Árvores Rubro-Negras

Letícia Rodrigues Bueno

UFABC

Árvores Rubro-Negras (*“Red-Black Trees”*): Introdução

- **Objetivo:** garantir que operações básicas demorem $O(\lg n)$ no pior caso.

Árvores Rubro-Negras (*"Red-Black Trees"*): Introdução

- **Objetivo:** garantir que operações básicas demorem $O(\lg n)$ no pior caso.
- **Árvores rubro-negras:** árvores binárias de busca com um bit extra por nó para cor vermelha ou preta;

Árvores Rubro-Negras (*“Red-Black Trees”*): Introdução

- **Objetivo:** garantir que operações básicas demorem $O(\lg n)$ no pior caso.
- **Árvores rubro-negras:** árvores binárias de busca com um bit extra por nó para cor vermelha ou preta;
- **Altura:** no máximo $2\lg(n+1)$ onde n é número de nós;

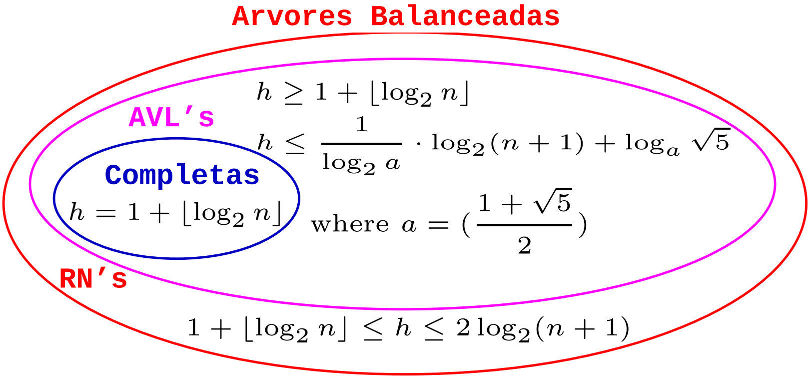
Árvores Rubro-Negras (*"Red-Black Trees"*): Introdução

- **Objetivo:** garantir que operações básicas demorem $O(\lg n)$ no pior caso.
- **Árvores rubro-negras:** árvores binárias de busca com um bit extra por nó para cor vermelha ou preta;
- **Altura:** no máximo $2\lg(n+1)$ onde n é número de nós;
- Inserção e remoção executadas em $O(\lg n)$.

Árvores Rubro-Negras (*“Red-Black Trees”*): Introdução

- **Objetivo:** garantir que operações básicas demorem $O(\lg n)$ no pior caso.
- **Árvores rubro-negras:** árvores binárias de busca com um bit extra por nó para cor vermelha ou preta;
- **Altura:** no máximo $2\lg(n+1)$ onde n é número de nós;
- Inserção e remoção executadas em $O(\lg n)$.
- Nenhum caminho é maior do que duas vezes o comprimento de qualquer outro caminho;

Comparação entre Árvores Balanceadas



Árvores Rubro-Negras: Definição

- Uma árvore rubro-negra é uma ABB que obedece às propriedades:

Árvores Rubro-Negras: Definição

- Uma árvore rubro-negra é uma ABB que obedece às propriedades:
    1. Todo nó externo é preto;

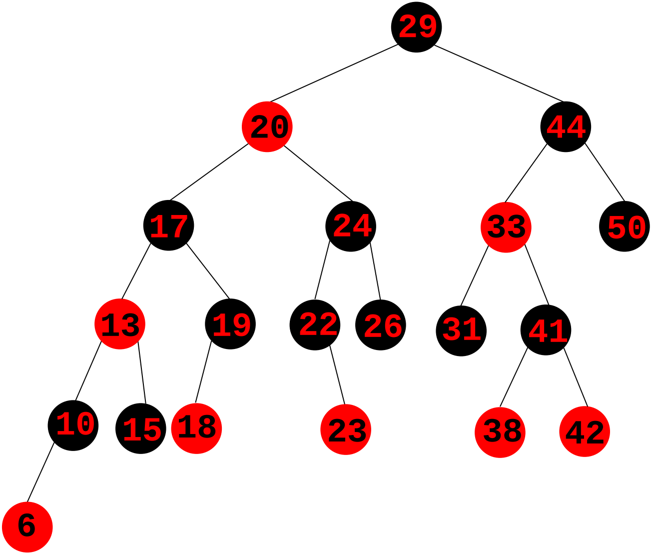
Árvores Rubro-Negras: Definição

- Uma árvore rubro-negra é uma ABB que obedece às propriedades:
    1. Todo nó externo é preto;
    2. Para cada nó, todos os caminhos de um nó até as folhas contêm mesmo número de nós pretos;

Árvores Rubro-Negras: Definição

- Uma árvore rubro-negra é uma ABB que obedece às propriedades:
  1. Todo nó externo é preto;
  2. Para cada nó, todos os caminhos de um nó até as folhas contêm mesmo número de nós pretos;
  3. Se um nó é vermelho, então ambos filhos são pretos.

Exemplo de Árvore Rubro-Negra

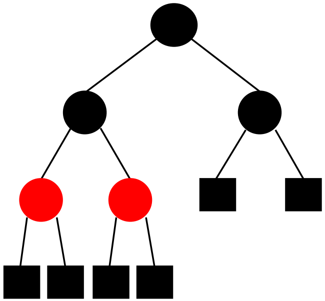


Árvores Rubro-Negras: Inserção

Nó inserido *q* é rubro. Possibilidades:

Caso 1: *v* é negro

Exemplo:

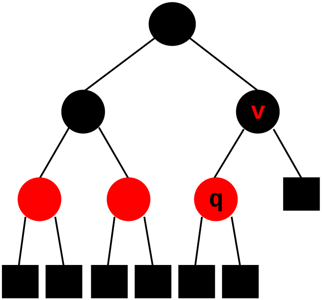


Árvores Rubro-Negras: Inserção

Nó inserido *q* é rubro. Possibilidades:

Caso 1: *v* é negro

Exemplo:



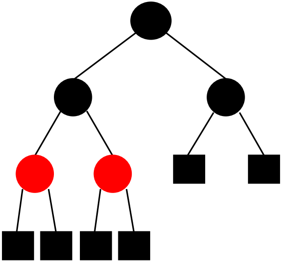
Árvores Rubro-Negras: Inserção

Nó inserido q é rubro. Possibilidades:

Caso 2: v é rubro. Então, w (pai de v) é preto.

Caso 2.1: t é rubro;
Alteramos a cor de v, t, w

Exemplo:



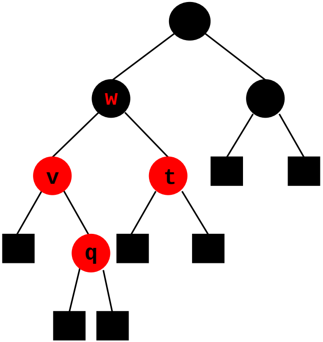
Árvores Rubro-Negras: Inserção

Nó inserido *q* é rubro. Possibilidades:

Caso 2: *v* é rubro. Então, *w* (pai de *v*) é preto.

Caso 2.1: *t* é rubro;
Alteramos a cor de *v*, *t*, *w*

Exemplo:



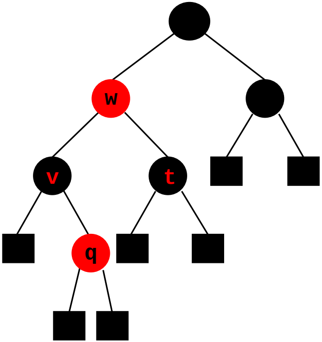
Árvores Rubro-Negras: Inserção

Nó inserido *q* é rubro. Possibilidades:

Caso 2: *v* é rubro. Então, *w* (pai de *v*) é preto.

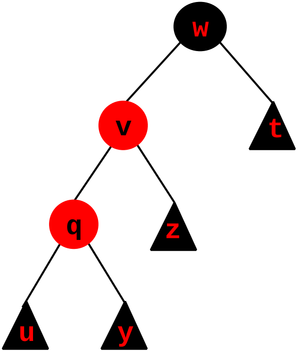
Caso 2.1: *t* é rubro;
Alteramos a cor de *v*, *t*, *w*

Exemplo:



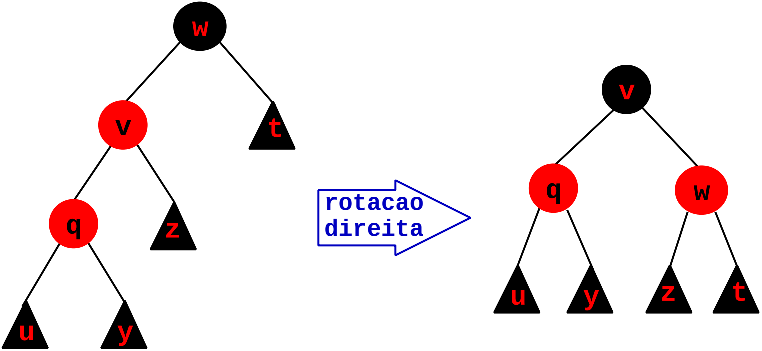
Rotações – Caso 2.2: *t* é preto

Caso 2.2.1: *q* é filho esquerdo de *v* e *v* é filho esquerdo de *w*.
Altera cor de *v* e *w*.



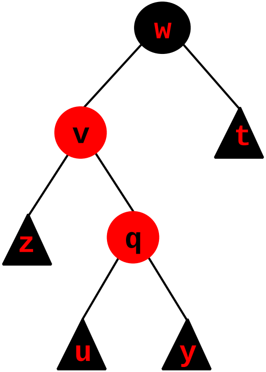
Rotações – Caso 2.2: *t* é preto

Caso 2.2.1: *q* é filho esquerdo de *v* e *v* é filho esquerdo de *w*.
Altera cor de *v* e *w*.



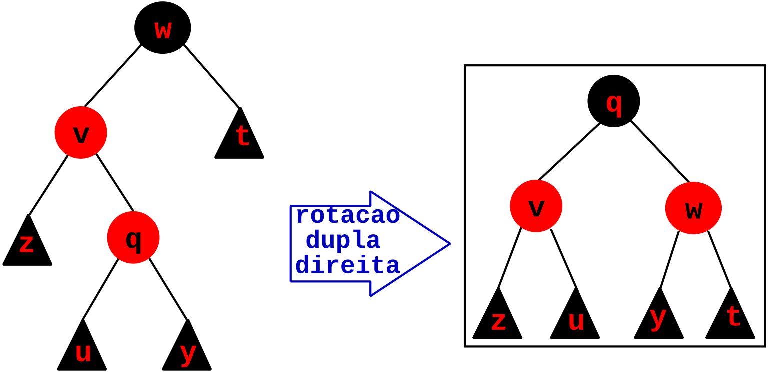
Rotações – Caso 2.2: *t* é preto

Caso 2.2.2: *q* é filho direito de *v* e *v* é filho esquerdo de *w*. Altera cor de *q* e *w*.



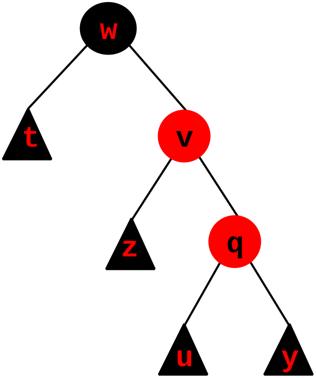
Rotações – Caso 2.2: *t* é preto

Caso 2.2.2: *q* é filho direito de *v* e *v* é filho esquerdo de *w*.
Altera cor de *q* e *w*.



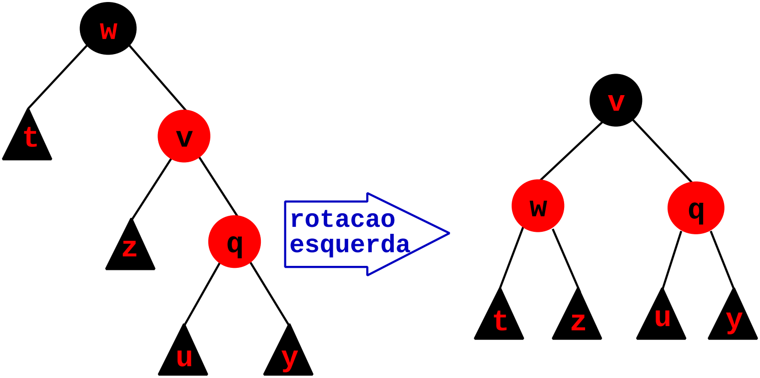
Rotações – Caso 2.2: *t* é preto

Caso 2.2.3: *q* é filho direito de *v* e *v* é filho direito de *w*.
Altera cor de *v* e *w*.



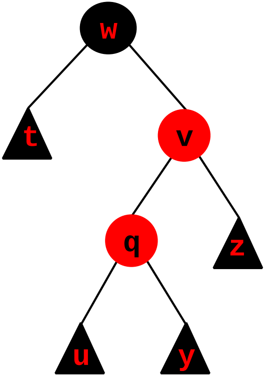
Rotações – Caso 2.2: *t* é preto

Caso 2.2.3: *q* é filho direito de *v* e *v* é filho direito de *w*.
Altera cor de *v* e *w*.



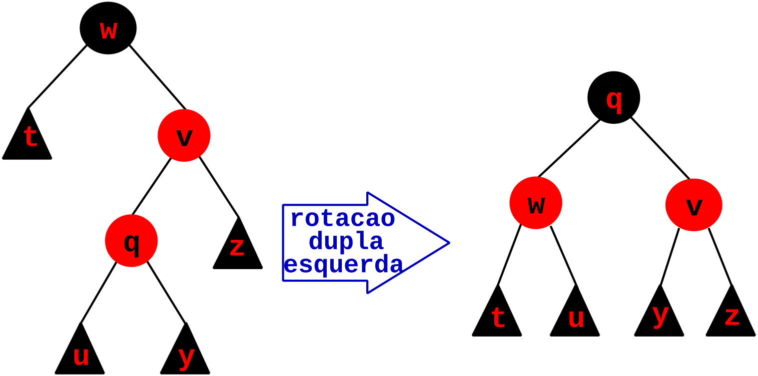
Rotações – Caso 2.2: *t* é preto

Caso 2.2.4: *q* é filho esquerdo de *v* e *v* é filho direito de *w*.
Altera cor de *q* e *w*.



Rotações – Caso 2.2: *t* é preto

Caso 2.2.4: *q* é filho esquerdo de *v* e *v* é filho direito de *w*. Altera cor de *q* e *w*.



Árvores Rubro-Negras: algoritmo para inserção

```
InsereRN(x, ptv, ptw, ptr, a):
    se ptv = externo então
        ocupar(ptv)
        ptv ↑ .esq ← ptv ↑ .dir ← externo
        ptv ↑ .chave ← x; ptv ↑ .cor ← R
        se ptraiz = externo então
            ptv ↑ .cor ← N; ptraiz ← ptv
        senão se x < ptw ↑ .chave então
            ptw ↑ .esq ← ptv
        senão ptw ↑ .dir ← ptv
    senão se x ≠ ptv ↑ .chave então
        se x < ptv ↑ .chave então ptq ← ptv ↑ .esq
        senão ptq ← ptv ↑ .dir
            InsereRN(x, ptq, ptv, ptw, a)
            se a = 1 então rota(ptq, ptv, ptw, ptr, a)
            senão se a = 0 então a = 1
            senão “Inserção Inválida”
```

Comparação: árvores AVL e rubro-negras

Comparação: árvores AVL e rubro-negras

1. **Árvores AVL**:

Comparação: árvores AVL e rubro-negras

1. **Árvores AVL**:

   1.1 primeira árvore binária de busca com balanceamento proposta por Adel’son-Vel’skii e Landis em 1962;

Comparação: árvores AVL e rubro-negras

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

Comparação: árvores AVL e rubro-negras

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

Comparação: árvores AVL e rubro-negras

1. **Árvores AVL**:
   1.1 primeira árvore binária de busca com balanceamento proposta por Adel'son-Vel'skii e Landis em 1962;
   1.2 **altura:** entre $\log_2(n+1)$ e $1.4404\log_2(n+2) - 0.328$, portanto, $O(\log n)$;
2. **Árvores rubro-negras**:
   2.1 proposta por Guibas e Sedgewick em 1978;

Comparação: árvores AVL e rubro-negras

1. **Árvores AVL**:
   1.1 primeira árvore binária de busca com balanceamento proposta por Adel'son-Vel'skii e Landis em 1962;
   1.2 **altura:** entre $\log_2(n+1)$ e $1.4404 \log_2(n+2) - 0.328$, portanto, $O(\log n)$;
2. **Árvores rubro-negras**:
   2.1 proposta por Guibas e Sedgewick em 1978;
   2.2 **altura:** $2 \log_2(n+1)$, portanto, $O(\log n)$;

Comparação: árvores AVL e rubro-negras

1. **Árvores AVL**:
   1.1 primeira árvore binária de busca com balanceamento proposta por Adel'son-Vel'skii e Landis em 1962;
   1.2 **altura:** entre $\log_2(n+1)$ e $1.4404 \log_2(n+2) - 0.328$, portanto, $O(\log n)$;
2. **Árvores rubro-negras**:
   2.1 proposta por Guibas e Sedgewick em 1978;
   2.2 **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;

Exercícios

1. Prove ou dê contra-exemplo:
   1.1 Toda árvore completa é AVL.
   1.2 Toda árvore AVL é completa.
   1.3 Toda árvore AVL é rubro-negra.
   1.4 Toda árvore rubro-negra é AVL.
   1.5 Toda árvore completa é rubro-negra.
   1.6 Toda árvore rubro-negra é completa.

Exercícios

2. Mostre que o caminho mais longo a partir de um nó x em uma árvore rubro-negra até uma folha descendente tem comprimento no máximo de duas vezes o comprimento do caminho mais curto a partir do nó x até uma folha descendente.

Exercícios

2. Mostre que o caminho mais longo a partir de um nó x em uma árvore rubro-negra até uma folha descendente tem comprimento no máximo de duas vezes o comprimento do caminho mais curto a partir do nó x até uma folha descendente.
3. Dê um exemplo de inserção em árvore rubro-negra cuja recoloração dos nós se propaga até a raiz.

Exercícios

2. Mostre que o caminho mais longo a partir de um nó x em uma árvore rubro-negra até uma folha descendente tem comprimento no máximo de duas vezes o comprimento do caminho mais curto a partir do nó x até uma folha descendente.
3. Dê um exemplo de inserção em árvore rubro-negra cuja recoloração dos nós se propaga até a raiz.
4. Escreva o procedimento de remoção de um nó em árvores rubro-negras.

Exercícios

2. Mostre que o caminho mais longo a partir de um nó x em uma árvore rubro-negra até uma folha descendente tem comprimento no máximo de duas vezes o comprimento do caminho mais curto a partir do nó x até uma folha descendente.
3. Dê um exemplo de inserção em árvore rubro-negra cuja recoloração dos nós se propaga até a raiz.
4. Escreva o procedimento de remoção de um nó em árvores rubro-negras.
5. Prove ou dê contra-exemplo: seja uma árvore rubro-negra cuja raiz possui a cor rubra. Se esta for alterada para negra, a árvore mantém-se rubro-negra.

Bibliografia Utilizada

SZWARCFITER, J. L. e MARKENZON, L. Estruturas de Dados e seus Algoritmos, LTC, 1994.