Árvore Auto-Balanceáveis

Prof. Rafael Guterres Jeffman

Motivação

- Numa árvore binária de pesquisa (ABP), o custo de procura de um elemento depende da altura da árvore.

- Numa árvore equilibrada, a altura necessária para armazenar todos os elementos é menor que numa árvore degenerada.

- ABP podem degenerar em listas encadeadas quando os elementos são inseridos já ordenados.

Árvores Auto-Balanceáveis

- Em uma árvore auto-balanceável, independente de como os elementos são adicionados, a árvore fica "equilibrada".

- Uma árvore auto-balanceável é uma árvore binária de pesquisa que se auto organiza, mantendo a altura da árvore próximo da altura ideal (árvore cheia) para o número de elementos armazenados.

Árvore AVL

- A árvore AVL foi originalmente proposta em 1962, por Georgy Maximovich Adelson-Velsky e Evgenii Mikhailovich Landis.

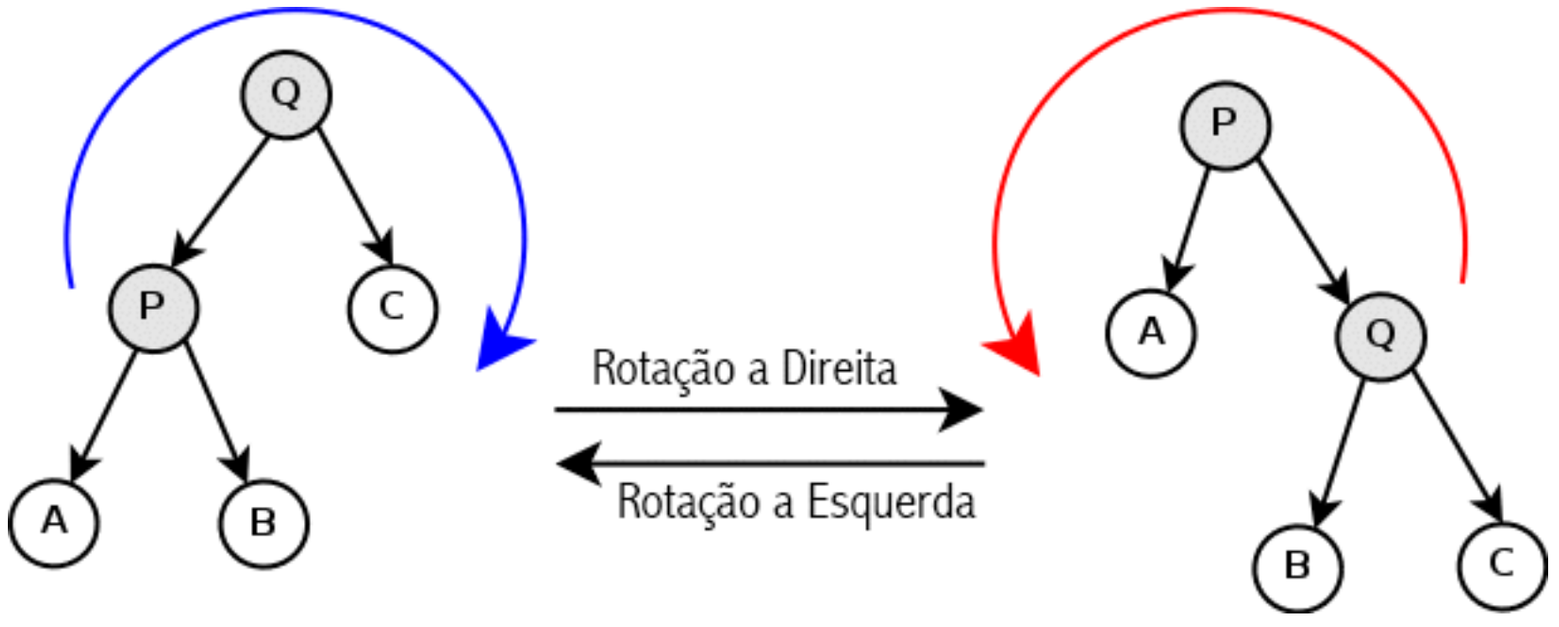
- Primeira proposta de uma árvore binária de pesquisa auto-balanceáveis.

- São as árvores auto-balanceáveis mais restritas, permitindo pouca diferença na altura dos dois lados de uma árvore binária.

Operações

- Procura:
  - A procura é idêntica à procura em uma árvore binária de pesquisa.
- Inserção e Remoção:
  - Tanto a inserção e a remoção são semelhantes as mesmas operações em uma ABP. Após a inserção na ABP, a árvore é balanceada, utilizando rotações de árvores.

Rotação de Árvores



Rotação de Árvores

- Rotação a Esquerda
  - Salva B.
  - Q vira filho à direita de P
  - B vira filho à esquerda de Q
  - P assume lugar de Q
- Rotação a Direita
  - Salva B.
  - P vira filho à esquerda de Q
  - B vira filho à direita de P
  - Q assume lugar de P

Altura de um Nodo

- A altura de uma folha é ZERO.
- A altura de um nodo é UM mais a altura do mais alto filho de um nodo.
- Para implementar a altura de um nodo, utilizamos UM como altura das folhas, desta forma a implementação fica:

$$h(n) = 1 + \max (h(n.esq), h(n.dir))$$

Fator de Balanceamento

- O fator de balanceamento de um nodo da Árvore AVL é utilizado para determinar se a árvore deve ser balanceada ou não.

- O fator de balanceamento de um nodo é a diferença da altura da árvore à esquerda do nodo com relação a árvore à direita.

$$FB(n) = h(n.esq) - h(n.dir)$$

Inserção em uma Árvore AVL

- Executar a inserção “normal” em uma ABP.

- A partir do nodo inserido, em direção à raiz, calcular a altura do nodo e o fator de balanceamento.

- Caso o módulo do fator de balanceamento seja maior que 1, executar o processo de balanceamento.

- Caso a nova altura do nodo não seja diferente da altura anterior, o algoritmo pode ser terminado.

Balanceamento em uma Árvore AVL

- Todo nodo em uma árvore AVL deve ter o valor absoluto do fator de balanceamento menor ou igual a UM.

$$|FB(n)| <= 1$$

- Caso |FB(n)| seja igual a DOIS, deve ser feita uma rotação no nodo para corrigir o balanceamento da árvore.

Rotação do nodo AVL

- Se FB(n) == +2,
  - Se FB(n.esq) == -1), rotaciona n.esq à esquerda
  - rotaciona n à direita
- Se FB(n) == -2,
  - Se FB(n.dir) == +1), rotaciona n.dir à direita
  - rotaciona n à esquerda
- Após a rotação do nodo “n”, a partir deste nodo, executar uma travessia pós-fixa recalculando a altura e o fator de balanceamento dos nodos.

Remoção na árvore AVL

- A remoção em uma árvore AVL segue os mesmos passos da remoção em uma ABP.
- Após a remoção do nodo, devem ser recalculados a altura e o fator de balanceamento dos nodos a partir do “pai” do nodo removido, e o processo de balanceamento executado quando necessário.

Árvores Red-Black

- É uma árvore binária de pesquisa auto-balanceável.
- O balanceamento da Red-Black é menos restrito, permitindo que a árvore fique levemente desbalanceada, no entanto, são executadas menos rotações em caso de inserção ou remoção de nodos.

Propriedades

- Uma árvore Red-Black possui cinco propriedades:
  - Um nodo de uma árvore Red-Black é preto ou vermelho.
  - A raiz de uma árvore Red-Black é um nodo preto.
  - Todas as folhas são pretas e não possuem “conteúdo” (são nulas).
  - Todo filho vermelho contém dois filhos pretos (ou seja, não contém filhos vermelhos).
  - Todo caminho de um nodo até as folhas possui o mesmo número de nodos pretos.

Inserção em uma Árvore RB

- O primeiro passo da inserção em uma árvore Red-Black, é realizar a inserção do nodo em uma árvore binária de pesquisa.

- O nodo inserido na árvore é vermelho.

- Após a inserção do nodo na árvore, é realizado o processo de inserção na árvore RB, iniciando pelo “caso 1”

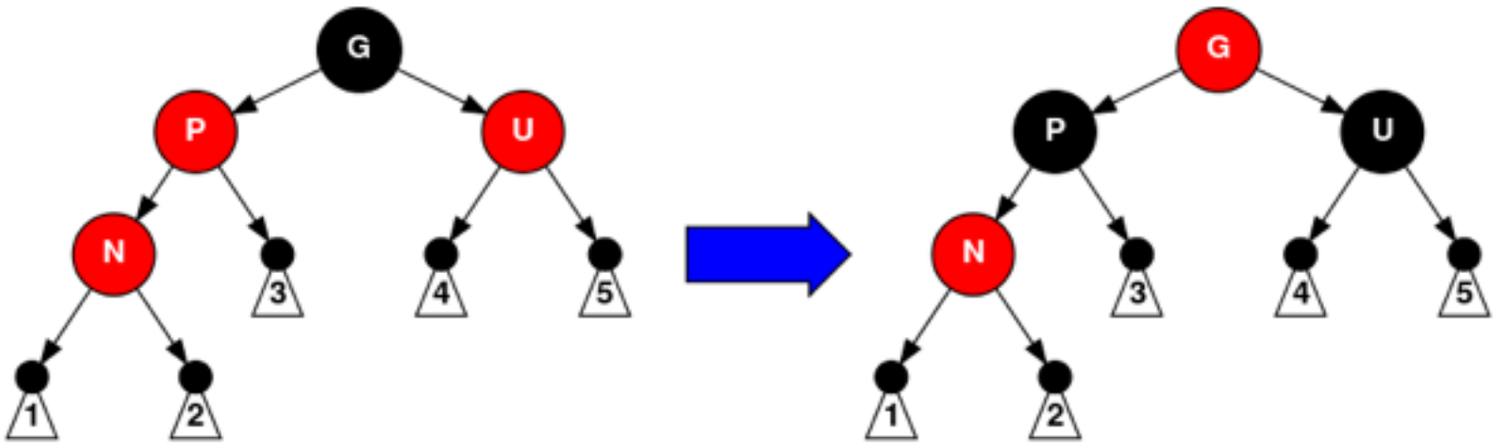
Inserção RB: Caso 1

- Se o nodo inserido é a raiz da árvore, pinta o nodo de preto e termina o algoritmo de inserção.

Inserção RB: Caso 2

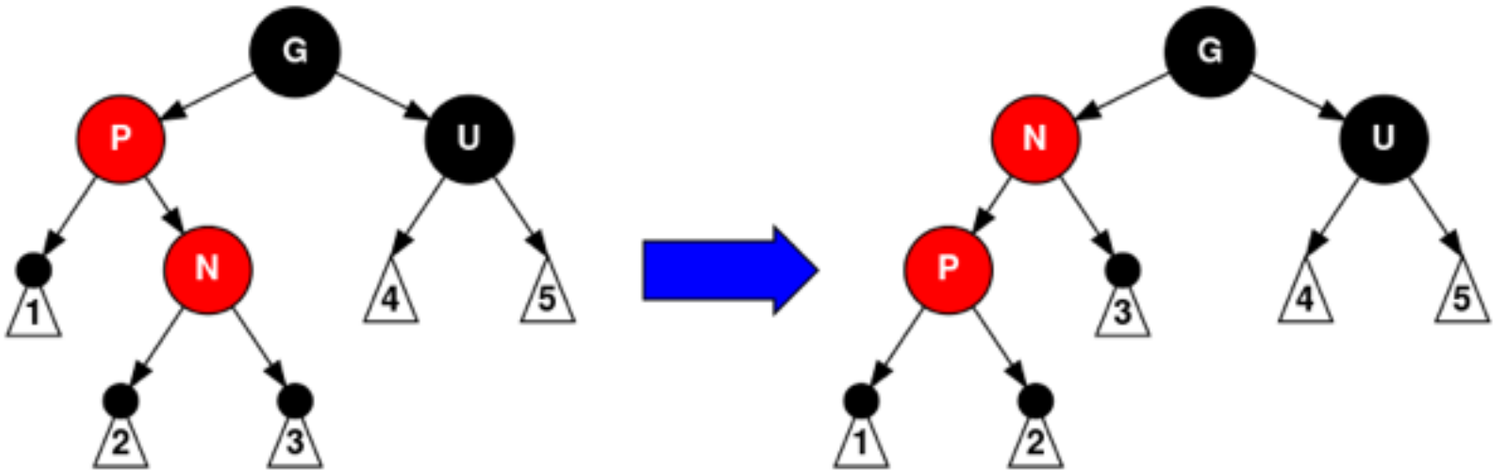
- Se a cor do pai do nodo inserido for preta, encerra o algoritmo.

Inserção RB: Caso 3



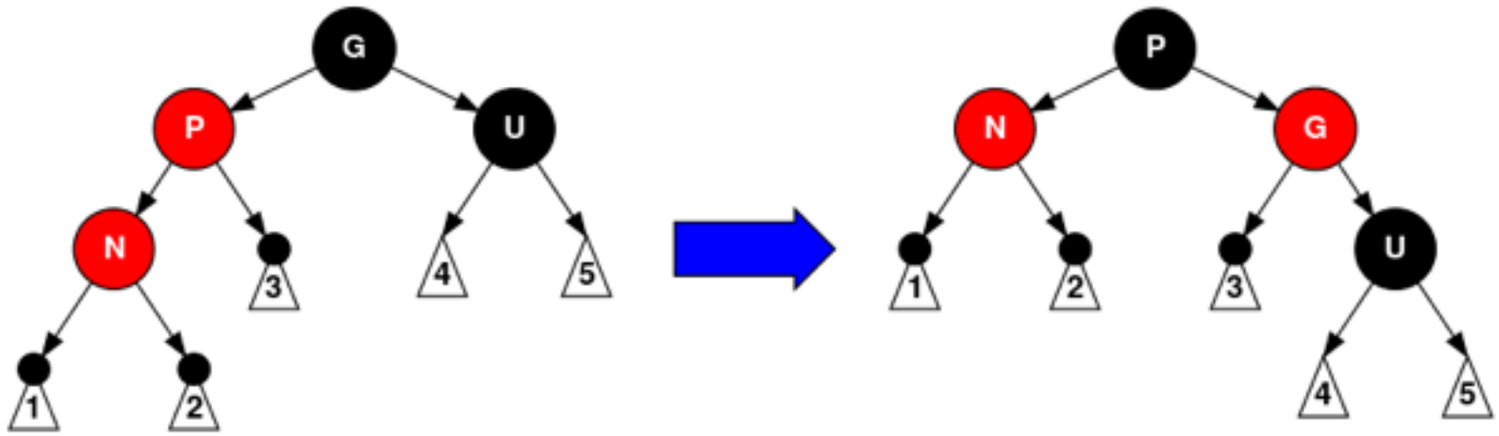
- Se o pai (P) e o tio (U) são vermelhos,
  - Pinta o pai e o tio de preto.
  - Pinta o avô (G) de vermelho.
  - Volta para o Caso 1 no avô (G).

Inserção RB: Caso 4



- Se o pai (P) é vermelho e o tio (U) é preto, e o nodo (N) é filho à esquerda e o pai (P) é filho à direita,
  - Rotaciona à direita em (P), segue para o caso 5 em P.
  - (Deve ser tratado o caso onde P e N estão do outro lado, invertendo a rotação.)

Inserção RB: Caso 5



- Se o pai (P) é vermelho e o tio (U) é vermelho, e P e N são filho à esquerda,
  - Pinta o pai e o tio de preto.
  - Pinta o avô (G) de vermelho.
  - Rotaciona à direita em G.
  - (Deve ser tratado o caso de P e N serem filhos à direita, rotacionando à esquerda em G.)

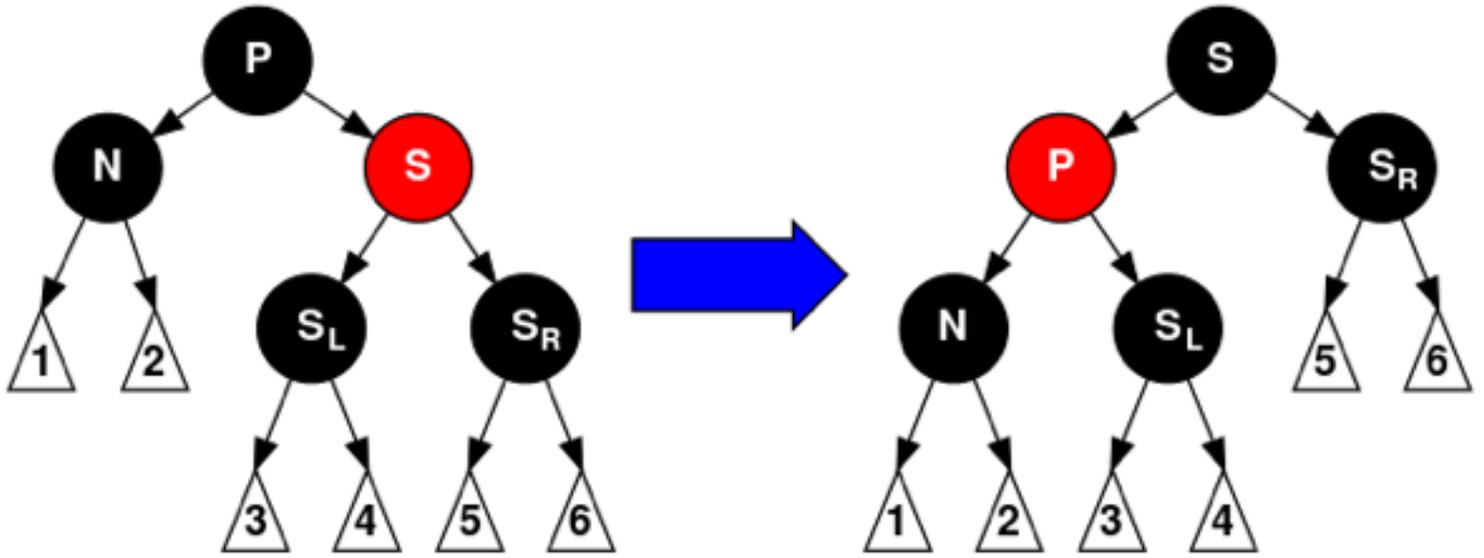
Remoção em Árvore RB

- Quando um nodo é removido, ou não tem filhos, ou tem apenas um filho.
- Se o nodo tem um único filho, este filho será VERMELHO, basta que o filho substitua o nodo sendo removido e seja pintado de PRETO.
- Se o nodo não tem filhos, devido à terceira propriedade das árvores RB, um nodo “preto” tomará lugar do nodo removido.
    - Se o nodo removido é “vermelho”, uma folha (Nil) irá tomar lugar do nodo, deixando a árvore válida.
    - Se o nodo removido é “preto”, haverá a diminuição de um nodo preto em um caminho, invalidando a quinta propriedade, nesse caso, chamamos de “N” a folha que substitui o nodo removido, e executamos os casos de remoção a partir do primeiro.

Remoção RB: Caso 1

- Se o nodo N é a raiz,
  - Termina o algoritmo.

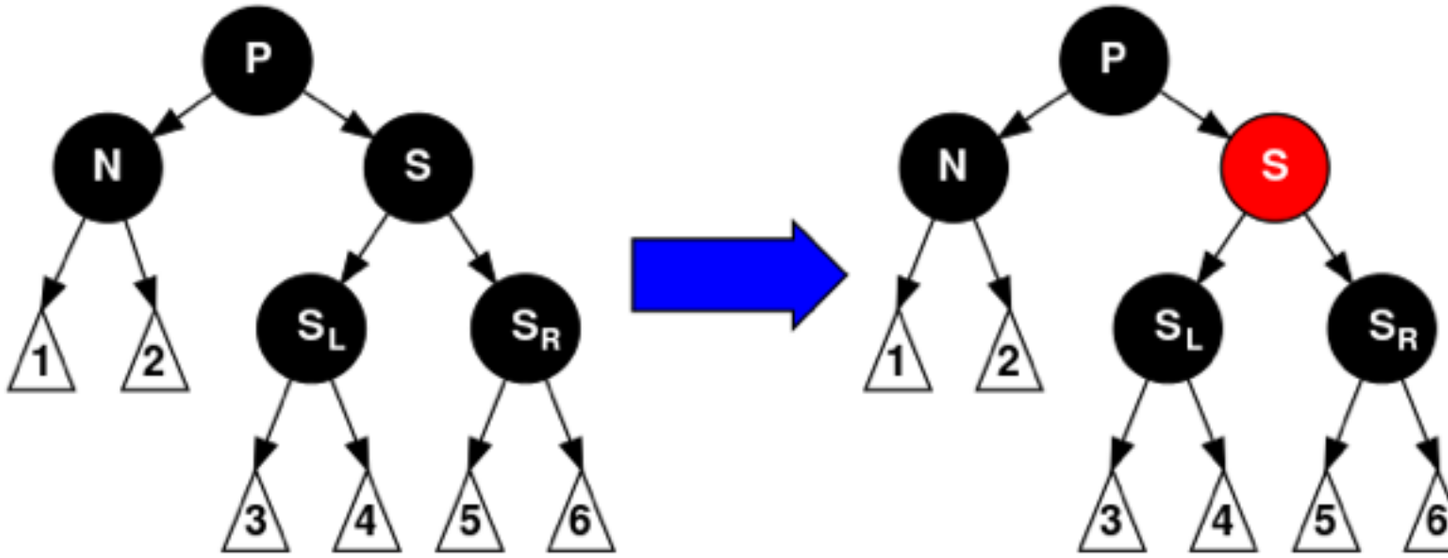
Remoção RB: Caso 2



- Se o nodo (N) e o pai (P) são pretos, e o irmão (S) é vermelho,
  - Pinta P de vermelho
  - Pinta S de Preto
  - Se N for filho à esquerda, rotaciona à esquerda em P
  - Se N for filho à direita, rotaciona à direita em P
  - Segue para o caso 3 em N

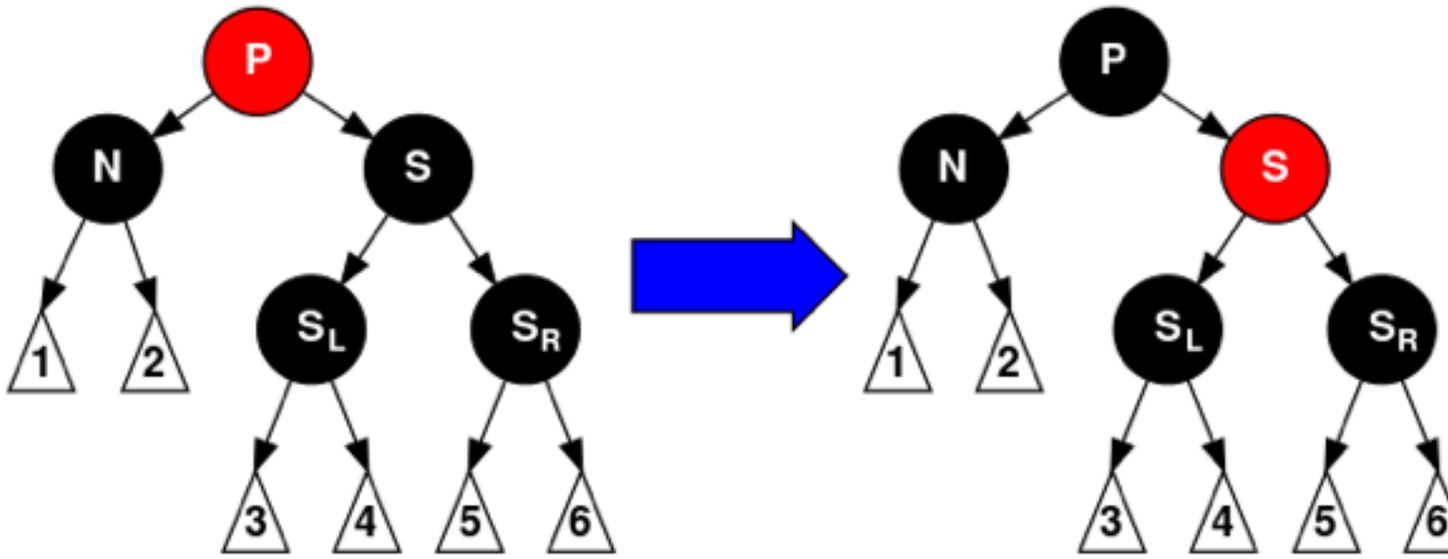
Fecomércio RS | Senac

Remoção RB: Caso 3



- Se o nodo (N), o pai (P), o irmão (S) e os filhos dos irmãos (S_L e S_R) são todos pretos,
  - Pinta o irmão (S) de vermelho
  - Volta para o caso 1 em P

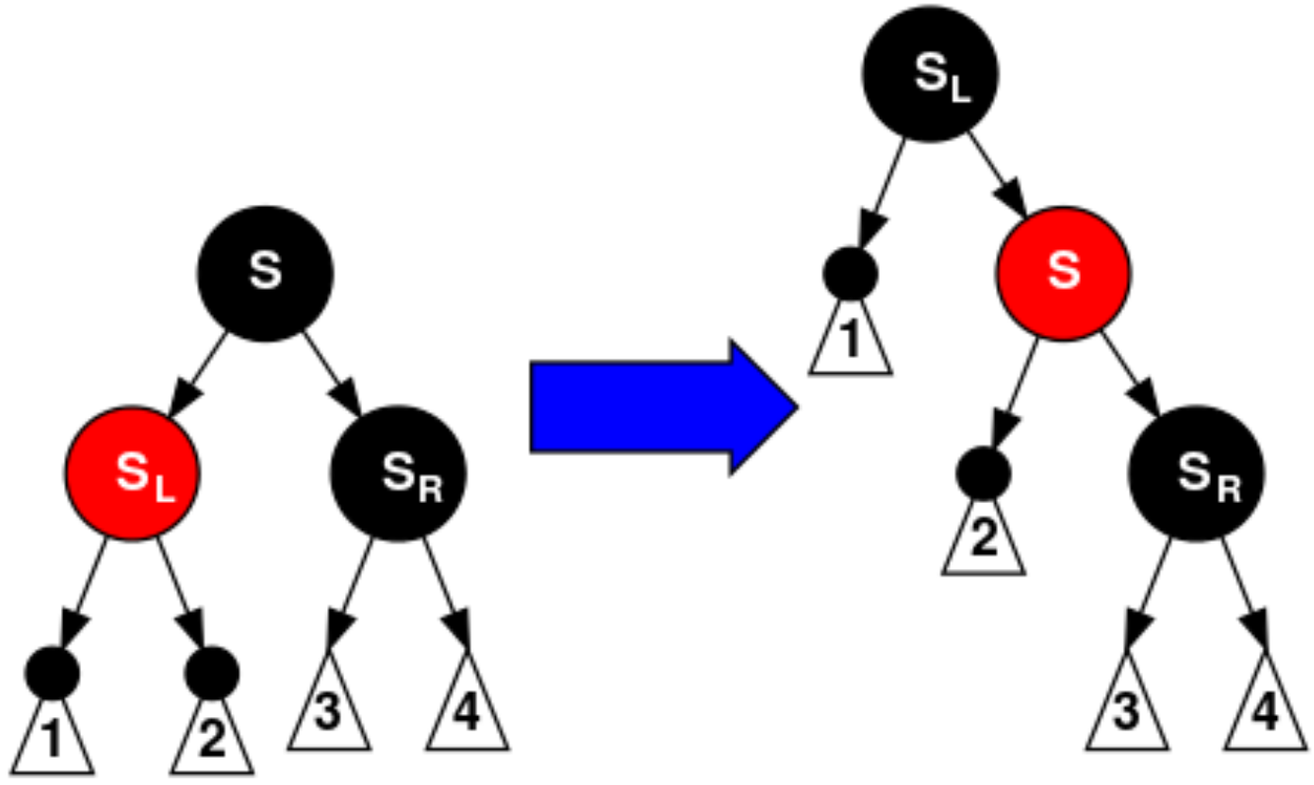
Remoção RB: Caso 4



- Se o nodo (N), o irmão (S) e os filhos do irmão (S_L e S_R) são pretos, e o pai (P) é vermelho,
  - Pinta o pai (P) de preto
  - Pinta o irmão (S) de vermelho
  - Termina o algoritmo

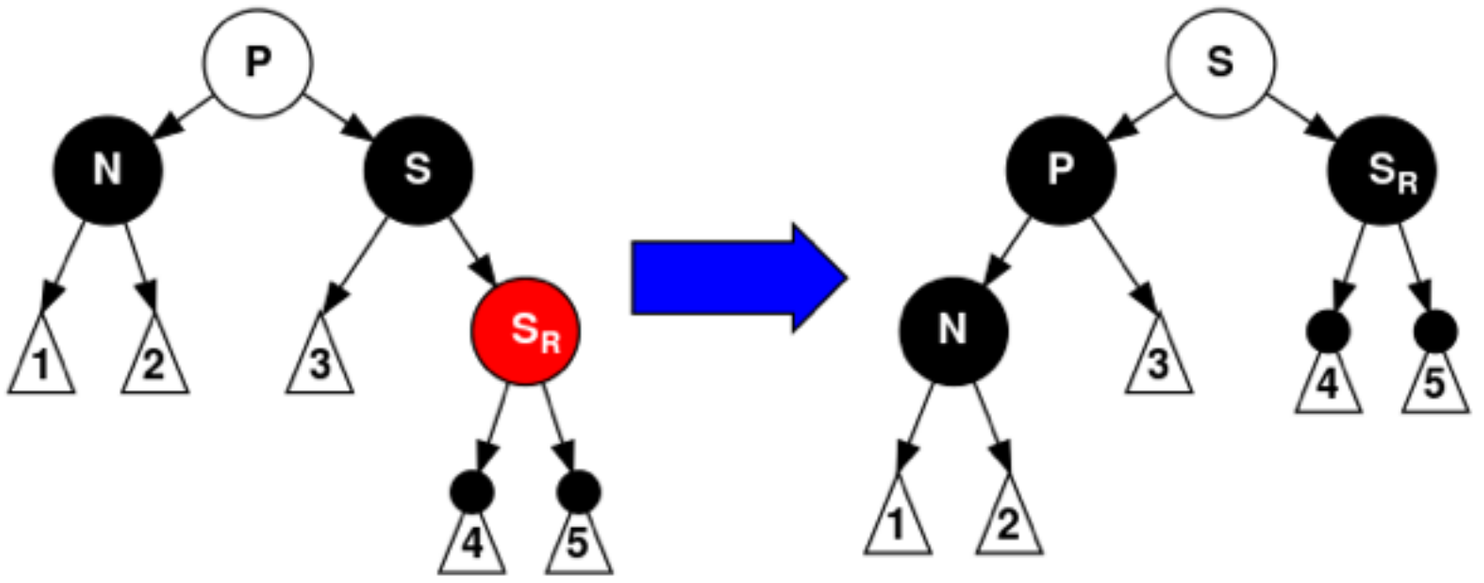
Fecomércio RS | Senac

Remoção RB: Caso 5



- Se N é o filho à esquerda, o irmão (S) e o filho à direita do irmão (S_R) são preto, e o filho à esquerda do irmão (S_L) é vermelho,
  - Pinta S de Vermelho,
  - Pinta SL de Preto,
  - Rotaciona à direita em S
  - Vai para o caso 6
  - (É preciso tratar o caso inverso da árvore, onde N é o filho à direita)

Remoção RB: Caso 6



- Se N é o filho à esquerda e o irmão (S) são pretos, e o filho à direita do irmão (S_R) é Vermelho,
  - Pinta S com a cor do pai (P)
  - Pinta P de Preto,
  - Pinta S_R de Preto,
  - Rotaciona à esquerda em P
  - (É preciso tratar o caso inverso da árvore, onde N é o filho à direita)