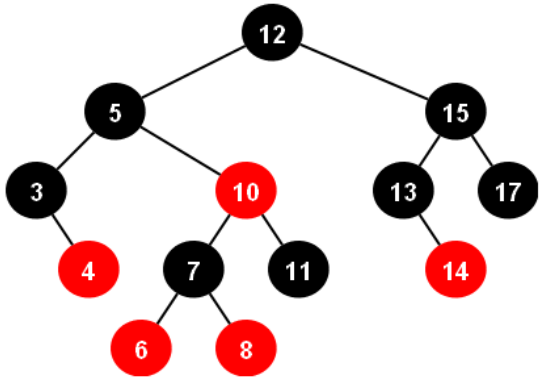
Governo Federal

Ministério da Educação

Universidade Federal do Maranhão

A Universidade que Cresce com Inovação e Inclusão Social

Árvore Rubro-Negra



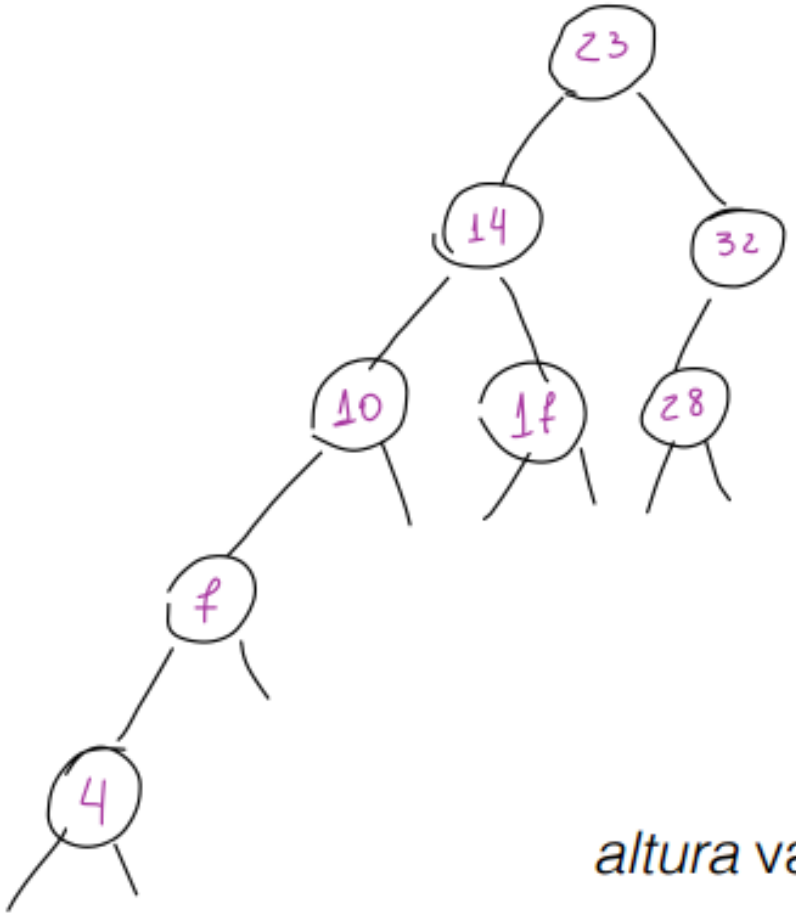
Estrutura de Dados II

Prof. João Dallyson

Email: Joao.dallyson@ufma.br

Referência: Notas de aula Prof. Rafael Fernandes DAI/IFMA

Introdução



qual o tempo
no pior caso
da operação de busca
(ou inserção) em
uma ABB com *n nós*?

- ☐ $\Theta(1)$
- ☐ $\Theta(\log n)$
- ☐ $\Theta(altura)$
- ☐ $\Theta(n)$

altura varia entre ~log *n* e ~*n*

FONTE: http://www.ic.unicamp.br/~rocha/teaching/2014s1/mc202/aulas/aula-arvores-rubro-negras.pdf

Introdução

- **As árvores de pesquisa binária já estudadas (ED1) de altura h podem implementar as operações básicas: BUSCA, INSERIR, REMOVER, MÍNIMO, MÁXIMO, SUCESSOR E PREDECESSOR no tempo O(h)**
  - Estas operações são rápidas se a árvore for pequena

- **A árvore rubro-negra é um tipo de árvore de pesquisa balanceada com o objetivo de garantir que as operações básicas sejam executadas em tempo O(lg n) no pior caso.**

Introdução

- **É uma árvore de pesquisa binária na qual cada nó possui um bit extra para cor do nó.**
  - Cor: **vermelho** ou **preto**
- **Criadas em 1972 com o nome de árvores binárias simétricas**
- **Todo caminho em uma árvore rubro-negra é maior que o dobro de qualquer outro caminho.**
  - Árvore aproximadamente balanceada.
- **Aplicações reais:**
  - Sistemas de bibliotecas: dicionários.
  - São usadas no TreeSet e TreeMap no Core Java API, como também no Standard C++ sets and maps.

Complexidade

- **A altura máxima de uma árvore Rubro-Negra com n nós internos é:**

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

- **Por serem balanceadas as operações levam tempo**

$$O(\log n)$$

- **Possui melhor desempenho em árvores com alturas maiores**

AVL x Rubro-Negras

- Árvores AVL:
  - primeira árvore binária de busca com balanceamento; proposta por Adel'son-Vel'skii e Landis em 1962
  - altura: entre $\log_2(n + 1)$ e $1.4404 \log_2(n + 2) - 0.328$, portanto, $O(\log n)$

- Árvores rubro-negras:
  - proposta por Guibas e Sedgewick em 1978
  - altura: $2 \log_2(n + 1)$, portanto, $O(\log n)$

- Comparação: árvores AVL são mais rigidamente balanceadas que árvores rubro-negras, levando a inserção e remoção mais lentas, porém recuperação (busca) mais rápida

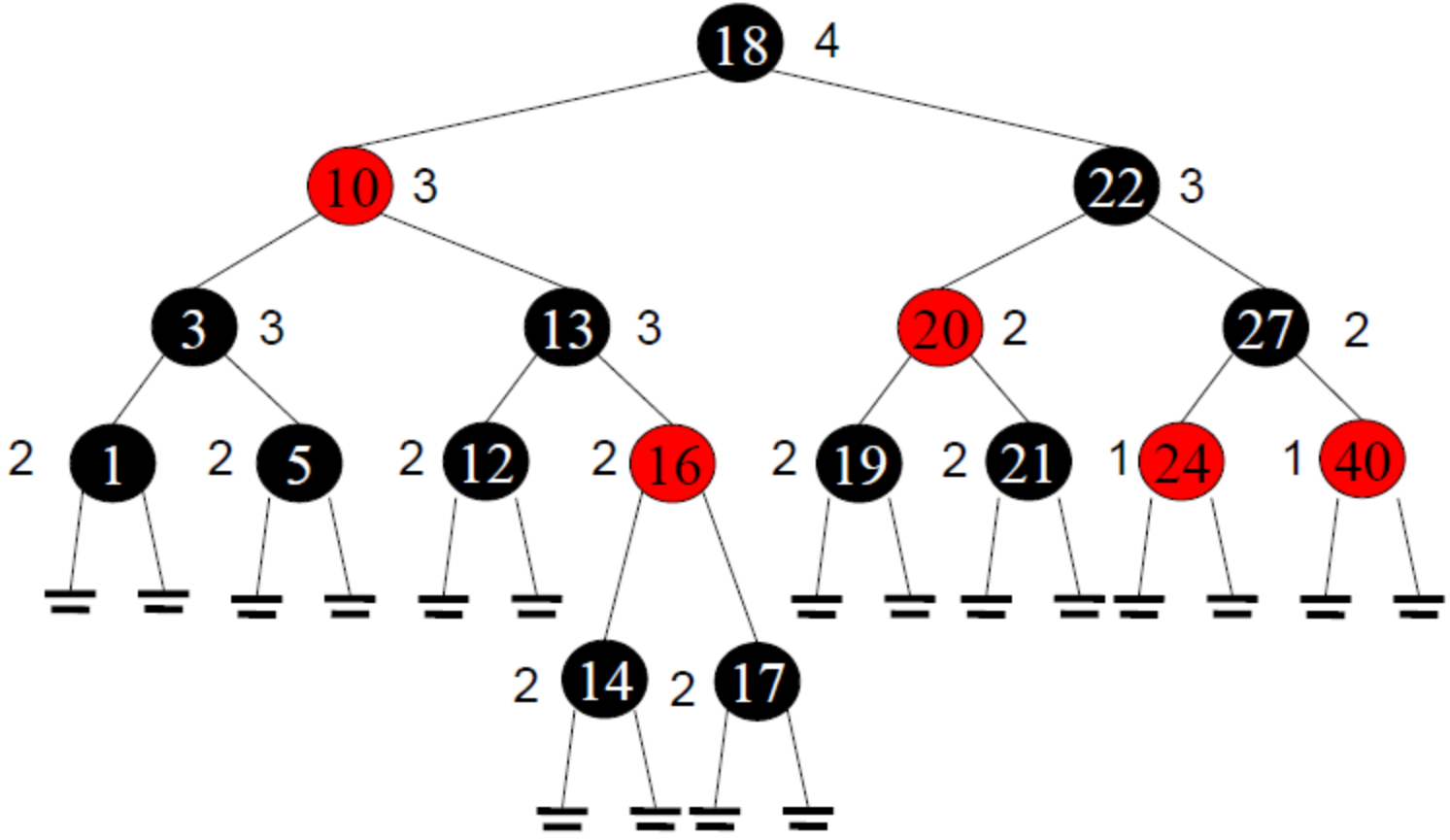
Nó da Árvore Rubro-Negra

```
public class NoRN <AnyType> {

      AnyType elemento;
      NoRN<AnyType> pai, esquerda, direita;
      private boolean cor;

      NoRN( AnyType e ){
          this (e, null, null );
      }

      NoRN(AnyType e, NoRN esq, NoRN  dir){
          elemento  = e;
          esquerda      = esq;
          direita     = dir;
          cor = ArvoreRN.BLACK;
      }
}
```

Propriedades

1. Todo nó é vermelho ou **preto**
2. A **raiz** da árvore necessariamente é preta
3. Toda **folha** null é preta
4. Nós vermelhos que não seja folhas possuem apenas **filhos** pretos
5. Todos os caminhos a partir da raiz até suas folhas passam pela mesma quantidade de nós pretos
6. Não podem existir dois nós vermelhos consecutivos

Exemplo

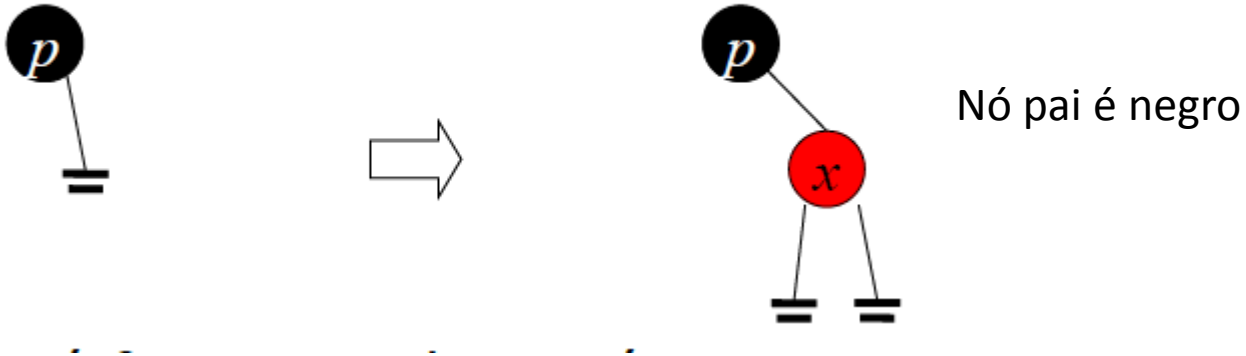


Operações

- **Toda vez que um nó é inserido ou removido todas as propriedades devem ser testadas**
- **Caso uma ou mais propriedades não sejam satisfeitas então rotações e/ou mudanças de cores devem ser feitas**
  - Rotações e mudanças de cores são realizadas para manter o balanceamento
- **O bh ou black-high de cada nó indica a quantidade de nós pretos encontrados entre a raiz e qualquer folha**

Inserção

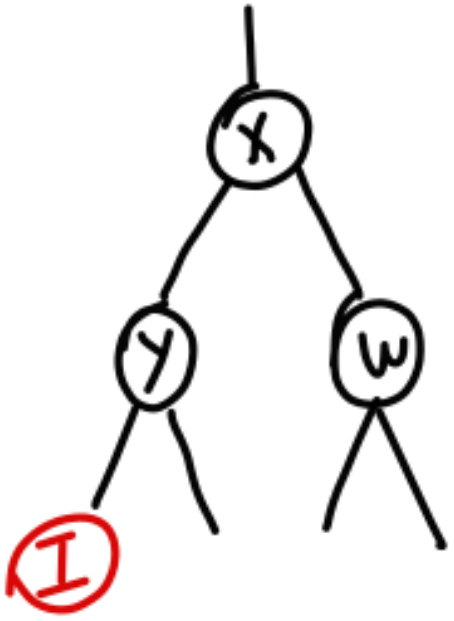
- **Todo nó inserido é inicialmente vermelho**
  - Objetiva não alterar o bh dos nós



- Se um nó for inserido na **árvore vazia** então basta mudar a cor do nó de vermelho para preto

Inserção

- **Noção de Tio**



o caso simétrico
é sempre aplicável

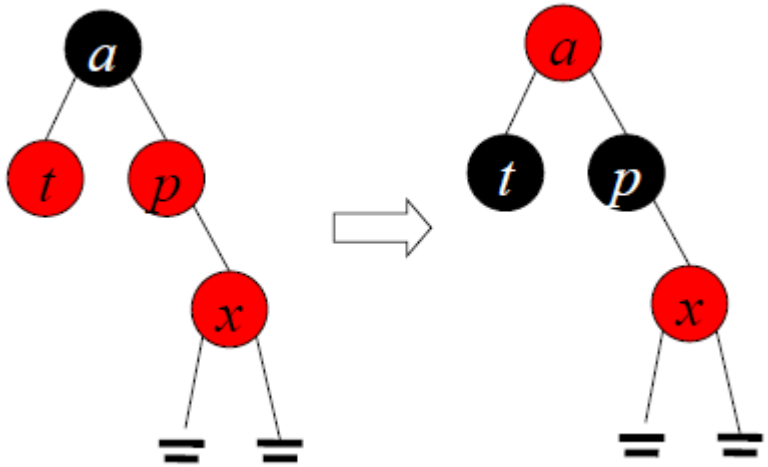
Inserção

Caso 1

– Se um **nó** for vermelho e não tiver pai, então ele é a **raiz** e deve ser transformado em preto

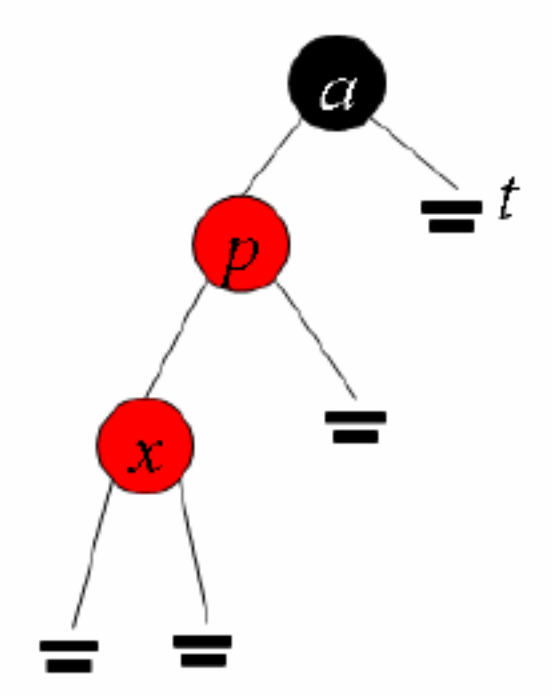
Caso 2

– Um nó x esta sendo inserido, e seu **tio** é vermelho então é necessário recolorir o pai, o tio e o avô



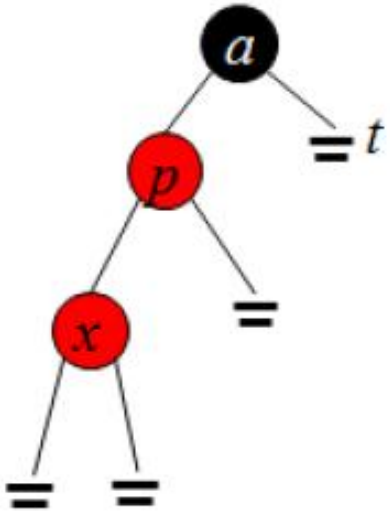
Inserção

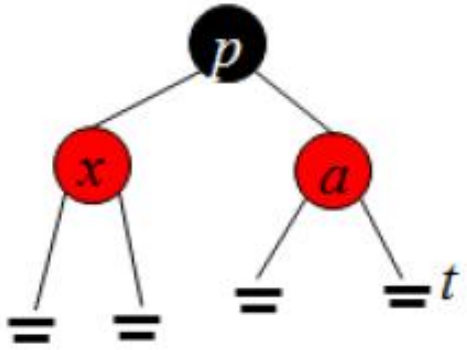
- **Caso 3**
  - Se o nó a tiver um **pai** vermelho uma nova operação deve ser feita
- **Caso 4**
  - Supondo que x esta sendo inserido em um **nó** vermelho e seu **tio** é null(preto), então deve ser feita uma rotação



Inserção

- **Case 3:**
  - Subcaso 1: Caso os nós estejam à esquerda deve-se fazer uma rotação à direita.
  - X é filho esquerdo, tem pai(p) vermelho e **avô(a)** preto
  - Os nós devem ser recoloridos.

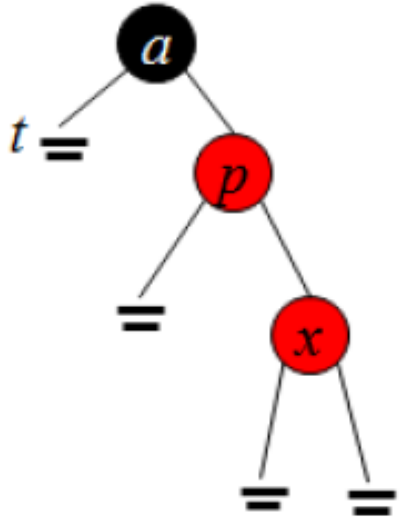


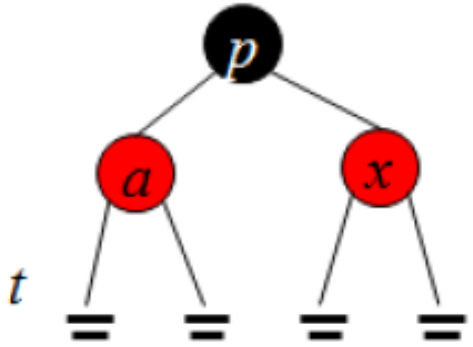


Recoloração de p e a

Inserção

- **Case 3:**
  - Subcaso2:
    - Caso os nós estejam à direita deve-se fazer uma rotação à esquerda
    - X é filho direita, tem pai(p) vermelho e **avô(a)** preto
    - Os nós devem ser recoloridos

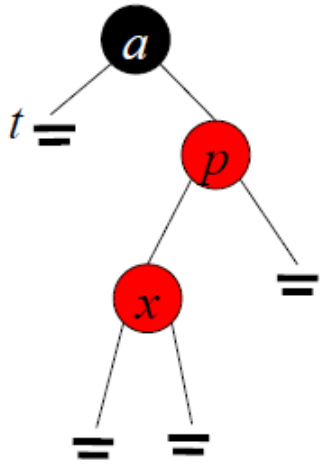


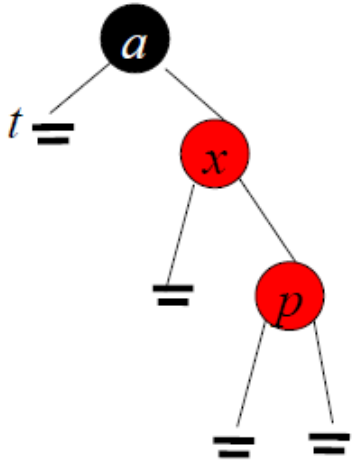


Recoloração de p e a

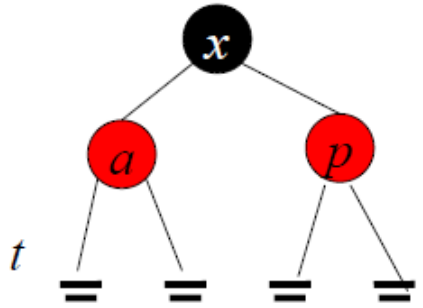
Inserção

- **Case 3:**
  - Subcaso3:
    - X é filho esquerda, tem pai(p) vermelho e **avô(a)** preto
    - Aplica-se uma rotação à direita com p e uma rotação à esquerda com a





Rotação simples à direita

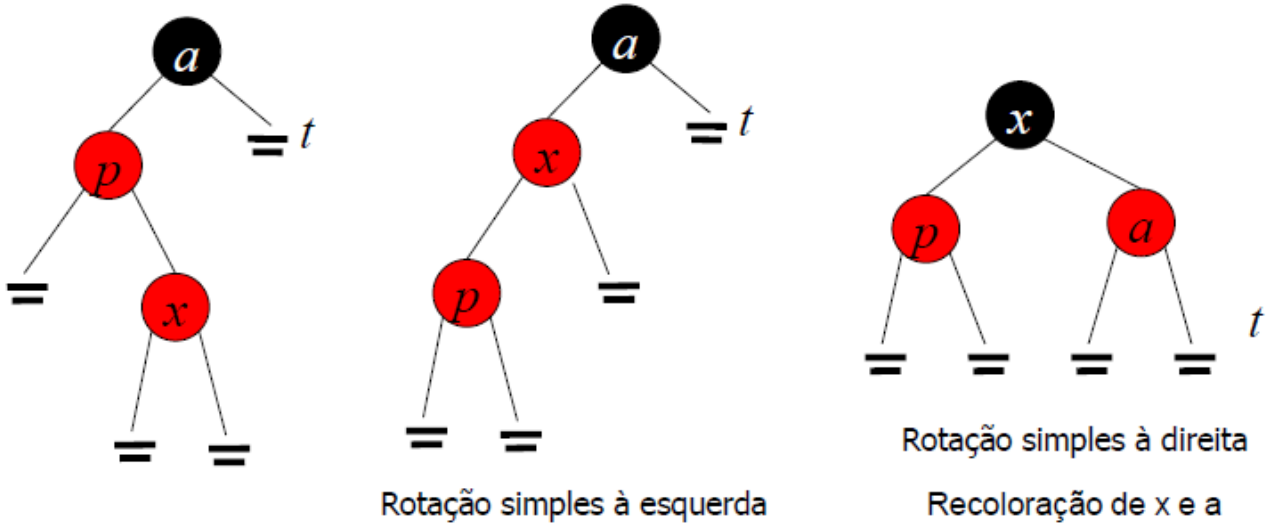


Rotação simples à esquerda

Recoloração de x e a

Inserção

- **Case 3:**
  - Subcaso4:
    - X é filho direta, tem pai(p) vermelho e **avô(a)** preto
    - Aplica-se uma rotação à esquerda com p e uma rotação à direita com a

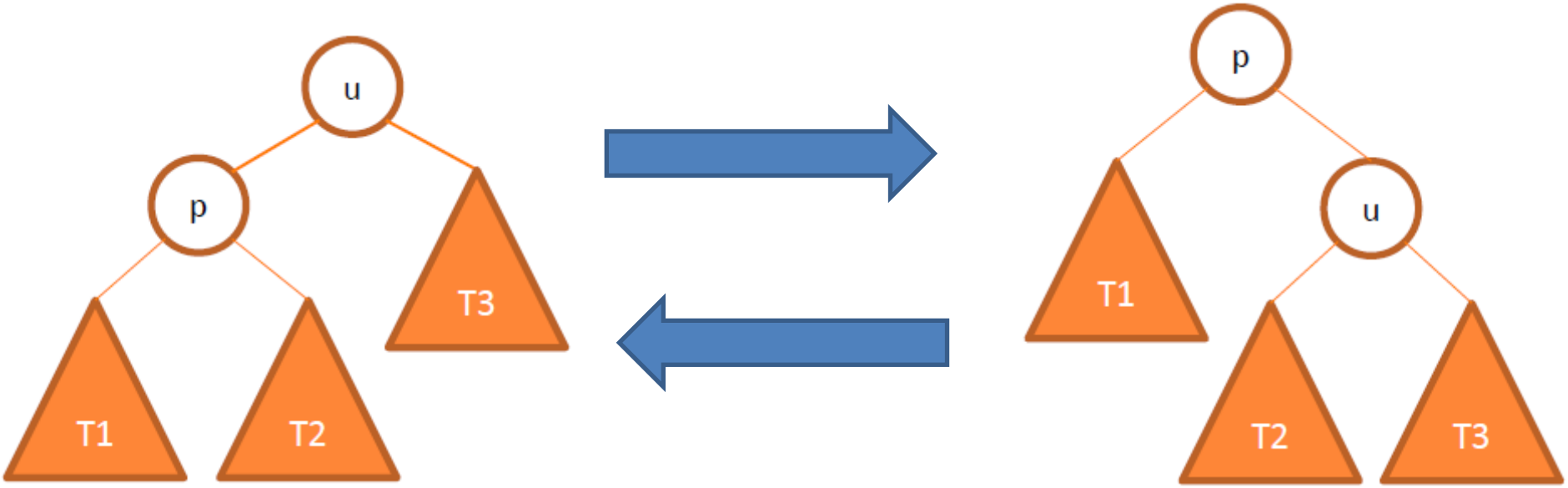


Rotação simples à esquerda

Rotação simples à direita

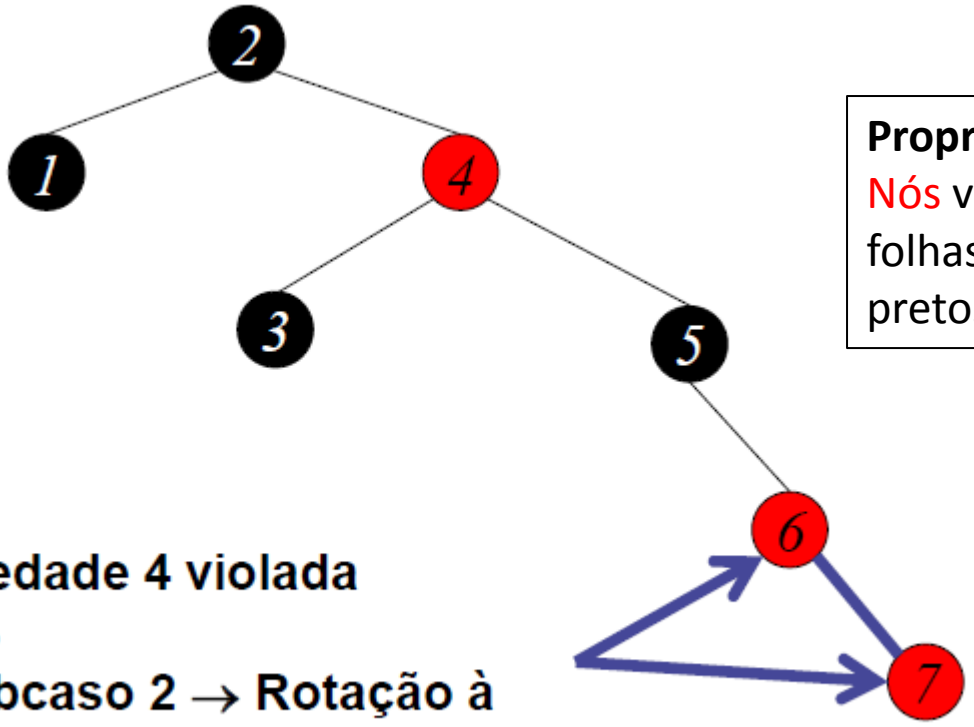
Recoloração de x e a

Rotações com Sub-Árvores



Exemplo

- **Inserindo o nó 7 na árvore abaixo**

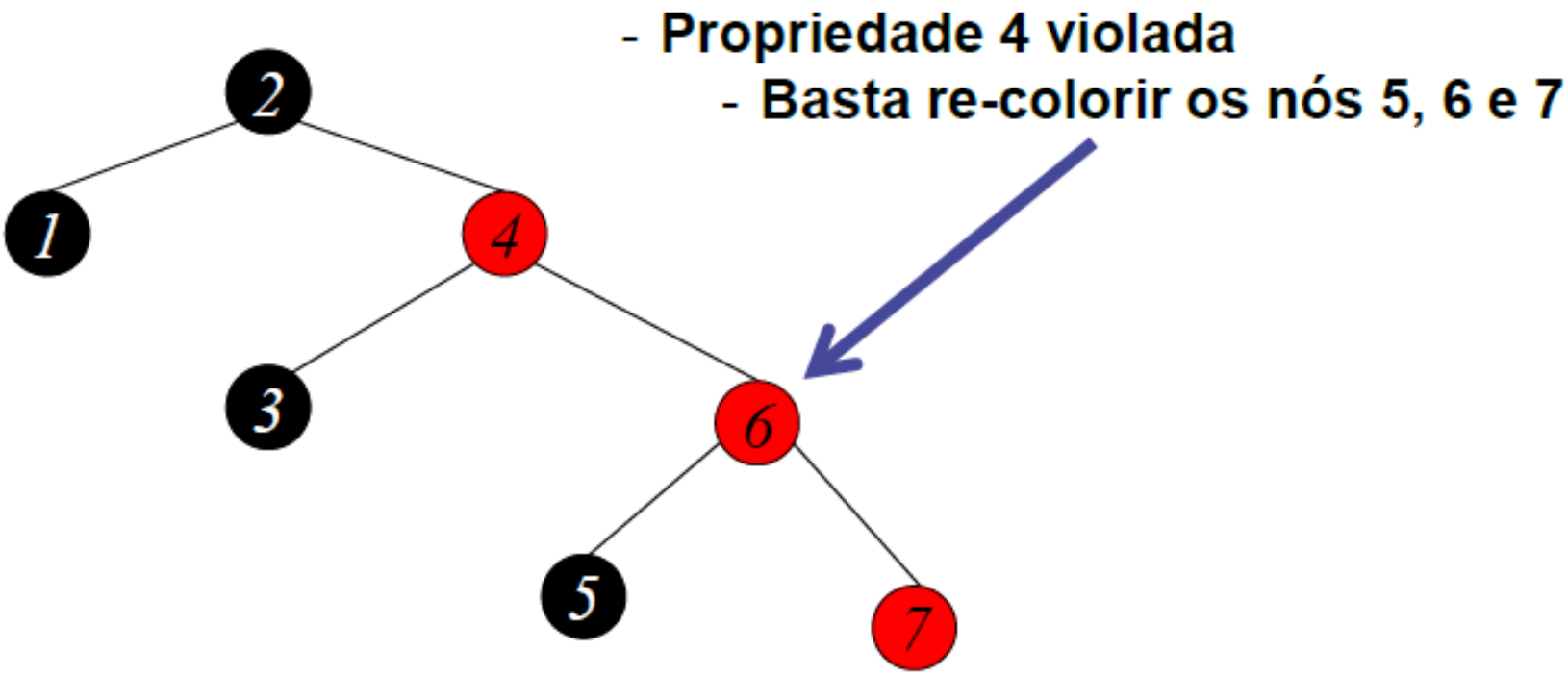


Propriedade 4:
Nós vermelhos que não sejam folhas possuem apenas filhos pretos

- Propriedade 4 violada
- Caso 3
  - Subcaso 2 → Rotação à esquerda no nó 5

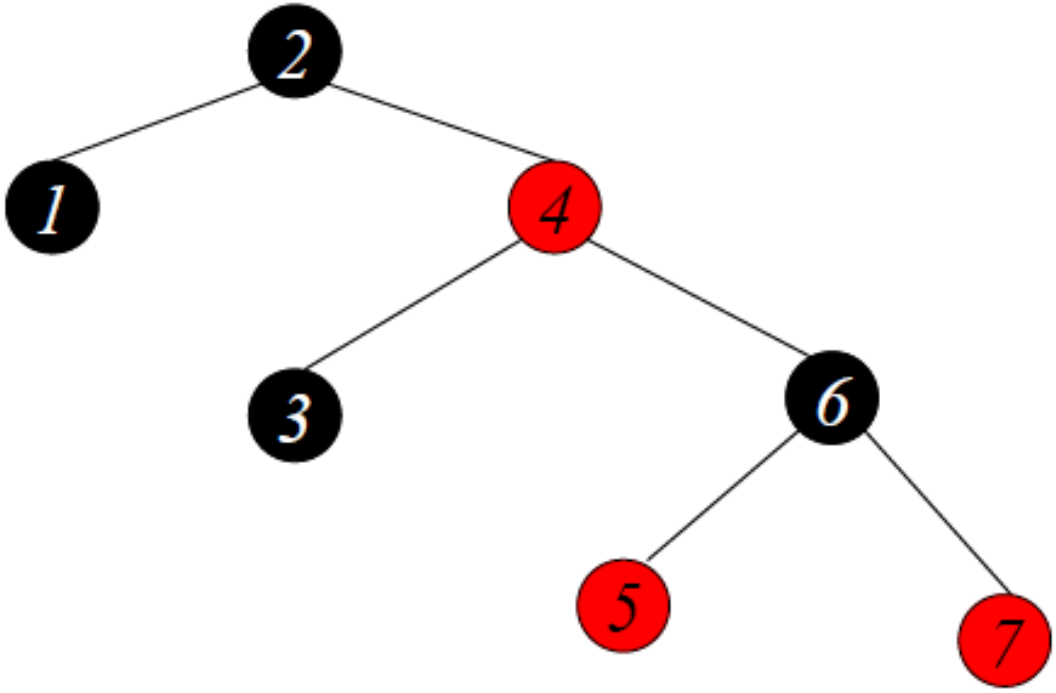
Exemplo

- **Inserindo o nó 7 na árvore abaixo**



Exemplo

- **Inserindo o nó 7 na árvore abaixo**

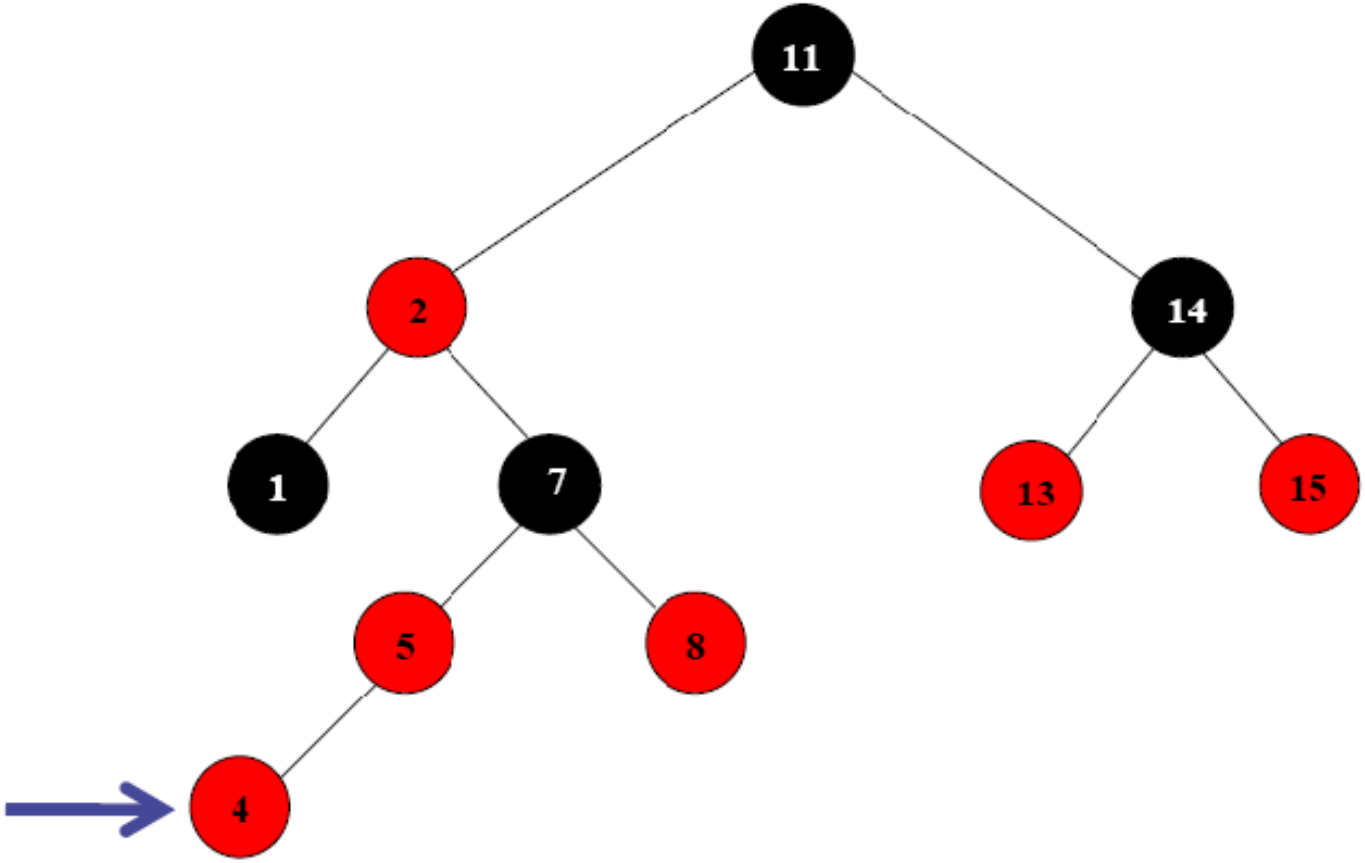


Casos para correção de cores

- **Existem 3 casos para correção das cores:**
  - Caso 1 – o tio do elemento inserido é vermelho
    - Mudar cores
  - Caso 2 – O tio do elemento inserido é preto e o elemento inserido é à direita
  - Caso 3 – O tio do elemento inserido é preto e o elemento inserido é à esquerda
    - Nos casos 2 e 3 - Mudam-se as cores e fazem-se uma ou duas rotações

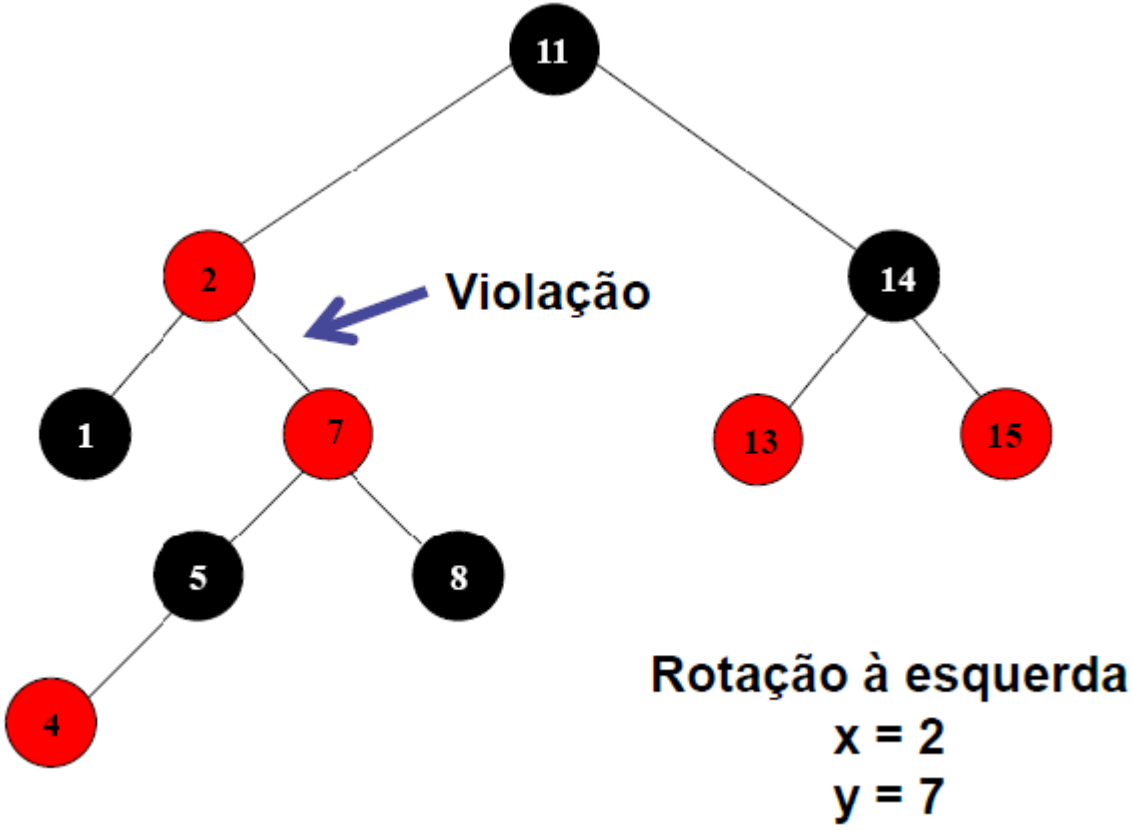
Exemplo: inserindo o nó 4

- **Caso 1: o tio do nó inserido é vermelho**



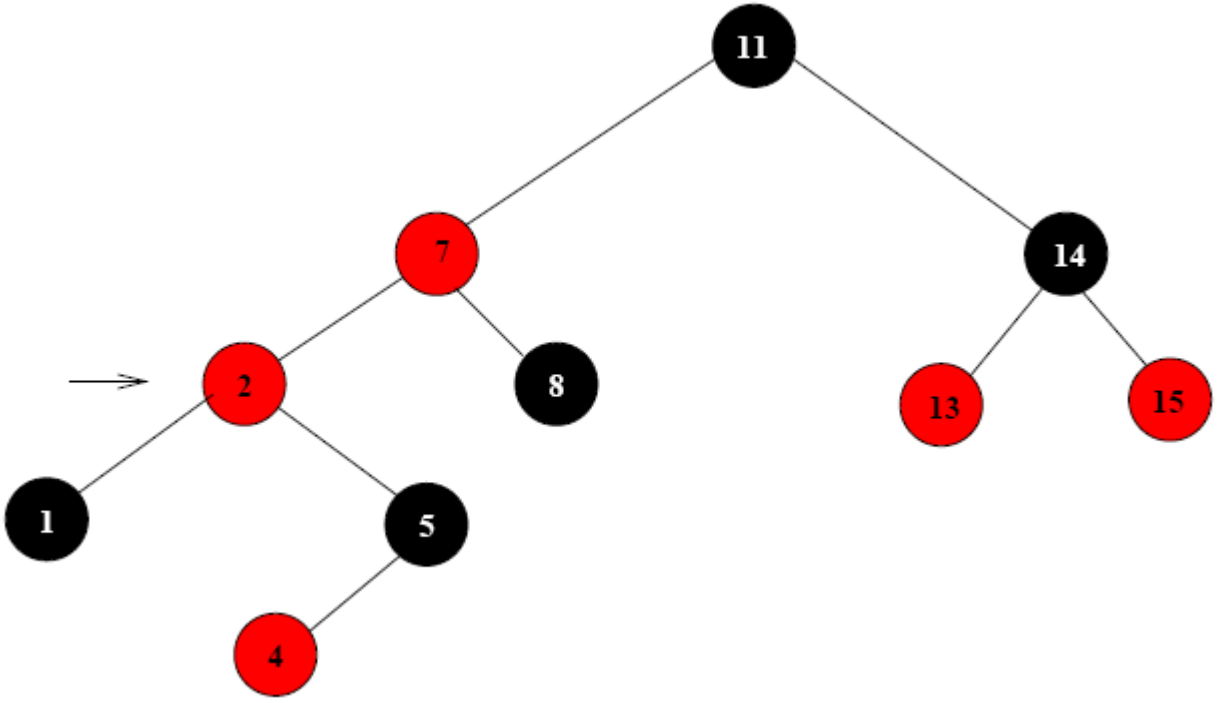
Exemplo: inserindo o nó 4

- **Caso 2: 7 é vermelho e 14 (tio) é preto**
- **Rotação nos elementos 2 e 7**



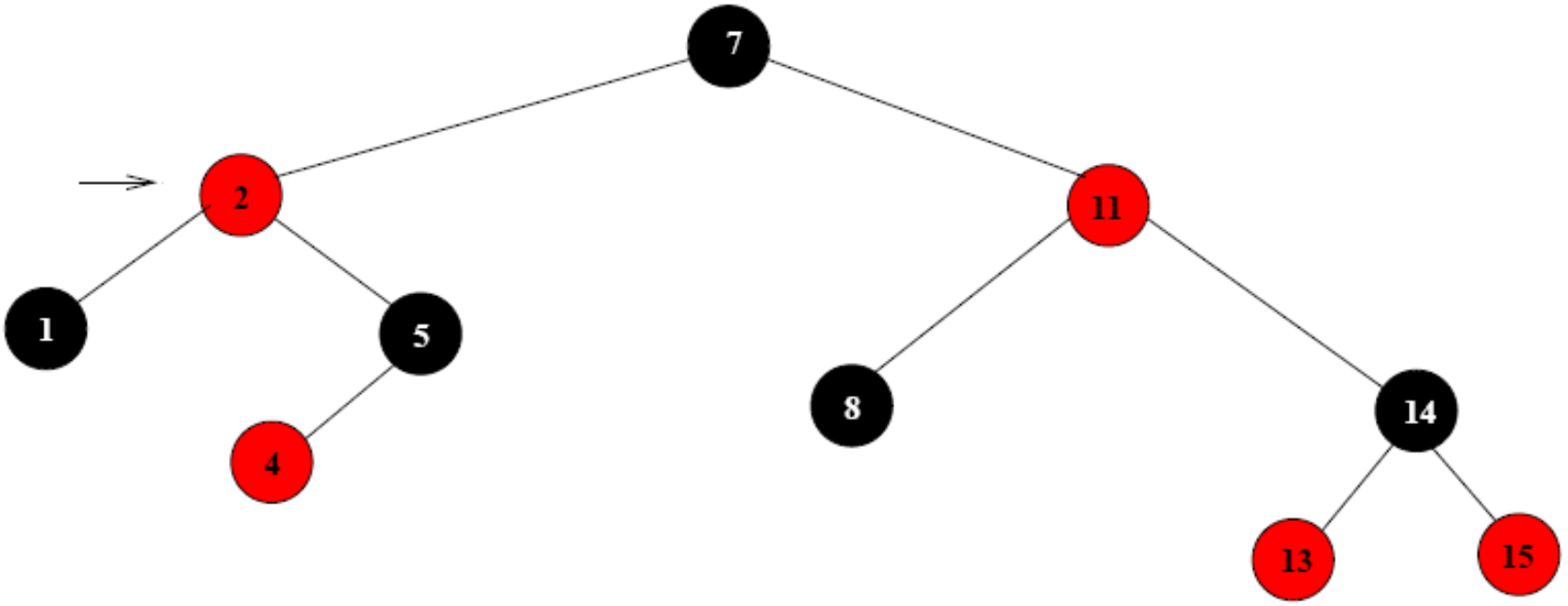
Exemplo: inserindo o nó 4

- Caso 3 o tio do elemento inserido à esquerda é preto; nós 7 e 2 continuam violando a propriedade 4
- Rotacionar 7 e 11 para a direita



Exemplo: inserindo o nó 4

- Pela propriedade 1 o nó 7 passa a ser preto
- O nós 8 e 14 são pretos, logo o nó 11 passa a ser vermelho



Animação

- **http://www.cs.usfca.edu/~galles/visualization/RedBlack.html**

Exercício

1) Insira os elementos 2, 1, 4, 5, 9, 3, 6, 7 em uma árvore Rubro-Negra

Referências

Básica

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: Teoria e Prática. Editora Campus, 2002
- Ziviani, N. Projeto de Algoritmos Com Implementações em Pascal e C, Cengage Learning, 2004.
- Notas de aula. Prof. Rafael Fernandes DAI/IFMA
- Notas de aula. Prof. Tiago A. E. Ferreira. DEINFO/UFRPE

Complementar

- ASCENCIO, Ana Fernanda Gomes; ARAUJO, Graziela Santos. Estruturas de Dados: Algoritmos, análise da complexidade e implementações em Java e C/C++. Pearson Prentice Hall, 2010
- DROZDEK, Adam. Adam Drozdek. Data Structures and Algorithms in Java. 2. Cengage Learning. 2004. 2. Cengage Learning. 2004
- GOODRICH, Michael T. Estruturas de dados e algoritmos em java. 4 ED. Porto Alegre: Bookman, 2007. 600.
- SKIENA, Steven S.. **The Algorithm Design Manual**. 2. Springer-Verlag. 2008

Perguntas....