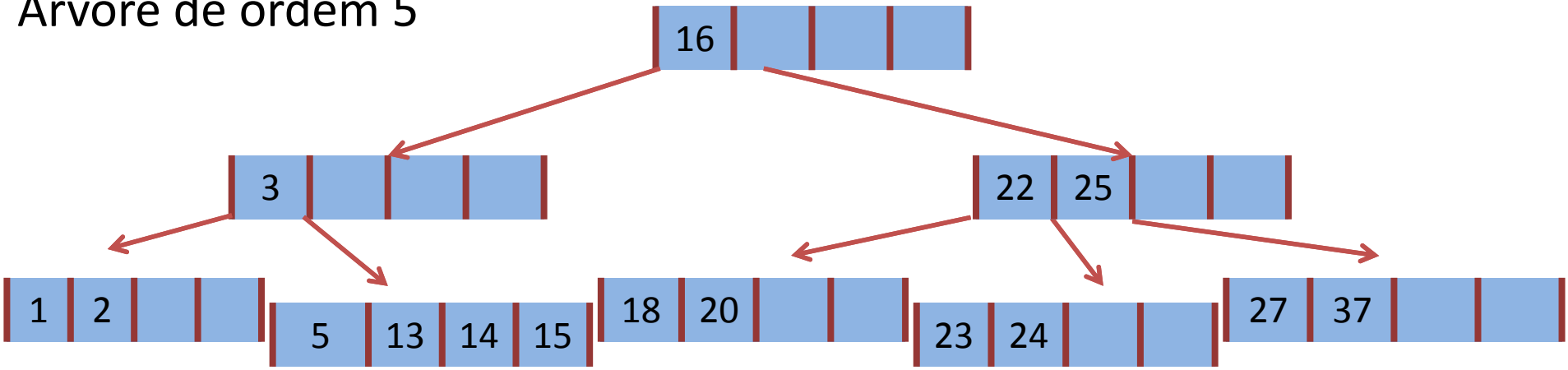


Excluir o 8 (Caso 1 – 1.2.2)



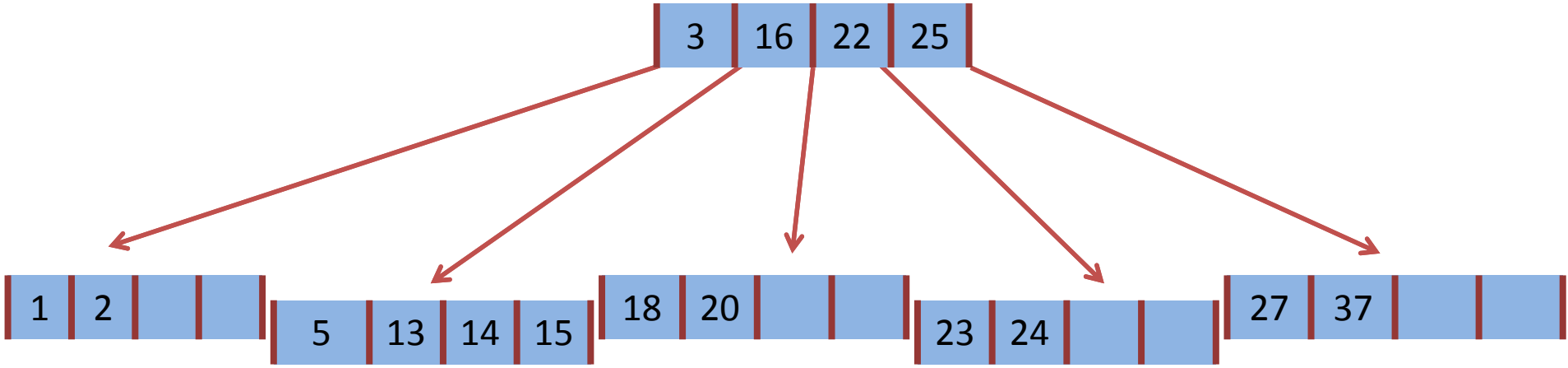
Exclusão

Árvore de ordem 5



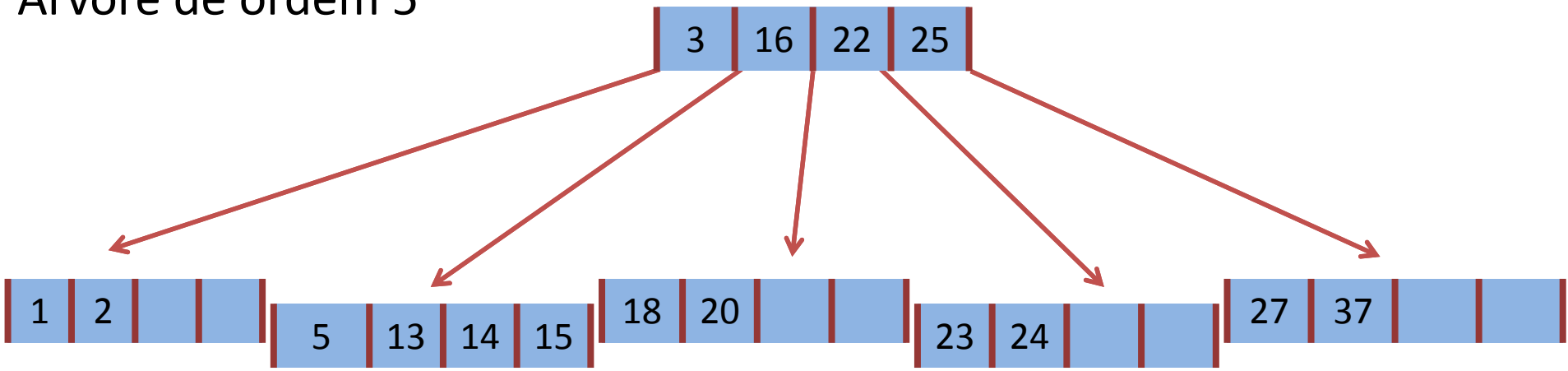
Continuação

Excluir o 8 (Caso 1 – 1.2.2**)



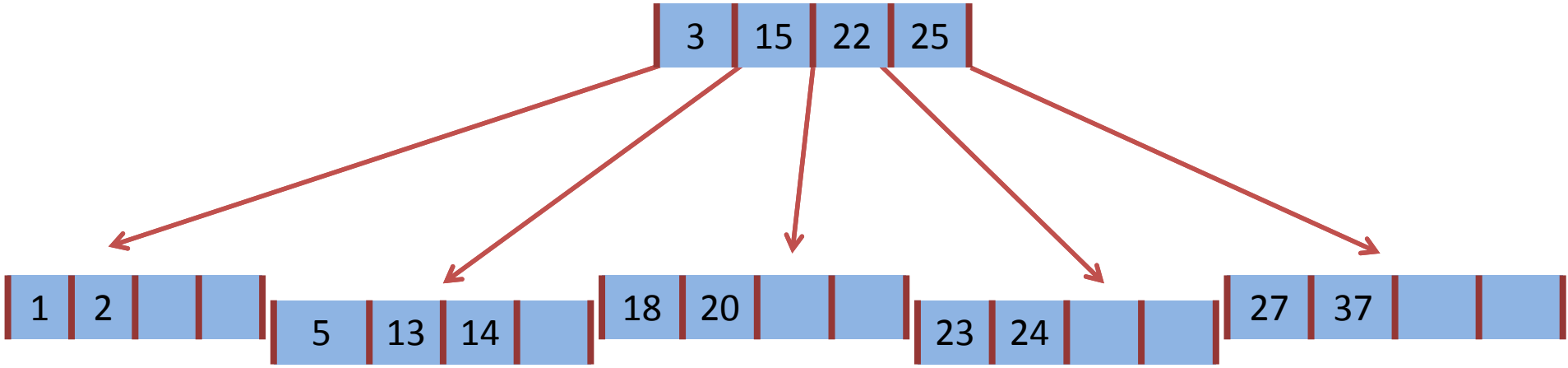
Exclusão

Árvore de ordem 5



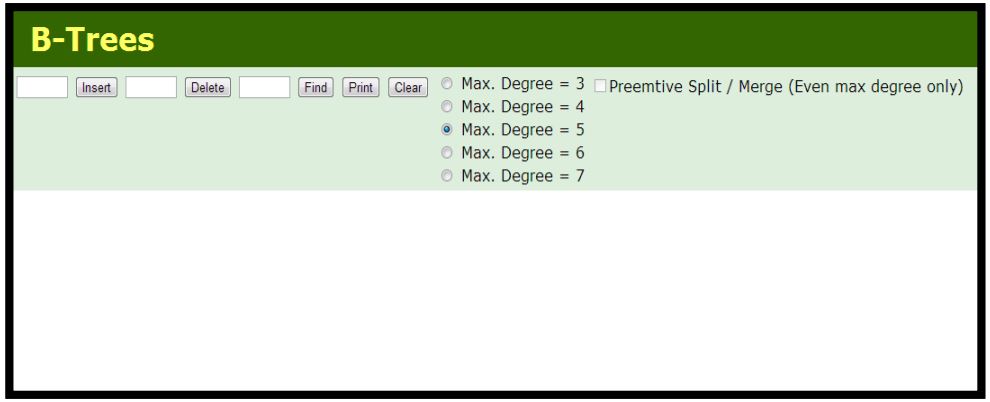
Continuação

Excluir o 16 (Caso 2)

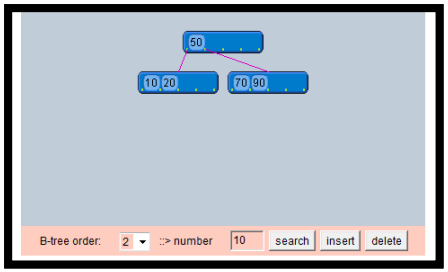


Animações da Árvore B

- http://www.nttvu.edu.cn/sec/kcjx/visualDataStructure/visualization/BTree.html



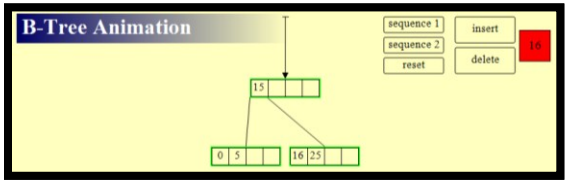
- http://slady.net/java/bt/view.php



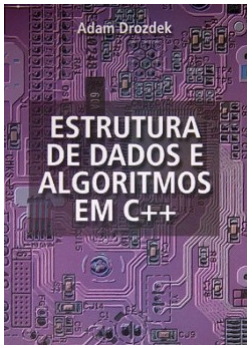
- Vivio B-Tree (necessita do plug-in vivio):

  https://www.cs.tcd.ie/Jeremy.Jones/vivio/trees/B-tree.htm

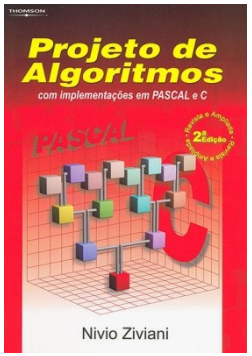
  Busca no Google: vivio b tree animation



Referência Bibliográfica



DROSDEK, A. Estrutura de dados e algoritmos em C++. Cengage: 2002.



ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C. 2.ed. Thomson, 2004.

Site:

http://www.lcad.icmc.usp.br/~nonato/ED/B_arvore/btree.htm
do professor Gustavo Nonato

Exercícios

1. Mostre os resultados de inserir as chaves a seguir em uma árvore B de ordem 5 inicialmente vazia: 14, 39, 1, 6, 41, 32, 8, 38, 43, 3, 36. Após essas inserções, mostre os resultados de remover as chaves ímpares nessa estrutura.

2. Mostre os resultados de inserir as chaves a seguir em uma árvore B de ordem 7 inicialmente vazia: F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E. Em seguida, mostre os resultados de remover todas as consoantes dessa árvore.

3. Dada a árvore B ordem 5 que contém todas as letras do alfabeto, mostre o que acontece com a árvore com a inserção das chaves $ (menor que A) e a seguir, da chave [(maior que Z).

4. Dada uma árvore B de ordem 256, qual o número máximo de descendentes por página? Qual o número mínimo de chaves (desconsiderando a raiz)? E da raiz? Quantas chaves têm uma página não-folha com 200 descendentes?

Fonte: http://www.lcad.icmc.usp.br/~jbatista/sce183/mat/listaArvoreB_2.pdf