Árvore 2-3

Luigi Wagner
Rafael Alessandro
Rafael Falcão

Ciências da Computação – Disciplina: Estruturas de Dados – 1
Instituto de Informática
Universidade Federal de Goiás

17 de novembro de 2017

Sumário

1 Introdução
2 Pesquisa
3 Inserção
4 Remoção
5 Códigos
6 Aplicações
7 Questionário
8 Referências Bibliográficas

Introdução

Uma Árvore 2-3 é uma árvore onde cada nó com filho (nó interno) tem também 2 filhos (2-node) e 1 elemento de dados (chave) ou 3 filhos (3-nodes) e 2 elementos de dados (chaves). Os nós externos a árvore (nós-folha) não tem filhos e possuem um ou dois elementos de dados (chaves).

Introdução

Propriedades

As principais `propriedades` de uma Árvore 2-3 são:

- Cada nó interno tem dois filhos (2-node) se tem uma chave, ou três filhos (3-node) se tem duas chaves;
- Cada nó não-folha tem 2 ou 3 filhos. Se tem 2 filhos tem 1 item de dados e se tem 3 filhos tem 2 itens de dados;
- Todos os dados são ordenados;
- Todas as folhas estão no mesmo nível- assim como na Árvore B;
- Cada nó folha tem 1 ou 2 campos.

Como, por exemplo:

Introdução

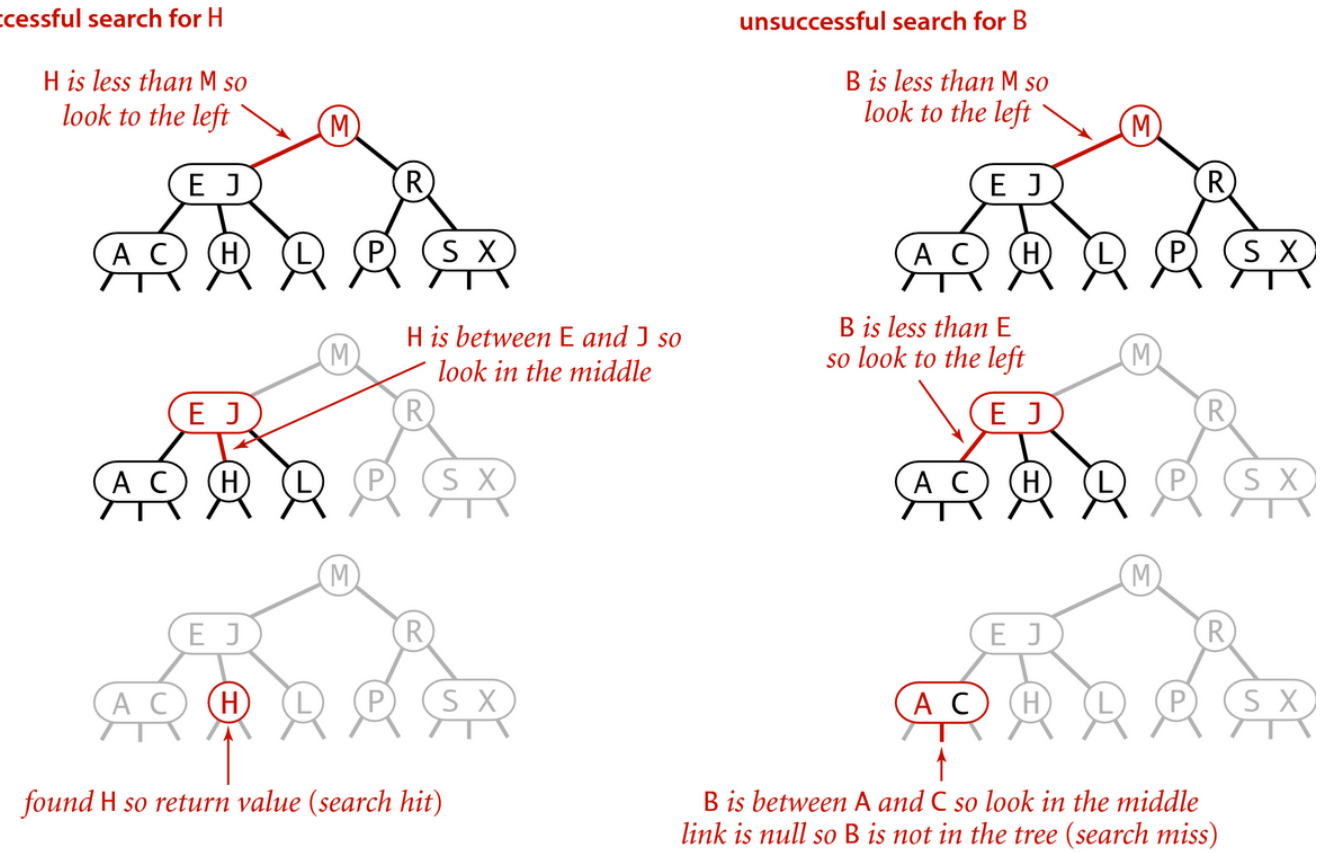


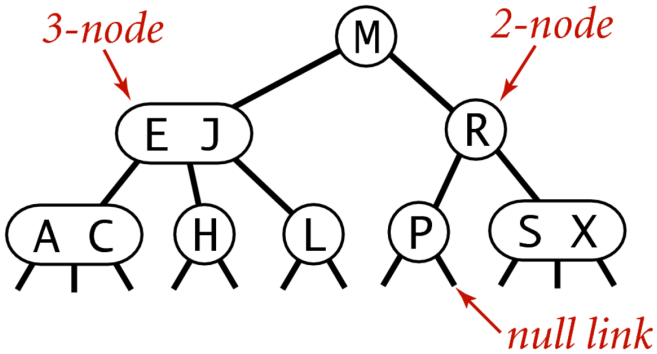
Figura: `https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/figuressw/Chapter3/TTsearch.png`

Introdução

Componentes ou Anatomia

Uma Árvore 2-3 é composta por:

1. Componente 1;
2. Componente 2;
3. Componente 3.



Anatomy of a 2-3 search tree

Figura: `https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/figuressw/Chapter3/TTanatomy.png`

Introdução

Bloco

Um `bloco` pode conter figuras, tabelas, listas, etc.

Sumário

1 Introdução

2 Pesquisa

3 Inserção

4 Remoção

5 Códigos

6 Aplicações

7 Questionário

8 Referências Bibliográficas

Pesquisa em uma Árvore 2-3

Vamos começar pelo mais básico procedimento a ser realizado ao lidar com uma Árvore.

Pesquisa em uma Árvore 2-3

Exemplo de uma Árvore 2-3

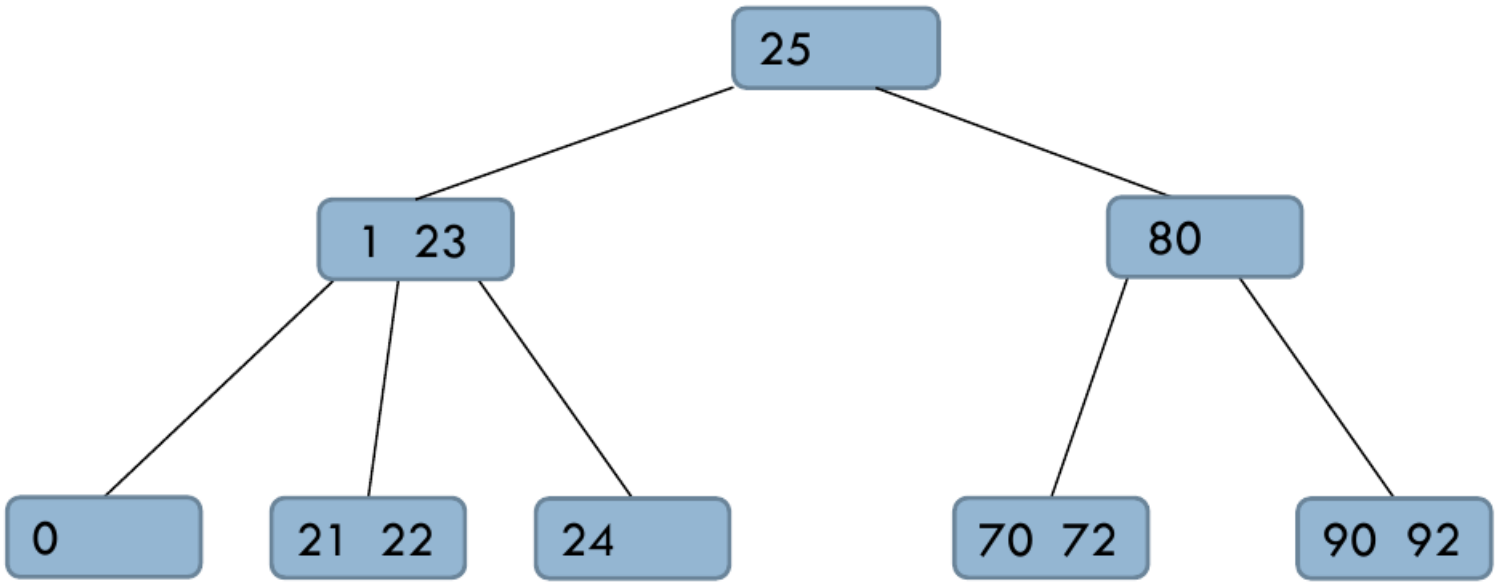


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Pesquisa em uma Árvore 2-3

Pesquisando pela chave 72 na Árvore:

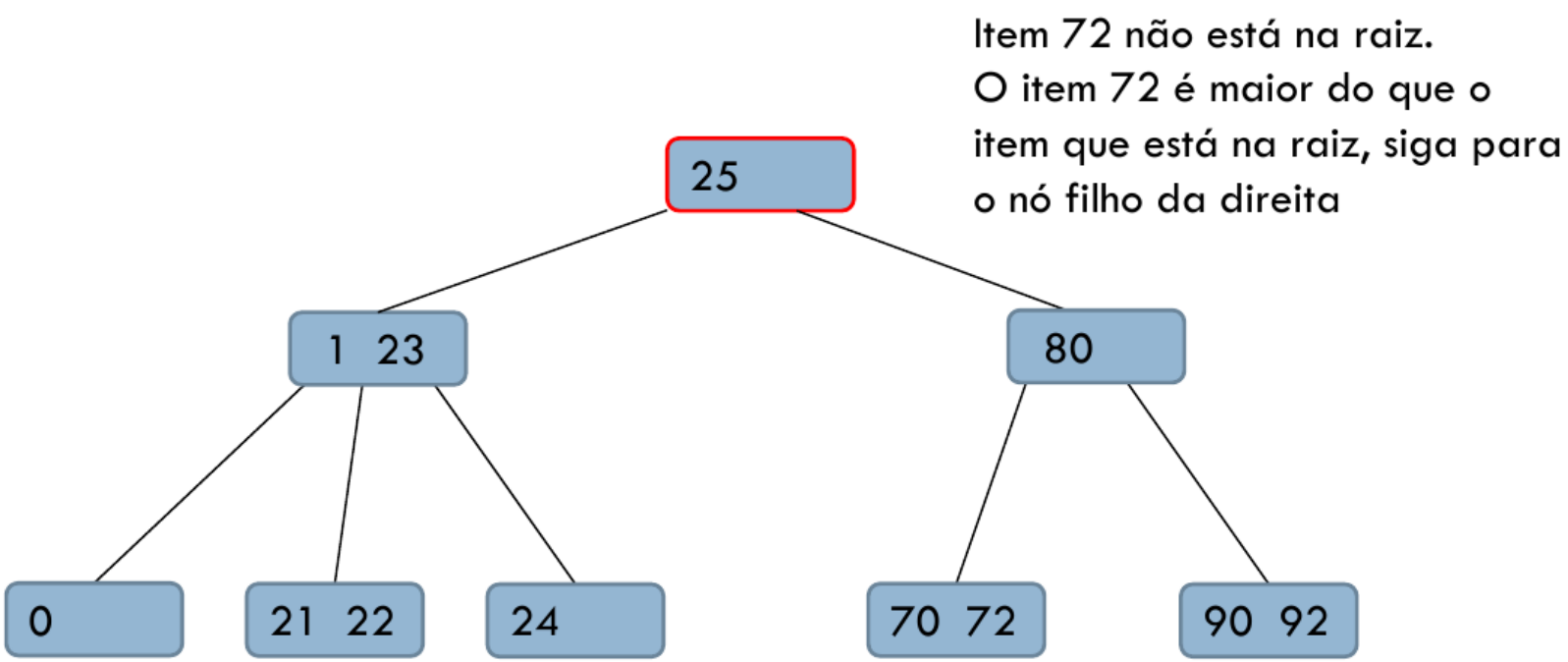


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Pesquisa em uma Árvore 2-3

Pesquisando pela chave 72 na Árvore:

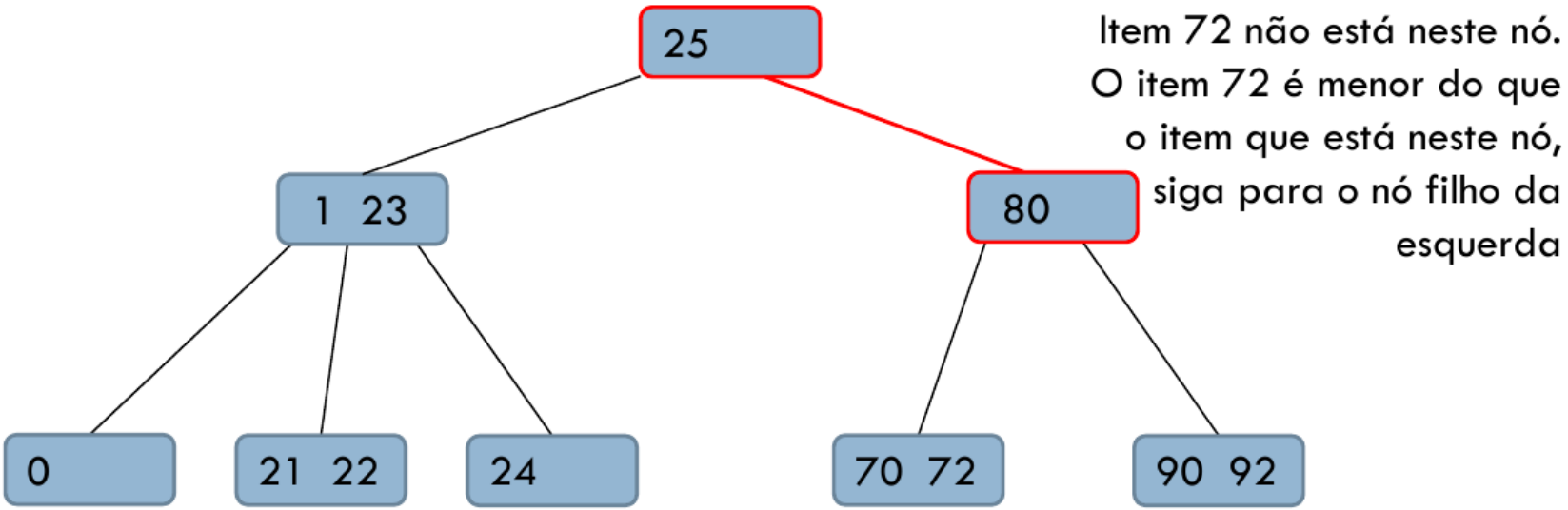


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Pesquisa em uma Árvore 2-3

Pesquisando pela chave 72 na Árvore:

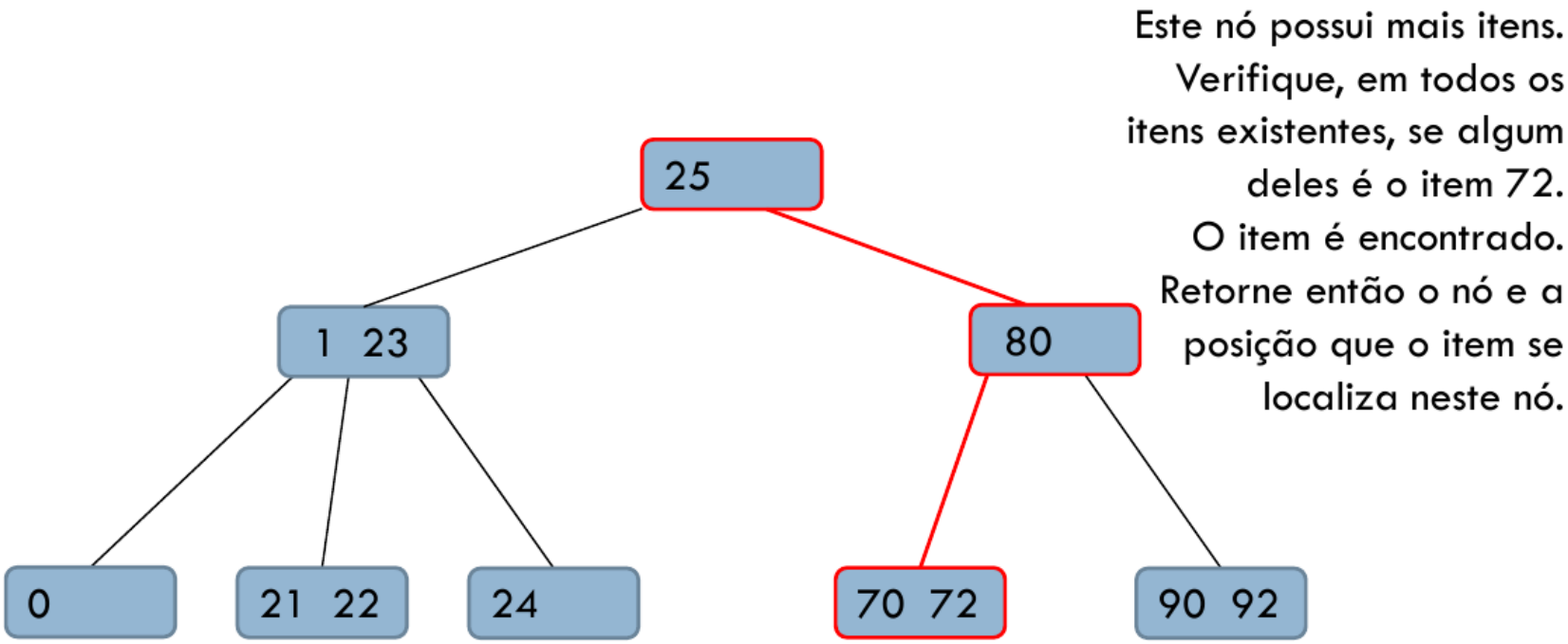


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Sumário

1 Introdução

2 Pesquisa

3 Inserção

4 Remoção

5 Códigos

6 Aplicações

7 Questionário

8 Referências Bibliográficas

Inserção em uma Árvore 2-3

Continuando os procedimentos, vamos à *Inserção* em uma Árvore 2-3.

Inserção em uma Árvore 2-3

Inserindo a chave 50 na Árvore:

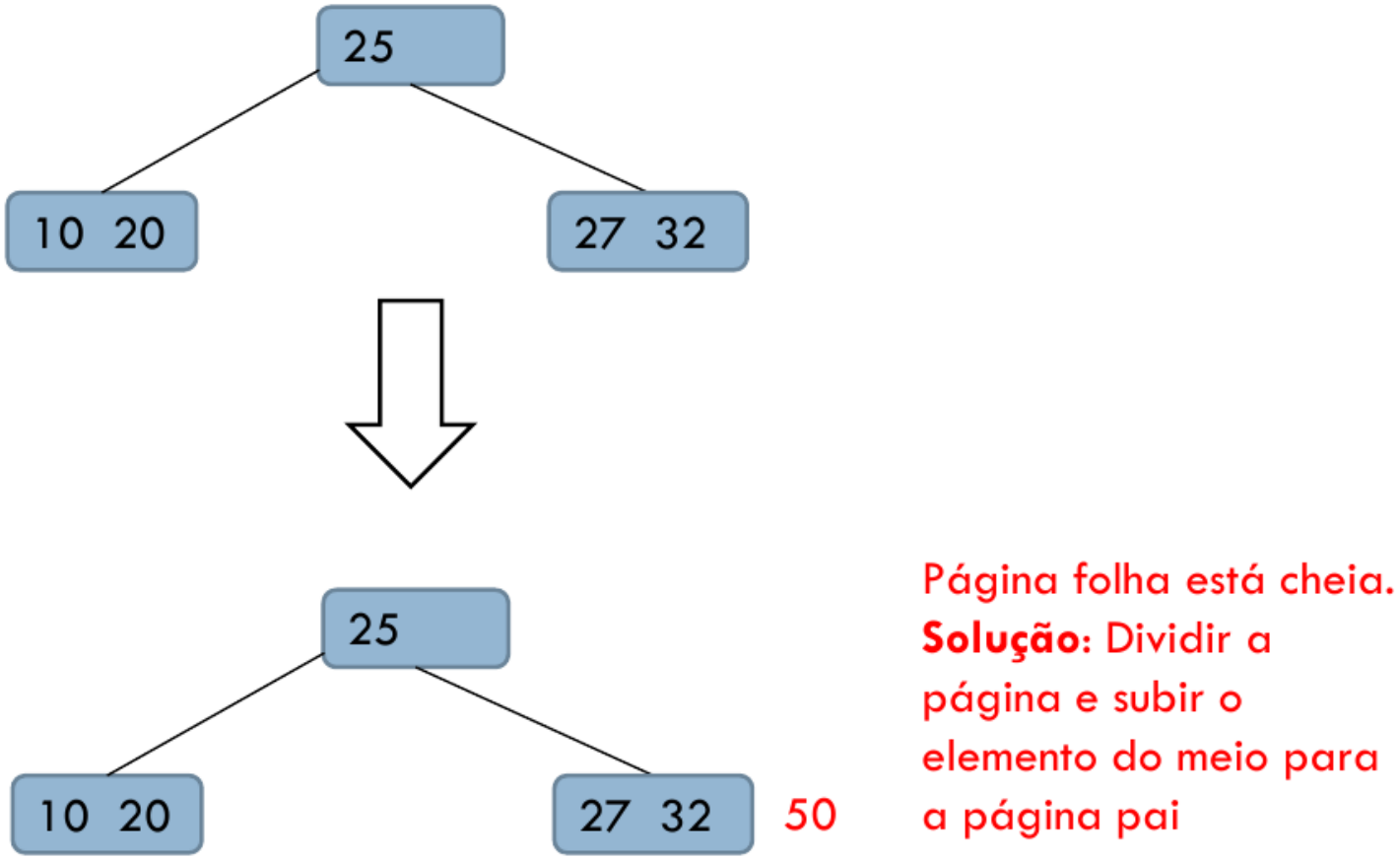


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Inserção em uma Árvore 2-3

Inserindo a chave 50 na Árvore:

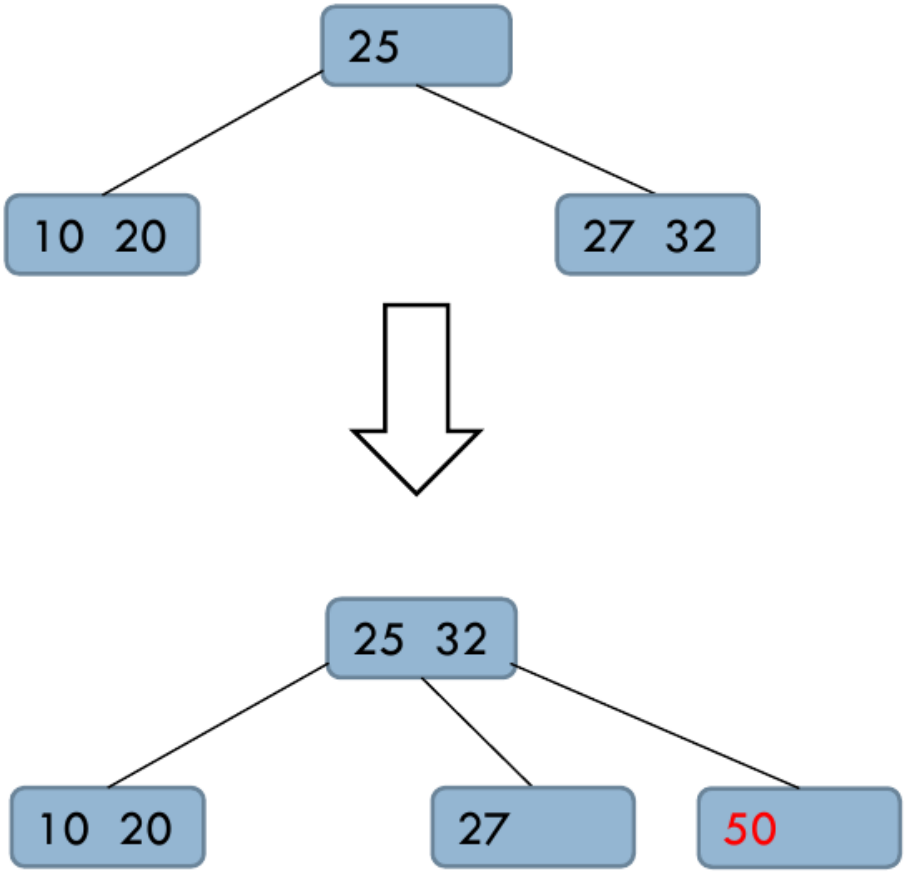


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Inserção em uma Árvore 2-3

Inserindo a chave 5 na Árvore:

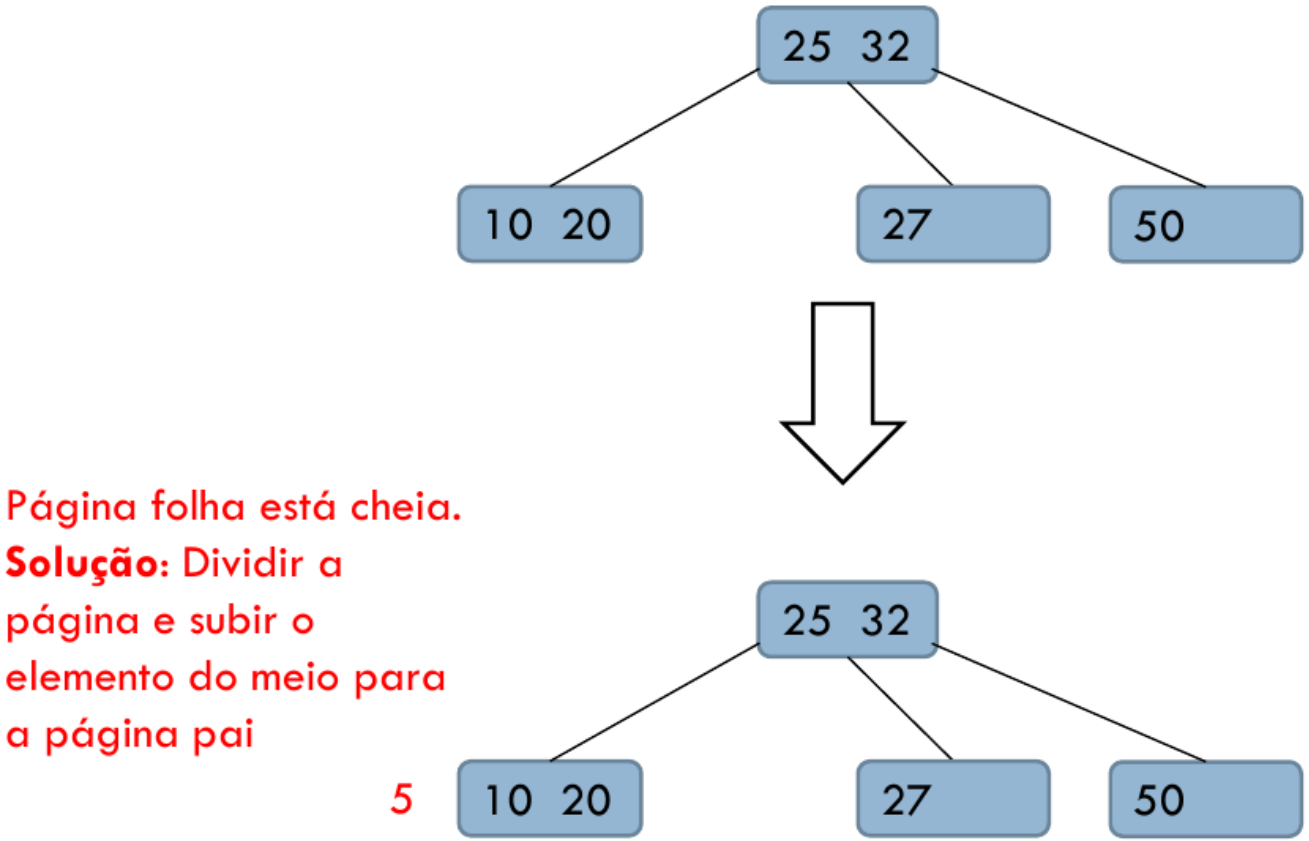


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Inserção em uma Árvore 2-3

Inserindo a chave 5 na Árvore:

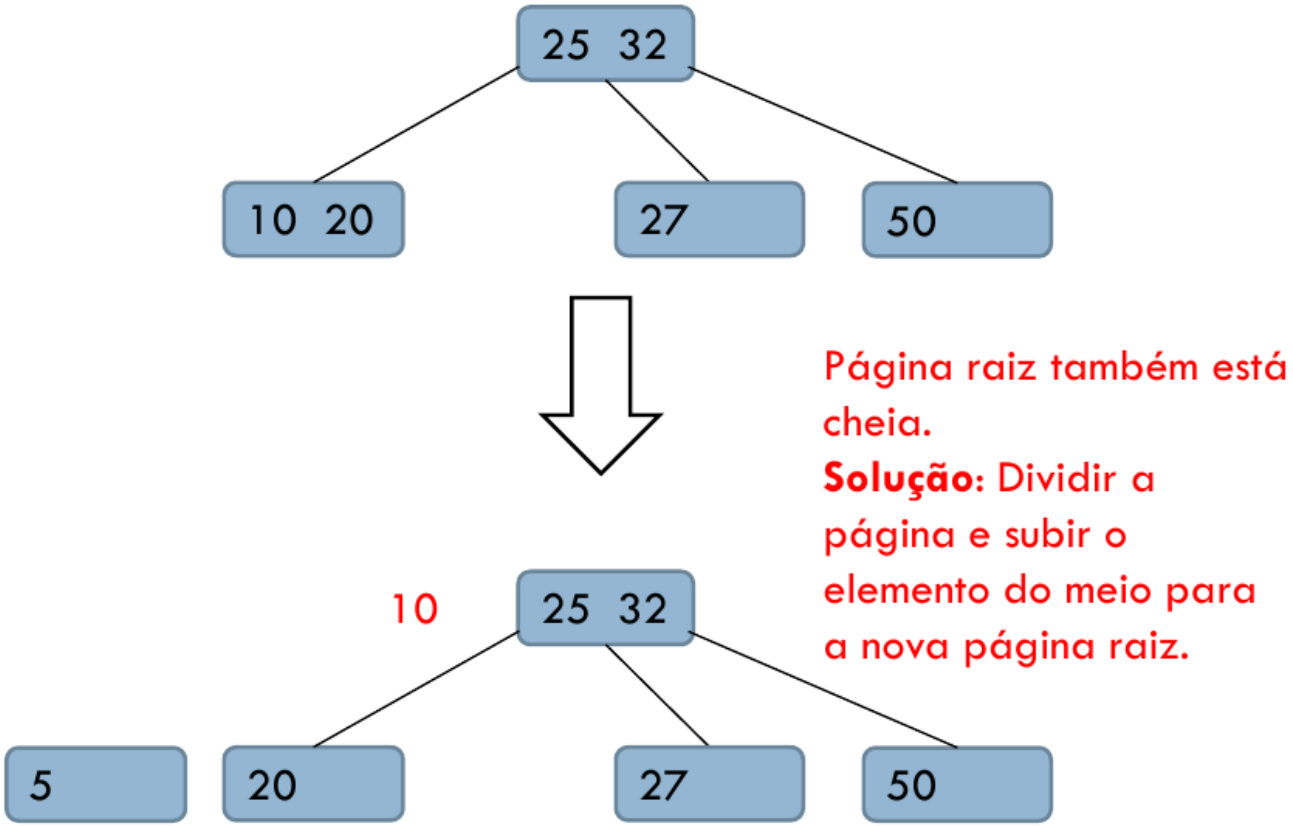


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Inserção em uma Árvore 2-3

Inserindo a chave 5 na Árvore:

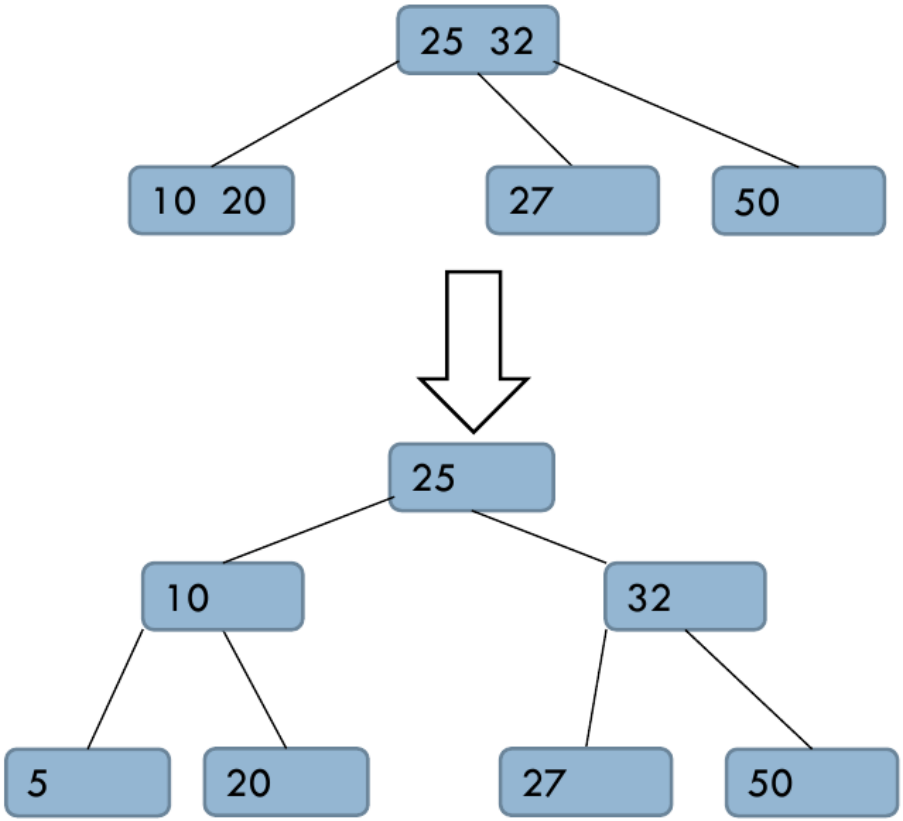


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Sumário

1 Introdução

2 Pesquisa

3 Inserção

4 Remoção

5 Códigos

6 Aplicações

7 Questionário

8 Referências Bibliográficas

Remoção em uma Árvore 2-3

Hora de tratarmos do mais complexo dos procedimentos relacionados à uma *Árvore 2-3*, a Remoção.

Remoção em uma Árvore 2-3

Começaremos pelo mais simples:
Exemplo 1:

- Remover item 73.

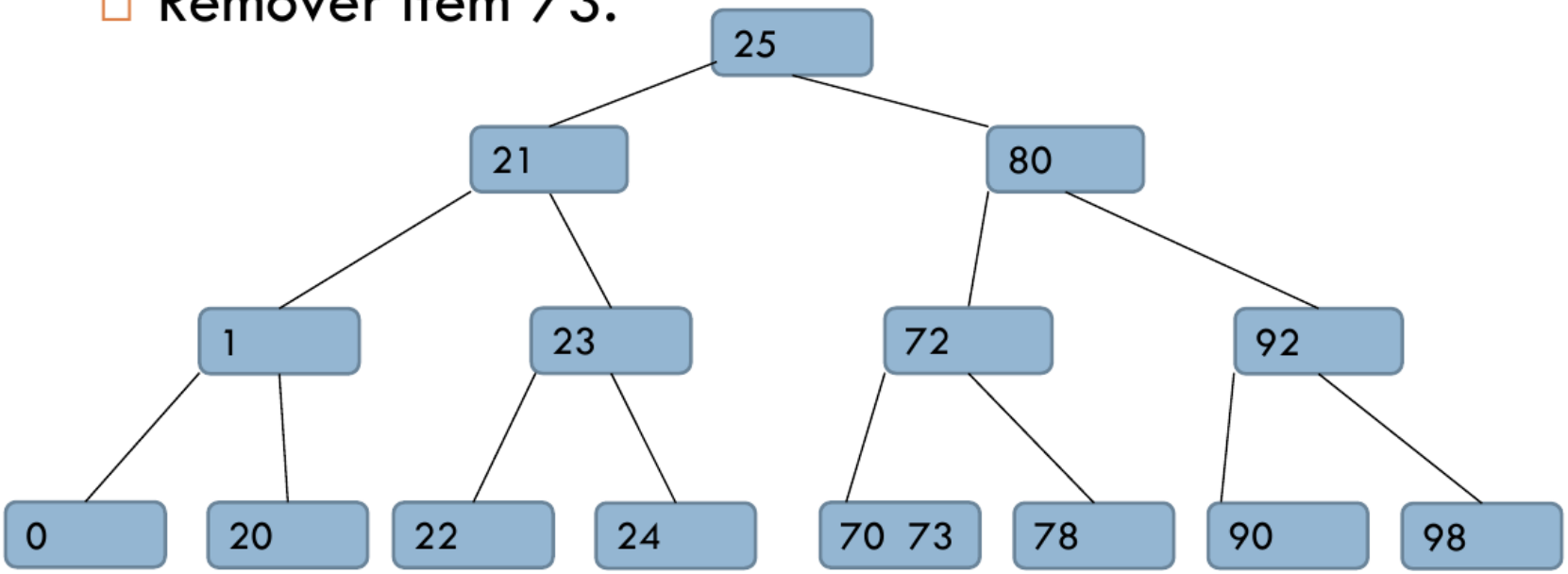


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

- Remover item 73.

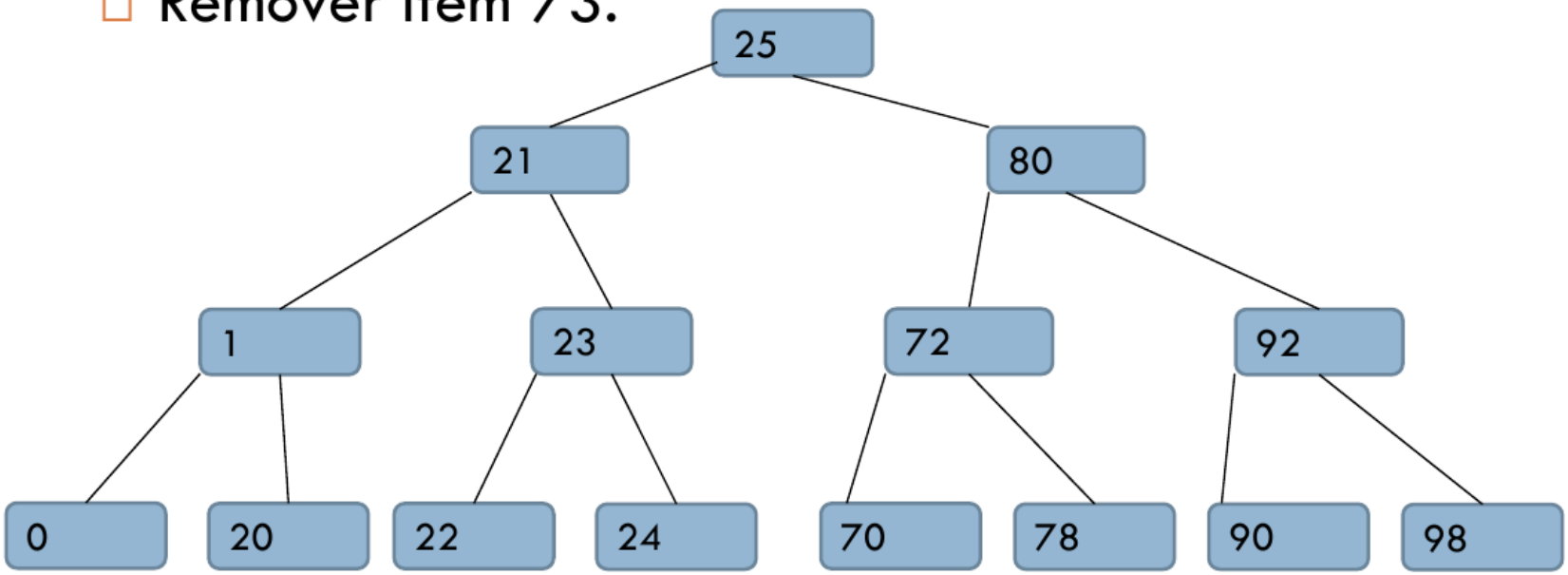


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

Exemplo 2:

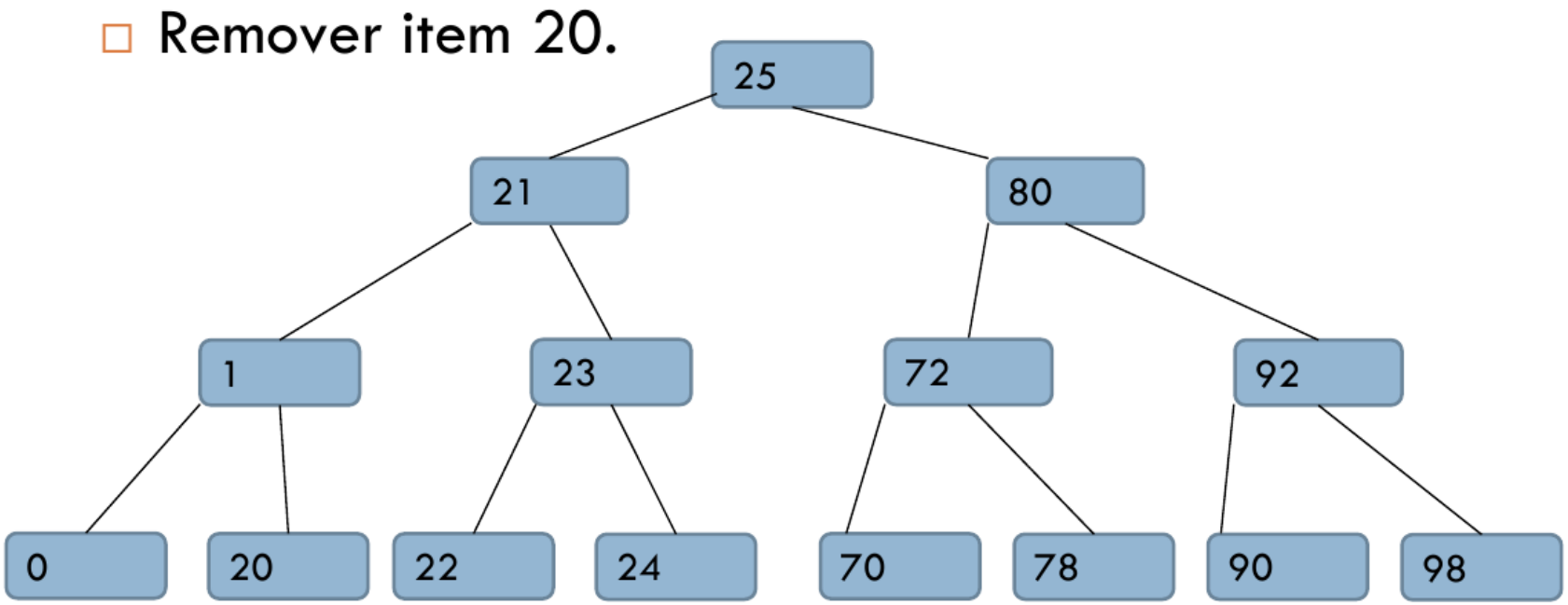


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

Problema!

Houston, we have a problem.
Repare que não podemos remover diretamente o 20, temos que reorganizar a Árvore para manter suas propriedades válidas.
No caso, a propriedade referida é a de um nó ter 2 ou três filhos, não é permitido apenas um filho.

Remoção em uma Árvore 2-3

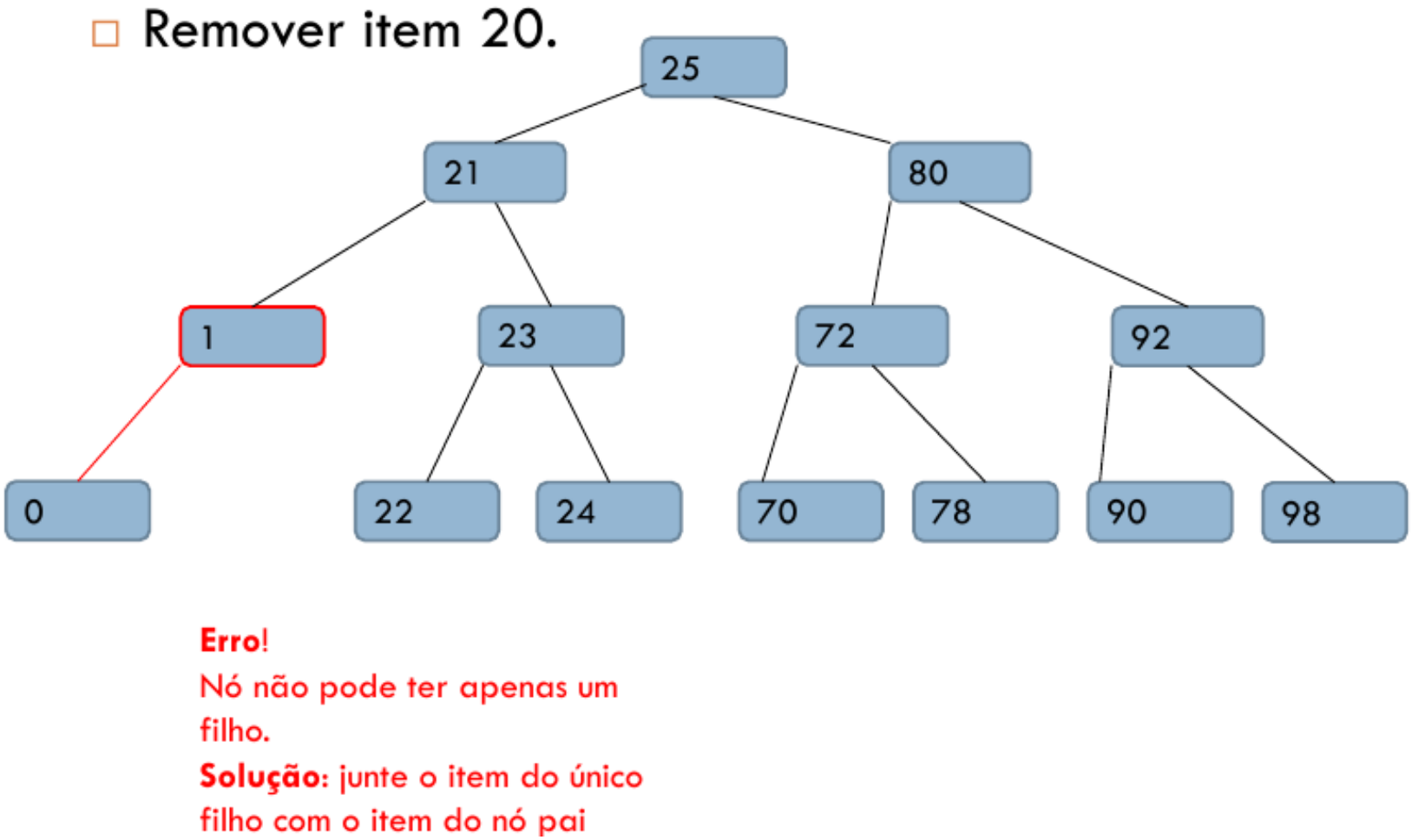


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

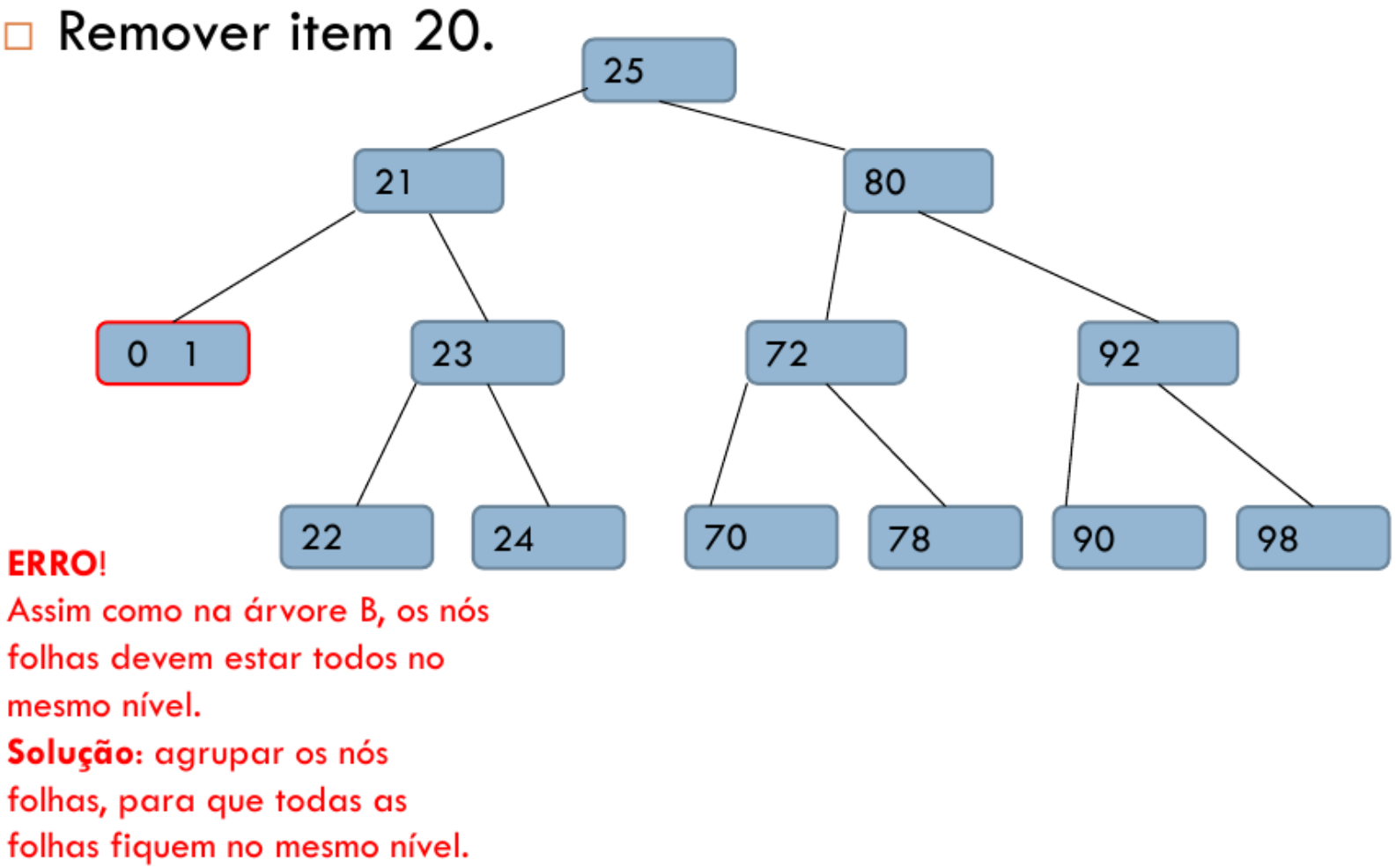


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

- Remover item 20.

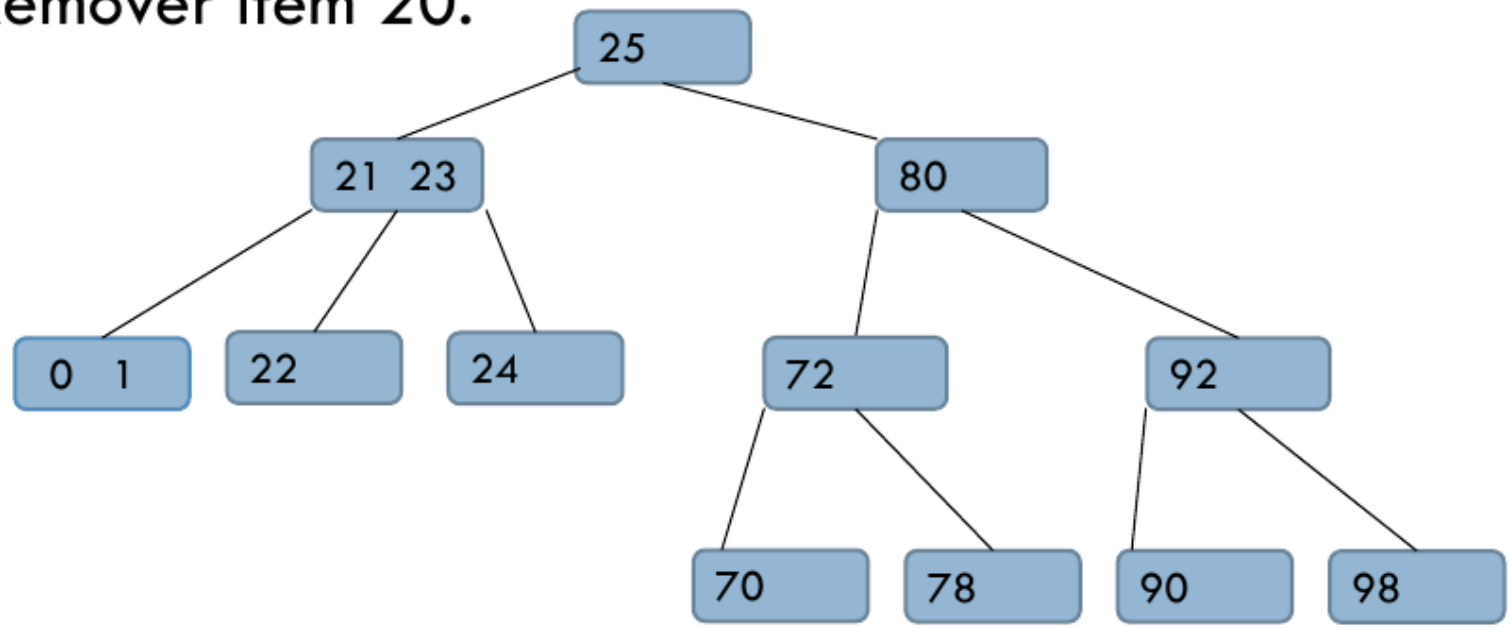
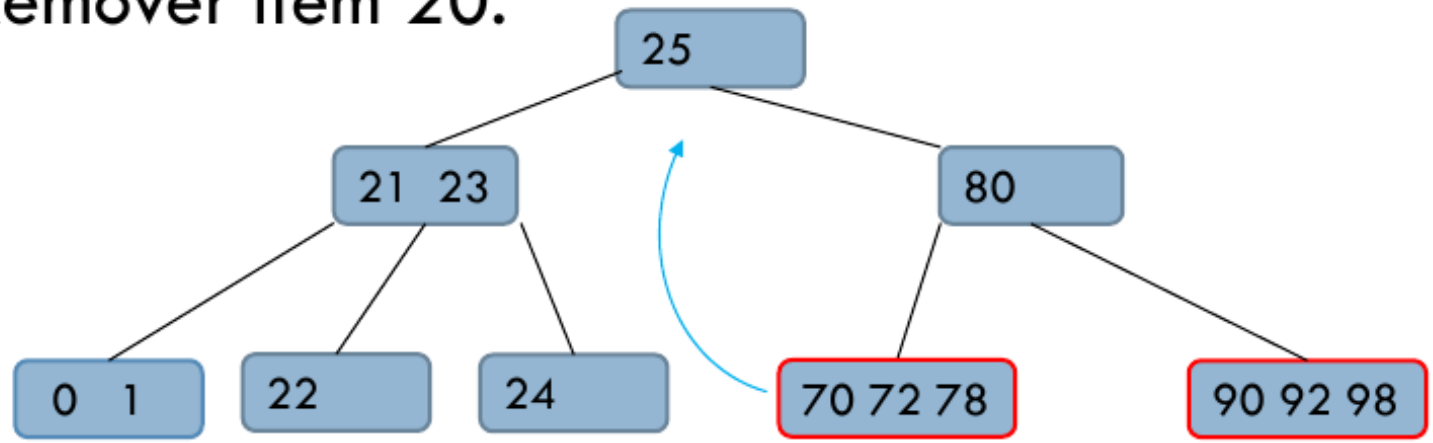


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

- Remover item 20.

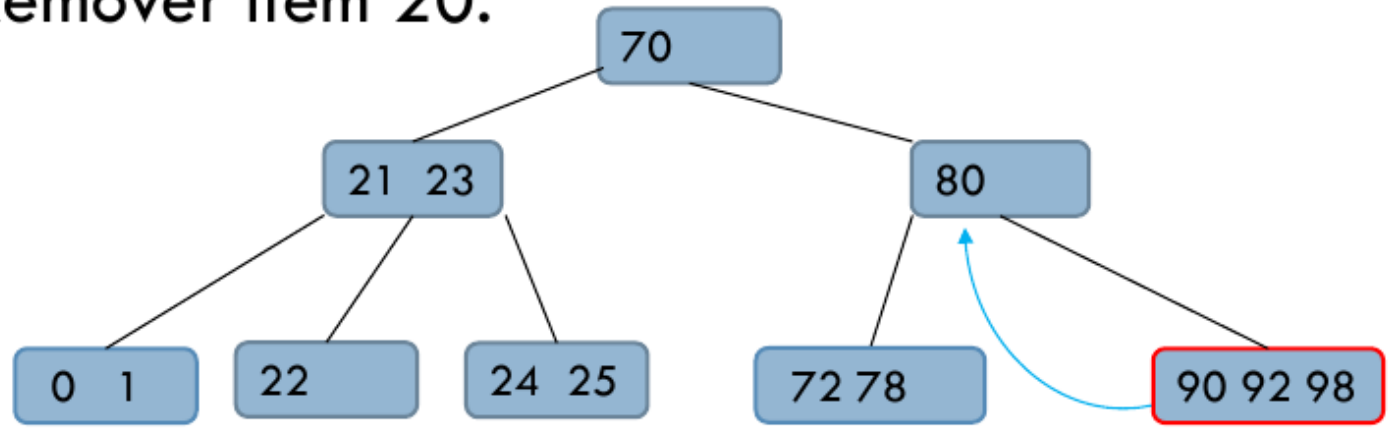


ERRO!
Os nós podem ter no máximo 2 itens de dados
Solução: levar o menor elemento da subárvore da direita para a raiz, e descer o item da raiz para a subárvore da esquerda

Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

- Remover item 20.

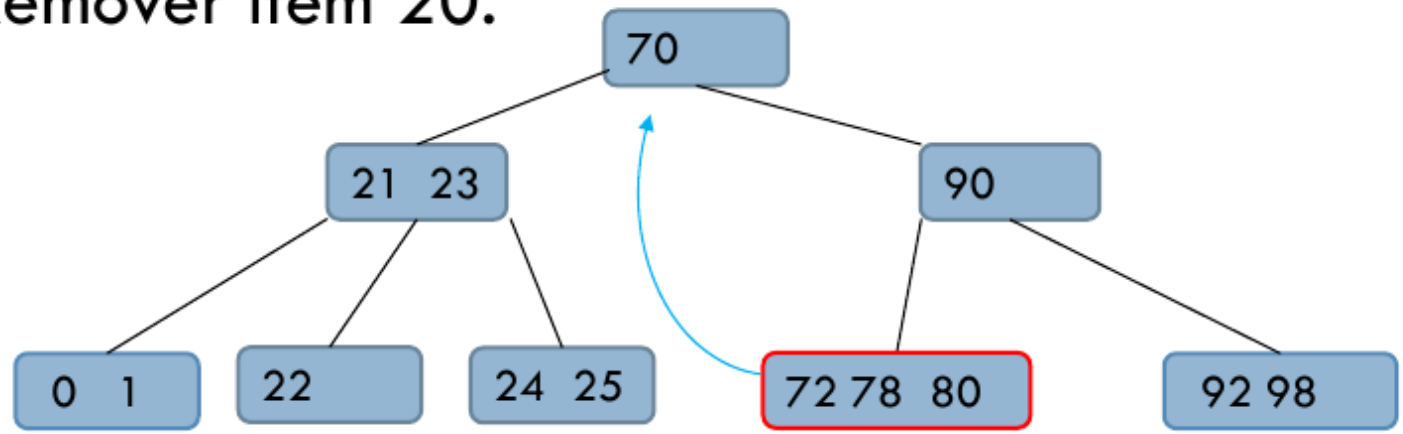


ERRO!
Os nós podem ter no máximo 2 itens de dados.
Solução: levar o menor elemento da subárvore da direita para a raiz, e descer o item da raiz para a subárvore da esquerda.

Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

- Remover item 20.

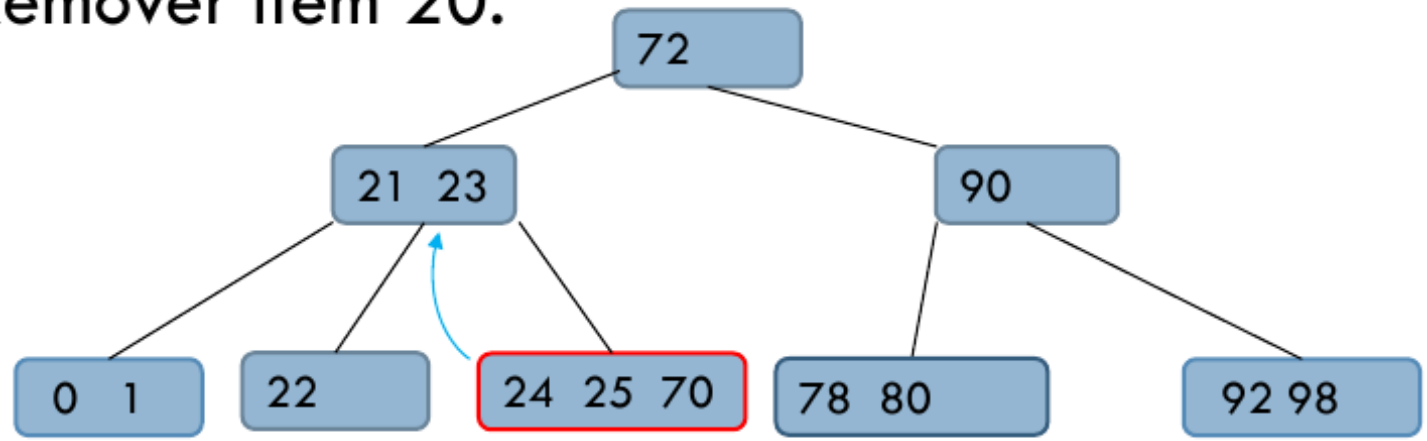


ERRO!
Os nós podem ter no máximo 2 itens de dados.
Solução: levar o menor elemento da subárvore da direita para a raiz, e descer o item da raiz para a subárvore da esquerda.

Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

- Remover item 20.



ERRO!
Os nós podem ter no máximo 2 itens de dados.
Solução: levar o menor elemento da subárvore da direita para a raiz, e descer o item da raiz para a subárvore da esquerda.

Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

- Remover item 20.

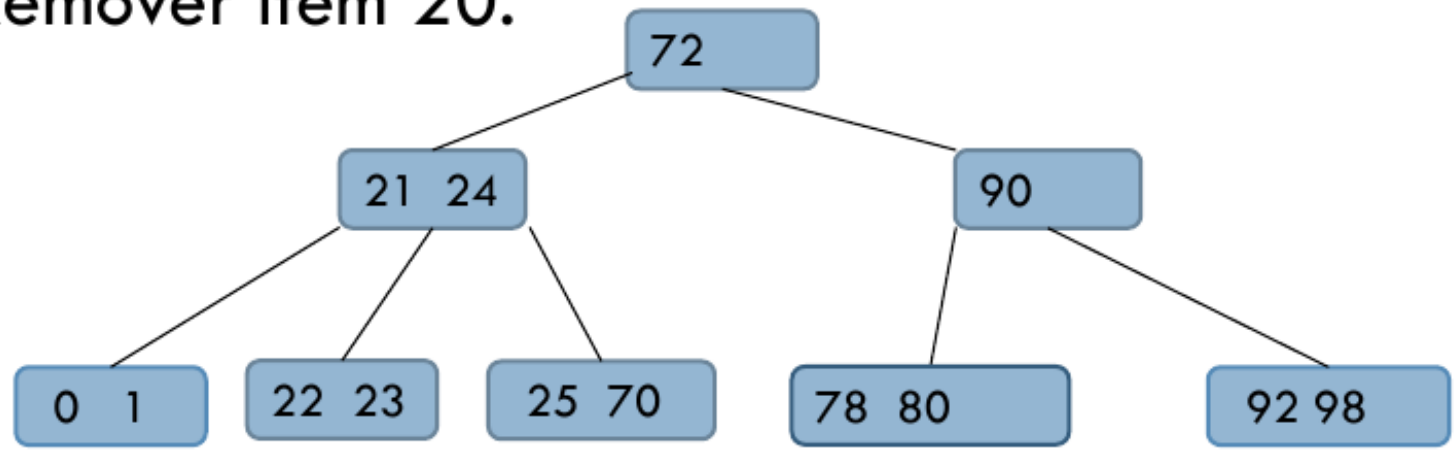


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

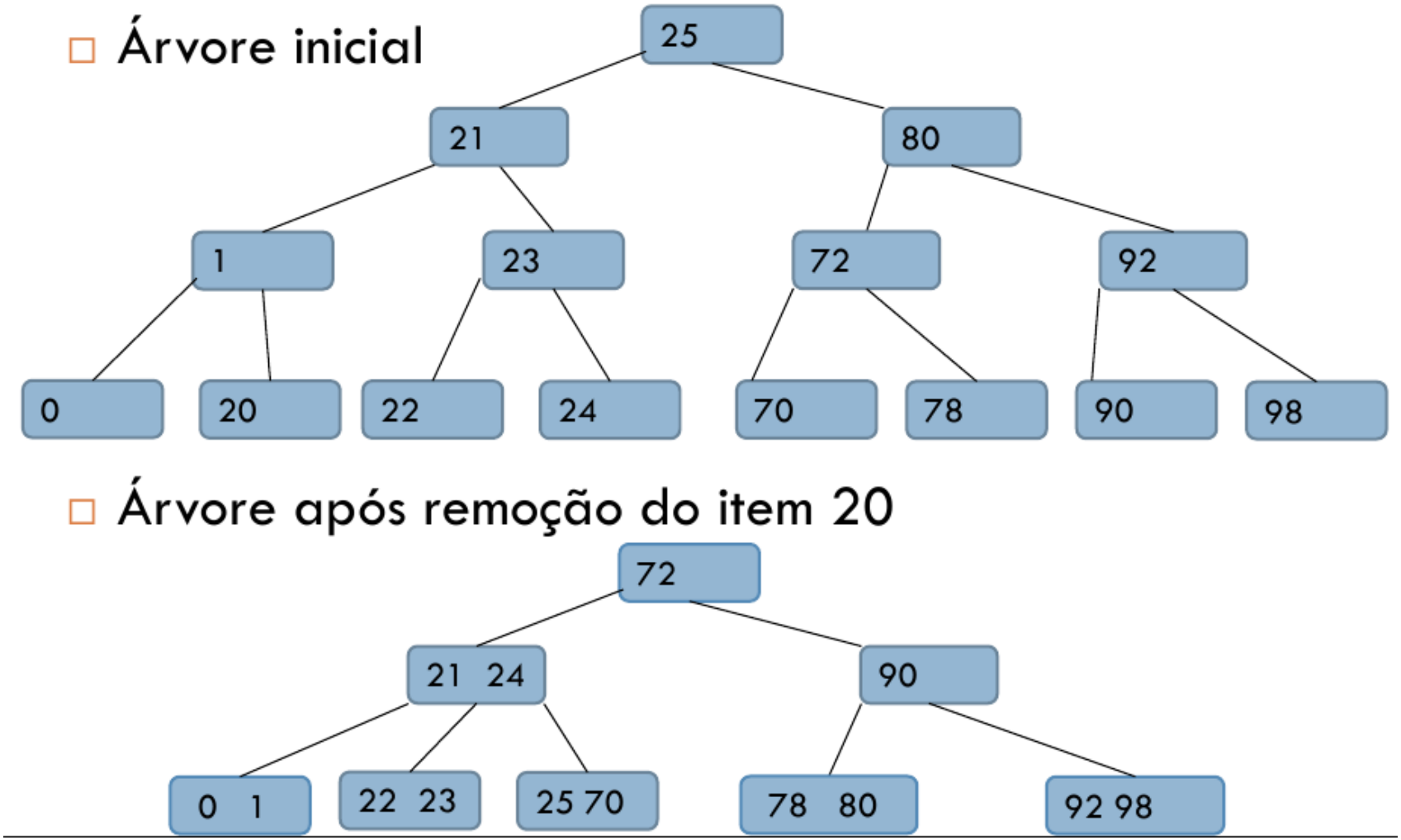


Figura: Imagem retirada de [2] (vide *Referências Bibliográficas*)

Remoção em uma Árvore 2-3

Entenderam o porquê do *complexo* agora, ou seja, tenham plena certeza de suas inserções ou sofrerão com sua remoção!

Sumário

1 Introdução

2 Pesquisa

3 Inserção

4 Remoção

5 Códigos

6 Aplicações

7 Questionário

8 Referências Bibliográficas

Programas

Agora que já vimos como funciona e os princípios básicos das Árvores 2-3, daremos uma olhada nas `estruturas` e `algoritmos` para a manipulação e armazenamento das supracitadas Árvores 2-3.

1. Estrutura básica (Struct);
2. Alocação da Árvore 2-3.

Código

Estrutura de uma Árvore 2-3

```
1  typedef struct _no23 {
2    int lkey, // chave esquerda
3        rkey, // chave direita
4        nkeys; // numero de chaves
5    struct _no23 *left, // ponteiro ao filho esquerdo
6                 *center, // ponteiro ao filho central
7                 *right; // ponteiro ao filho direito
8  } no23;
```

Código

Alocação de uma Árvore 2-3

```
1   23tree *23tree_alloc() {
2     23tree *t;
3     23tree_node *r;
4     t = malloc(sizeof(23tree));
5     t->n = 0;
6     t->min_item = NULL;
7     t->stack = malloc(STACK_SIZE * sizeof(23tree_node *));
8     r = t->root = malloc(sizeof(23tree_node));
9     r->key1 = r->key2 = NULL;
10    r->link_kind = LEAF;
11    r->left.item = r->middle.item = r->right.item;
12
13    return t;
14  }
```

Código

Busca em uma Árvore 2-3

```
1   no23 *find(no23 *raiz, int key) {
2     if(raiz==NULL)
3       return NULL; // nao encontrou
4     if(key == raiz->lkey)
5       return raiz; // retorna chave esquerda
6     if((raiz->nkeys == 2) && (key == raiz->rkey))
7       return raiz; // retorna a chave direita
8     if(key < raiz->lkey)
9       return find(raiz->left, key);
10    else if(raiz->nkeys == 1)
11      return find(raiz->center, key);
12    else if (key < raiz->rkey)
13      return find(raiz->center, key);
14    else
15      return find(raiz->right, key);
16  }
```

Quebra Nó

```
1   no23 *quebraNo(no23 *no, int val, int *rval, no23 *subarvore){
2     no23 *paux;
3     if (val > no->rkey) { // val esta mais a direita
4       *rval = no->rkey; // promove a antiga maior
5       paux = no->right;
6       no->right = NULL; // elimina o terceiro filho
7       no->nkeys = 1; // atualiza o numero de chaves
8       return criaNo(val, 0, 1, paux, subarvore, NULL);
9     }else if(val >= no->lkey){ // val esta no meio
10      *rval = val; // continua sendo promovido
11      paux = no->right;
12      no->right = NULL;
13      no->nkeys = 1;
14      return criaNo(no->rkey, 0, 1, subarvore, paux, NULL);
15    }else{ // val esta a mais a esquerda
16      *rval = no->lkey; // primeiro cria o noh a direita
17      paux = criaNo(no->rkey, 0, 1, no->center, no->right, NULL);
18      no->lkey = val; // em seguida arruma o noh a esquerda
19      no->nkeys = 1;
20      no->right = NULL;
21      no->center = subarvore;
22      return paux;
23    }
24  }
```

Inserção

No âmbito das Árvores 2-3, daremos uma olhada na inserção.

- Assemelha-se a inserção em uma Árvore 2-3 à inserção em uma Árvore binária de busca.
- Estrutura básica (Struct);
- Alocação da Árvore 2-3.

Sumário

1 Introdução

2 Pesquisa

3 Inserção

4 Remoção

5 Códigos

6 Aplicações

7 Questionário

8 Referências Bibliográficas

Aplicações

Objetivo

- Árvores 2-3 foram inventadas para garantir um bom desempenho no pior caso.
- Se você estiver satisfeito com um bom desempenho médio, basta usar um Árvore Binária de Busca

Aplicações

Altura

Considere uma árvore 2-3 com N nós.

- Se temos apenas nós simples, a altura da árvore será lg N
- Se temos apenas nós duplos, a altura da árvore será $\log_3 n$ que é igual a [0.63 lg N]

Conclusão: A altura nunca passa de lg N.

Aplicações

Proposição

Em uma árvore 2-3 com N nós, busca e inserção nunca visitam mais que lg N nós - mesmo no pior caso.

- Cada visita faz no máximo 2 comparações de chaves.

Aplicações

Observação

Numa árvore 2-3, alguns nós envolvem uma comparação entre chaves enquanto outros envolvem três comparações.

- Assim, o número de comparações de chaves é no máximo o dobro do número de nós visitados

Complexidade

Time complexity in big O notation

Algorithm	Average	Worst Case
Space	O(n)	O(n)
Search	O(log n)	O(log n)
Insert	O(log n)	O(log n)
Delete	O(log n)	O(log n)

Figura: Imagem retirada de [1] (vide *Referências Bibliográficas*)

Sumário

1 Introdução

2 Pesquisa

3 Inserção

4 Remoção

5 Códigos

6 Aplicações

7 Questionário

8 Referências Bibliográficas

Questionário

Deve haver, ao final, uma seção dedicada à apresentação de um questionário com pelo menos 05 (cinco) questões, e suas respectivas respostas esperadas.

Cada questão deverá ser um `bloco`, o mesmo ocorrendo com a resposta esperada.

Veja o exemplo....

Questionário

Questão 01

Enunciado da primeira questão, se necessário com figuras, tabelas, e tudo mais.

Questionário

Questão 01 – Resposta Esperada

Bloco com a resposta esperada para a primeira questão.

Sumário

1 Introdução

2 Pesquisa

3 Inserção

4 Remoção

5 Códigos

6 Aplicações

7 Questionário

8 Referências Bibliográficas

Referências Bibliográficas

1. https://pt.wikipedia.org/wiki/%C3%81rvore_2-3
2. https://www.passeidireto.com/disciplina/algoritmos-e-estrutura-de-dados-iii?type=6&materialid=1012744
3. https://www.ime.usp.br/~gold/cursos/2002/mac2301/aulas/b-arvore/b-arvore.html
4. COOPER, K. and TORCZON, L. – 2011 : *Chapter 2 – Scanners*;
5. GRUNE, *et al.* – 2012 : *Chapter 2 – Program Text to Tokens – Lexical Analysis*.

Referências Bibliográficas

Se desejar, pode encerrar com uma imagem, uma citação, etc.

Figura: by Mark Kostabi