AED2 Algoritmos e Estruturas de Dados II

Aula 13 – Árvores Rubro-Negras

Prof. Aléssio Miranda Júnior

alessio@cefetmg.br

2Q-2017

Árvores de Busca Binária

Por que ABBs?

São estruturas eficientes de busca (se a árvore estiver balanceada).

Permitem minimizar o tempo de acesso no pior caso.

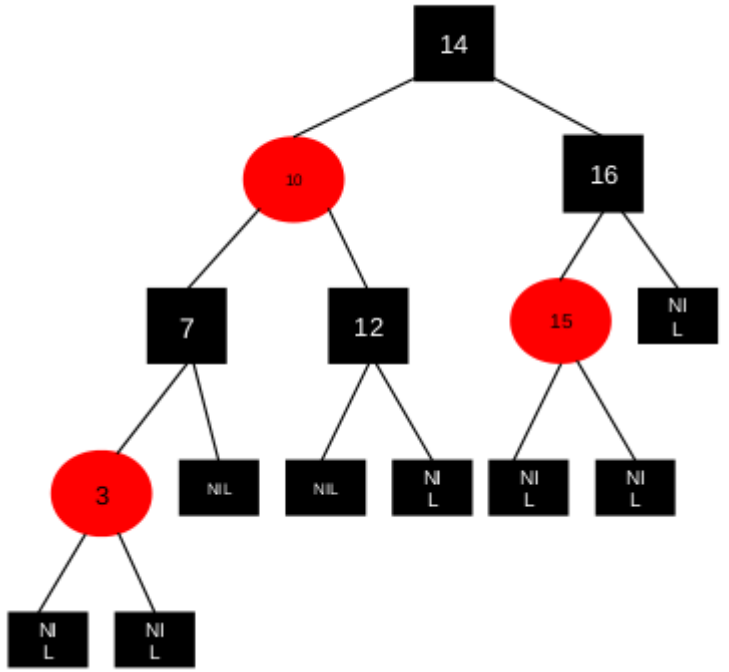
Complexidade das operações de busca, inserção, remoção:

Se balanceada → O(lg(n)) Senão → O(n)

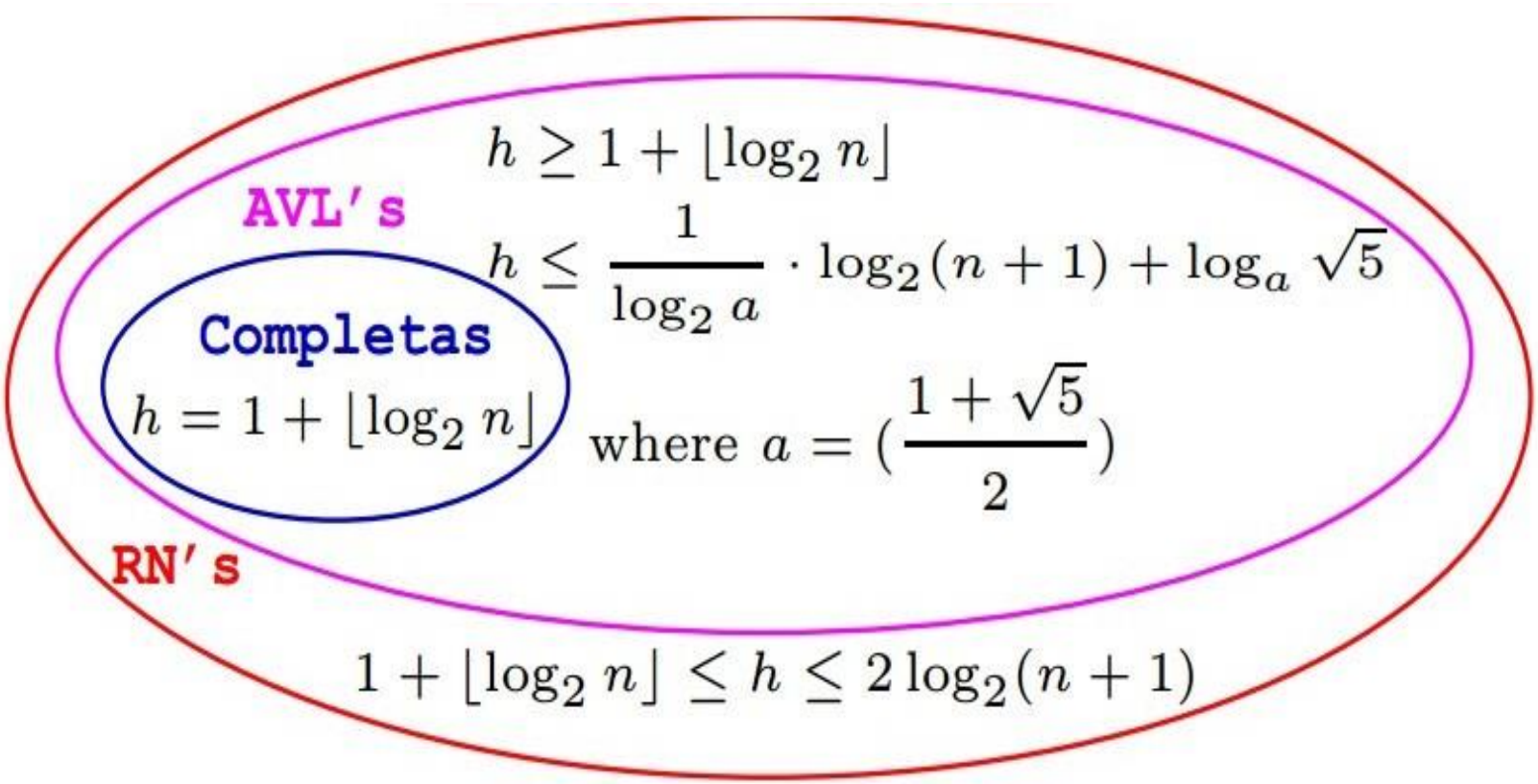
Árvores balanceadas

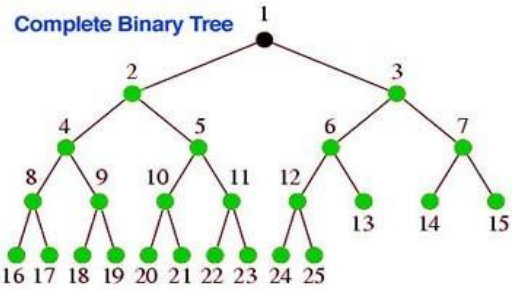
- As seguintes árvores são as ditas balanceadas – com altura O(lg(n)):
  - AVL
  - Rubro-negras / vermelho-preto / red-black tree
  - B

- As árvores Rubro-negras apresentam uma altura de, no máximo, igual a 2 lg(n+1).



Árvores balanceadas





Árvore Rubro-Negra

Acta Informatica 1, 290—306 (1972)

Symmetric Binary B-Trees: Data Structure and Maintenance Algorithms*

R. Bayer

Received January 24, 1972

Summary. A class of binary trees is described for maintaining ordered sets of data. Random insertions, deletions, and retrievals of keys can be done in time proportional to $\log N$ where N is the cardinality of the data-set. Symmetric B-Trees are a modification of B-trees described previously by Bayer and McCreight. This class of trees properly contains the balanced trees.

This paper will describe a further solution to the following well-known problem in information processing: Organize and maintain an index, i.e. an ordered set of keys or virtual addresses, used to access the elements in a set of data, in such a way that random and sequential insertions, deletions, and retrievals can be performed efficiently.

Other solutions to this problem have been described for a one-level store in [1, 3–5, 7] and for a two-level store with a pseudo-random access backup store in [2]. All these techniques use trees to represent the data sets. The class of trees to be described in this paper is a generalization of the trees described in [1, 3–5], but it is not comparable with the BB-trees described in [7]. The following technique is suitable for a one-level store.

Rudolf Bayer
Computer scientist

Rudolf Bayer	
Born	May 7, 1939 (age 75)
Nationality	German
Institutions	Technical University Munich
Alma mater	University of Illinois at Urbana–Champaign
Thesis	*Automorphism Groups and Quotients of Strongly Connected Automata and Monadic Algebras* (1966)
Doctoral advisor	Franz Edward Hohn[1]
Known for	B-tree UB-tree red-black tree
Notable awards	Cross of Merit, First class (1999), SIGMOD Edgar F. Codd Innovations Award (2001)

Árvore Rubro-Negra

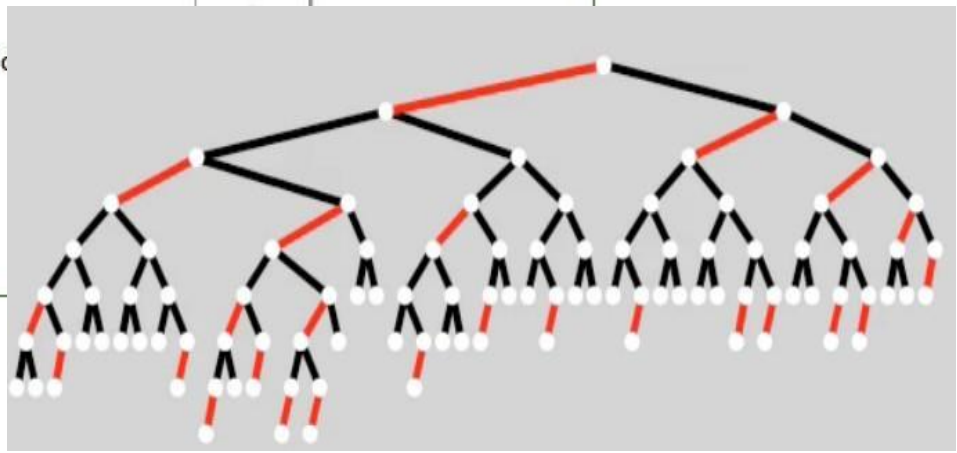
Somente em 1978, Leo Guibas e Robert Sedgewick, atribuiram a 'coloração' na árvore.

A dichromatic framework for balanced trees

Authors: Leo J. Guibas
Robert Sedgewick

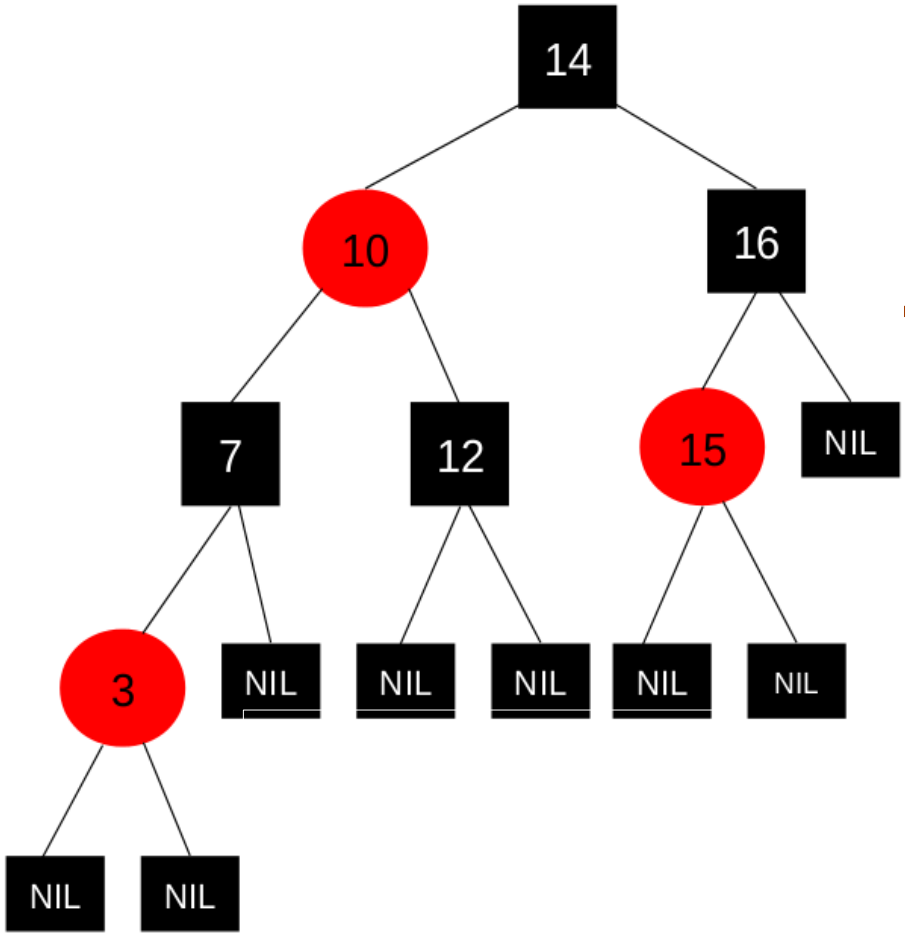
1978 Article

Published in:
· Proceeding
SFCS '78 Proceedings of the 19th Annual Symposium on Foundati
Computer Science
Pages 8-21
IEEE Computer Society Washington, DC, USA ©1978
table of contents doi>10.1109/SFCS.1978.3



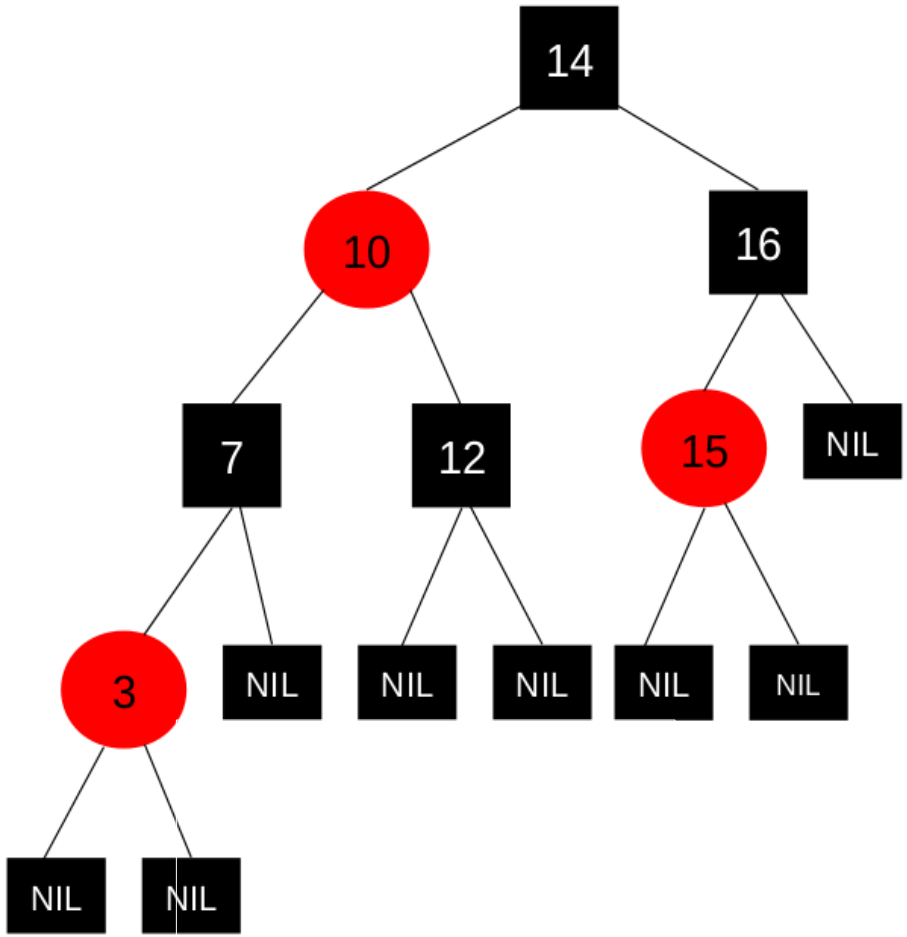
A cor "vermelho" foi escolhida porque era a mais bonita produzida pela impressora laser a cores disponíveis para os autores, enquanto trabalhavam na Xerox PARC.

Árvore Rubro-Negra



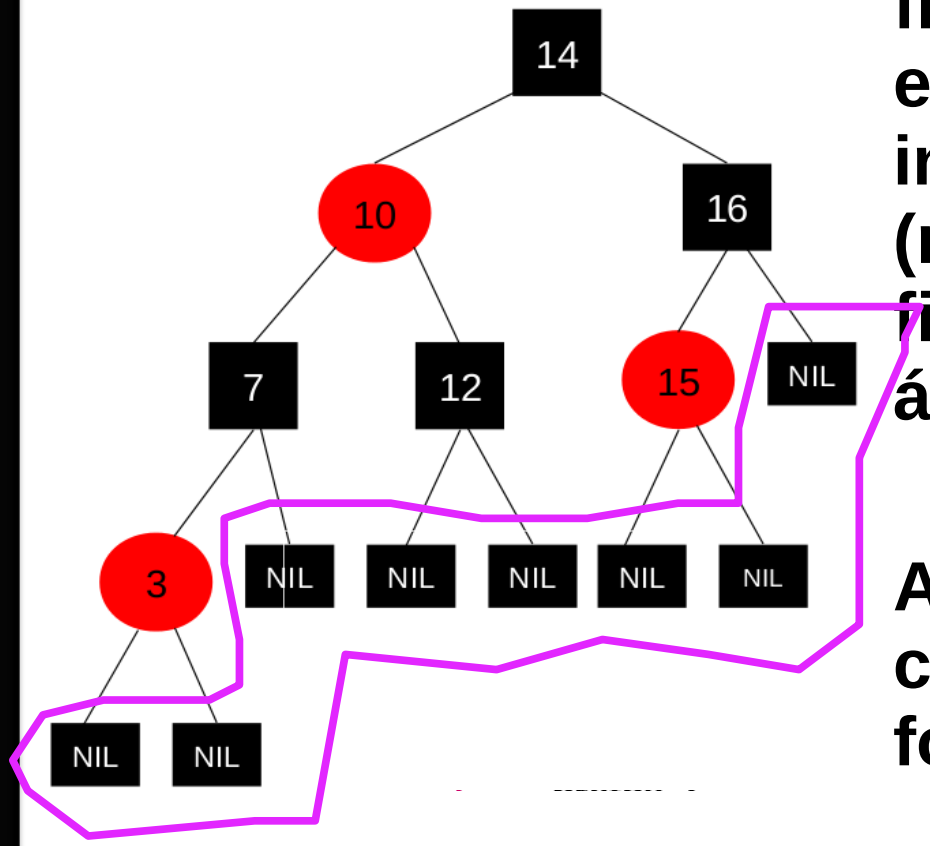
- As ARN possuem um bit extra para armazenar a cor de cada nó, que pode ser VERMELHO ou PRETO.

Árvore Rubro-Negra




```
struct cel {
    int chave;
    int conteudo;
    struct cel *esq;
    struct cel *dir;
    bool cor;
};
```

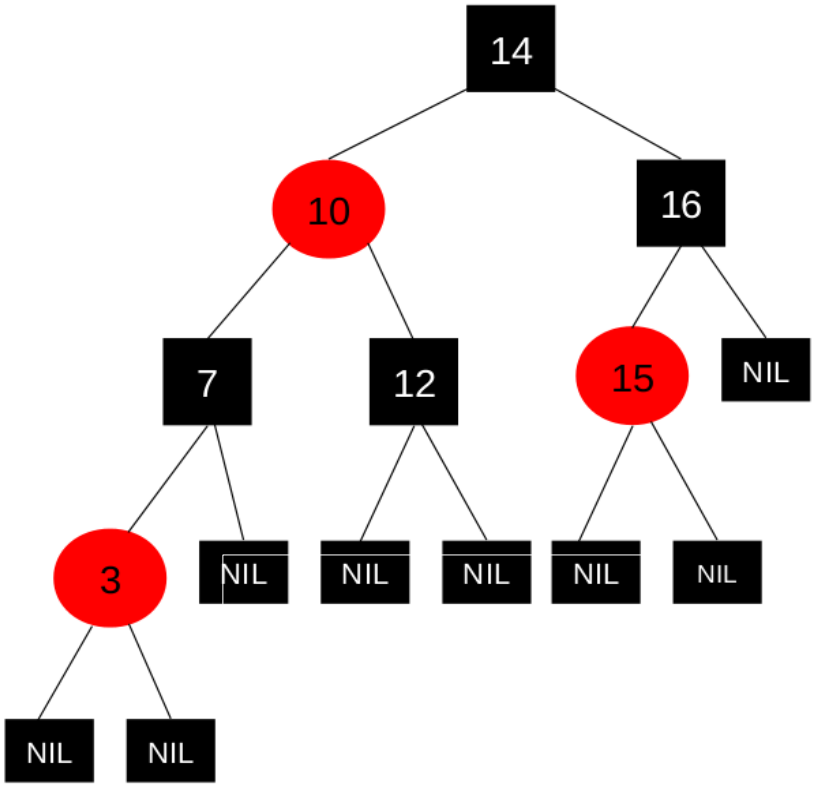
Propriedades de Árvore Rubro-Negra



- Quando um nó não possui um filho (esquerdo ou direito) então podemos supor que ao invés de apontar para nulo (nil), ele aponta para um nó fictício, que será uma folha da árvore.

Assim, todos os nós internos contêm chaves e todas as folhas são nós fictícios.

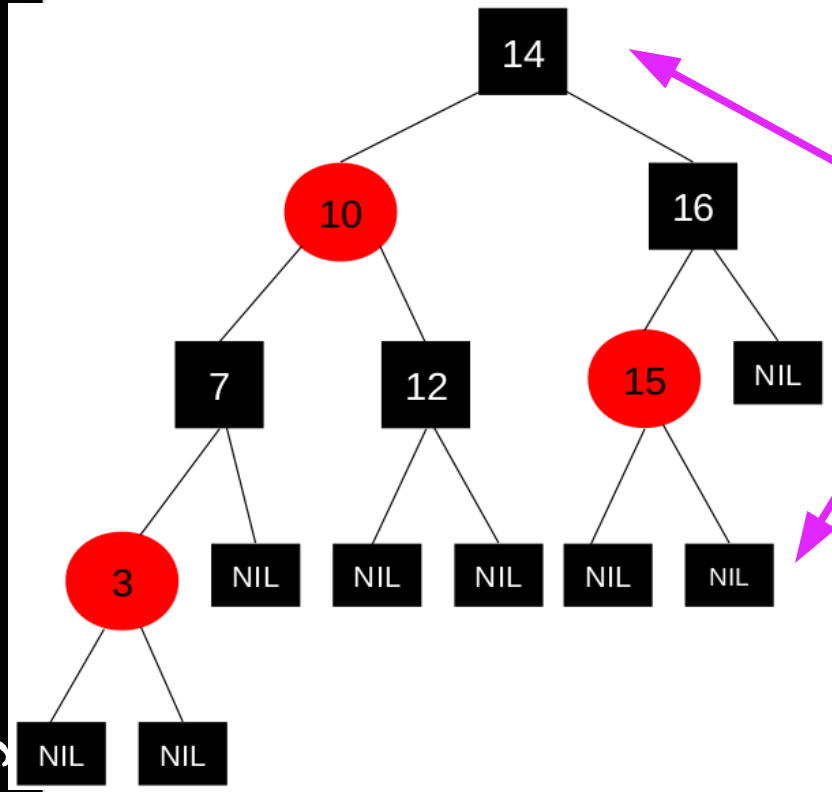
Propriedades de Árvore Rubro-Negra



As propriedades da árvore rubro- negra são:

- 1- Todo nó da árvore ou é vermelho ou é preto
- 2A raiz e as folhas (nil) são pretas
- 3Se um nó é vermelho, então seus filhos são pretos
- 4Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.

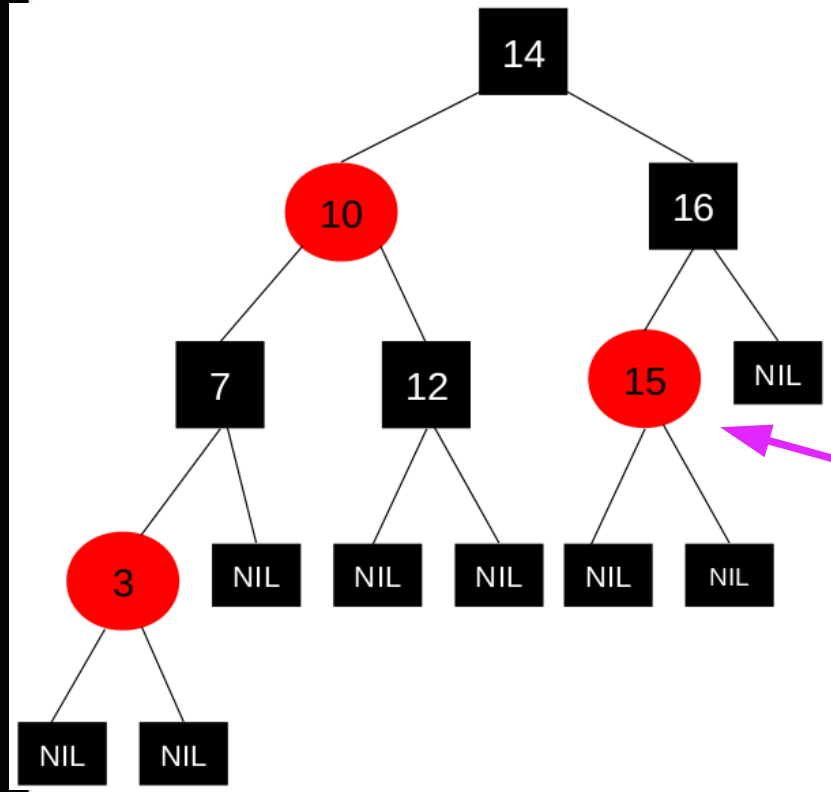
Propriedades de Árvore Rubro-Negra



As propriedades da árvore rubro- negra são:

- 1Todo nó da árvore ou é vermelho ou é preto
- 2A raiz e as folhas (nil) são pretas
- 3Se um nó é vermelho, então seus filhos são pretos
- 4Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.

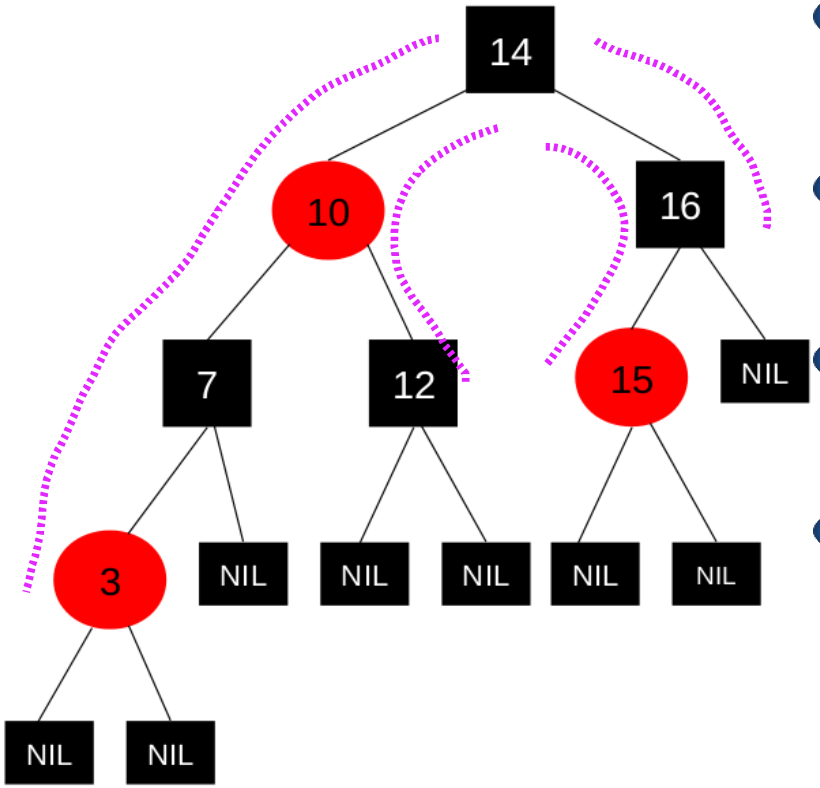
Propriedades de Árvore Rubro-Negra



As propriedades da árvore rubro- negra são:

- 1Todo nó da árvore ou é vermelho ou é preto
- 2A raiz e as folhas (nil) são pretas
- 3- Se um nó é vermelho, então seus filhos são pretos
- 4- Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.

Propriedades de Árvore Rubro-Negra

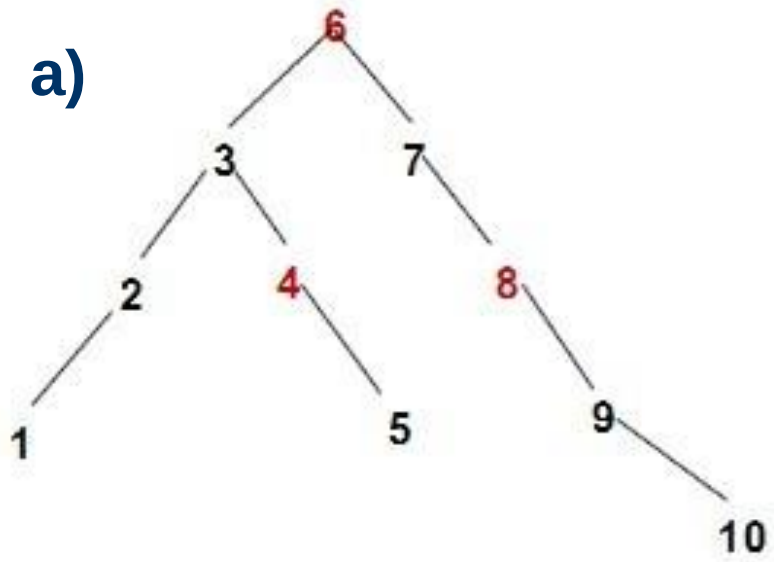


As propriedades da árvore rubro- negra são:

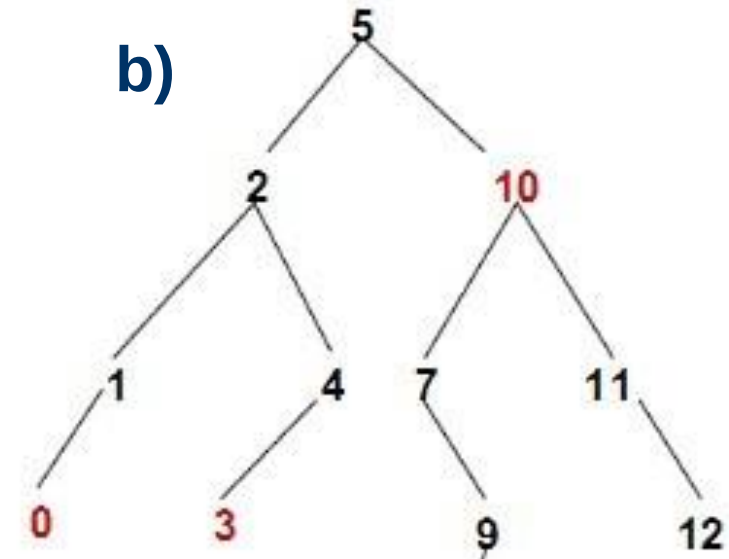
- 1Todo nó da árvore ou é vermelho ou é preto
- 2A raiz e as folhas (nil) são pretas
- 3Se um nó é vermelho, então seus filhos são pretos
- 4- Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.

Quais das árvores abaixo são rubro-negras?

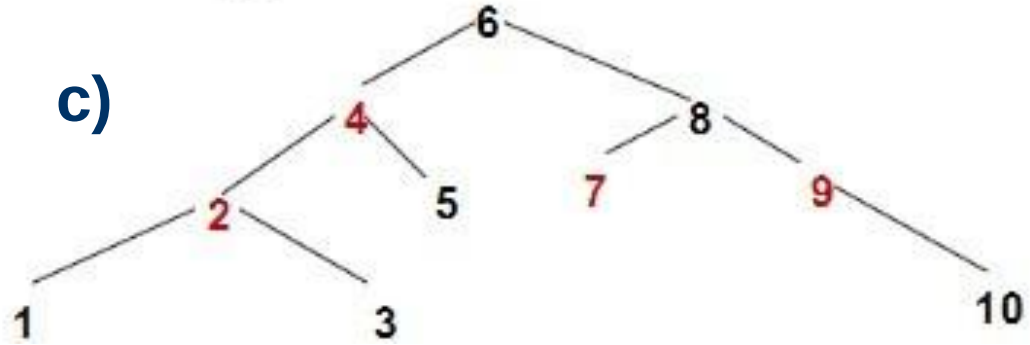
a)



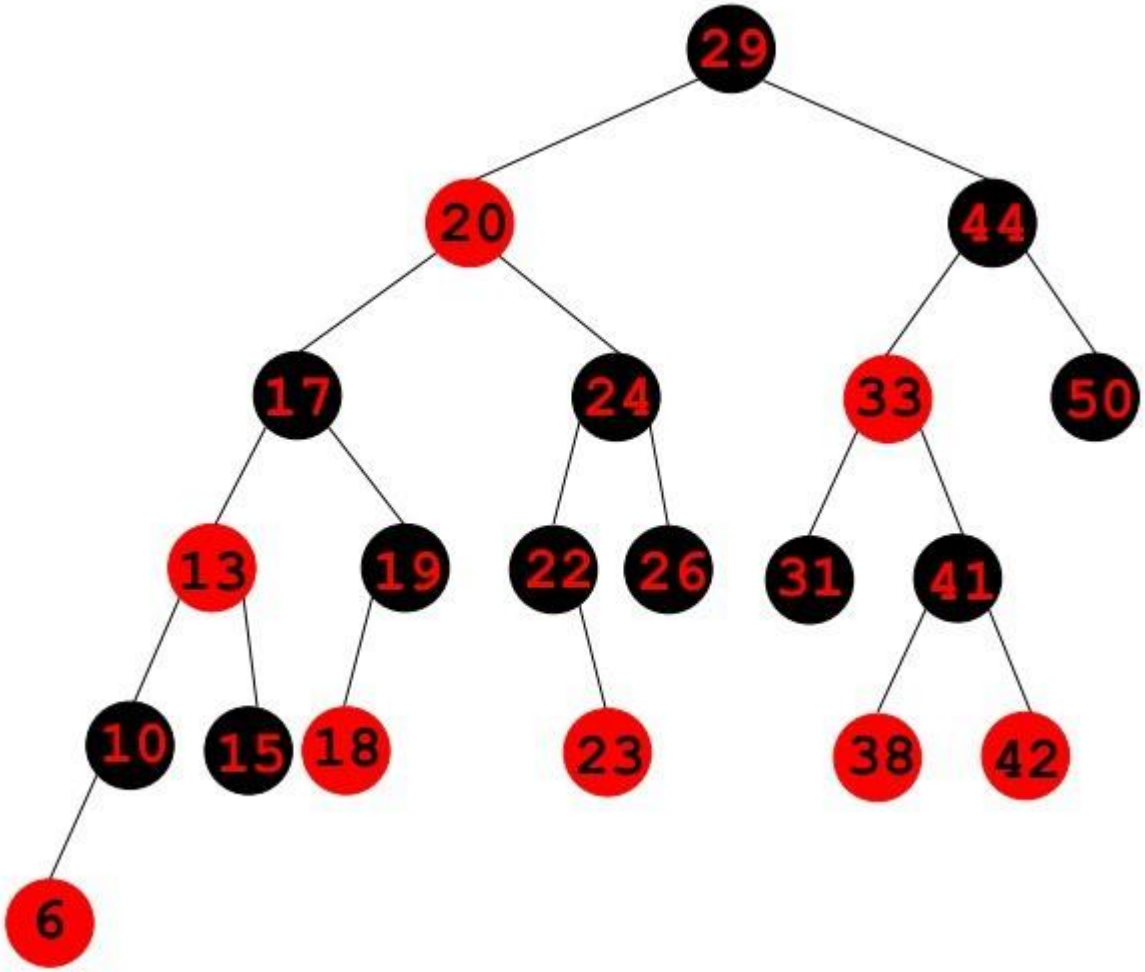
b)



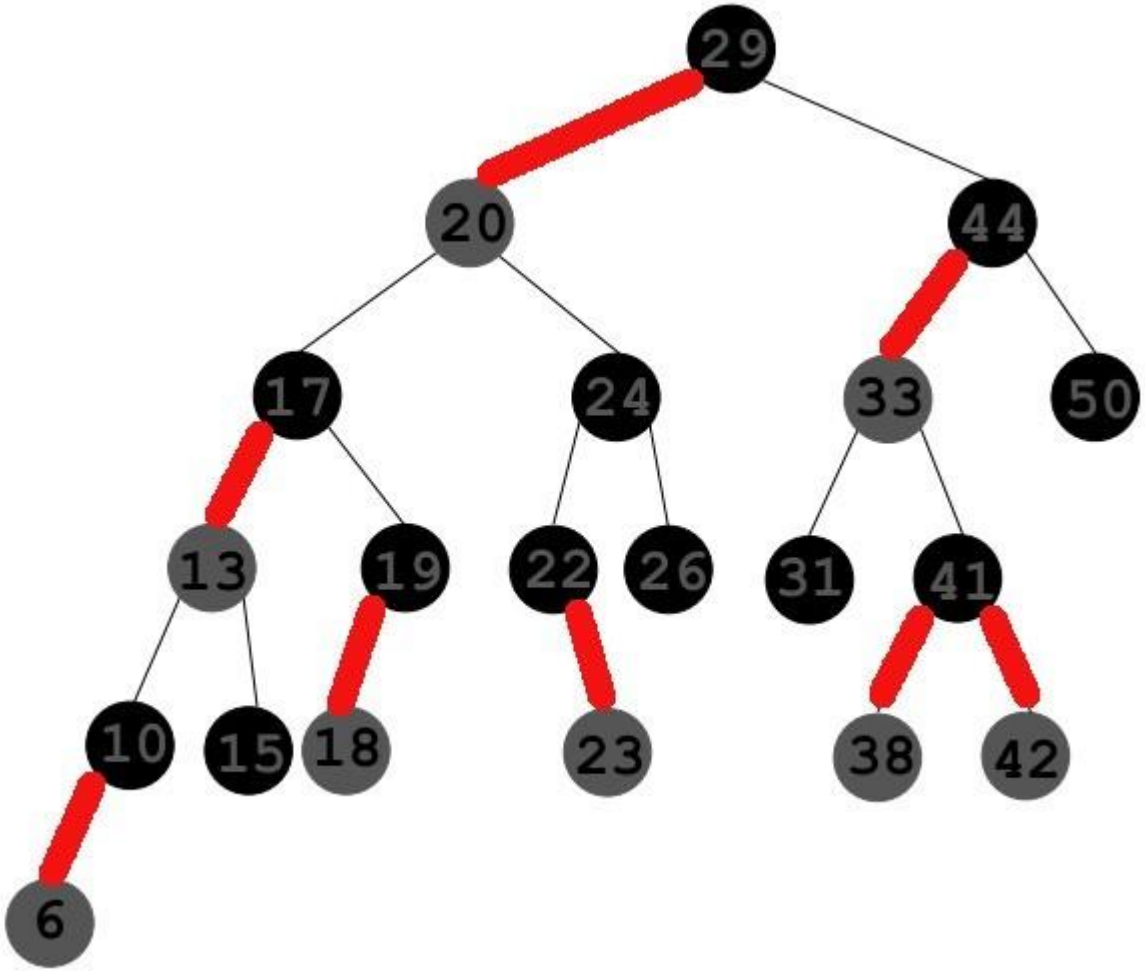
c)



Exemplo de Árvore Rubro-Negra



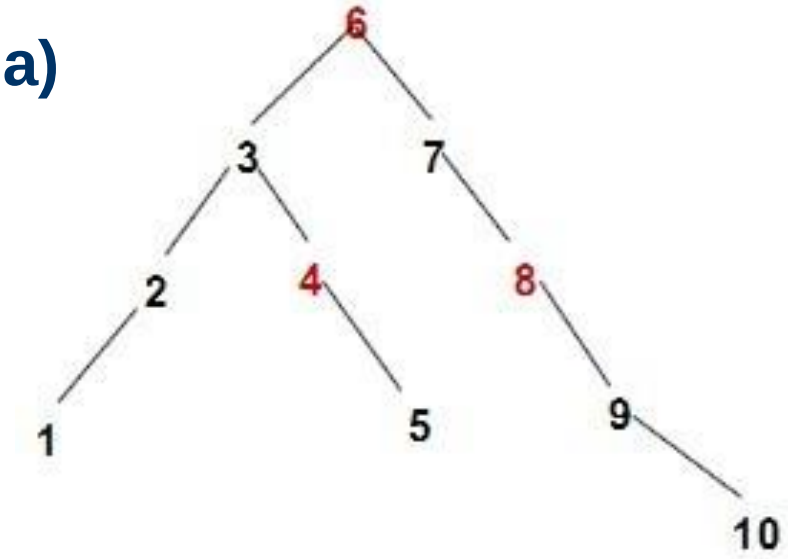
Exemplo de Árvore Rubro-Negra



Quais das árvores abaixo são rubro-negras?

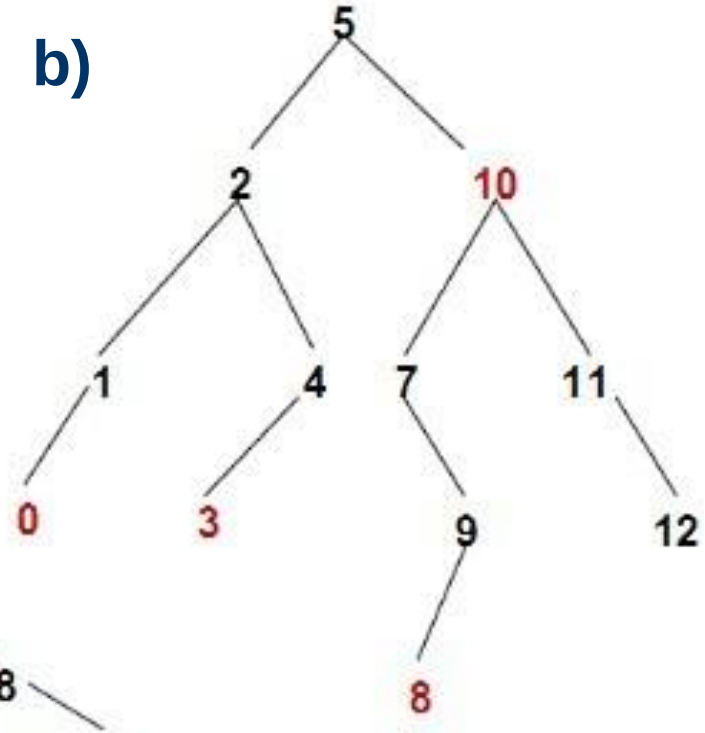
Raiz vermelha!
Caminhos da raiz as folhas devem ter o mesmo número de pretos

a)



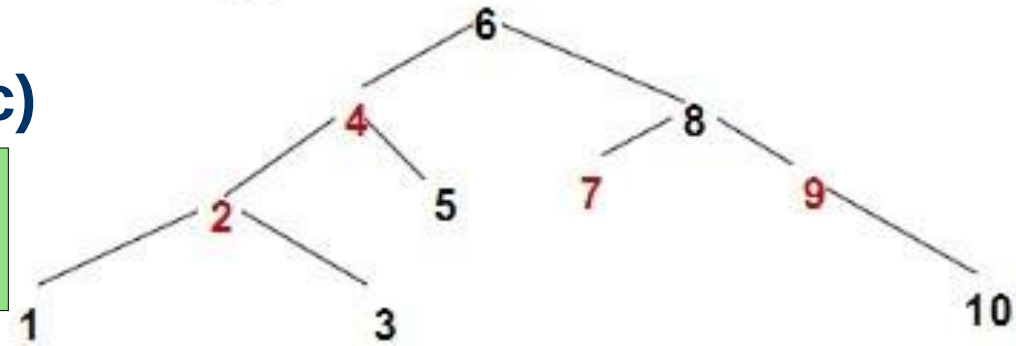
ok

b)



Não deve existir
Dois nós
Vermelhos consecutivos

c)

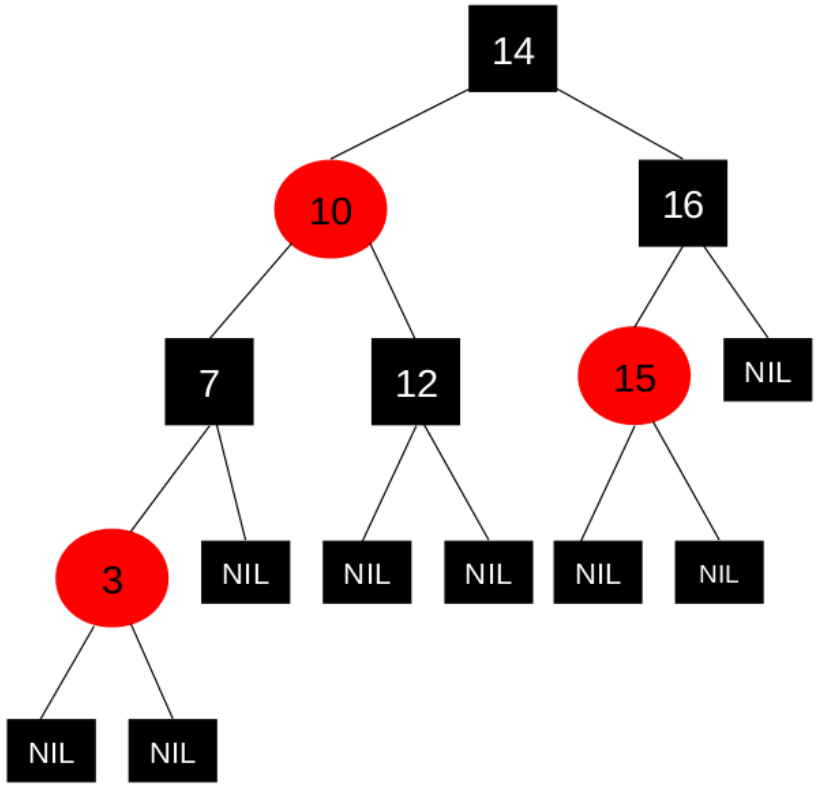


Nó vermelho deve ter filhos pretos!
Caminhos da raiz as folhas devem ter o mesmo número de pretos

Altura de uma Árvore Rubro-Negra

- **As ARNs com n nós/chaves internas tem altura, no máximo, igual a $2 \lg(n+1) = O(\log(n))$**

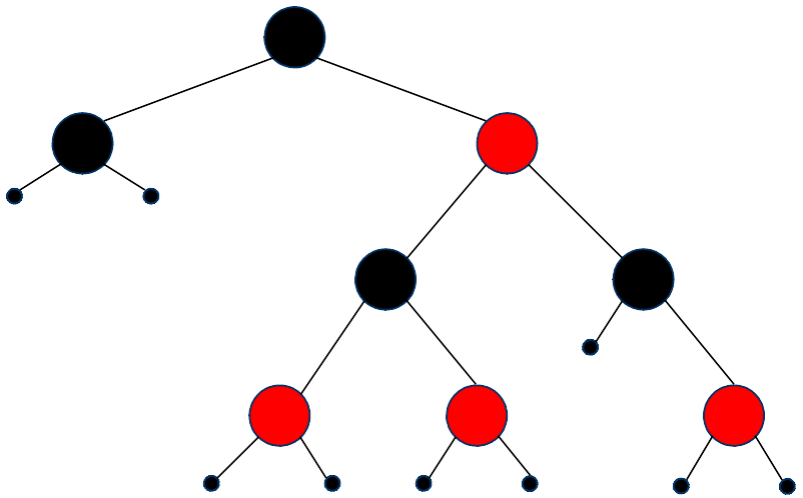
A prova é por indução. Ver detalhes no livro de Cormen et al.



Altura de uma Árvore Rubro-Negra

Intuição

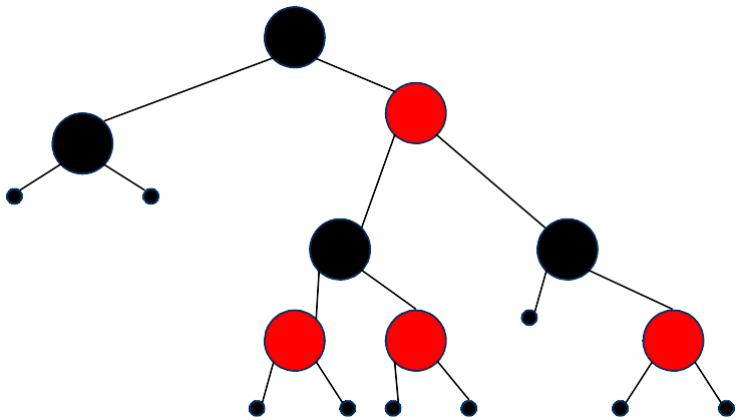
- **Junte os nós vermelhos aos seus pais pretos.**



Altura de uma Árvore Rubro-Negra

Intuição

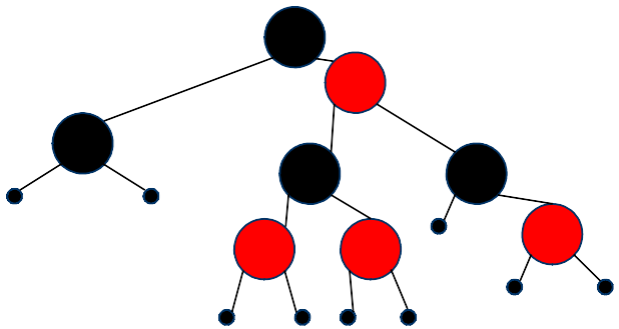
- **Junte os nós vermelhos aos seus pais pretos.**



Altura de uma Árvore Rubro-Negra

Intuição

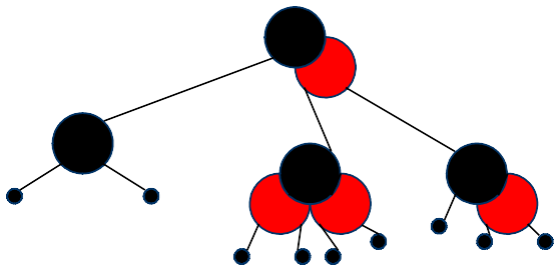
- **Junte os nós vermelhos aos seus pais pretos.**



Altura de uma Árvore Rubro-Negra

Intuição

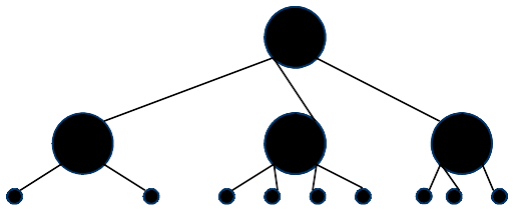
- **Junte os nós vermelhos aos seus pais pretos.**



Altura de uma Árvore Rubro-Negra

Intuição

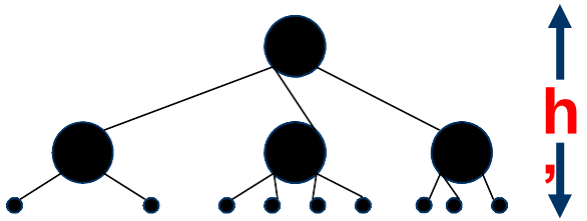
- **Junte os nós vermelhos aos seus pais pretos.**



Altura de uma Árvore Rubro-Negra

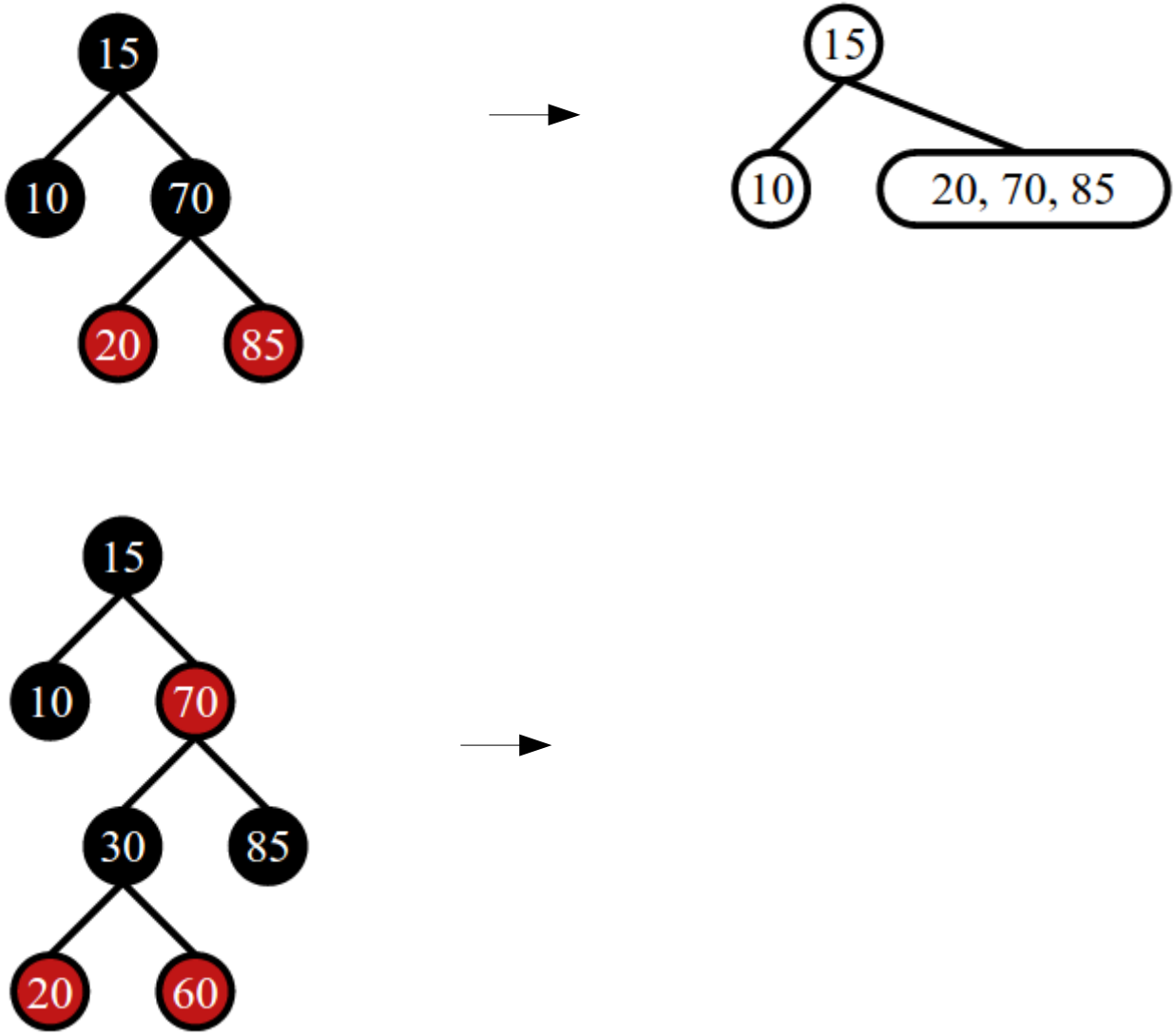
Intuição

- **Junte os nós vermelhos aos seus pais pretos.**

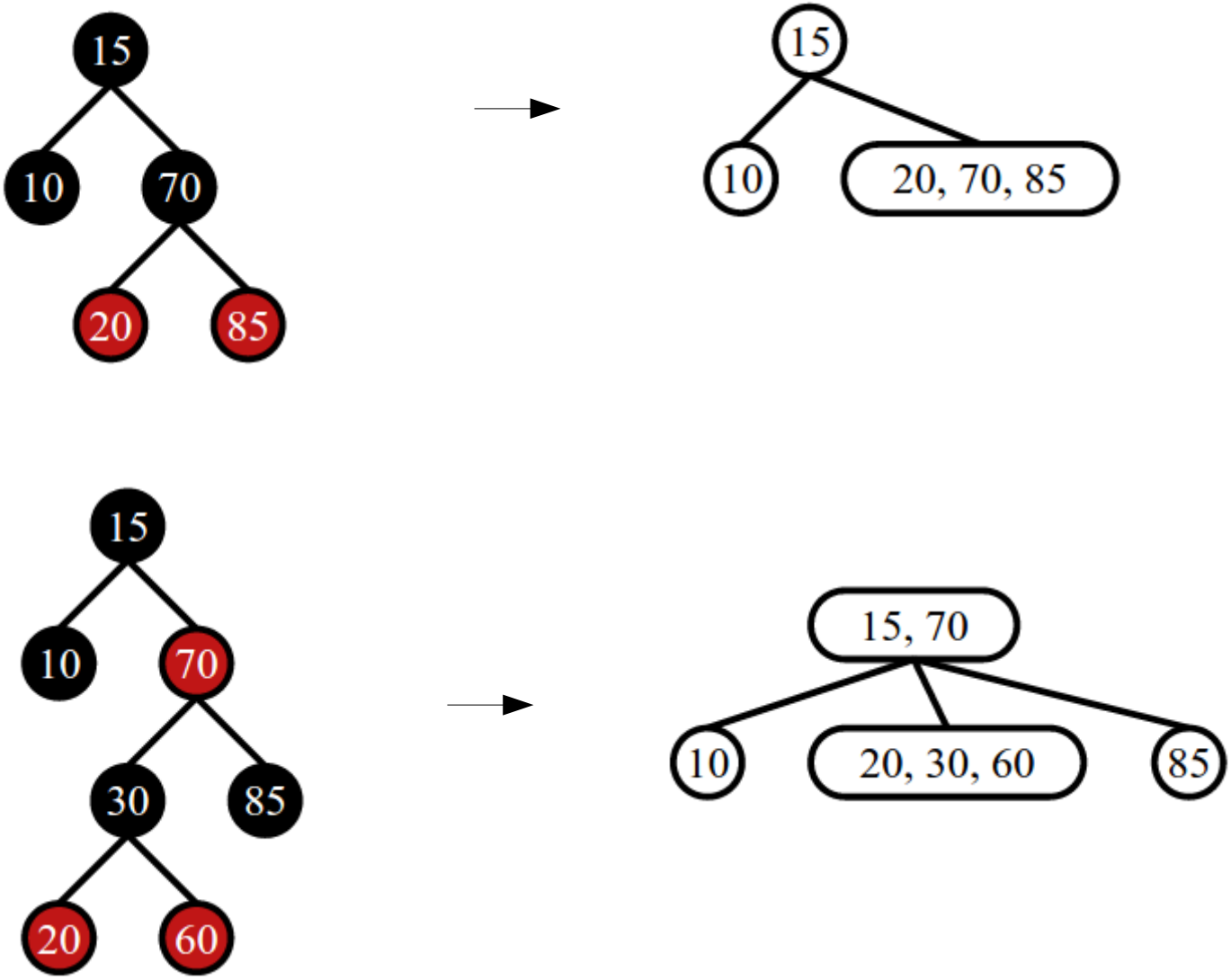


Tal processo produz uma árvore em que cada nó possui 2, 3 ou 4 filhos (árvore 2-3-4)

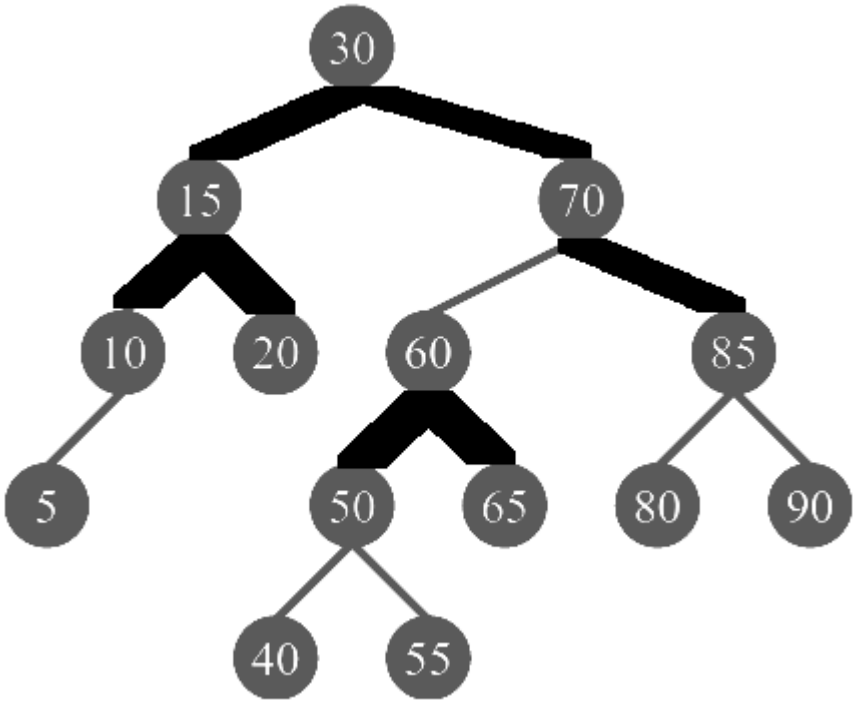
Árvore 2-3-4



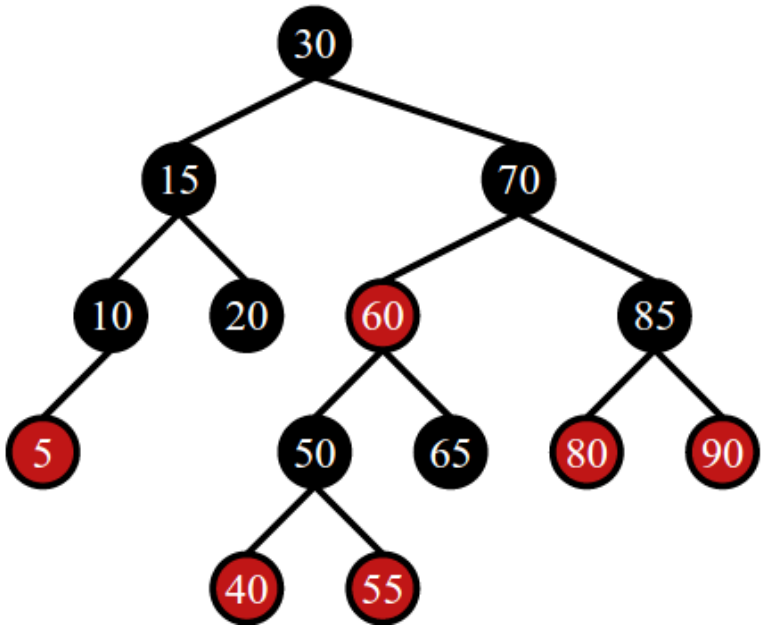
Árvore 2-3-4



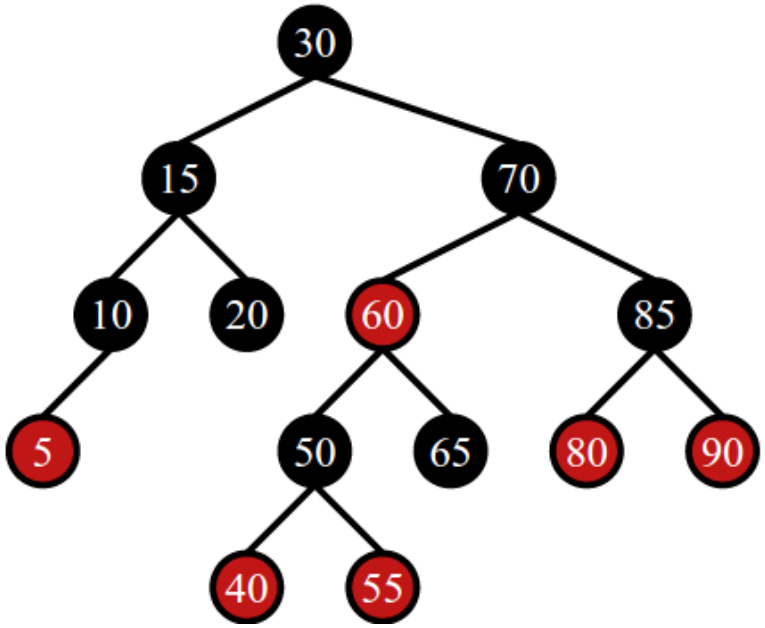
Atividade: Árvore 2-3-4

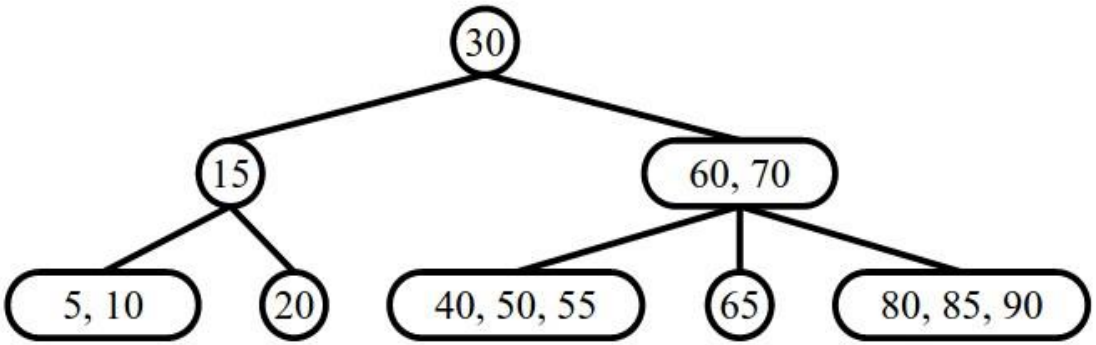


Atividade: Árvore 2-3-4



Atividade: Árvore 2-3-4

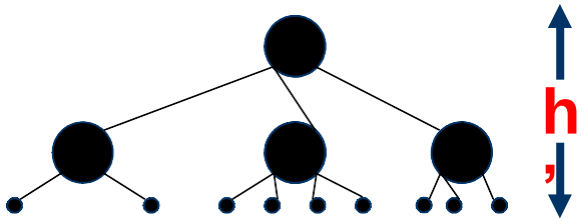




Altura de uma Árvore Rubro-Negra

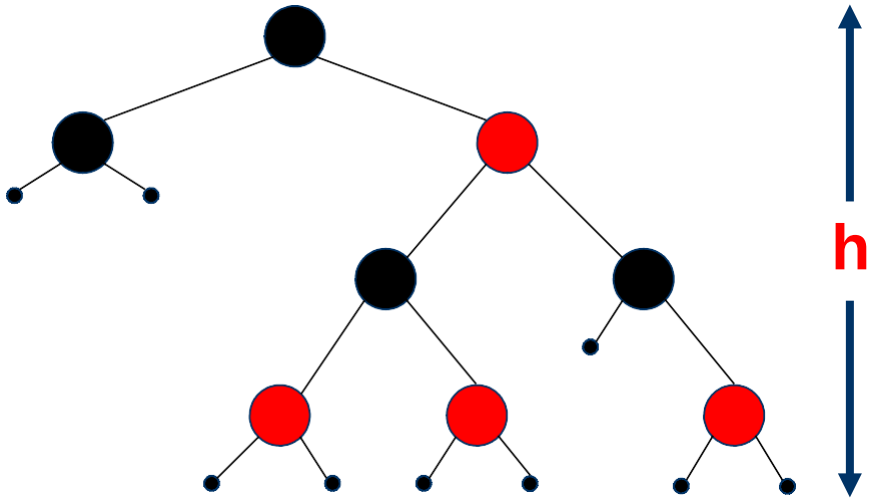
Intuição

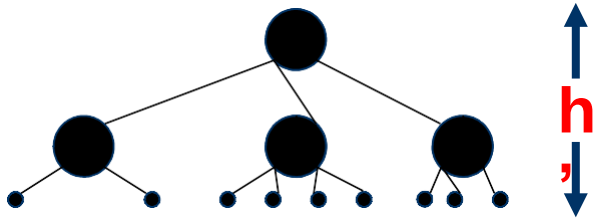
- **Junte os nós vermelhos aos seus pais pretos.**



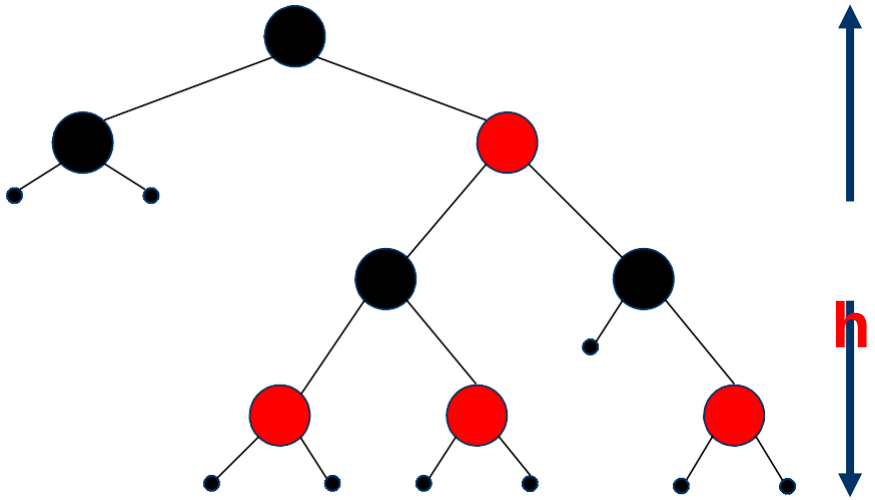
Tal processo produz uma árvore em que cada nó possui 2, 3 ou 4 filhos (árvore 2-3-4)

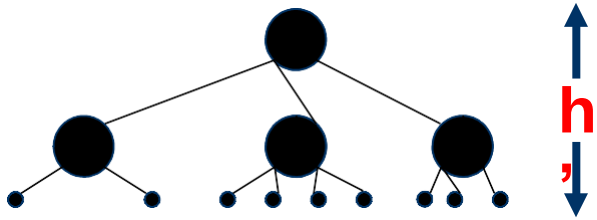
Altura de uma Árvore Rubro-Negra





Altura de uma Árvore Rubro-Negra

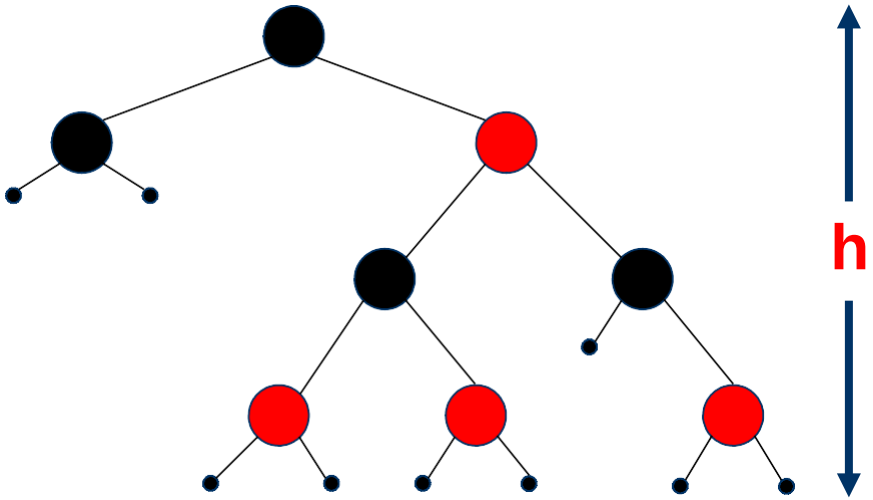


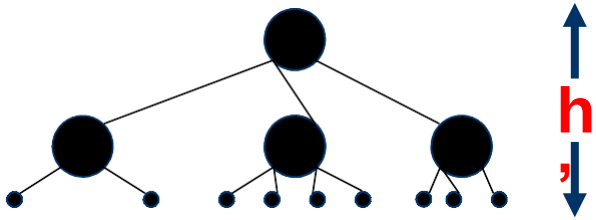


$$h' \geq \frac{1}{2}h$$

← Já que no máximo metade nos nós de qualquer caminho são vermelhos.

Altura de uma Árvore Rubro-Negra

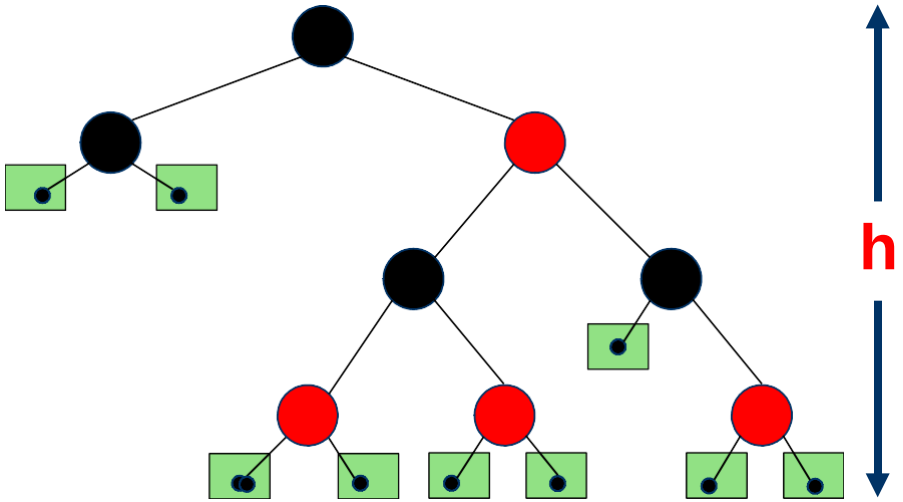


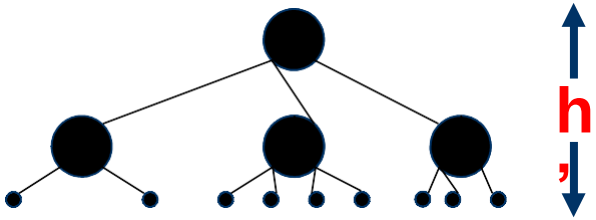


$$h' \geq \frac{1}{2}h$$

Número de folhas

Altura de uma Árvore Rubro-Negra

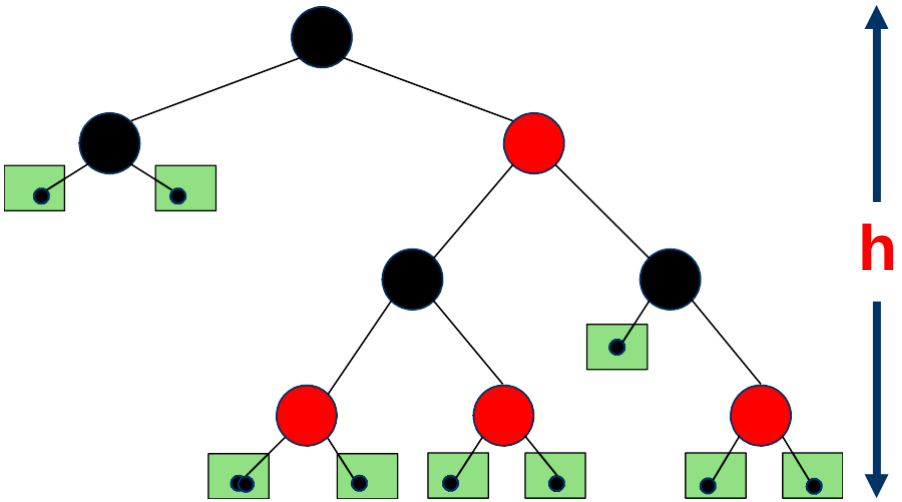


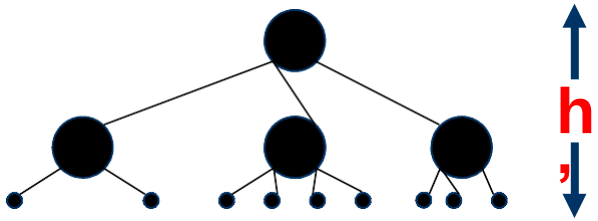


$$h' \geq \frac{1}{2}h$$

Número de folhas = n+1

Altura de uma Árvore Rubro-Negra





$$h' \geq \frac{1}{2}h$$

Número de folhas = n+1

$$n + 1 \geq 2^{h'}$$

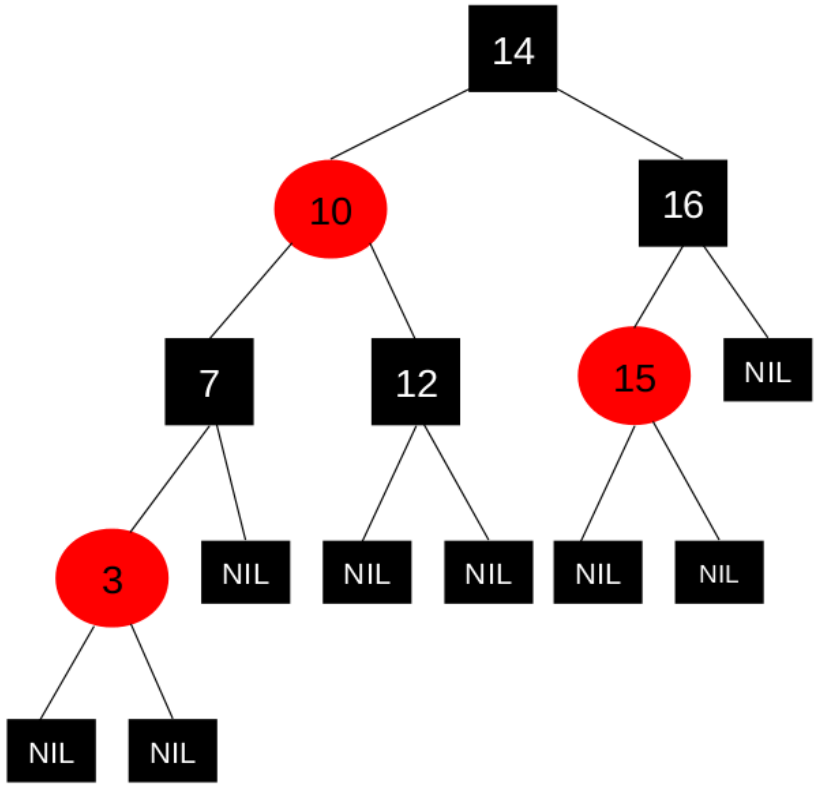
$$\lg(n+1) \geq h' \geq \frac{1}{2}h$$

$$h \leq 2\lg(n+1)$$

Altura de uma Árvore Rubro-Negra

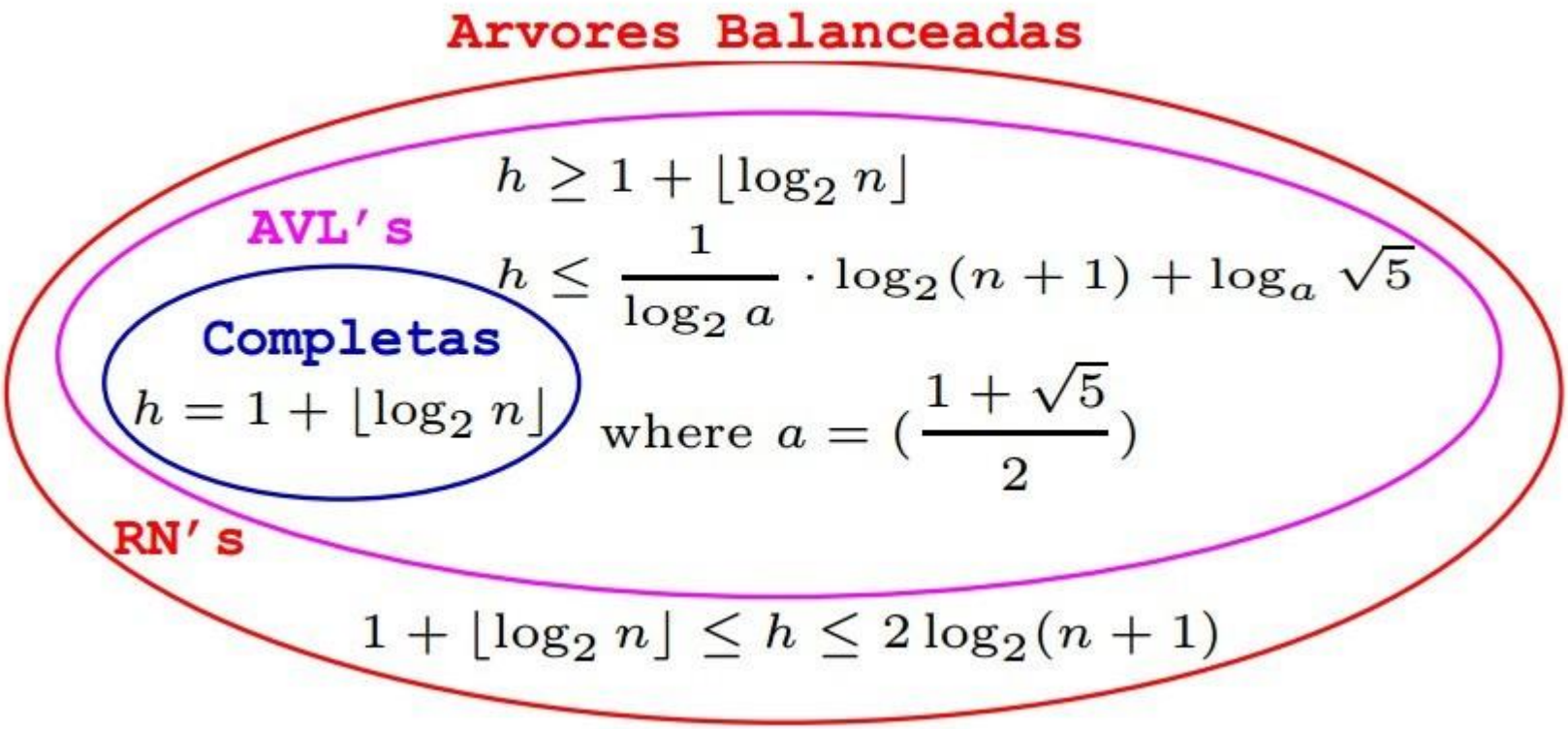
- As ARNs com n nós/chaves internas tem altura, no máximo, igual a $2 \lg(n+1) = O(\log(n))$

A prova é por indução. Ver detalhes no livro de Cormen et al.



Altura de uma Árvore Rubro-Negra

Nenhum caminho é maior do que duas vezes o comprimento de qualquer outro caminho

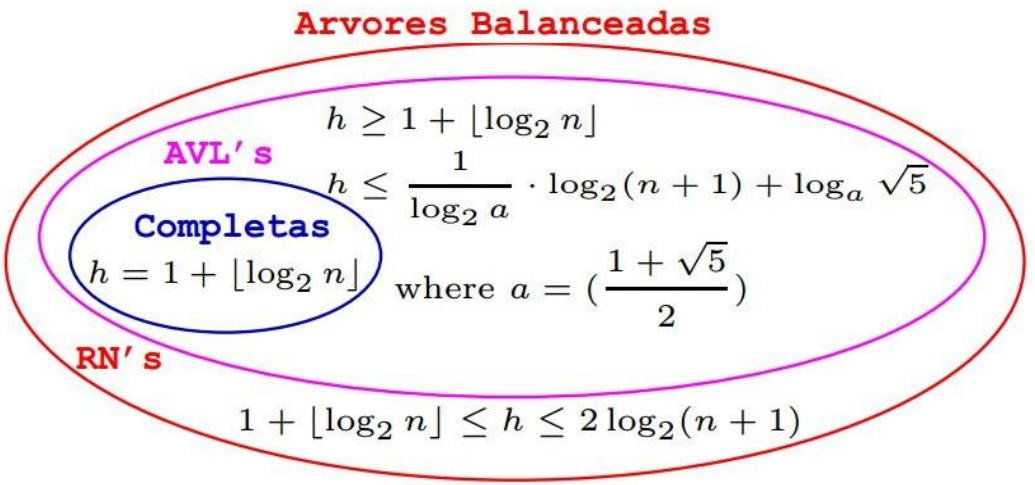


Operações em uma Árvore Rubro-Negra

- Esse resultado mostra a importância e utilidade de uma ARN, pois a busca, inserção e remoção têm complexidade de tempo de O(h) = O(log n)

- A busca em uma ARN é idêntica à busca em ABBs simples.

AVL x RN ?


Arvores Balanceadas
AVL's
Completas
$h = 1 + \lfloor \log_2 n \rfloor$
RN's
$h \geq 1 + \lfloor \log_2 n \rfloor$
$h \leq \frac{1}{\log_2 a} \cdot \log_2(n+1) + \log_a \sqrt{5}$
where $a = (\frac{1+\sqrt{5}}{2})$
$1 + \lfloor \log_2 n \rfloor \leq h \leq 2 \log_2(n+1)$

AVL x RN

- Altura de uma árvore AVL:
- ~1.44lg(n) Altura de uma árvore RN: ~2lg(n)
- Inserções de elementos: Na prática menos rotações são realizadas nas árvores RN.
- As rotações em árvores AVL são mais difíceis de implementar.

```c
#include <stdio.h>
#include <stdbool.h>


struct celAVL {
    int chave;              // 4b
    int conteudo;           // 4b
    struct celAVL *esq;     // 8b
    struct celAVL *dir;     // 8b
    int altura;             // 4b
};
typedef struct celAVL noAVL;

struct celRB {
    int chave;              // 4b
    int conteudo;           // 4b
    struct celRB *esq;      // 8b
    struct celRB *dir;      // 8b
    bool cor;               // 1b

};
typedef struct celRB noRB;


int main(int argc, char *argv[])
{
    printf("\nAVL: %zu", sizeof(noAVL));
    printf("\nRB : %zu", sizeof(noRB));
}
```

```
AVL: 32
RB : 32
```

Alinhado para endereçamento de memória

```
struct Test
{
   char AA;
   int BB;
   char CC;
};
```

```
|   1   |   2   |   3   |   4   |

| AA(1) | pad.................. |
| BB(1) | BB(2) | BB(3) | BB(4) |
| CC(1) | pad.................. |
```

12 bytes

```
#pragma pack(1)
```

```
|   1   |

| AA(1) |
| BB(1) |
| BB(2) |
| BB(3) |
| BB(4) |
| CC(1) |
```

6 bytes

```
#pragma pack(2)
```

```
|   1   |   2   |

| AA(1) | pad.. |
| BB(1) | BB(2) |
| BB(3) | BB(4) |
| CC(1) | pad.. |
```

8 bytes

```c
#include <stdio.h>
#include <stdbool.h>
#pragma pack(1)

struct celAVL {
    int chave;              // 4b
    int conteudo;           // 4b
    struct celAVL *esq;     // 8b
    struct celAVL *dir;     // 8b
    int altura;             // 4b
};
typedef struct celAVL noAVL;

struct celRB {
    int chave;              // 4b
    int conteudo;           // 4b
    struct celRB *esq;      // 8b
    struct celRB *dir;      // 8b
    bool cor;               // 1b

};
typedef struct celRB noRB;


int main(int argc, char *argv[])
{
    printf("\nAVL: %zu", sizeof(noAVL));
    printf("\nRB : %zu", sizeof(noRB));
}
```

```
AVL: 28
RB : 25
```

```
struct celRB {
    int chave;              // 4b
    int conteudo;           // 4b
    struct celRB *esq;      // 8b
    struct celRB *dir;      // 8b
    bool cor;               // 1b

};
typedef struct celRB noRB;
```

25 bytes

```
struct celRB {
    int chave;              // 4b
    int conteudo;           // 4b
    struct celRB *esq;      // 8b
    struct celRB *dir;      // 8b
    bool cor;               // 1b

};
typedef struct celRB noRB;
```

25 bytes

```
struct celRB {
    int chave;              // 4b
    int conteudo;           // 4b
    struct celRB *esq;      // 8b
    struct celRB *dir;      // 8b


};
typedef struct celRB noRB;
```

24 bytes

Assumindo que as chaves serão todas positivas: O sinal pode indicar Red / Black

Operações nas árvore RN

Operações em uma Árvore Rubro-Negra

Inserções e remoções feitas em uma ARN podem modificar a sua estrutura.

Precisamos garantir que nenhuma das propriedades seja violada.

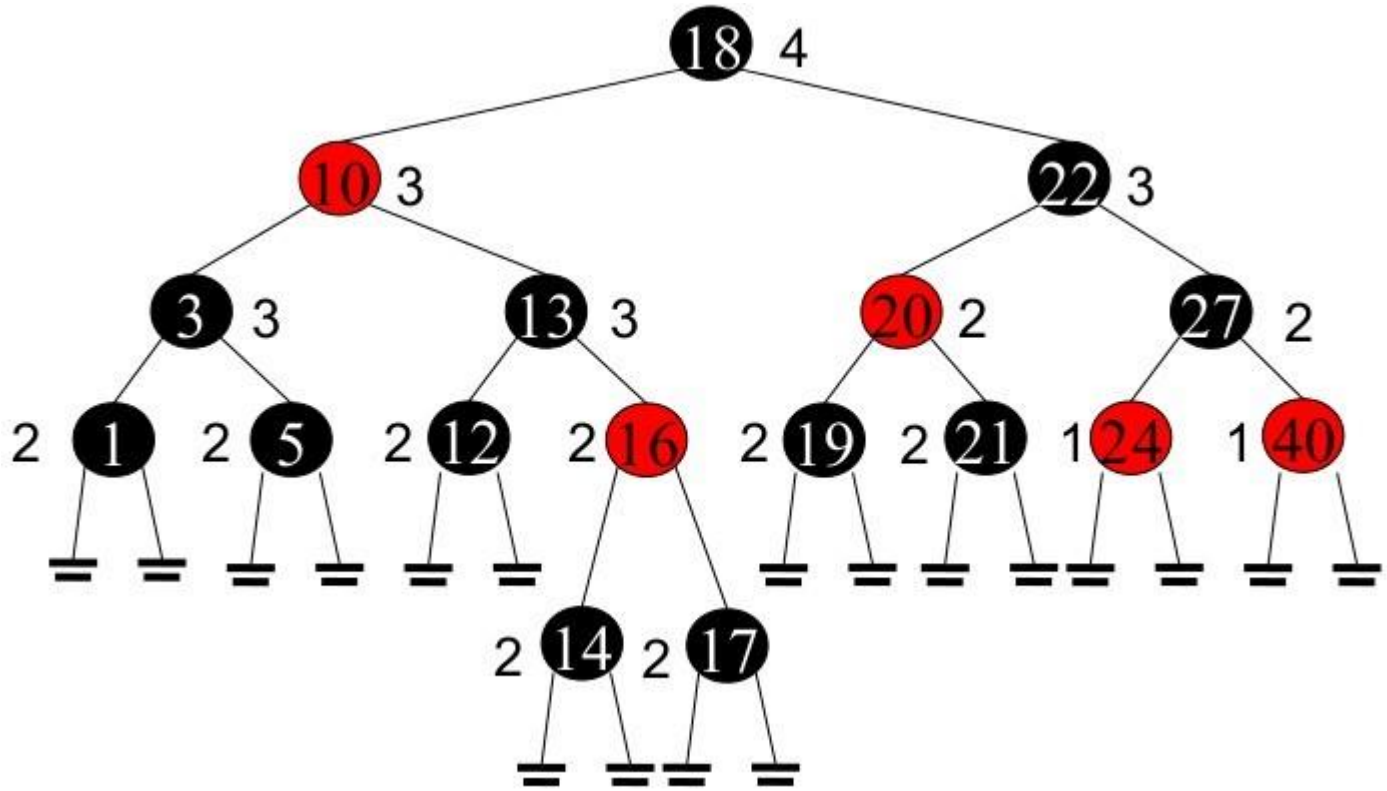
- Para isso podemos ter que mudar a estrutura da árvore
- e as cores de alguns dos nós da árvore.

A mudança da estrutura da árvore é feita por dois tipos de rotações em ramos da árvore:

- Rotação à esquerda
- Rotação à direita

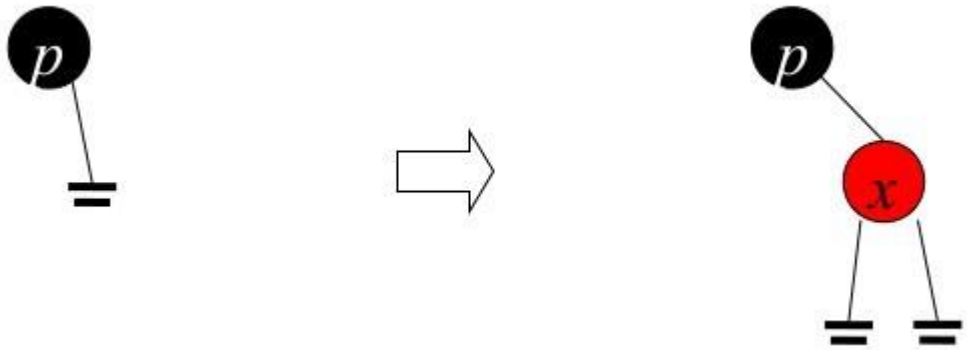
Altura Negra

É o número de nós negros encontrados até qualquer nó folha descendente.

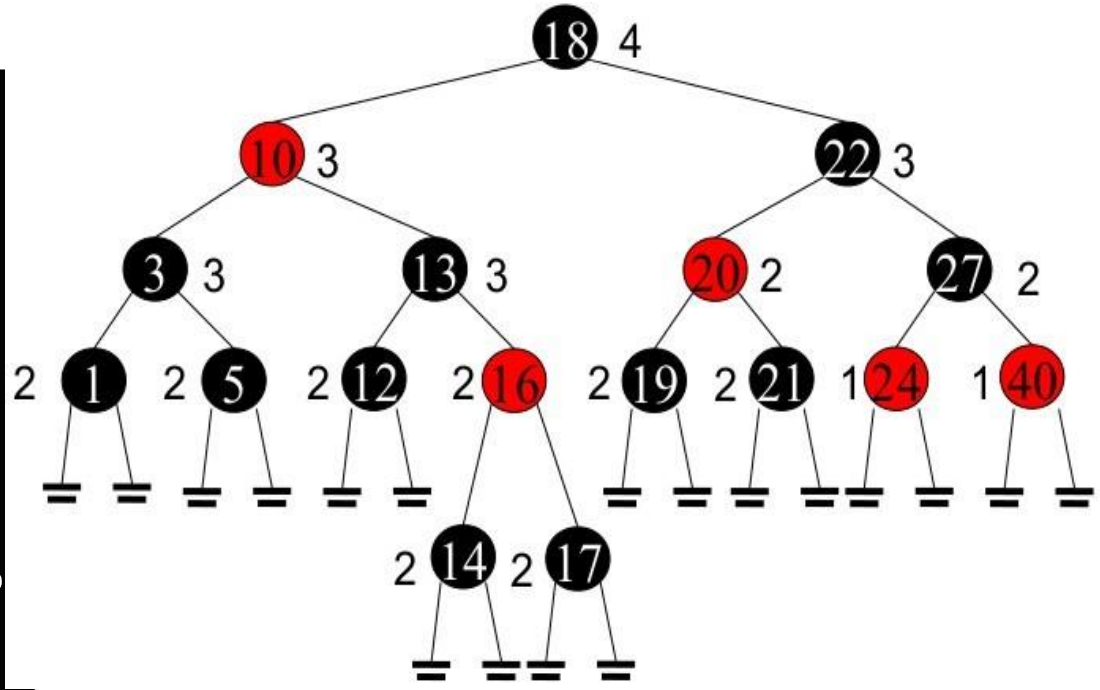


Inserção de elementos

Um nó sempre é inserido na cor vermelha, assim não altera a altura negra da árvore.



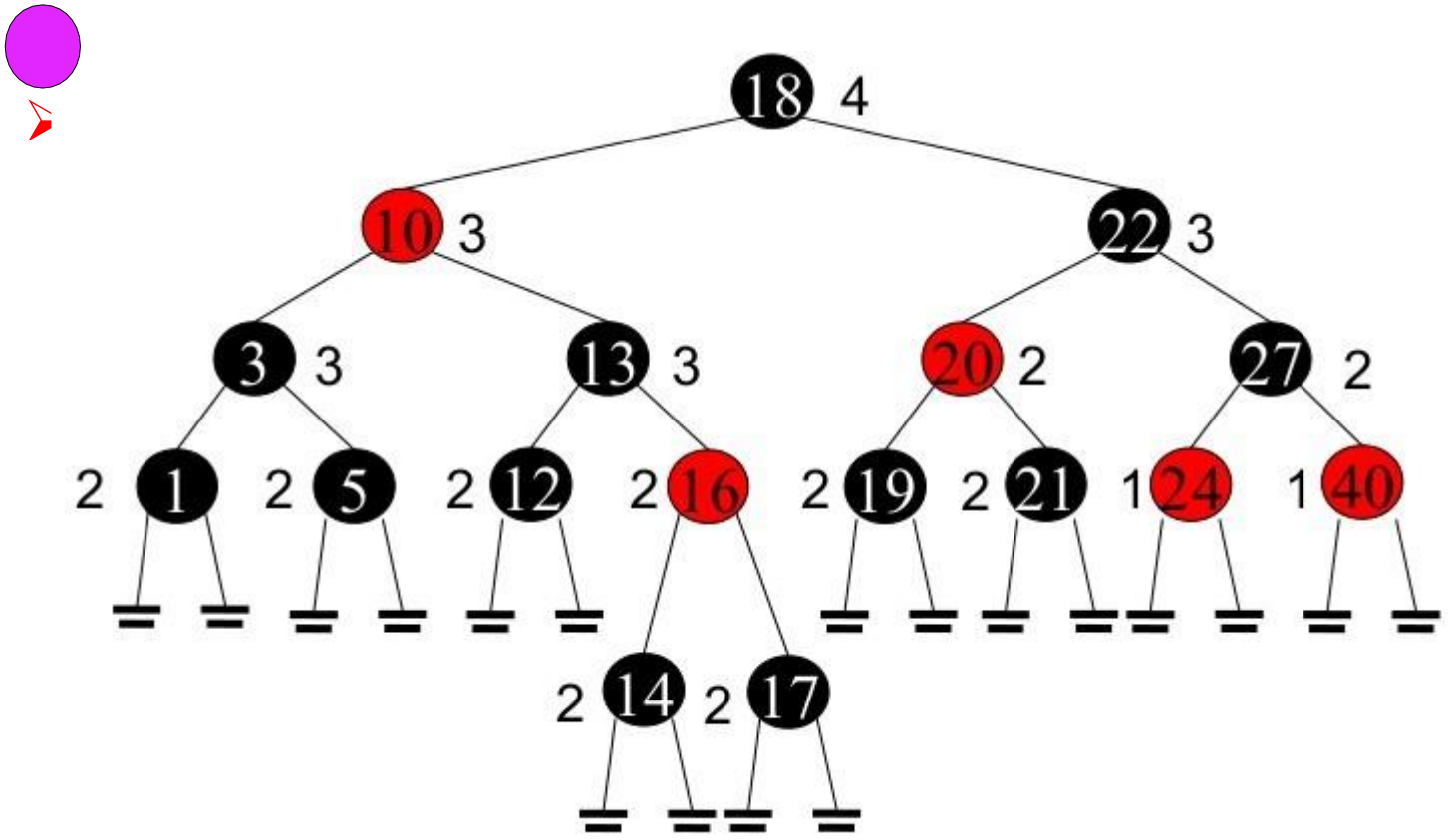
Inserção de elementos



Se o nó fosse inserido na cor preta, invalidaria a propriedade 4, pois haveria um nó preto a mais em um dos caminhos

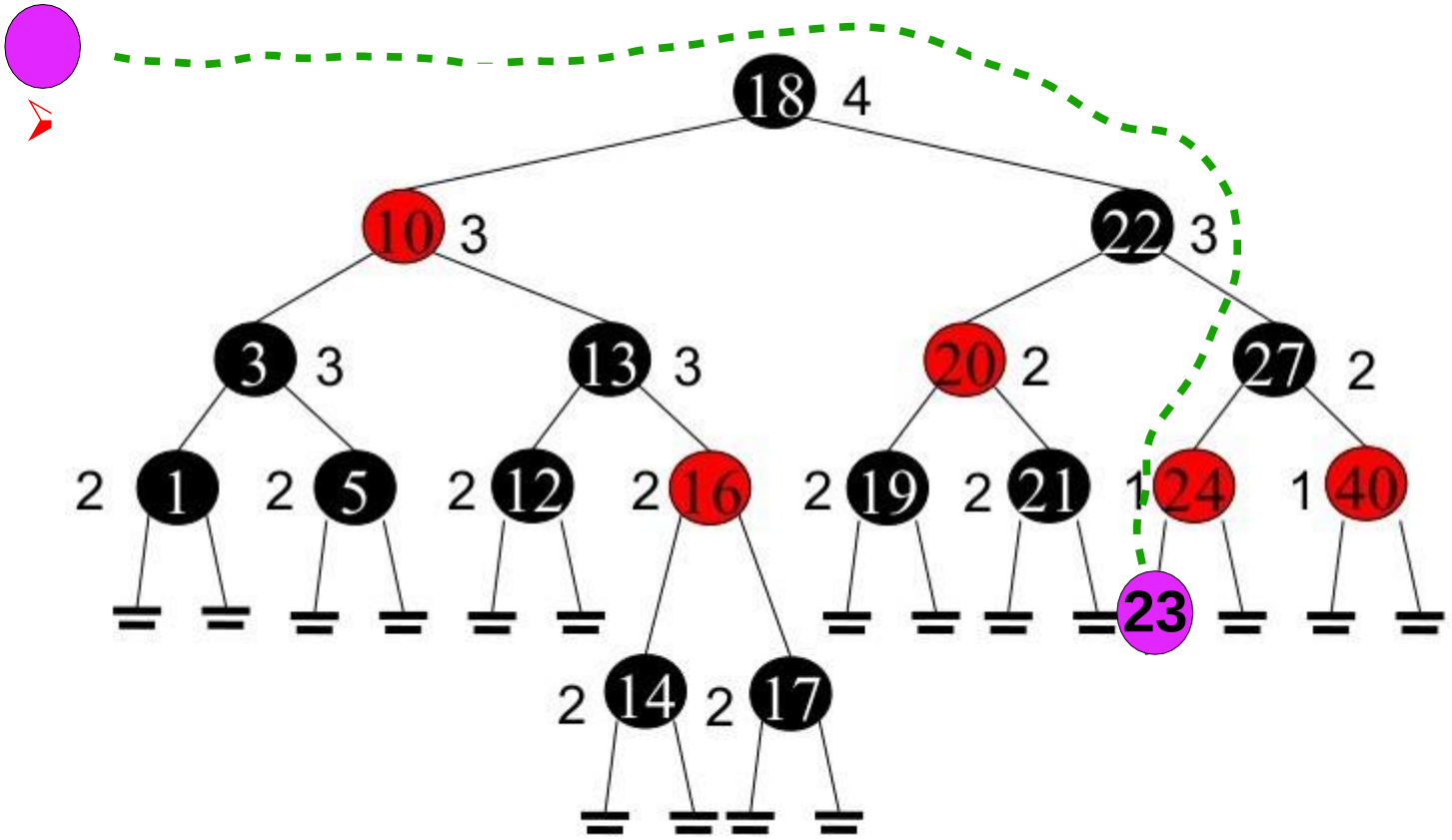
Inserção de elementos

- A inserção começa por uma busca da posição onde o nó deve ser inserido, partindo-se da raiz em direção aos nós que possuam o valor mais próximo do qual vai ser inserido.



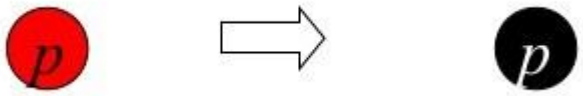
Inserção de elementos

- A inserção começa por uma busca da posição onde o nó deve ser inserido, partindo-se da raiz em direção aos nós que possuam o valor mais próximo do qual vai ser inserido.



Inserção

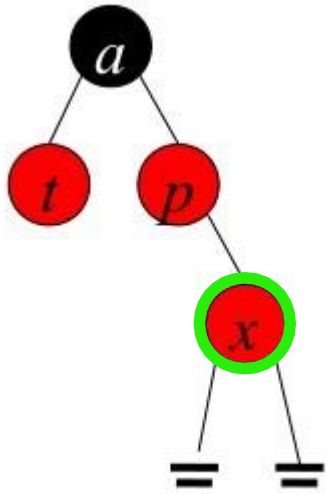
- CASO 1:
- Caso a inserção seja feita em uma árvore vazia, basta alterar a cor do nó para preto, satisfazendo assim a propriedade 2.



Inserção

CASO 2:

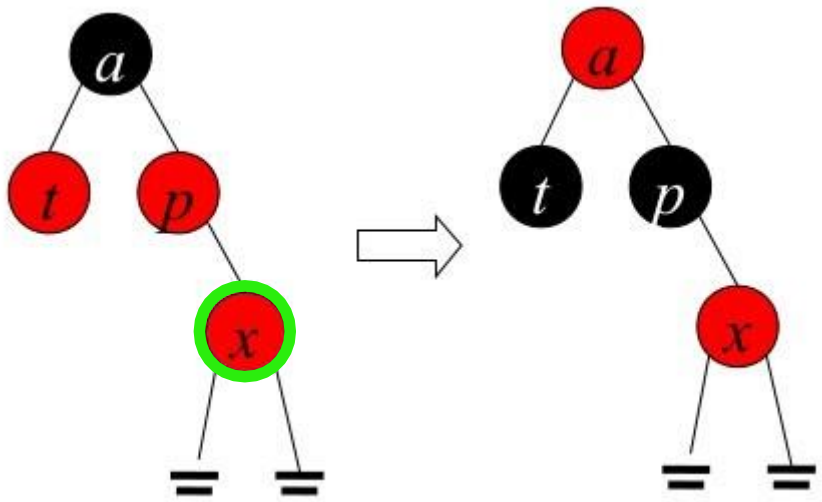
Ao inserir x, se o tio de x é vermelho, é necessário fazer a recoloração de a, t, p.



Inserção

CASO 2:

Ao inserir x, se o tio de x é vermelho, é necessário fazer a recoloração de a, t, p.

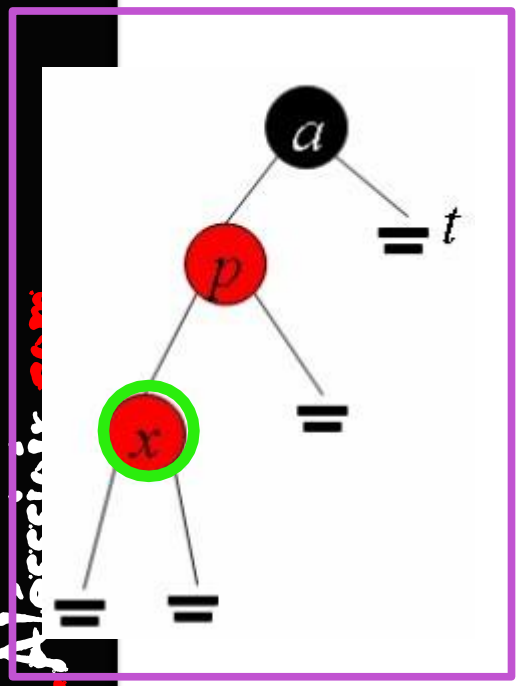


Obs1:
Se o pai de a é vermelho o rebalanceamento tem que ser feito novemente.

Obs2:
Se a é raiz, então ele deve ser preto.

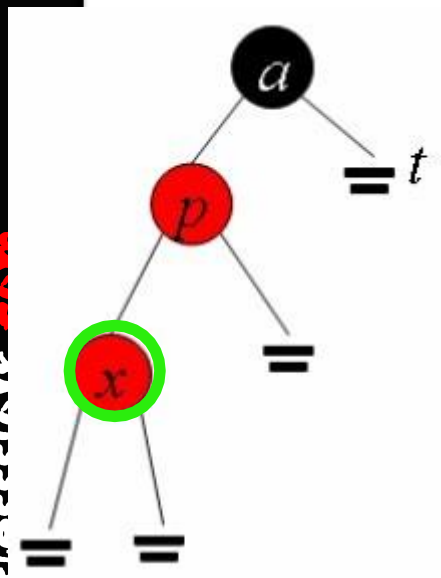
Inserção

- CASO 3:
- Suponha que o tio de x é preto.
- Nesse caso, para manter a propriedade 3 é preciso fazer rotações envolvendo x, p, t, a.

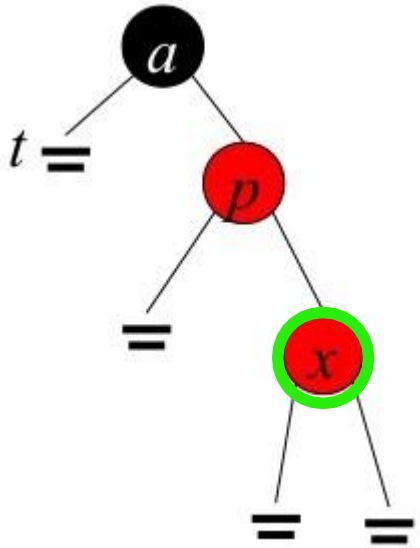


Inserção

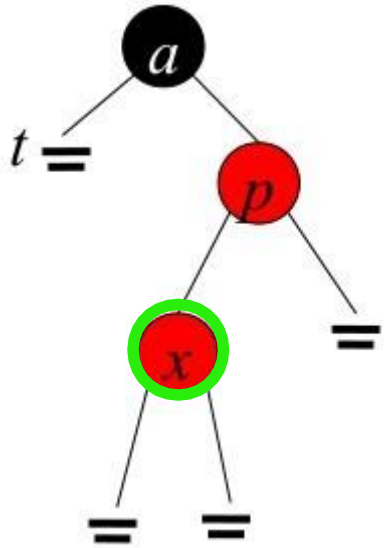
- CASO 3:
- Suponha que o tio de x é preto.
- Nesse caso, para manter a propriedade 3 é preciso fazer rotações envolvendo x, p, t, a.



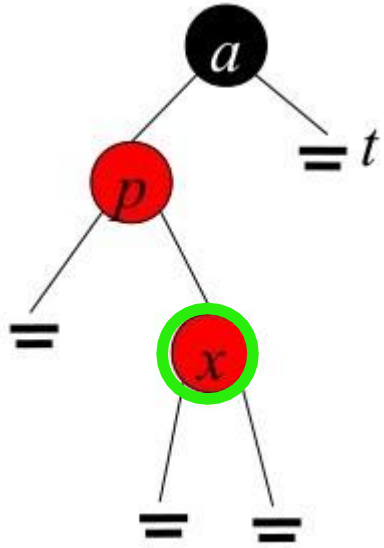
3a



3 b



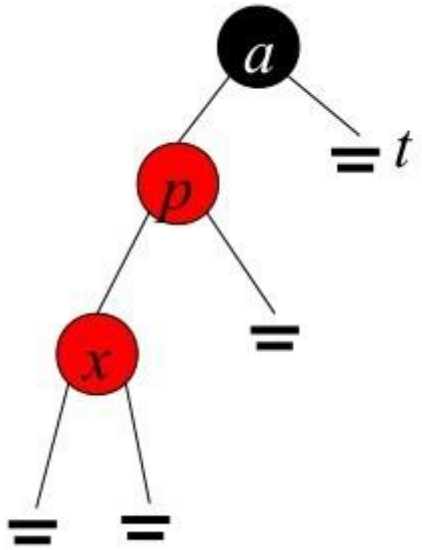
3 c



3 d

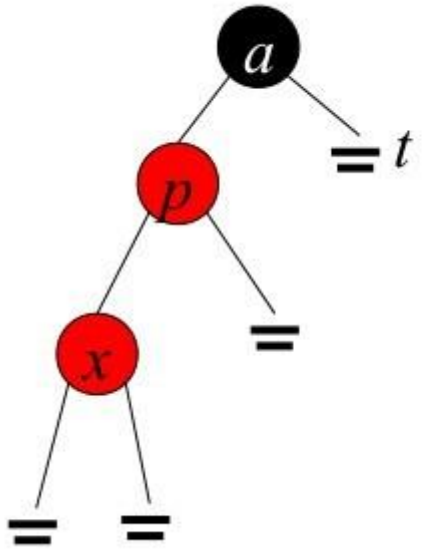
Inserção

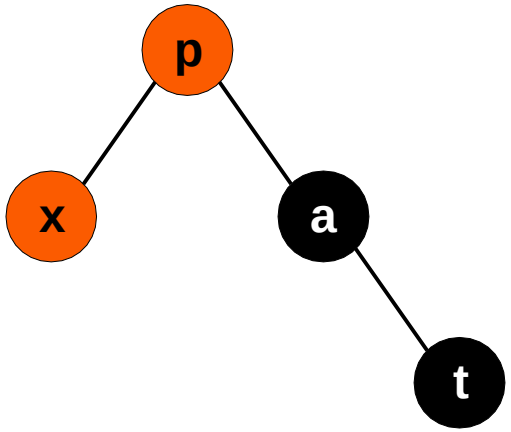
CASO 3a: Rotação à direita



Inserção

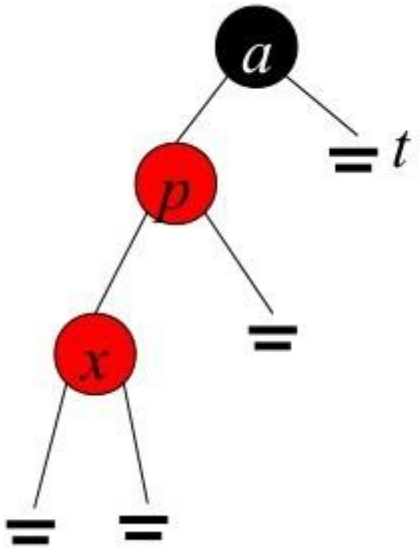
CASO 3a: Rotação à direita

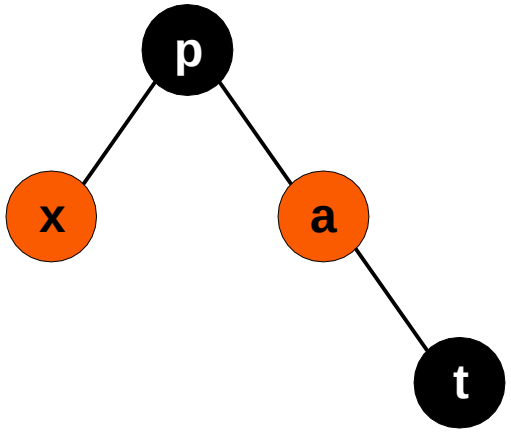




Inserção

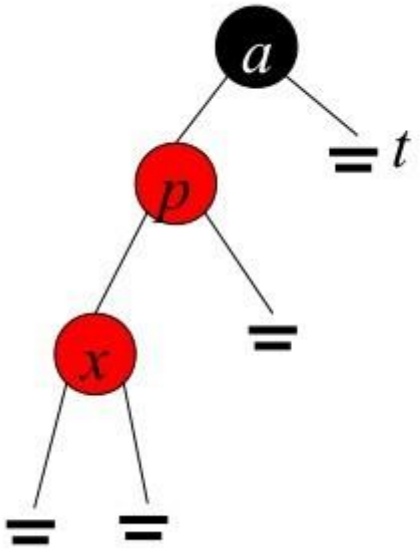
CASO 3a: Rotação à direita

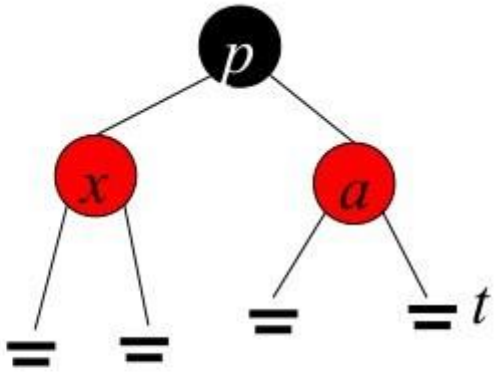




Inserção

CASO 3a: Rotação à direita

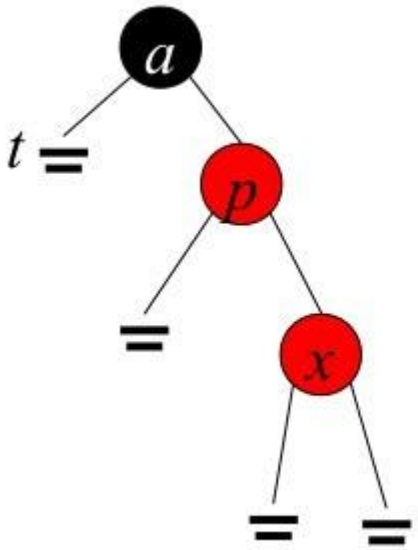


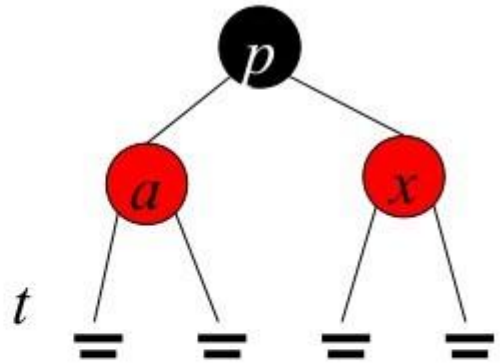


Recoloração de p e a

Inserção

CASO 3b: Rotação à esquerda

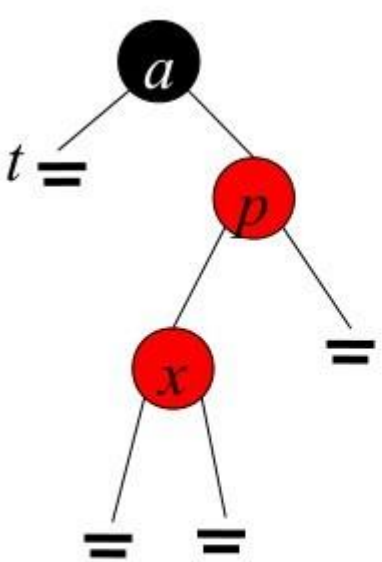


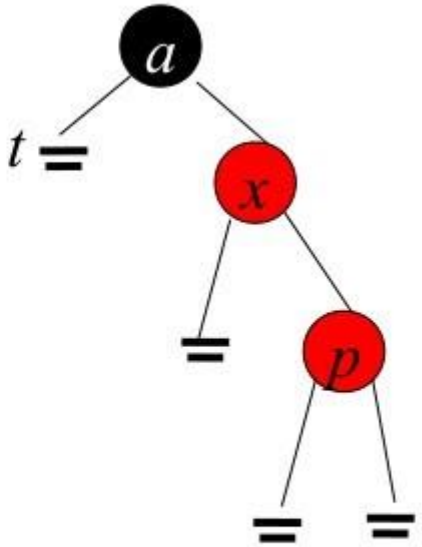


Recoloração de p e a

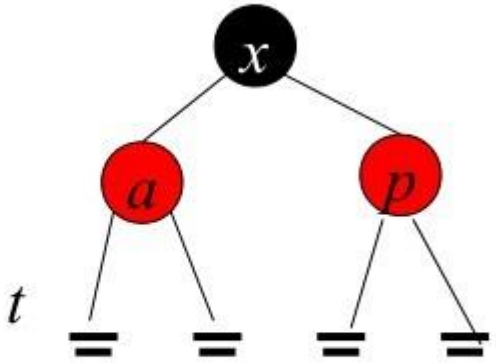
Inserção

CASO 3c: Rotação dupla esquerda





Rotação simples à direita

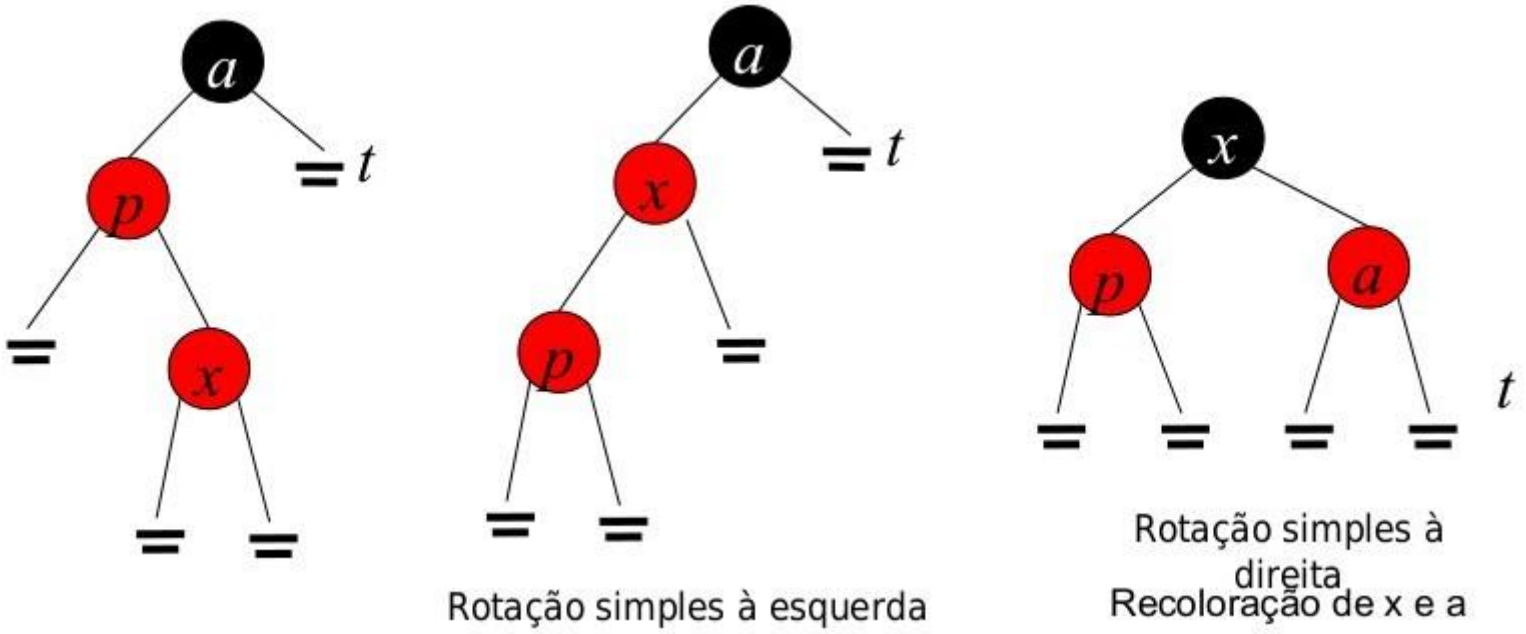


Rotação simples à esquerda

Recoloração de x e a

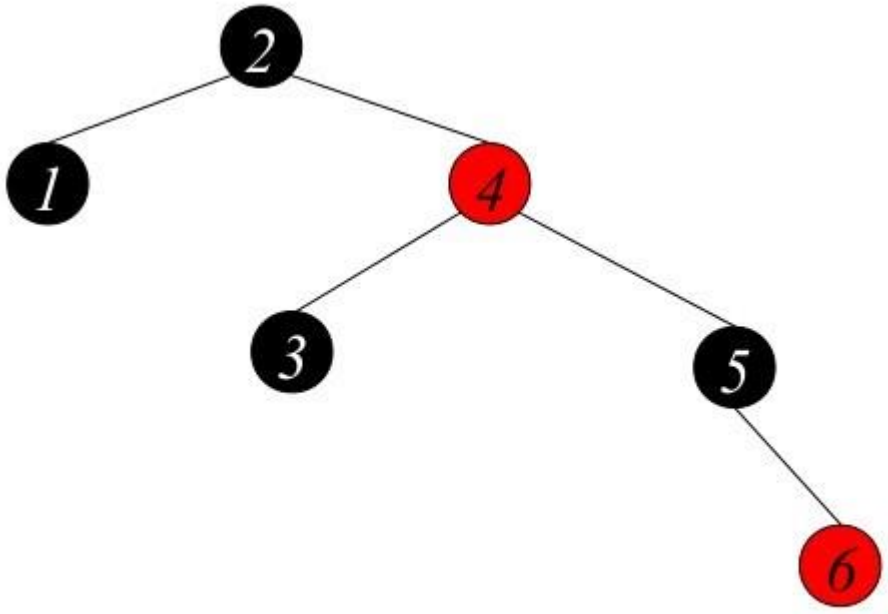
Inserção

CASO 3d: Rotação dupla direita



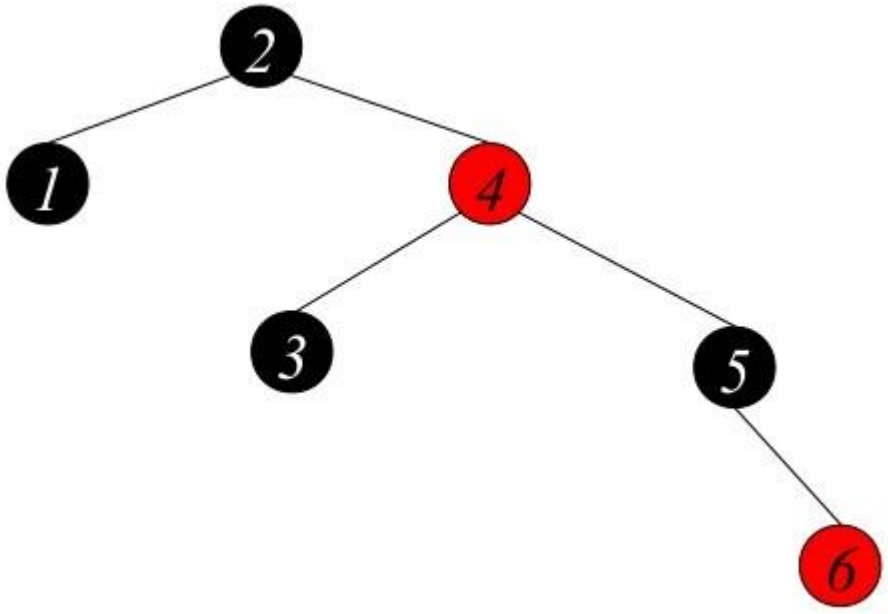
Exemplo 1

Estado inicial da árvore:



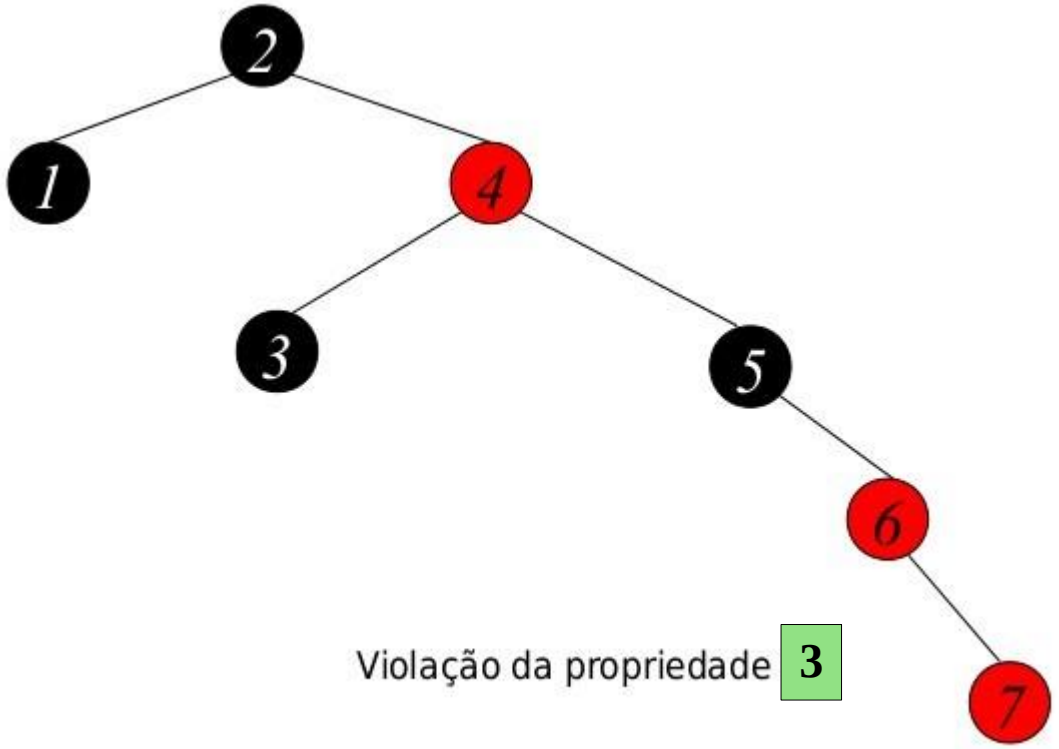
Exemplo 1

- Inserção do nó '7'.



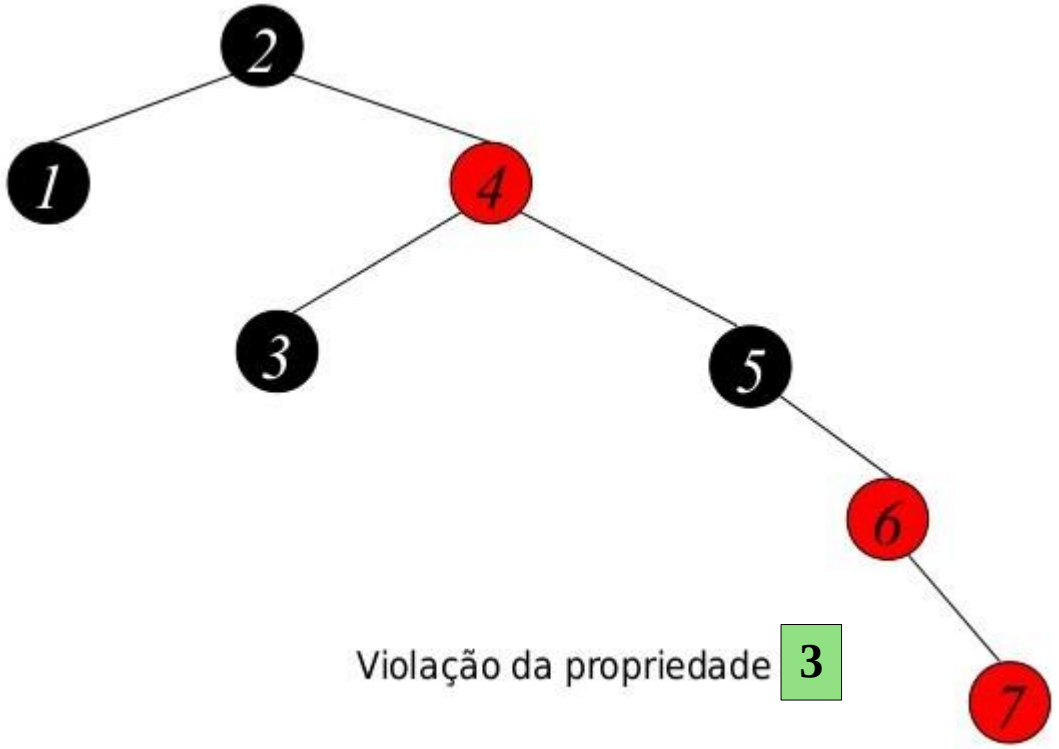
Exemplo 1

- Inserção do nó '7'.



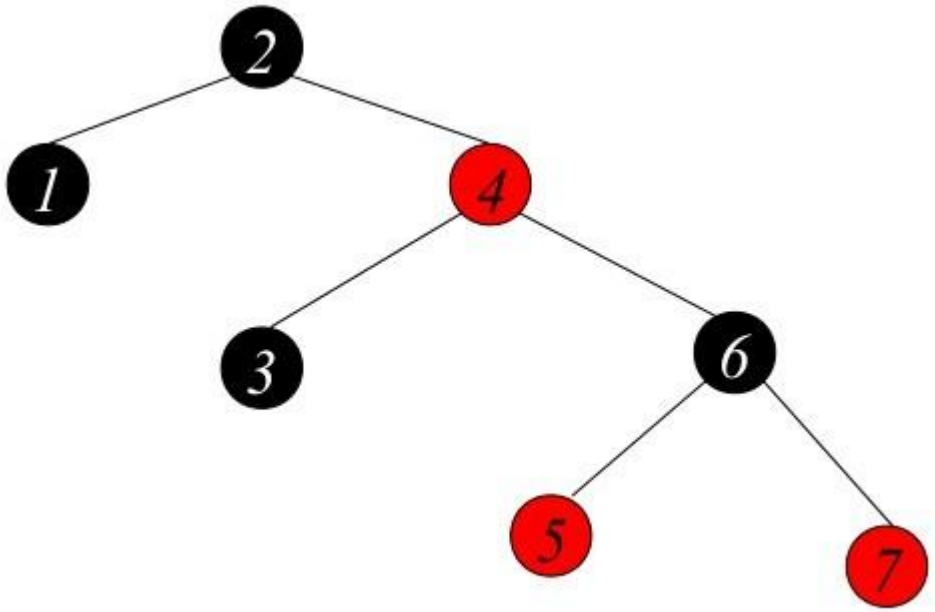
Exemplo 1

- Rotação à esquerda: nós 5,6 e 7.



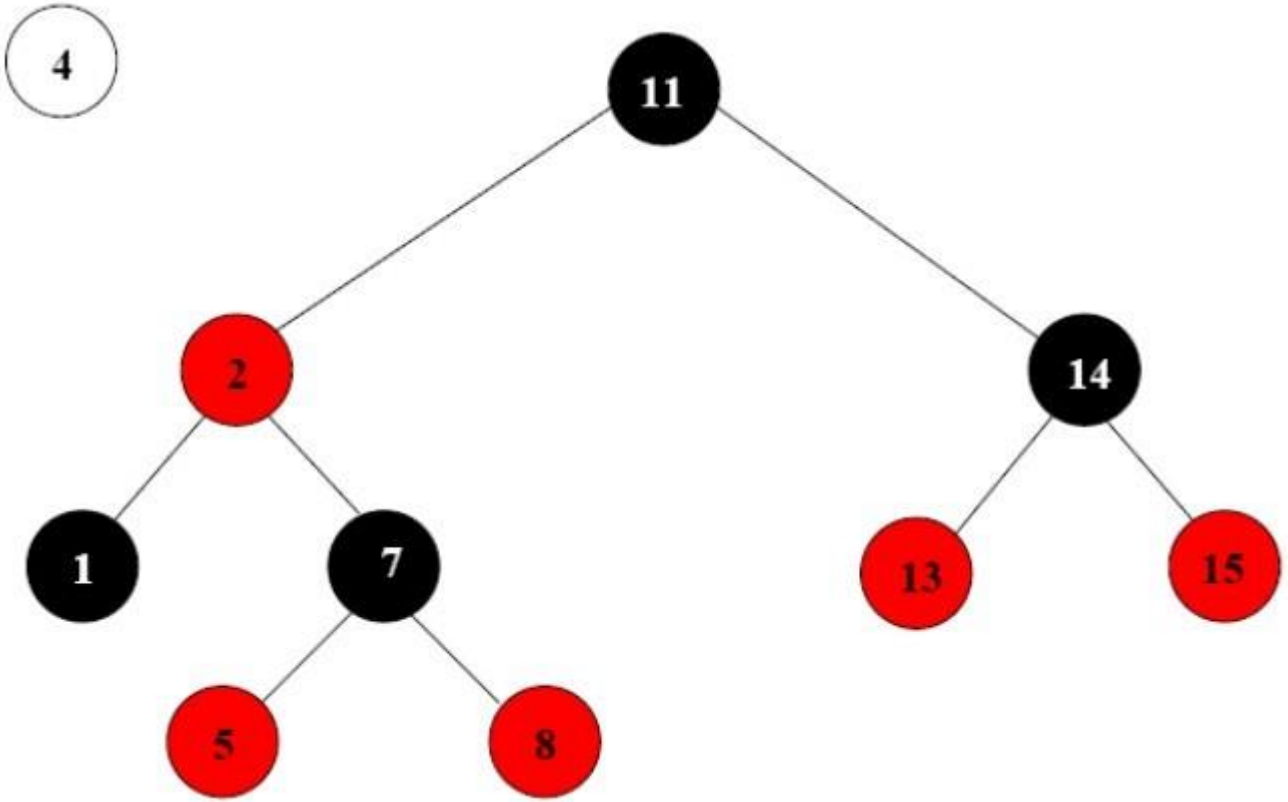
Exemplo 1

- Rotação à esquerda: nós 5,6 e 7.
- Alteração de cor dos nós 5 e 6.

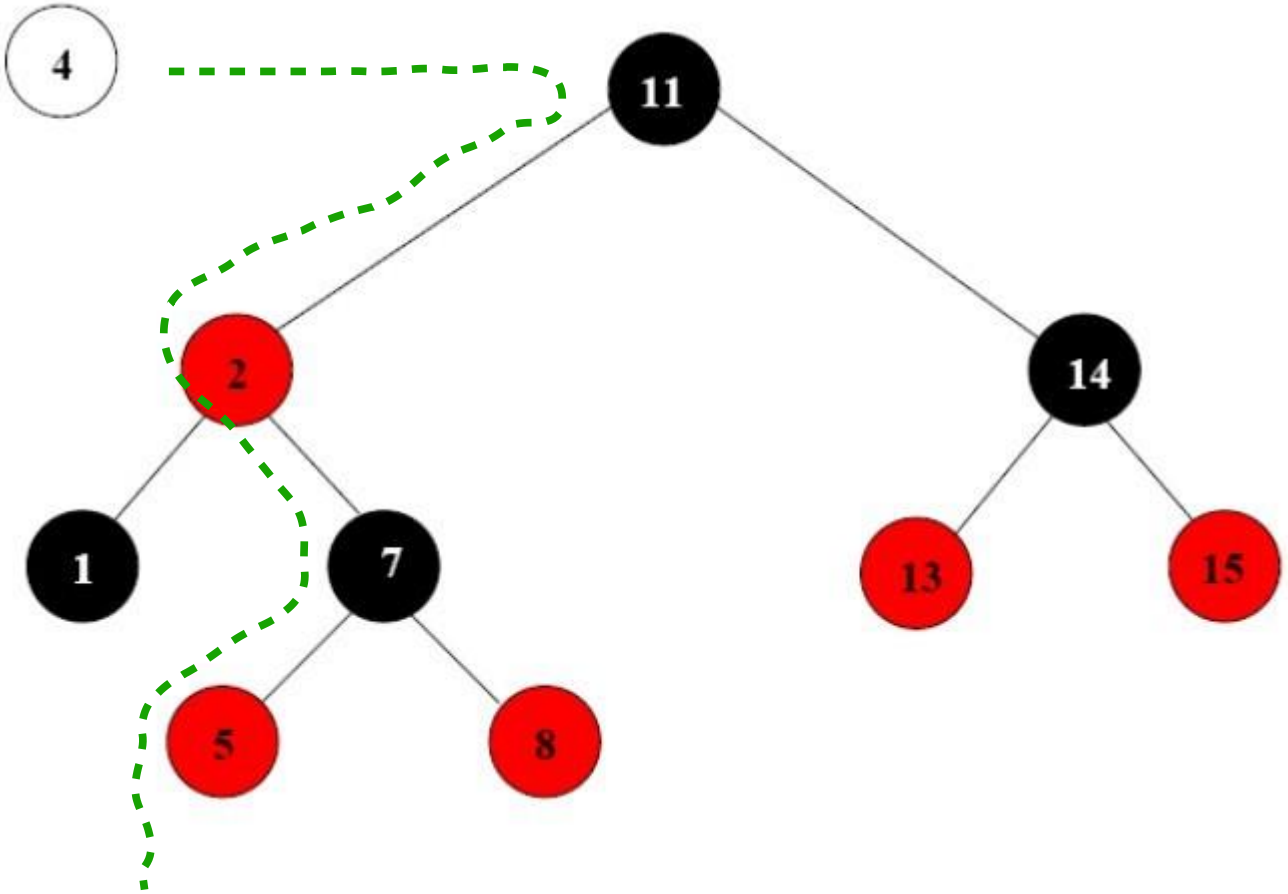


Exemplo 2

Estado inicial da árvore:

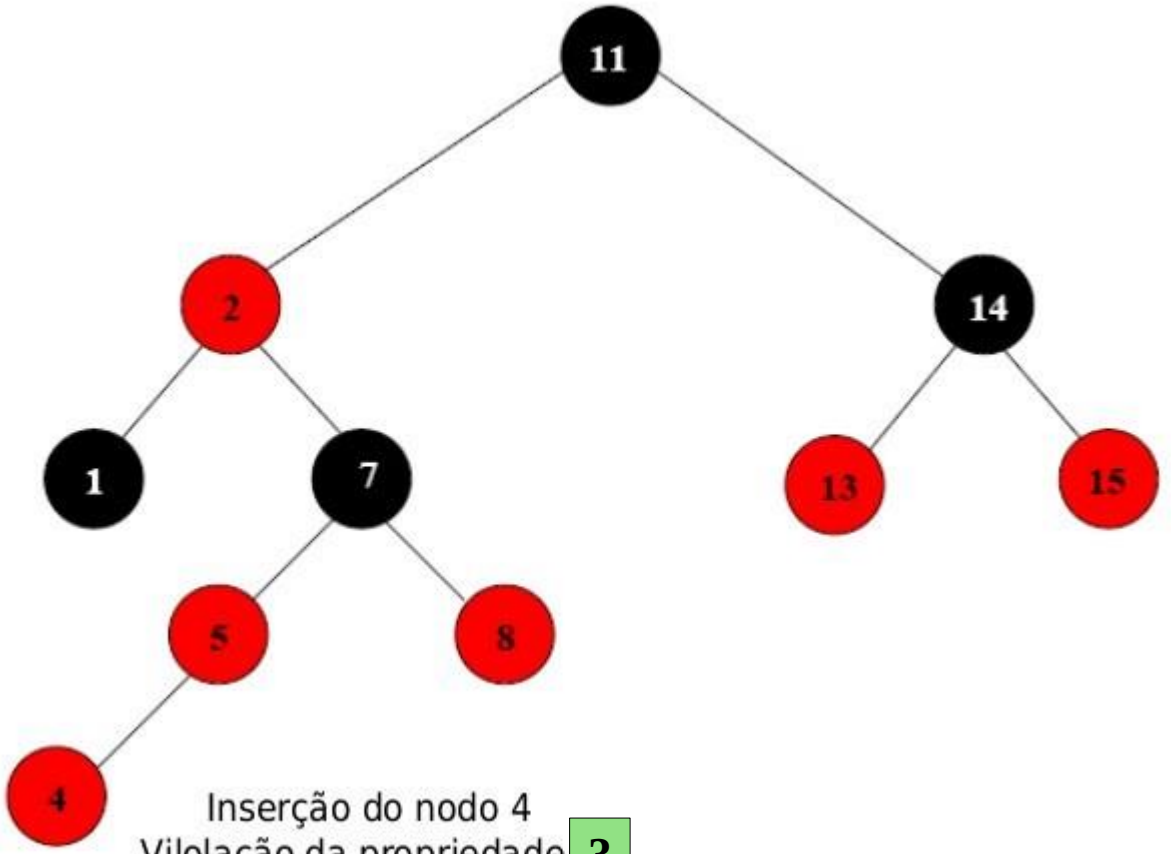


Exemplo 2



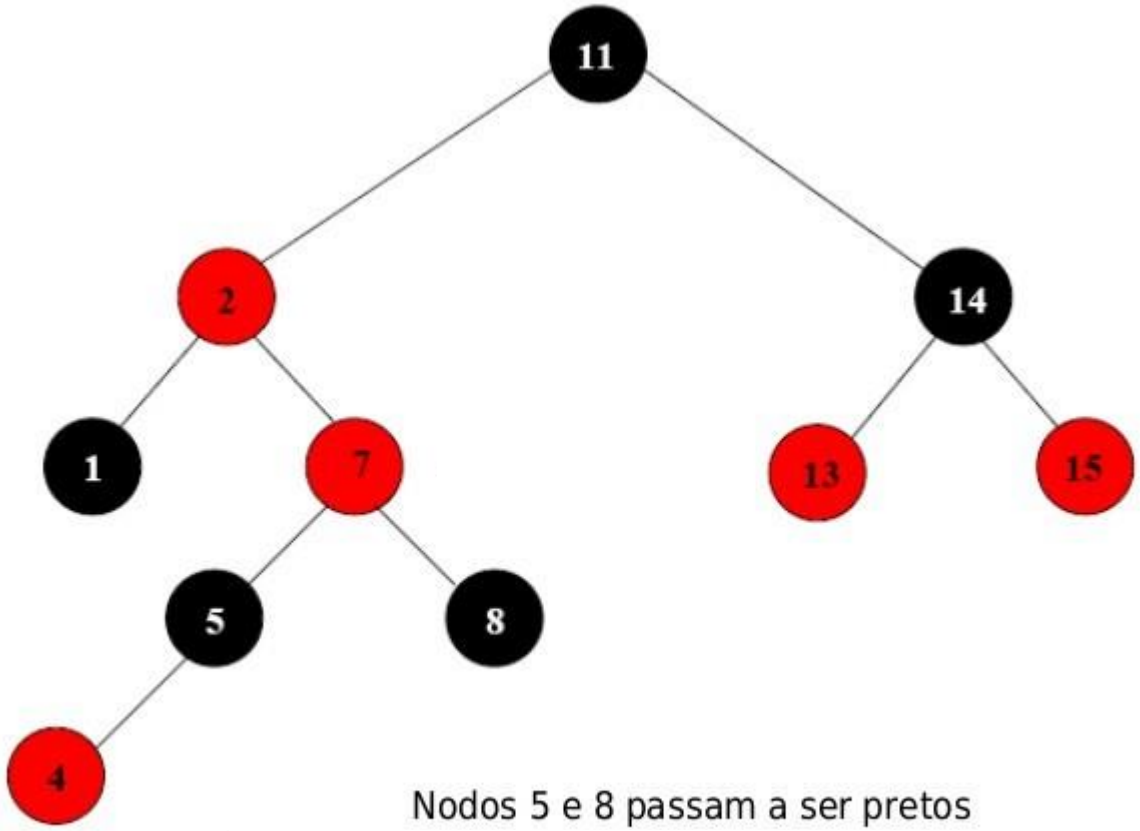
Exemplo 2

CASO 2: O tio do elemento inserido é vermelho.



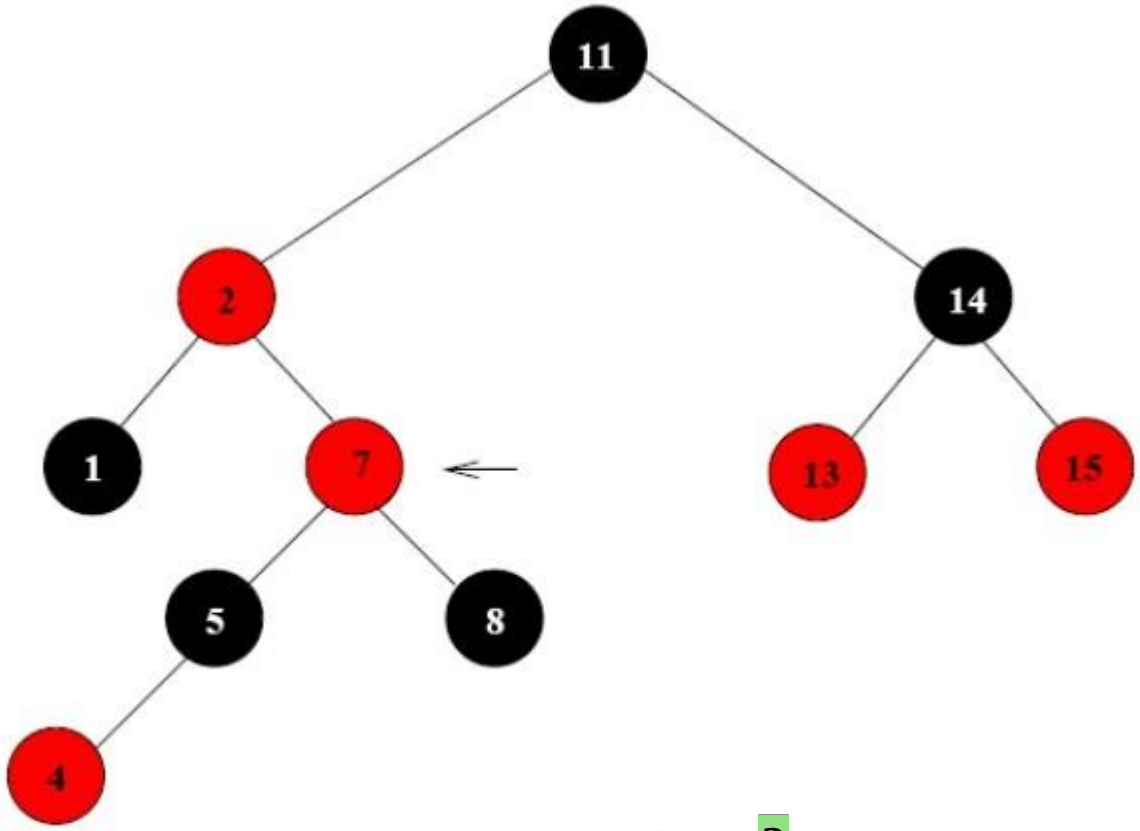
Exemplo 2

CASO 2: O tio do elemento inserido é vermelho.



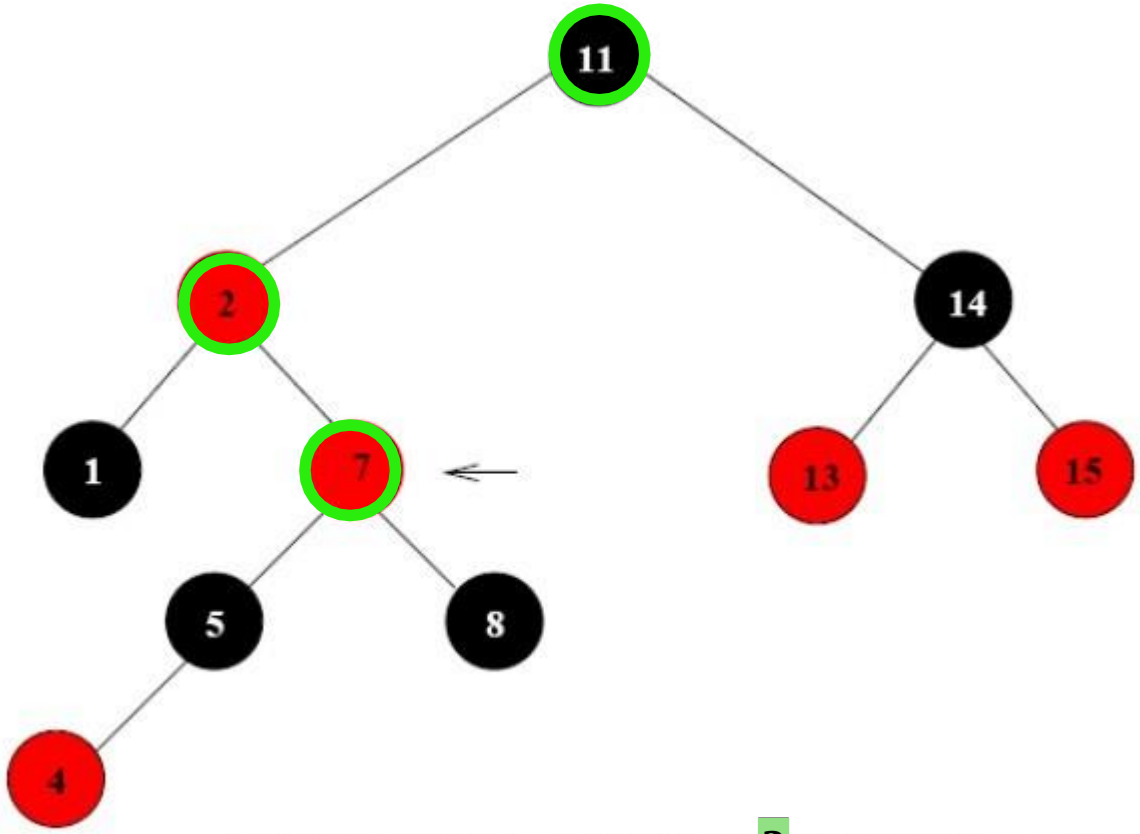
Nodos 5 e 8 passam a ser pretos

Exemplo 2



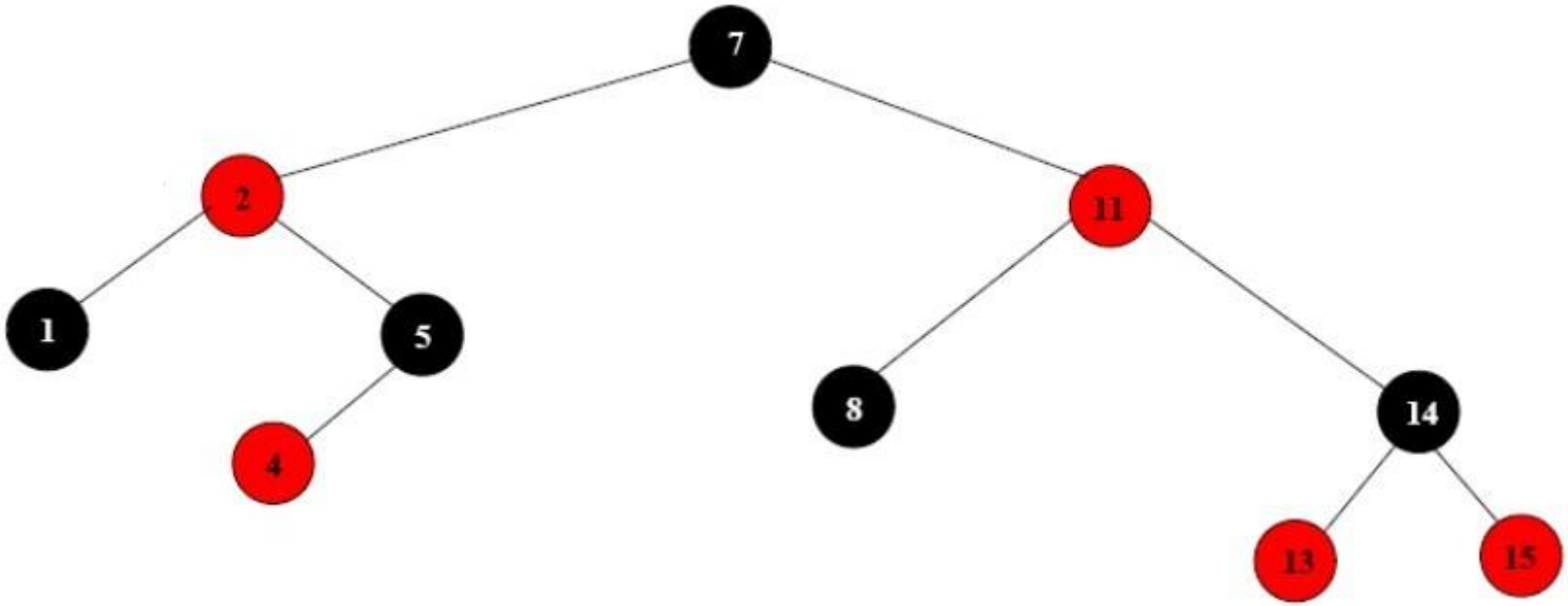
Outra violação da propriedade 3 entre os nodos 2 e 7
Necessária uma rotação

Exemplo 2



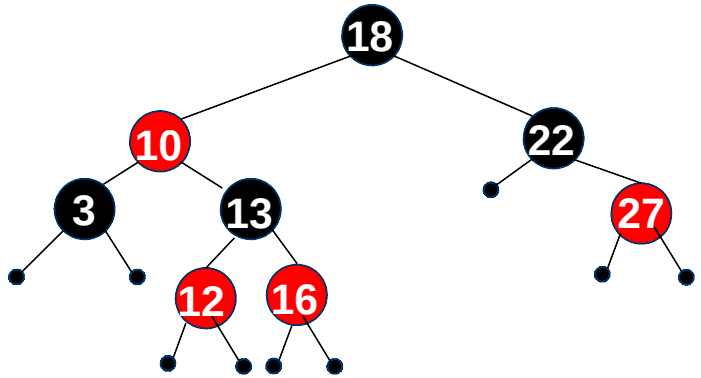
Outra violação da propriedade 3 entre os nodos 2 e 7
Necessária uma rotação

Exemplo 2



Lista 03

(1) O que acontece com a inserção do valor 14 na árvore abaixo? Após isso, o que acontece após a eliminação do valor 18? Forneça apenas as árvores rubro-negra resultantes.



(2) Forneça a árvore rubro-negra resultante da inserção da sequência 10, 6, 4, 5, 0, 3, 9, 2, 1, 8, 7, -1, -3.

Lista 03

(3) Para cada uma das seguintes afirmações indique se é verdadeira ou falsa. Justifique ou exemplifique.

- As sub-árvores esquerda e direita da raíz de uma árvore RN são sempre árvores RN.
- A recoloração de um dado nó pode-se propagar até a raiz da árvore RN.