

Árvore 2-3

Luigi Wagner
Rafael Alessandro
Rafael Falcão

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

14 de novembro de 2017

Sumário

- 1 Introdução
- 2 Códigos
- 3 Terceiro Tópico
- 4 Questionário
- 5 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

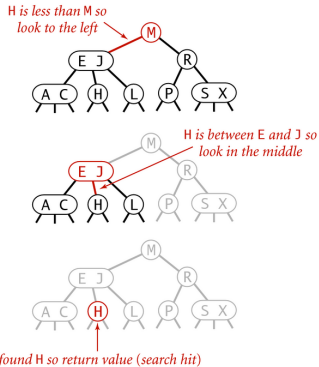
As principais propriedades de uma Árvore 2-3 são:

- 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;
- Cada nó folha tem 1 ou 2 campos.

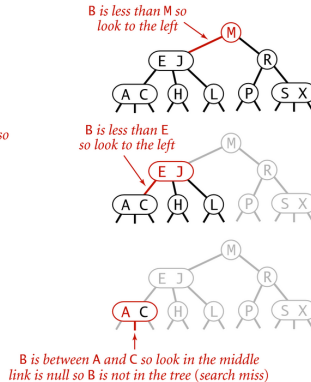
Como, por exemplo:

Introdução

successful search for H



unsuccessful search for B



Search hit (left) and search miss (right) in a 2-3 tree

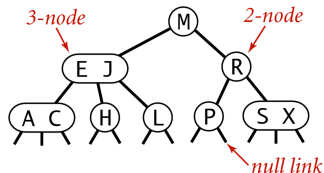
Figura: <https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/figuressw/Chapter3/TTsearch.png>

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 Códigos
- 3 Terceiro Tópico
- 4 Questionário
- 5 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 _23tree_link {  
2      struct 23tree_node *node; //internal node  
3      int item;  
4  } 23tree_link;
```

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 Códigos
- 3 Terceiro Tópico**
- 4 Questionário
- 5 Referências Bibliográficas

Terceiro Tópico

Tabelas

Qualquer recursos disponível no LaTeX e seus pacotes complementares podem ser utilizados, desde que ao final a equipe gere um arquivo compactado (extensão .zip) contendo toda a “*pasta*” com os *slides* desenvolvidos.

A pasta gerada também deverá conter um arquivo .pdf com os *slides* utilizados.

Terceiro Tópico

Exemplo 01

Exemplos devem utilizar um bloco para serem apresentados, como o que está acontecendo neste momento, ou seja, um bloco está sendo utilizado e seu título é Exemplo 01.

Você também pode utilizar [cores](#) para dar destaque no texto.

Sumário

- 1 Introdução
- 2 Códigos
- 3 Terceiro Tópico
- 4 Questionário**
- 5 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 Códigos
- 3 Terceiro Tópico
- 4 Questionário
- 5 Referências Bibliográficas

Referências Bibliográficas

- https://pt.wikipedia.org/wiki/%C3%81rvore_2-3
- COOPER, K. and TORCZON, L. – 2011 : *Chapter 2 – Scanners*;
- 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