

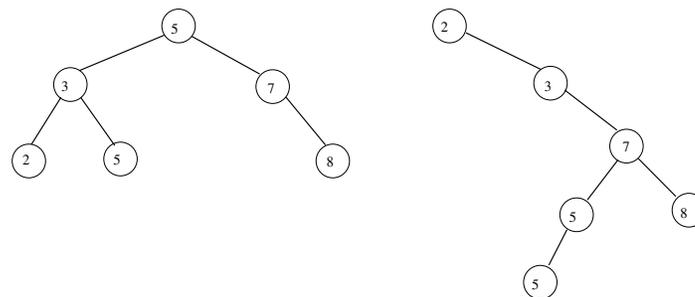
Aula 13

Árvores de Busca Balanceadas

Prof. Marco Aurélio Stefanos
marco em dct.ufms.br
www.dct.ufms.br/~marco

Árvore Binária de Busca

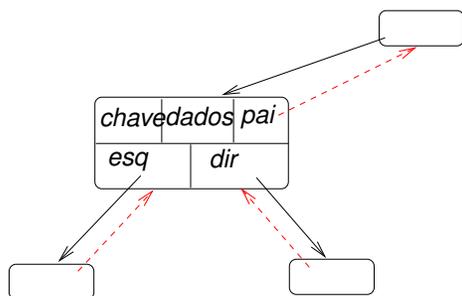
- Propriedades
 - Se y está na subárvore esquerda de x então $chave(y) \leq chave(x)$
 - Se y está na subárvore direita de x então $chave(y) \geq chave(x)$



Árvore Binária de Busca

Objetivo: Implementar operações de Busca, Inclusão, Remoção, Mínimo/Máximo e Predecessor/Sucessor em um conjunto de dados

- Representação da árvore binária Estrutura Ligada



Operações Busca e Sucessor

$Busca(T, k)$

Comece na raiz e compare k com a chave do nó corrente x

1. Se a chave é encontrada pare
2. Se a $k < chave(x)$ busque na subárvore esquerda de x
3. Se a $k > chave(x)$ busque na subárvore direita de x

$Sucessor(x)$: maior elemento menor que $chave(x)$

- Caso 1 $dir(x) \neq NULO$
sucessor é o mínimo na subárvore direita de x
- Caso 2 $dir(x) = NULO$
Sucessor é primeiro nó à direita subindo pelos ancestrais de x

Operações na Árvore Binária de Busca

Complexidade das operações

- Busca $O(h)$
- Inclusão $O(h)$
- Remoção $O(h)$
- Mínimo $O(h)$
- Máximo $O(h)$
- Predecessor $O(h)$
- Sucessor $O(h)$

Uma Árvore Binária de Busca é **Balanceda** se $h = O(\lg n)$

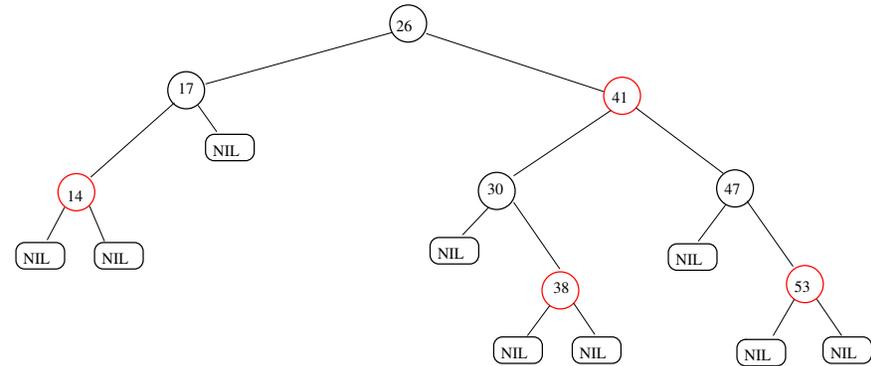
Árvore Rubro-Negra

- Variação da árvore binária de busca (balanceada)
- Cada nó possui um campo adicional: a cor do nó, que pode ser **rubro** ou **negro**
- Operações tomam $O(\lg n)$ no pior caso

Propiedades de Árvores Rubro-Negras

1. Cada nó é rubro ou negro
2. A raiz é negra
3. Cada folha (NIL) é negra
4. Se um nó é rubro seus dois filhos são negros
5. Todo caminho de um nó até uma folha contém o mesmo número de nós negros

Exemplo de um Árvore RN



NIL[T]: Sentinela para todas as folhas e pai da raiz

Lema

Def.: a altura negra de um nó x , $h_n(x)$ é o número de nós negros de um caminho até uma folha descendente incluindo-se x .

Afirmção Uma subárvore com raiz em um nó x tem pelo menos $2^{h_n(x)} - 1$ nós internos

Prova: Indução em $h(x)$

Base: $h(x) = 0 \Rightarrow x$ é folha $\Rightarrow h_n(x) = 0$ e $2^0 - 1 = 0$ nós internos na subárvore

PI: $h(x) > 0$. A altura negra de cada filho de x é $h_n(x)$ ou $h_n(x) - 1$, mas a altura de cada um de seus filhos é menor que $h(x)$. Portanto usando a HI, o número de nós internos na subárvore com raiz x é pelo menos

$$2 \cdot (2^{h_n(x)-1} - 1) + 1 = 2^{h_n(x)} - 1.$$

Lema

Lema Qualquer árvore RN com n nós internos tem altura $\leq 2\lg(n+1)$.

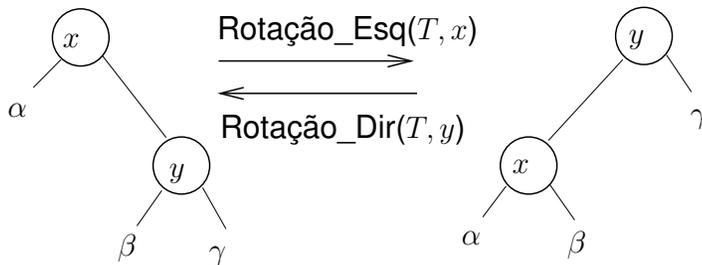
Prova: Seja h a altura da árvore RN. Pela Prop. 4 a altura negra da árvore é pelo menos $h/2$. Portanto, pela Afirmação temos $n \geq 2^{h/2} - 1 \Rightarrow h = 2\lg(n+1) - 1$

- Portanto a árvore RN é balanceada
- Como executar as operações de Inclusão e Remoção sem alterar as propriedades?

Aula 13 - p. 9

Rotações

Há dois tipos de rotação para reestruturar a árvore:
Esquerda e Direita
Hipótese para uma rotação esquerda em um nó x
Pai da raiz é NIL
Filho direito y de x não é NIL



Aula 13 - p. 10

Rotação_Esq(T, x)

```
1:  $y = Dir[x]$ 
2:  $Dir[x] = Esq[y]$ 
3: if  $Esq[y] \neq NIL$  then
4:    $p[Esq[y]] = x$ 
5:  $p[y] = p[x]$ 
6: if  $p[x] = NIL$  then
7:    $root[T] = y$ 
8: else
9:   if  $x = Esq[p[y]]$  then
10:     $Esq[p[y]] = y$ 
11:  else
12:     $Dir[p[y]] = y$ 
13:  $Esq[y] = x$ 
14:  $p[x] = y$ 
```

Aula 13 - p. 11

Inclusão na Árvore Rubro-Negra

Idéia

1. Insira um nó z como em uma árvore binária de busca normal
2. Pinte o nó de vermelho
3. Restaure as propriedades rubro-negras

Devemos subir na árvore até restaurar as propriedades.
Olharemos para dois nós, em especial:

Notação:
 x nó rubro
 y tio de x

Aula 13 - p. 12

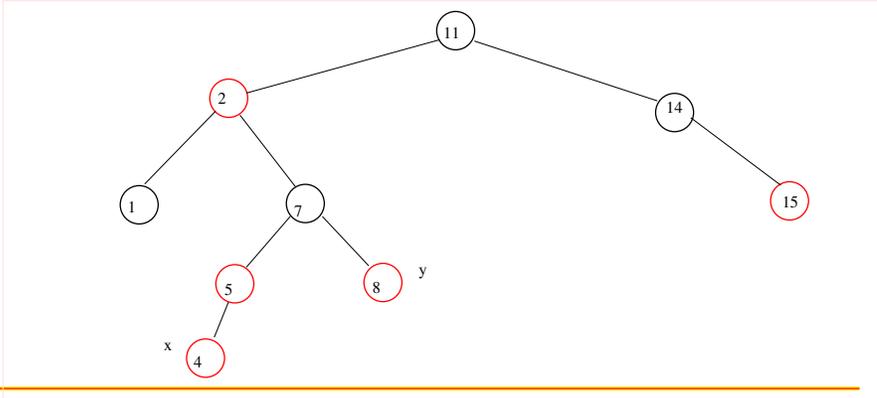
Inclusão na Árvore Rubro-Negra

Caso 1: Se y é rubro

$cor[y]=\text{negro}$

$cor[p[x]]=\text{negro}$

$cor[p[p[x]]]=\text{rubro}$



Aula 13 – p. 13

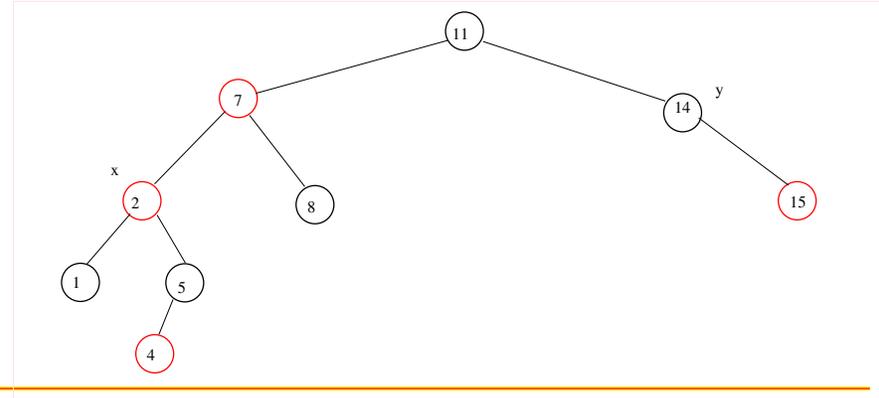
Inclusão na Árvore Rubro-Negra

Caso 3: y é negro e x é filho esquerdo

$cor[p[x]] = \text{negro}$

$cor[p[p[x]]] = \text{rubro}$

Rotação_Dir($T, p[p[x]]$)



Aula 13 – p. 13

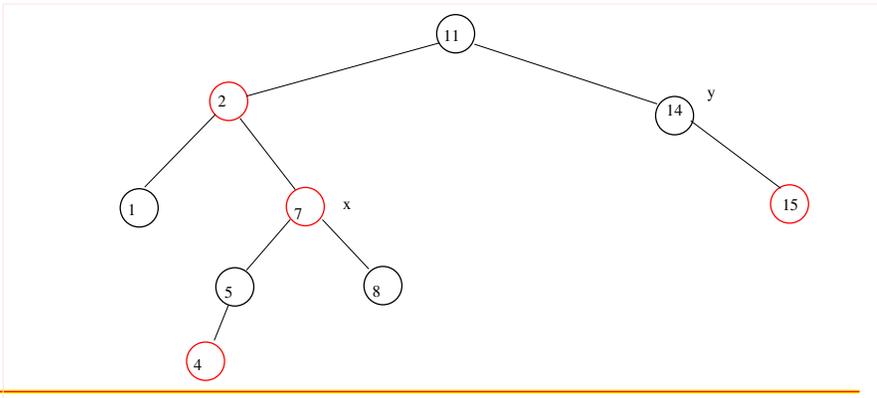
Inclusão na Árvore Rubro-Negra

Caso 2: Se y é negro

x é filho direito

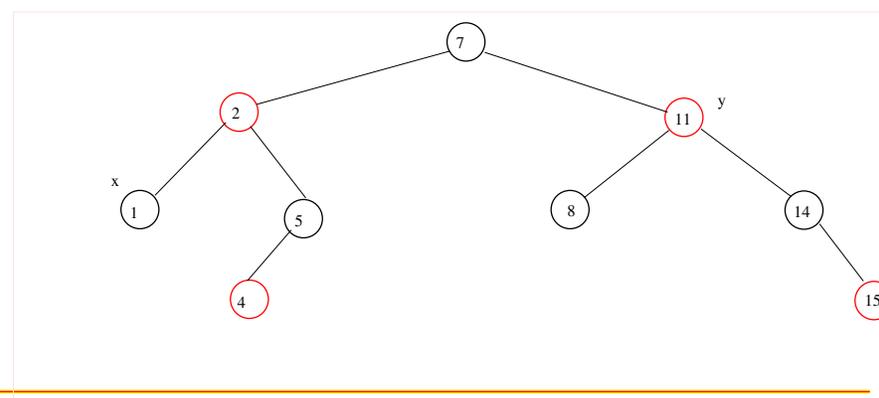
$x = p[x]$

Rotação_Esq(T, x)



Aula 13 – p. 13

Inclusão na Árvore Rubro-Negra



Aula 13 – p. 13

Inserer_RN(T, x)

```
1: Inserir( $T, x$ )
2:  $cor[x] = \text{rubro}$ 
3: while  $x \neq \text{root}[T]$  e  $cor[p[x]] = \text{rubro}$  do
4:   if  $p[x] = \text{Esq}[p[p[x]]]$  then
5:      $y = \text{Dir}[p[p[x]]]$ 
6:     if  $cor[y] = \text{rubro}$  then
7:        $cor[p[x]] = \text{negro}$   $\triangleright$  Caso 1
8:        $cor[y] = \text{negro}$ 
9:        $cor[p[p[x]]] = \text{rubro}$ 
10:       $x = p[p[x]]$ 
11:    else
12:      if  $x = \text{Dir}[p[p[x]]]$  then
13:         $x = p[p[x]]$   $\triangleright$  Caso 2
14:        Rotacao_Esq( $T, x$ )
15:         $cor[p[x]] = \text{negro}$   $q$   $\triangleright$  Caso 3
16:         $cor[p[p[x]]] = \text{rubro}$ 
17:        Rotacao_Dir( $T, p[p[x]]$ )
18:      else
19:        simétrico ao anterior
20:  $cor[\text{root}[T]] = \text{negro}$ 
```

Aula 13 – p. 14

Análise da Inserção

- Inserir um novo nó na árvore: $O(\lg n)$
- Restaurar propriedades
 - O laço se repete se somente o Caso 1 é executado
 - Caso 2 é seguido sempre do Caso 3
 - Caso 3 resolve o problema
 - O número de vezes que o laço é executado é $O(\lg n)$
- Tempo total $O(\lg n)$

Aula 13 – p. 15

Remoção na Árvore Rubro-Negra

Qual a cor do nó a ser deletado? **rubro**?

1. Todo nó é rubro ou negro. Ok!
2. A raiz é negra. Ok!
3. Toda folha é negra. Ok!
4. Se um nó é rubro, seus dois filhos são negros
 - Não criamos dois nós rubros seguidos. Ok!
5. Para cada nó, todo caminho descendente de uma folha tem o mesmo número de nós negros.
 - Não mudamos nenhuma altura negra. Ok!

Aula 13 – p. 16

Remoção na Árvore Rubro-Negra

Qual a cor do nó a ser deletado? **negro**?

1. Todo nó é rubro ou negro. Ok!
2. A raiz é negra.
 - Se removermos a raiz seu substituto pode ser rubro
3. Toda folha é negra. Ok!
4. Se um nó é rubro, seus dois filhos são negros
 - Podemos criar dois nós rubros seguidos.
5. Para cada nó, todo caminho descendente de uma folha tem o mesmo número de nós negros.
 - Podemos mudar alguma altura negra.

Aula 13 – p. 17

Remoção na Árvore Rubro-Negra

1. Qual a cor do nó a z ser deletado? **negro**
2. Troque z pelo seu sucessor y
3. Remove y

Idéia:

1. Passar a cor de y para seu filho x . Viola a Prop. 1, pois x tem duas cores
2. Mover negro extra para cima
 - até encontrar um rubro ou até atingir a raiz
 - fazer rotações quando necessárias
 - x sempre aponta para nó duplo negro

Complexidade: $O(\lg n)$
