Árvores Vermelho-Preto

SCC0202 - Algoritmos e Estruturas de Dados I

Prof. Fernando V. Paulovich
http://www.icmc.usp.br/~paulovic
paulovic@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

4 de novembro de 2010

Sumário

Sumário

Introdução

- As árvores vermelho-preto são árvores binárias de busca "aproximadamente" balanceadas
- Também conhecidas como rubro-negras ou *red-black trees*
- Foram inventadas por Bayer sob o nome "Árvores Binárias Simétricas" em 1972, 10 anos depois das árvores AVL

Introdução

- As árvores vermelho-preto possuem um *flag* extra para armazenar a cor de cada nó, que pode ser **Vermelho** ou **Preto**

- Além deste, cada nó será composto ainda pelos seguintes campos
  - info (os "dados" do nó)
  - fesq (filho esquerdo)
  - fdir (filho direito)
  - pai

Introdução

- Restringindo o modo como os nós são coloridos desde a raiz até uma folha, assegura-se que **nenhum caminho será maior que duas vezes o comprimento de qualquer outro**, dessa forma, a árvore é aproximadamente balanceada

Introdução

- Uma árvore vermelho-preto com n nós tem altura máxima

  $$2\log(n+1)$$

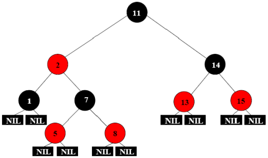
- Por serem "balanceadas" as árvores vermelho-preto possuem complexidade logarítmica em suas operações

  $$O(\log n)$$

Sumário

Propriedades

- Todo nó é vermelho ou preto
- A raiz é preta
- Toda folha externa (nó **NIL**) é preta
- Se um nó é vermelho, então ambos seus filhos são pretos
- Todos os caminhos a partir da raiz da árvore até suas folhas passa pelo mesmo número de nós pretos

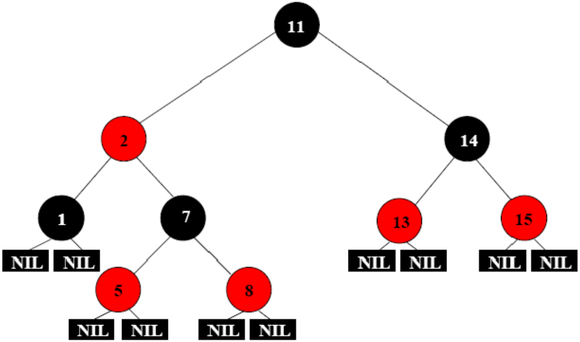


Propriedades

- Um nó que satisfaz as propriedades anteriores é denominado equilibrado, caso contrário é dito desequilibrado
  - Em uma árvore vermelho-preta todos os nós estão equilibrados
- Uma condição óbvia obtida das propriedades é que num caminho da raiz até uma sub-árvore vazia **não pode existir dois nós vermelhos consecutivos**

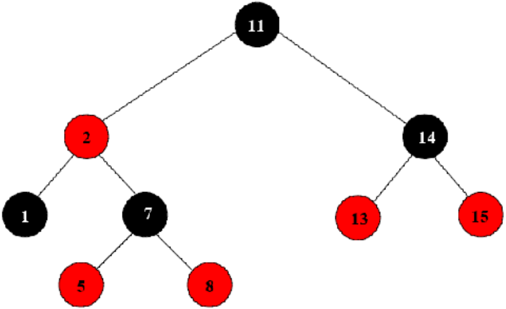
Propriedades

- Formas de representação



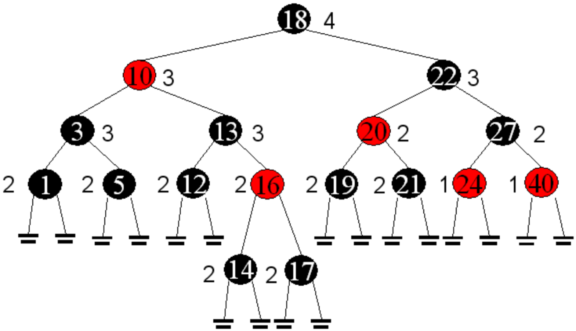
Propriedades

- Formas de representação



Propriedades

- **Altura negra**: é número de nós negros encontrados até qualquer nó folha externo



Propriedades

Lema 1

- Seja x a raiz de uma (sub)árvore vermelho-preta, então essa terá no mínimo $2^{an(x)} - 1$ nós internos, onde $an(x)$ é a altura negra de x

---

- Prova por indução
    - Caso base: Um nó de altura 0 (i.e., nó-folha externo) tem $0 = 2^0 - 1$ nós internos
    - Caso genérico: Um nó x de altura $h > 0$ tem 2 filhos com altura negra $an(x)$ ou $an(x) - 1$, conforme x seja vermelho ou negro. No pior caso, x é negro e as subárvores enraizadas em seus 2 filhos têm $2^{an(x)-1} - 1$ nós internos cada e x tem $2(2^{an(x)-1} - 1) + 1 = 2^{an(x)} - 1$ nós internos

Propriedades

Lema 2

- Uma árvore vermelho-preta com n nós tem no máximo altura $2 \times \log_2(n+1)$

- Prova
  - Se uma árvore tem altura h, a altura negra de sua raiz será no mínimo $h/2$ (pelo propriedade **(4)**) e a árvore terá $n \geq 2^{h/2} - 1$ nós internos (Lema 1)
  - Como consequência, a árvore tem altura $O(\log n)$ e as operações de busca, inserção e remoção podem ser feitas em $O(\log n)$

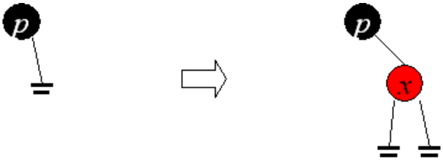
Sumário

Inserção

- A operação de inserção em uma árvore vermelho-preta começa por uma busca da posição onde o novo nó deve ser inserido
- Essa inserção inicial segue os princípios de uma inserção em *Árvore Binária de Busca*
- Após a inserção um conjunto de propriedades é testado, e se a árvore não satisfizer essas propriedades, são realizadas rotações e/ou ajustes de cores, de forma que a árvore permaneça balanceada

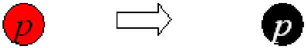
Inserção

- Um nó é inserido sempre na cor vermelha, assim, não altera a altura negra da árvore
- Se o nó fosse inserido na cor preta, invalidaria a propriedade **(5)**, pois haveria um nó preto a mais em um dos caminhos



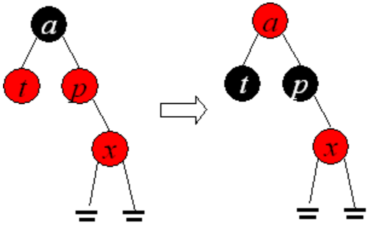
Inserção

- Caso a inserção seja feita em uma árvore vazia, basta alterar a cor do nó para preto, satisfazendo assim a propriedade número **(2)**



Rotações e Recolorações

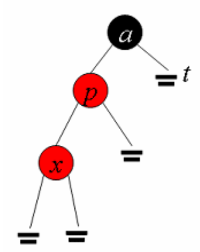
- **Caso 1**: Suponha agora que p é vermelho e a, o pai de p (e avô de x) é preto. Se t, o irmão de p (tio de x) é vermelho, ainda é possível manter o critério **(4)** apenas fazendo a recoloração de a, t e p



- Obs.: Se o pai de a é vermelho, o rebalanceamento tem que ser feito novamente considerando a como nó inserido

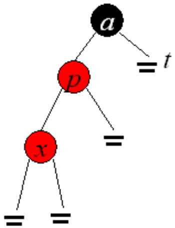
Rotações e Recolorações

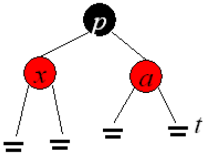
- **Caso 2**: Suponha que p é vermelho, seu pai a é preto e seu irmão t é preto. Neste caso, para manter o critério **(4)** é preciso fazer rotações envolvendo a, t, p e x.
  - Há 4 subcasos que correspondem às 4 rotações possíveis



Rotações e Recolorações

- **Caso 2a**: O nó *x* é filho esquerdo de *p*, e *p* é filho esquerdo de *a*
  - Aplicar *Rotação Direita*

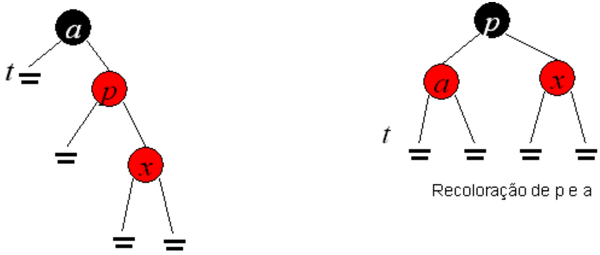




Recoloração de p e a

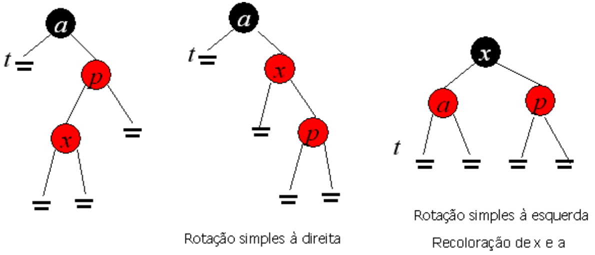
Rotações e Recolorações

- **Caso 2b**: O nó x é filho direito de p, e p é filho direito de a
    - Aplicar *Rotação Esquerda*



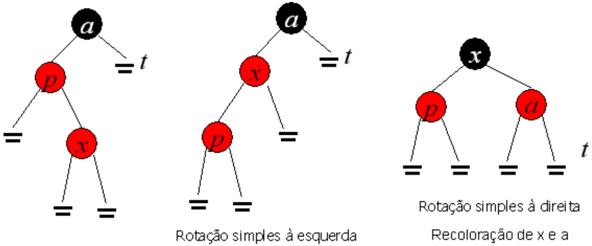
Rotações e Recolorações

- **Caso 2c**: O nó x é filho esquerdo de p, e p é filho direito de a
    - Aplicar *Rotação Dupla Esquerda*



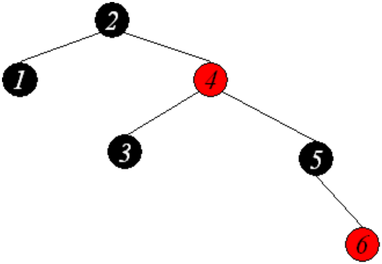
Rotações e Recolorações

- **Caso 2d**: O nó x é filho direito de p, e p é filho esquerdo de a
    - Aplicar *Rotação Dupla Direita*



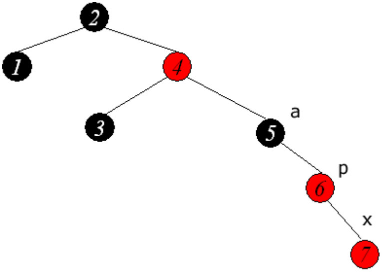
Exemplo 1

- Estado inicial da árvore
- Vamos inserir um nó com valor 7



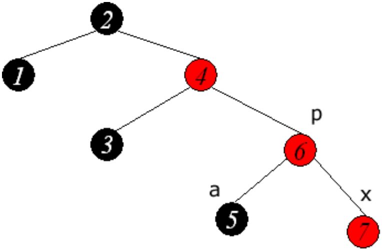
Exemplo 1

- O tio t do elemento inserido x é **preto**, seu pai p é filho direito de a e x é filho direito de p
  - **Caso 2b**: requer rotação esquerda em a



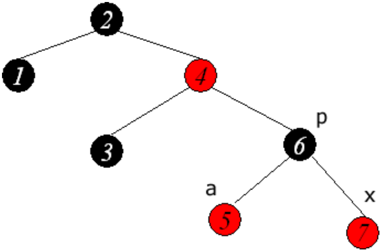
Exemplo 1

- Violação da propriedade pelos nós p e x – recoloração



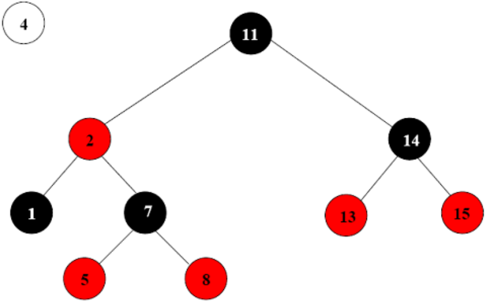
Exemplo 1

- Recoloração dos nós p e x



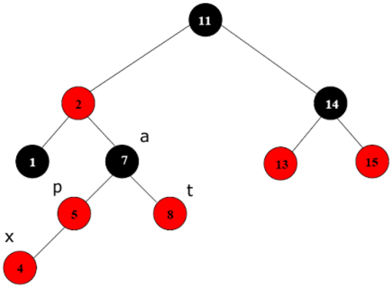
Exemplo 2

- Estado inicial da árvore
- Vamos inserir o nó com valor 4



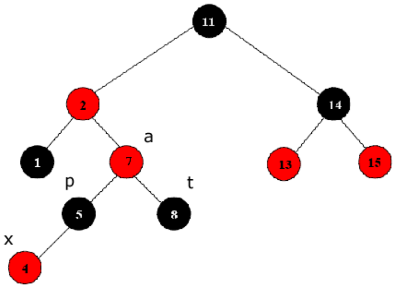
Exemplo 2

- O tio t do elemento inserido x é vermelho
  - **Caso 1**: requer a recoloração dos nós a, t e p
- Violação da propriedade **(4)**



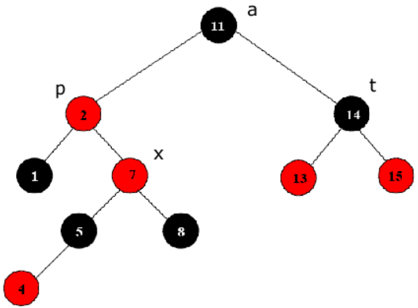
Exemplo 2

- Nós p e t passam a ser pretos e o nó a passa a ser vermelho
- Violação da propriedade **(4)** entre os nós a e seu pai



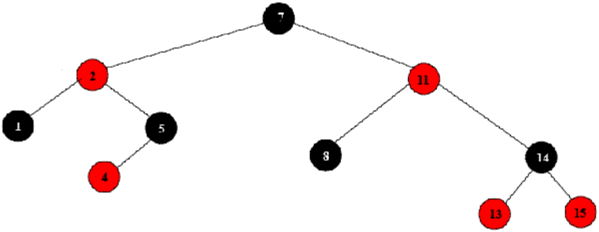
Exemplo 2

- O tio t do elemento inserido x é **preto** e o elemento inserido é um filho da direita de p
    - **Caso 2d**: requer rotação dupla direita – rotação esquerda em p e rotação direita em x



Exemplo 2

- Processo termina porque já atingiu a raiz da árvore



Complexidade da Inserção

- Rebalanceamento tem custo $O(1)$
- Rotações têm custo $O(1)$
- Inserção tem custo $O(\log n)$

Exercício

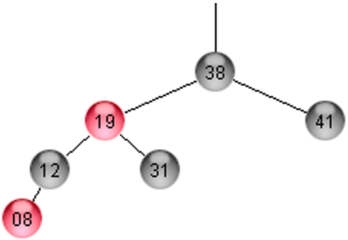
Inserir

- 41 - 38 - 31 - 12 - 19 - 8

Exercício

Inserir

- 41 - 38 - 31 - 12 - 19 - 8



Sumário

Remoção

- A remoção nas árvores vermelho-pretas se inicia com uma etapa de busca e remoção como nas árvores binárias de busca convencionais

- Então se alguma propriedade vermelho-preta for violada, a árvore deve ser rebalanceada

Remoção

- Caso a **remoção efetiva** seja de um nó **vermelho**, esta é realizada sem problemas, pois todas as **propriedades** da árvore se manterão **intactas**
- Se o nó a ser **removido** for **preto**, a quantidade de nós pretos em pelo menos um dos caminhos da árvore foi alterado, o que implica em que algumas operações de rotação e/ou alteração de cor sejam feitas para **manter o balanceamento da árvore**

Remoção

Remoção Efetiva

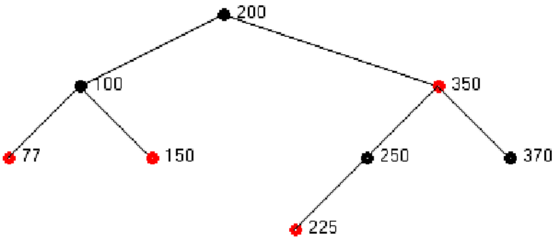
- Após as operações de rotação/alteração de cor necessárias, a remoção do nó é efetivamente realizada, restabelecendo-se as propriedades da árvore

Remoção Preguiçosa

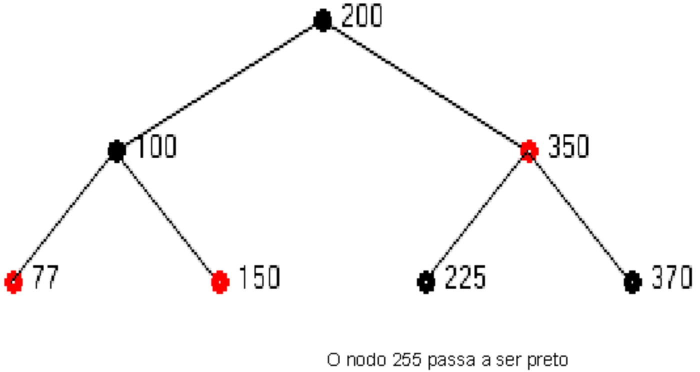
- Consiste em apenas marcar um determinado nó como removido, sem efetivamente retirá-lo da árvore

Exemplo 1

- Remover o nó 250

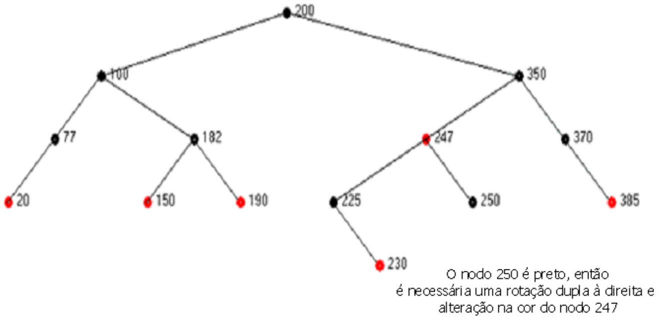


Exemplo 1

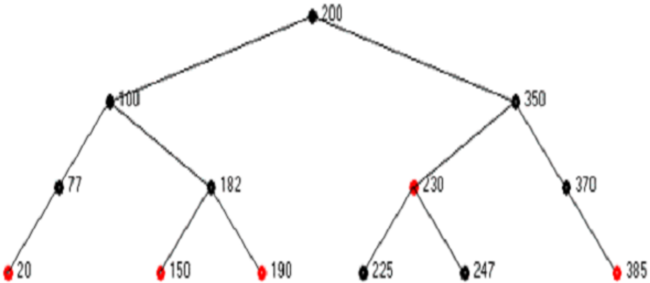


Exemplo 2

- Remover o nó 250



Exemplo 2



Rotação dupla à direita