Algoritmos e Estrutura de Dados II

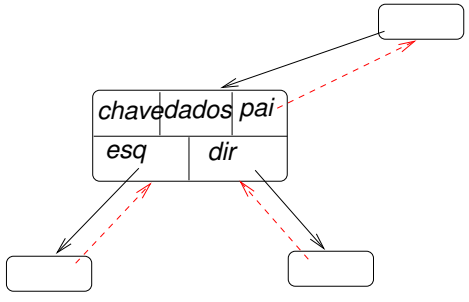
Árvores Rubro-Negras

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

Árvore Binária de Busca

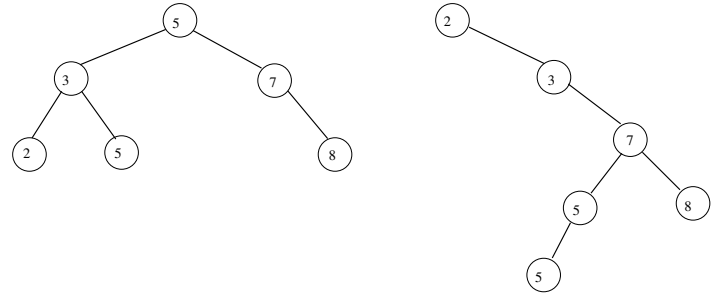
Objetivo: Implementar operações de Busca, Inclusão, Remoção, Mínimo/Máximo e Predecessor/Sucessor em um conjunto de dados

- Representação da árvore binária
  Estrutura Ligada



Árvore Binária de Busca

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



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

- Caso 1 $dir(x) \neq$ NULO
  sucessor é o mínino 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 Balanceada se $h = O(\lg n)$

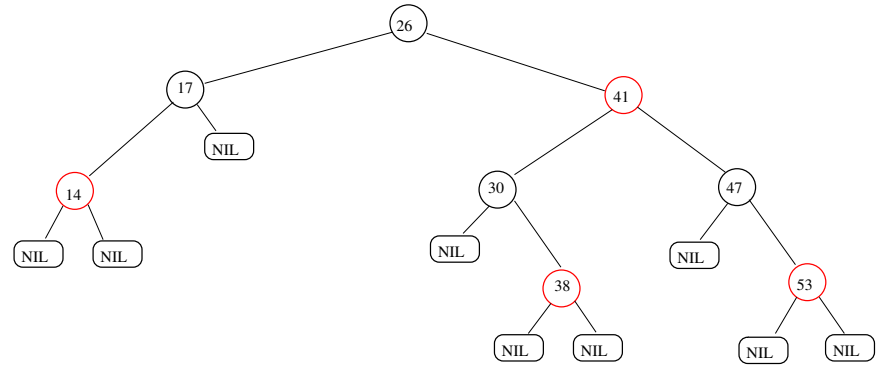
Árvore Rubro-Negra

- 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 $O(lgn)$ 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 excluindo-se x.
Afirmação Uma subárvore com raiz em um nó x tem pelo menos $2^{h_n(x)} - 1$ nós internos
Prova: Indução em $h(x)$
Base: $h(x) = 1 \Rightarrow x$ é folha $\Rightarrow h_n(x) = 0$ e $2^0 - 1 = 0$ nós internos na subárvore
PI: $h(x) > 1$. 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
$2.(2^{h_n(x)-1} - 1) + 1 = 2^{h_n(x)} - 1$.

Lema

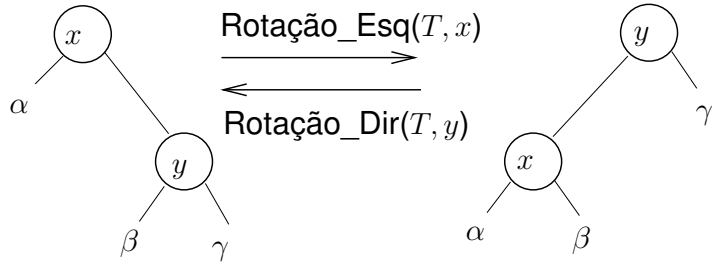
Lema Qualquer árvore RN com n nós internos tem altura $\leq 2\lg(n+1)$.

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 $n \geq 2^{h/2} - 1 \Rightarrow h \leq 2\lg(n+1) - 1$

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

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



Rotação_Esq(T, x)

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

Inclusão na Árvore Rubro-Negra

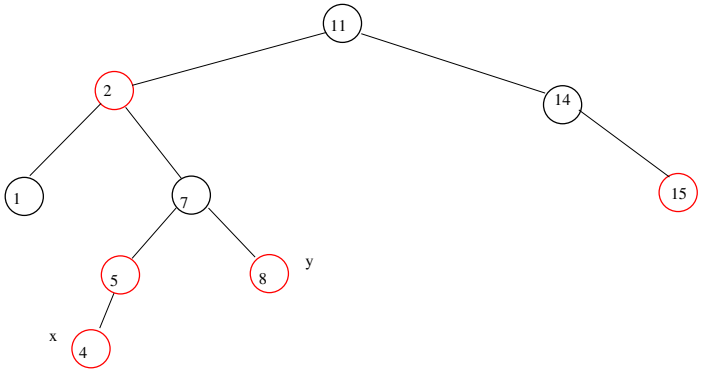
Idéia

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

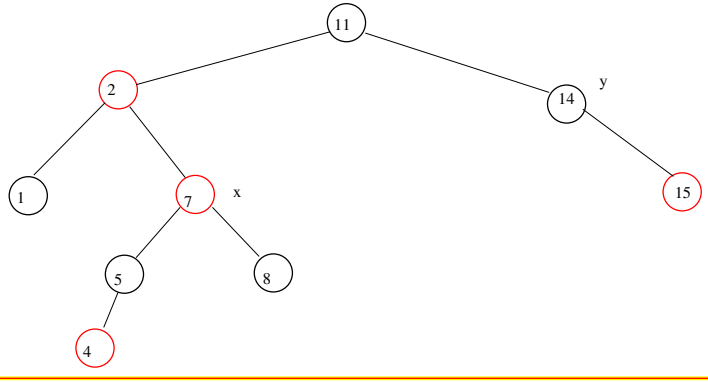
Inclusão na Árvore Rubro-Negra

Caso 1: Se y é rubro: Fazer
$cor[y]$=negro
$cor[p[x]]$=negro
$cor[p[p[x]]]$=rubro



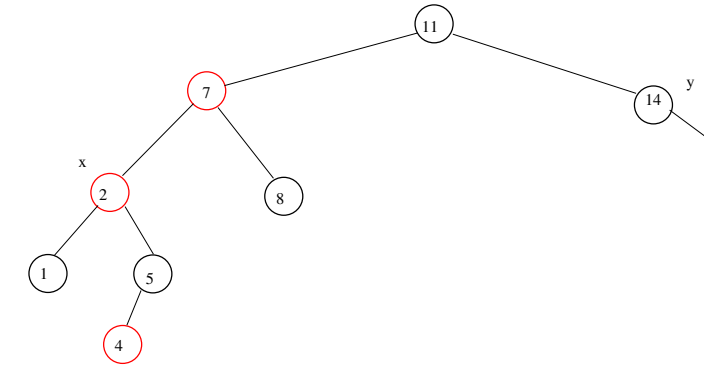
Inclusão na Árvore Rubro-Negra

Caso 2: Se y é negro e além disso
x é filho direito: Fazer
$x = p[x]$
Rotação_Esq(T, x)

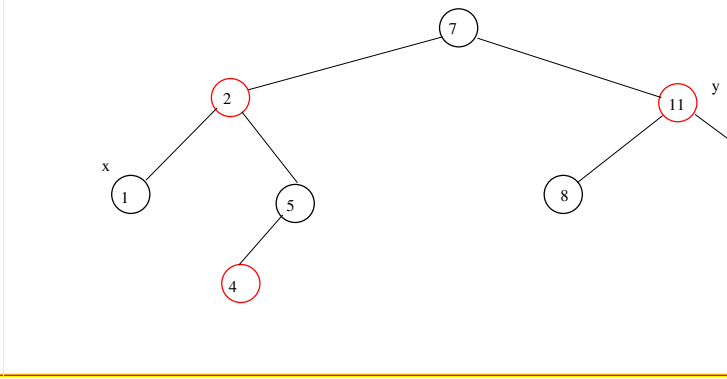


Inclusão na Árvore Rubro-Negra

Caso 3: y é negro e x é filho esquerdo: Fazer
$cor[p[x]]$ =negro
$cor[p[p[x]]]$ =rubro
Rotação_Dir($T, p[p[x]]$)



Inclusão na Árvore Rubro-Negra



Insere_RN(T, x)

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


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 $O(\lg n)$

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!

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.

Remoção na Árvore Rubro-Negra

1. Qual a cor do nó a z ser deletado? **negro**
2. Troque o conteúdo de z pelo de seu sucessor y
3. Remova 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 e mudanças de cores quando necessárias(Há 4 casos)
   - x sempre aponta para nó duplo negro

Complexidade: $O(\lg n)$