RED-BLACK TREES

anaeliza.moura@unicap.br

1

Red-black trees

- **Introdução**
  - Originalmente proposta por Rudolf Bayer em 1972, com o nome de “árvore B binária simétrica.”
  - Adquiriu seu nome atual em 1978 em um artigo proposto por Leonidas Guibas e Robert Sedgewick.

anaeliza.moura@unicap.br

2

Red-black trees

- **Definição**
  - É uma **árvore binária de busca** com um bit extra de armazenamento por nó: a cor do nó, que pode ser vermelha ou preta.
  - Restringindo-se as cores dos nós em qualquer caminho simples da raiz até uma folha, por meio das **propriedades vermelho-preto**, assegura-se que nenhum caminho tenha comprimento maior que duas vezes o de qualquer outro, de modo que a árvore é aproximadamente **balanceada**.

anaeliza.moura@unicap.br

3

Red-black trees

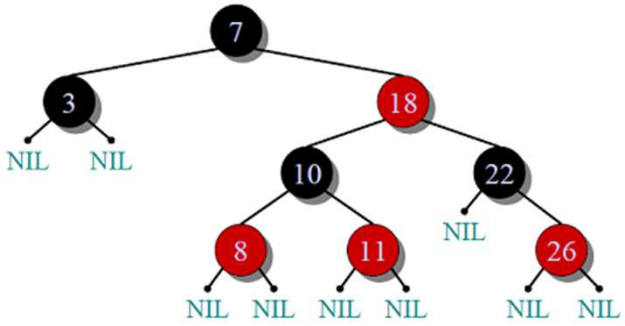
- **Propriedades vermelho-preto**
  - Uma árvore vermelho-preto é uma árvore de busca binária que satisfaz as seguintes propriedades vermelho-preto:
    1. Todo nó é vermelho ou preto.
    2. A raiz é preta (propriedade da raiz).
    3. Toda folha (NULL, sentinela) é preta.
    4. Se um nó é vermelho, então os seus filhos são pretos (propriedade vermelha).
    5. Para cada nó, todos os caminhos simples do nó até folhas descendentes contém o mesmo número de nós pretos (propriedade preta).

anaeliza.moura@unicap.br

4

Red-black trees

- **Exemplo 1**

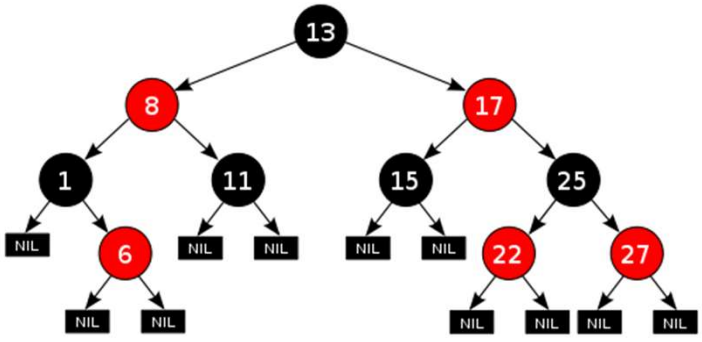


anaeliza.moura@unicap.br

5

Red-black trees

- **Exemplo 2**



anaeliza.moura@unicap.br

6

Red-black trees

- **Características**
  - Folhas
    - Nós folha em uma árvore vermelho-preto não contém dados.
  - Cada nó deve guardar o endereço do pai. Ou seja, cada nó da árvore contém os seguintes atributos: cor, chave, esquerda, direita, pai.

anaeliza.moura@unicap.br

Red-black trees

- **Definições**
  - Altura preta (black-height) de um nó
    - Denominamos o número de nós pretos em qualquer caminho simples de um nó X (não incluindo ele) até uma folha, por altura preta do nó, denotado por bh (x).
  - Altura preta (black-height) de uma árvore
    - Definimos a altura preta de uma árvore vermelho-preto como sendo a altura preta de sua raiz.

anaeliza.moura@unicap.br

Red-black trees

- **Exemplo de aplicações**
  - São utilizadas para implementar a tabela de símbolos em C++, Java, Pythom, BSD Unix e alguns outros sistemas operacionais.

anaeliza.moura@unicap.br

9

Red-black trees

- **Comparação AVL x Red-black trees**
  - As árvores AVL são mais balanceadas se comparadas às árvores vermelho-preto, mas com elas temos que realizar mais rotações durante as operações de inserção e remoção.
  - Se a aplicação envolve várias e freqüentes inserções e remoções, então as árvores vermelho-preto são mais adequadas.
  - Se as inserções e remoções são menos freqüentes e as operações de busca são mais freqüentes, então as árvores AVL são mais adequadas.

anaeliza.moura@unicap.br

10

Red-black trees

- Operações com árvores vermelho-preto
  1. Inserir um novo valor
  2. Buscar um valor
  3. Remover um valor
  4. Exibir todos os valores em uma determinada ordem.
- Como uma árvore vermelho-preto é uma árvore binária de busca, as operações (2) e (4) são herdadas sem alterações.

anaeliza.moura@unicap.br

11

Red-black trees

- Operações com árvores vermelho-preto
  - Como as operações de inserção e remoção modificam a estrutura da árvore, o resultado pode violar as propriedades vermelho-preto.
  - Para restabelecer as propriedades, temos duas possíveis ações:
    - Mudar a cor de alguns nós da árvore (recoloring).
    - Mudar as estruturas dos ponteiros por meio de rotações (rotation).

anaeliza.moura@unicap.br

12

Red-black trees

- Ações para restabelecer as propriedades vermelho-preto
  - Primeiro, tentamos recolorir.
  - Se a ação de recolorir não funcionar, realizamos as rotações.

anaeliza.moura@unicap.br

13

Red-black trees

- **Operação de inserção**
  - O objetivo da operação de inserção é inserir a chave K na árvore T, mantendo as propriedades vermelho-preto.
  - Se T é vazia, inserir um nó preto contendo K.
  - Se T não é vazia:
    - Inserir K em T usando o algoritmo de inserção em uma árvore binária de busca.
    - O nó contendo K deve ser “pintado” de vermelho.
    - Se necessário, executar operações para restabelecer as propriedades vermelho-preto.

anaeliza.moura@unicap.br

14

Red-black trees

- **Operação de inserção**

  Lembrete: em uma ABB, novos nós sempre são inseridos na condição de folha.

  – Adicionar um nó folha não irá afetar a propriedade da raiz.

  – Adicionar uma folha vermelha não irá afetar a propriedade preta.

  – Adicionar uma folha vermelha pode afetar a propriedade vermelha, caso isso aconteça, deve-se restabelecer a propriedade.

  – Ao restabelecer a propriedade vermelha, devemos garantir que a propriedade da raiz e a propriedade preta não sejam violadas.

anaeliza.moura@unicap.br

15

Red-black trees

- **Operação de inserção**

  Seja K o nó inserido e seja P o pai de K

  Caso 1: P é preto

  Se P é preto, a propriedade vermelha não é violada e não há nada a ser feito.

  Caso 2: P é vermelho

  Se P é vermelho, ele agora tem um filho vermelho, o que viola a propriedade vermelha. Devemos, então, restabelecer esta propriedade.

anaeliza.moura@unicap.br

16

Red-black trees

- **Operação de inserção – reestruturação**

  Seja G o avô de K e seja S o tio de K.

  Caso 2a: O tio S de K é preto ou nulo.

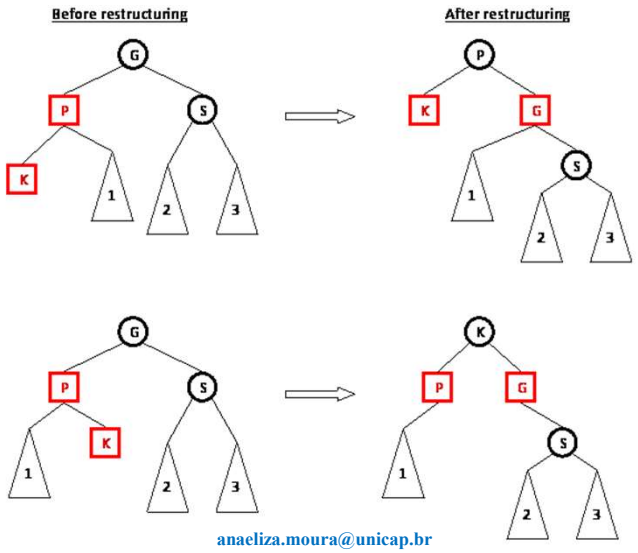
  Se o tio S de K é preto ou nulo, então faremos uma reestruturação envolvendo os nós K, P e G.

  Dependendo da configuração dos nós K, P e G, teremos quatro possibilidades.

anaeliza.moura@unicap.br

17

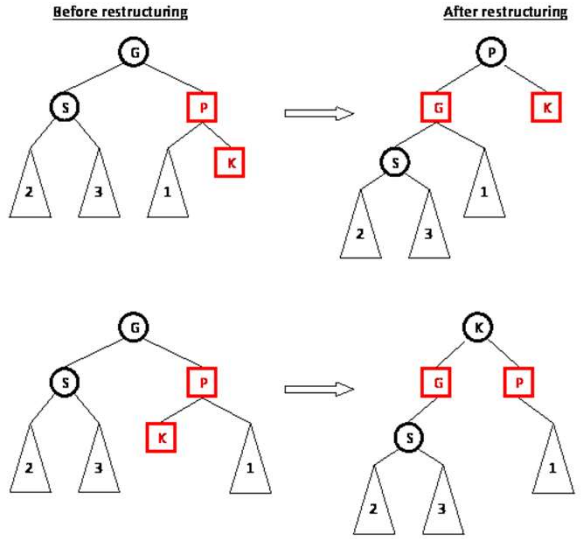
Rotações à Direita



anaeliza.moura@unicap.br

18

Rotações à Esquerda



anaeliza.moura@unicap.br

Red-black trees

- **Operação de inserção**

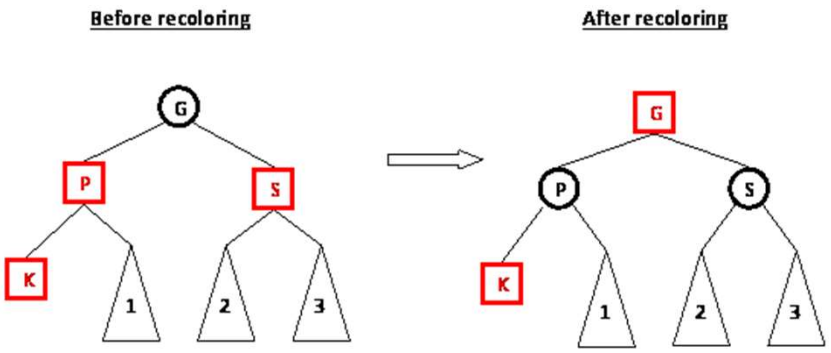
  Seja G o avô de K e seja S o tio de K.

  Caso 2b: O tio S de K é vermelho.

  Se o tio S de K é vermelho, temos que recolorir P, S e G: mudamos a cor de P e S para preto e a cor de G para vermelho (a menos que G seja a raiz, porque, se for G for a raiz, deixamos sua cor preta, para preservar a propriedade preta).

anaeliza.moura@unicap.br

Red-black trees



21

Red-black trees

- **Recolorir**
  - Recolorir não afeta a propriedade preta:o número de nós pretos no caminho não muda. Mas, recolorir pode introduzir uma quebra da propriedade vermelha. O que deve ser tratado.
  - Ou seja, a operação de recoloração é propagável.

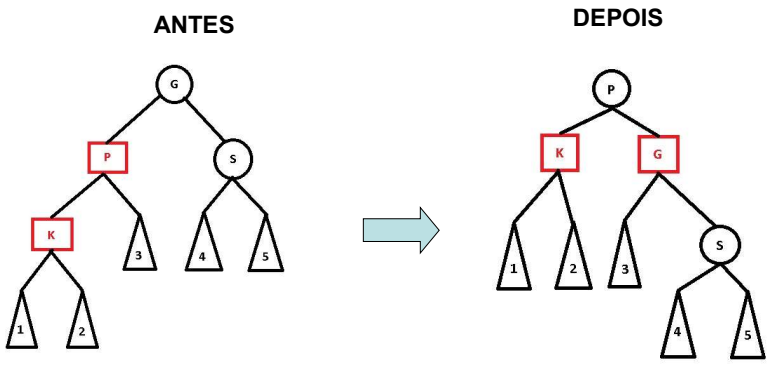
22

Red-black trees

- **Propagação da recoloração**
  - A propagação da recoloração pode ser outra recoloração ou uma rotação.
  - No caso de ser uma rotação, precisamos definir novos K, P, G e S.
  - O nó G passa a ser o novo K.
  - Redefinimos P, G e S a partir do novo K.

anaeliza.moura@unicap.br

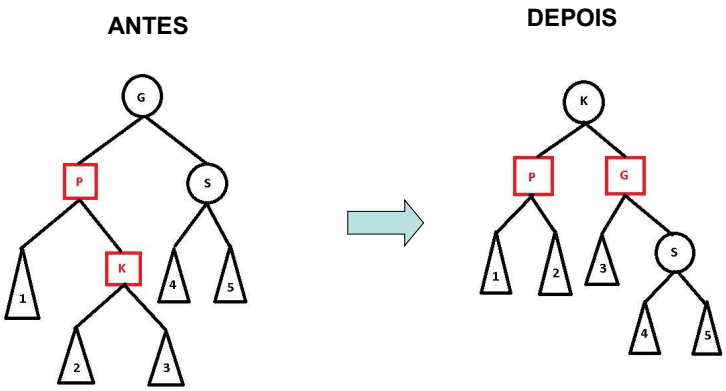
Rotação Simples à Direita



(*) Na propagação

anaeliza.moura@unicap.br

Rotação Dupla à Direita

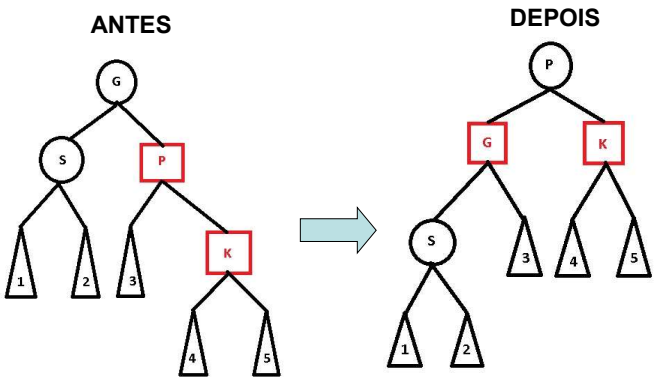


(*) Na propagação

anaeliza.moura@unicap.br

25

Rotação Simples à Esquerda

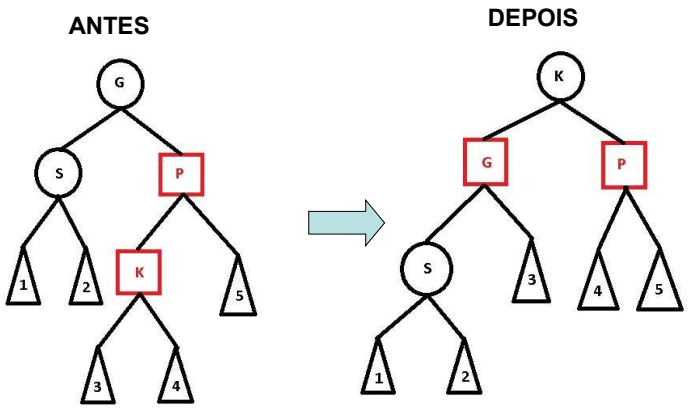


(*) Na propagação

anaeliza.moura@unicap.br

26

Rotação Dupla à Esquerda



(*) Na propagação

anaeliza.moura@unicap.br

27

Red-black trees

- Operação de Remoção
  - Remoção preguiçosa (virtual)
  - Remoção efetiva (física)

anaeliza.moura@unicap.br

28

Red-black trees

- **Operação de Remoção**
  - **Remoção preguiçosa (virtual)**
    - O nó é marcado como removido
    - Nenhuma remoção é efetivamente realizada
    - Não muda a estrutura da árvore
    - Não necessita de operações de reestruturação
    - Necessita de algoritmos de inserção e busca que tratem o nó como “ausente”.
    - A adoção desta solução é possível quando as árvores são usadas no contexto de uma aplicação com poucas operações de remoção.

anaeliza.moura@unicap.br

29

Red-black trees

- **Operação de Remoção**
  - **Remoção efetiva (física)**
    - **Passo 1**: Remover o nó utilizando o algoritmo de remoção de árvore binária de busca.
    - **Passo 2**: Verificar se alguma propriedade vermelho-preta foi violada. Em caso afirmativo, a árvore deve ser rebalanceada.

anaeliza.moura@unicap.br

30

Red-black trees

- **Operação de Remoção**:
  - Remoção: Considere que o elemento a ser removido:
    - **Caso 1**: Não possui filhos: remover.
    - **Caso 2**: Possui um único filho: remover, substituindo-o por seu filho.
    - **Caso 3**: Possui dois filhos: substituir o elemento a ser removido por seu antecessor (maior elemento na subárvore esquerda e remover o antecessor.

anaeliza.moura@unicap.br

Red-black trees

- **Operação de Remoção Efetiva**

  Remoção de nó vermelho

  Se o nó efetivamente removido for vermelho, todas as propriedades da árvore vermelho-preto serão preservadas e nenhuma operação de reestruturação será necessária.

anaeliza.moura@unicap.br

Red-black trees

- **Operação de Remoção Efetiva**
  - **Remoção de nó preto**
    - Se o nó efetivamente removido for preto, a quantidade de nós pretos em pelo menos um dos caminhos da árvore será alterada, o que implica que algumas operações de rotação e/ou alteração de cor sejam feitas para manter o balanceamento da árvore.

anaeliza.moura@unicap.br