Projeto e Análise de Algoritmos

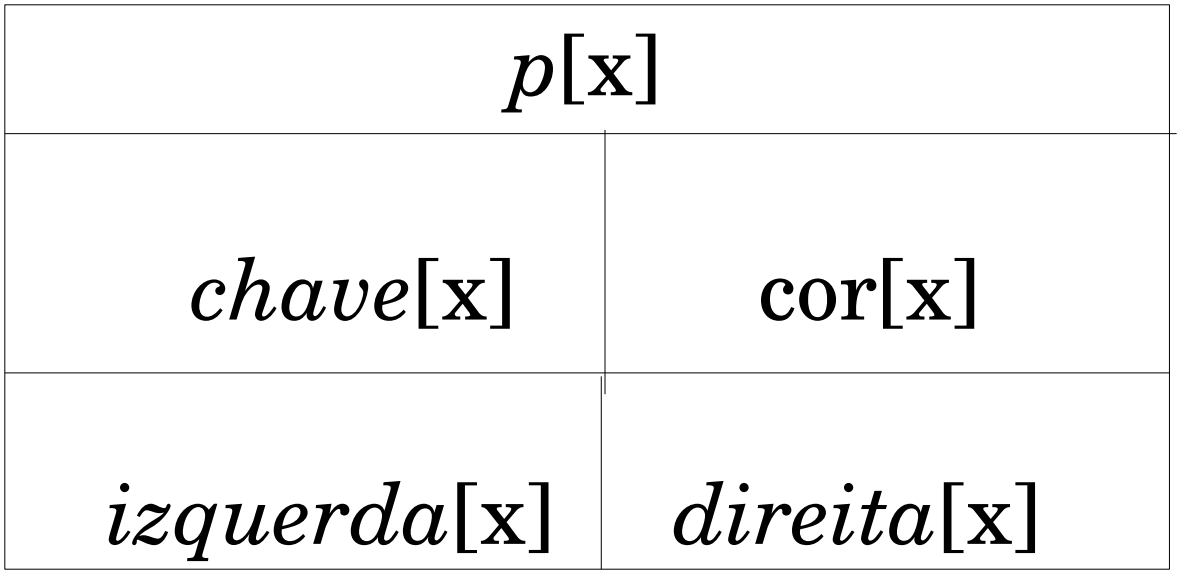
Árvores vermelho-preto
Aula 13 & 14

Diane Castonguay
diane@inf.ufg.br

Instituto de Informática
Universidade Federal de Goiás

Árvores vermelho-preto

objeto x



Árvores vermelho-preto: Propriedades

- É um árvore de pesquisa binária
- Todo nó é vermelho ou preto
- A raiz é preta
- Toda folha (NIL) é preta
- Se um nó é vermelho, então ambos os seus filhos são pretos
- Para cada nó, todos os caminhos até as folhas descendentes contêm o mesmo número de nós pretos.

Algoritmos de árvores de pesquisa binária

Sem modificação

INORDER-TREE-WALK(x)
TREE-SEARCH(x, k)
ITERATIVE-TREE-SEARCH(x, k)
TREE-MINIMUM(x)
TREE-MAXIMUM(x)
TREE-SUCESSOR(x)

Algoritmos de árvores de pesquisa binária

A modificar

TREE-INSERT(T, z)
TREE-DELETE(T, z)

Porque complicar as coisas?

Os algortimos de árvores de pesquisa binária sõ todos O(h) onde h é a altura da árvore.

A altura de um árvore de pesquisa binária varia entre lg n e n. O pior caso dos algoritmos é O(n)!

A altura de um árvore vermelho-preto varia entre lg n e 2 lg(n+1). O pior caso dos algoritmos é O(lg n).

(esperando que as modificações no demoram mais)

Representação de árvores vermelho-preto

objeto *nil*[T] é pai de *raiz*[T]

ultimo pai	
NIL	PRETO
raiz[T]	*raiz*[T]

Inserção - 1

RB-INSERT(T, z)

1. $y \leftarrow nil$[T]
2. $x \leftarrow raiz$[T]
3. enquanto $x \neq nil$[T] faça
4. $\quad y \leftarrow x$
5. $\quad$ se $chave$[z] < $chave$[x]
6. $\quad\quad$ então $x \leftarrow esquerda$[x]
7. $\quad\quad$ senão $x \leftarrow direita$[x]
8. $\quad$ fim-se
9. fim-enquanto
10. p[z] $\leftarrow y$
11. se $y = nil$[T]
12. $\quad$ então $raiz$[T] $\leftarrow$ z

Inserção - 2

RB-INSERT(T, z) (continação)

13. senão se *chave*[z] < *chave*[*x*]
14. então *esquerda*[*y*] ← z
15. senão *direita*[*y*] ← z
16. fim-se
17. fim-se
18. *esquerda*[z] ← *nil*[T]
19. *direita*[z] ← *nil*[T]
20. *cor*[z] ← VERMELHO
21. RB-INSERT-FIXUP(T, z)

Eliminação - 1

RB-DELETE(T, z)

1. se *esquerda*[z] = *nil*[T] ou *direita*[z] = *nil*[T]
2. então $y \leftarrow$ z
3. senão $y \leftarrow$ TREE-SUCESSOR(z)
4. fim-se
5. se *esquerda*[y] $\neq$ *nil*[T]
6. então $x \leftarrow$ *esquerda*[y]
7. senão $x \leftarrow$ *direita*[y]
8. fim-se
9. // não verificamos se $x \neq$ *nil*[T] para fazer a 10.
10. $p[x] \leftarrow p[y]$
11. se $p[y]$ = *nil*[T]
12. então *raiz*[T] $\leftarrow x$

Eliminação - 1

RB-DELETE(T, z) (continuação)

```
13. senão se y = esquerda[p[y]]
14.                 então esquerda[p[y]] ← x
15.                 senão direita[p[y]] ← x
16.             fim-se
17. fim-se
18. se y ≠ z
19. então chave[z] ← chave[y]
20.             copiar dados satélite de y em z
21. fim-se
22. se cor[y] = PRETO
23.    então RB-DELETE-FIXUP(T, x)
24. fim-se
25. retorna y
```

Rotação - 1

/*
 * *Assume que direita*[x] ≠ *nil*[T] e *p*[*raiz*[T]] = *nil*[T]
 */

LEFT-ROTATE(T, x)
1. $y \leftarrow direita[x]$
2. $direita[x] \leftarrow esquerda[y]$
3. $p[esquerda[y]] \leftarrow x$
4. $p[y] \leftarrow p[x]$

Rotação - 2

LEFT-ROTATE(T, x) (continuação)

5. se p[x] = *nil*[T]
6. então *raiz*[T] ← y
7. senão se x = *esquerda*[p[x]]
8. então *esquerda*[p[x]] ← y
9. senão *direita*[p[x]] ← y
10. fim-se
11. fim-se
12. *esquerda*[y] ← x
13. p[x] ← y

Inserção – Assertar as cores! - 1

```
RB-INSERT-FIXUP(T, z)
1. enquanto cor[p[z]] = VERMELHO faça
2.       se p[z] = esquerda[p[p[z]]]
3.            então y ← direita[p[p[z]]]
4.                 se cor[y] = VERMELHO
5.                      então cor[p[z]] ← PRETO              //c.1
6.                            cor[y] ← PRETO                 //c.1
7.                            cor[p[p[z]]] ← VERMELHO        //c.1
8.                            z ← p[p[z]]]                   //c.1
```

Inserção – Assertar as cores! - 2

RB-INSERT-FIXUP(T, z) (continuação)

```
9.      senão   se z = direita[p[z]]
10.                 então   z ← p[z]                    //c.2
11.                         LEFT-ROTATE(T, z)           //c.2
12.                 fim-se
13.                 cor[p[z]] ← PRETO                   //c.3
14.                 cor[p[p[z]]] ← VERMELHO             //c.3
15.                 RIGHT-ROTATE(T, p[p[z]])            //c.3
16. fim-se
```

Inserção – Assertar as cores! - 3

RB-INSERT-FIXUP(T, z) (continuação)

17. senão *(igual a cláusula então*
18. *trocando "direita" com "esquerda")*
19. fim-se
20. fim-enquanto
21. *cor*[*raiz*[T]] ← PRETO

Eliminação – Assertar as cores! 1

```
RB-DELETE-FIXUP(T, x)
1. enquanto x ≠ raiz[T]  e cor[x] = PRETO faça
2.      se x = esquerda[p[x]]
3.          então w ← direita[p[x]]
4.              se cor[w] = VERMELHO
5.                  então   cor[w] ← PRETO              //c.1
6.                          cor[p[x]] ← VERMELHO        //c.1
7.                          LEFT-ROTATE(T, p[x])        //c.1
8.                          w ← direita[p[x]]           //c.1
9.              fim-se
```

Eliminação – Assertar as cores! 2

```
10. se cor[esquerda[w]] = PRETO e cor[direita[w]] = PRETO
11. então cor[w] ← VERMELHO                                  //c.2
12.         x ← p[x]                                         //c.2
13. senão se cor[direita[w]] = PRETO
14.             então cor[esquerda[w]] ← PRETO               //c.3
15.                   cor[w] ← VERMELHO                      //c.3
16.                   RIGHT-ROTATE(T, w)                     //c.3
17.                   w ← direita[p[x]]                      //c.3
18.         fim-se
19.         cor[w] ← cor[p[x]]                               //c.4
20.         cor[p[x]] ← PRETO                                //c.4
21.         cor[direita[w]] ← PRETO                          //c.4
22.         LEFT-ROTATE(T, p[x])                             //c.4
23.         x ← raiz[T]
24.     fim-se
```

Eliminação – Assertar as cores! 3

RB-DELETE-FIXUP(T, x) (continuação)

25. senão *(igual a cláusula então*
26. *trocando "direita" com "esquerda")*
27. fim-se
28. fim-enquanto
29. *cor*[x] ← PRETO