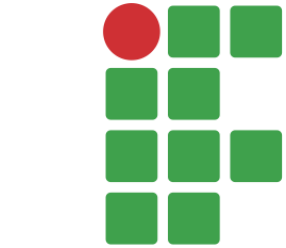
ÁRVORES
AVL – B – PV



INSTITUTO
FEDERAL
Paraíba

Professor Msc Paulo de Tarso F. Júnior

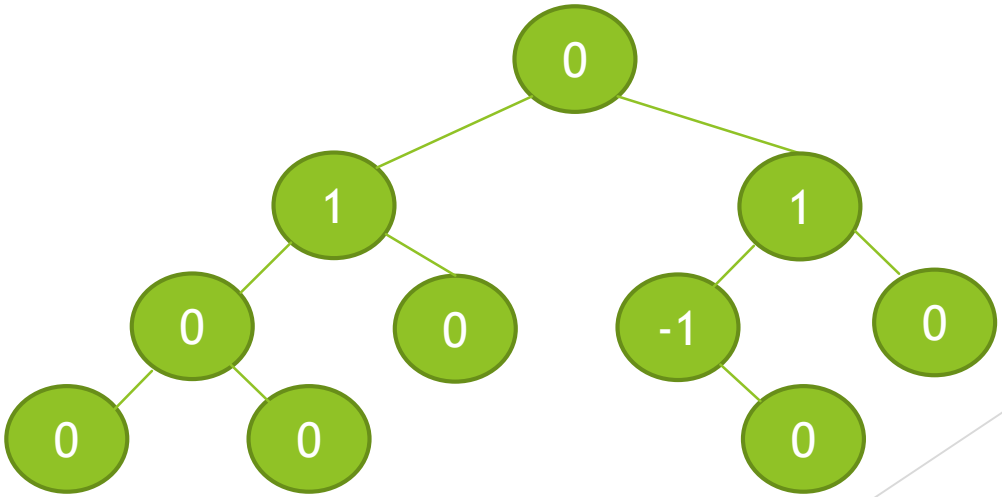
paulodt@gmail.com

Árvores AVL

- A altura de uma árvore binária é o nível máximo de suas folhas (profundidade)
- Uma árvore binária balanceada (AVL) é uma árvore binária na qual as alturas das duas sub-árvores de todo nó nunca difere em mais de 1.
- O balanceamento de um NÓ é definido como a altura de sua sub-árvore esquerda menos a altura de sua subárvore direita.

Árvores AVL

- Cada nó numa árvore binária balanceada (AVL) tem balanceamento de 1, -1 ou 0.
- Se o valor do balanceamento do nó for diferente de 1, -1 e 0. Essa árvore não é balanceada (AVL).
- Observe o exemplo a seguir:

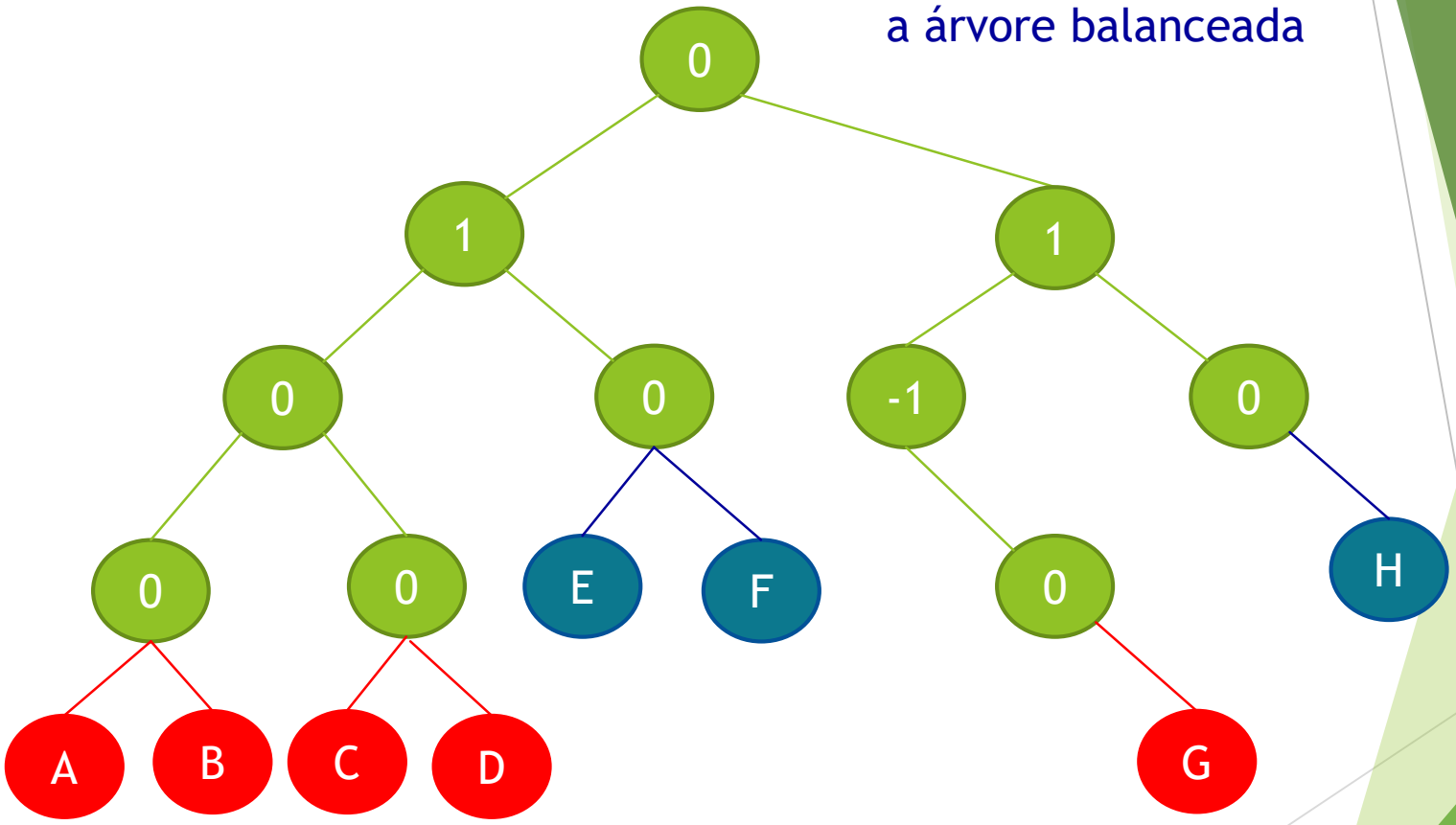


Árvores AVL

- Se a probabilidade de pesquisar um dado for a mesma para todos os dados, uma árvore binária balanceada determinará a busca mais eficiente.
- Mas os algoritmos de inserção e remoção já vistos até agora não garantem que essa árvore permanecerá balanceada.

Árvores AVL

Inserções que mantém a árvore balanceada



Inserções que desbalanceam a árvore

Árvores AVL

- O desbalanceamento ocorre quando:
- O NÓ inserido é um descendente esquerdo de um nó que tinha balanceamento de 1 (Nós de A-D)
- Se ele for um descendente direito de um nó que tinha balanceamento de –1 (Nó G).

Árvores AVL

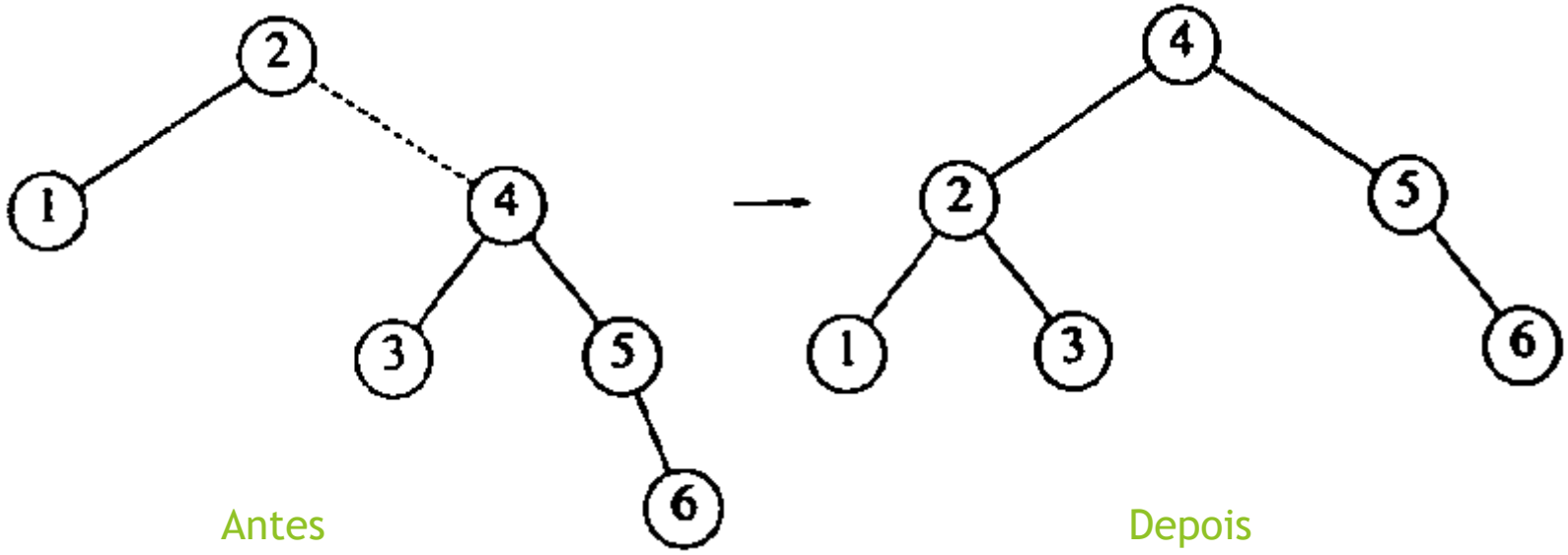
- Para manter uma árvore balanceada, é necessário fazer uma transformação na árvore tal que:

1. O percurso em ordem da árvore transformada seja o mesmo da árvore original (isto é, a árvore transformada continue sendo um árvore de busca binária)
2. A árvore transformada fique balanceada.

Árvores AVL

- A transformação a ser feita na árvore tal que ela se mantenha balanceada é chamada de rotação.
- A rotação poderá ser feita à esquerda ou à direita dependendo do desbalanceamento que tiver que ser solucionado.
- A rotação deve ser realizada de maneira que as regras 1 e 2 do slide anterior sejam respeitadas.
- Dependendo do desbalanceamento a ser solucionado, apenas uma rotação não será suficiente para resolvê-lo.

Árvores AVL



Antes

Depois

Árvores AVL

- Para o rebalanceamento da árvore é necessário calcular o Fator de Balanceamento para verificar qual rotação deve ser efetuada afim de rebalanceá-la.
- FB = h da sub-árvore direita - h da sub-árvore esquerda
- Se FB é negativo, as rotações são feitas à direita
- Se FB é positivo, as rotações são feitas à esquerda

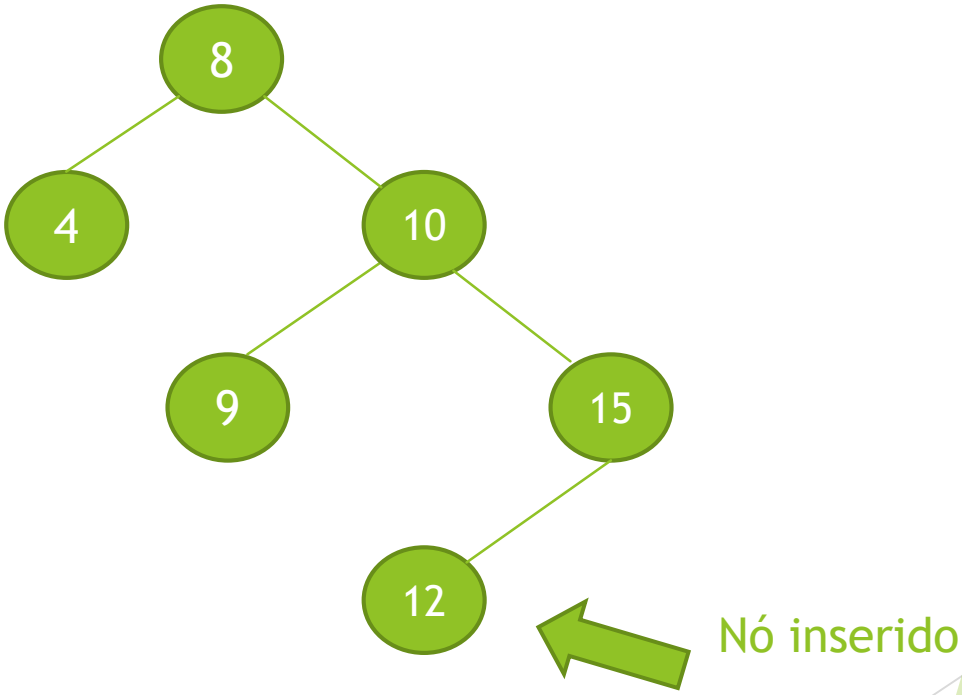
Árvores AVL

- Há dois tipos de ocorrências nos casos de balanceamento:
- **Caso1:** Nó raiz com FB 2 ou –2 com um filho (na direção de onde houve a inserção) com FB 1 ou –1 com o mesmo sinal, neste caso a solução é uma rotação simples

Árvores AVL

FB(raiz) = Hd – He
3 - 1 = 2

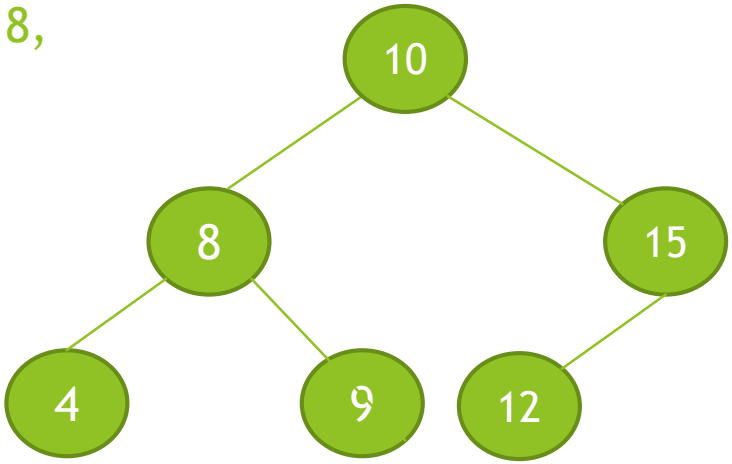
FB(10) = Hd – He
2 - 1 = 1



Solução: rotação à esquerda do nó 8, ou raiz.

Árvores AVL

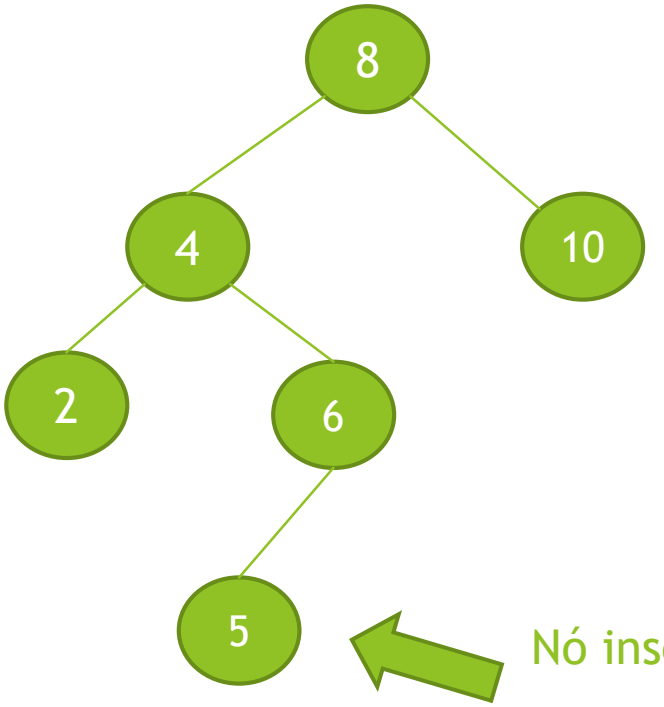
Solução: rotação à esquerda do nó 8, ou raiz.



Árvores AVL

- **Caso 2:** Nó raiz com FB 2 ou –2 com um filho (na direção de onde houve a inserção) com FB -1 ou 1 os quais possuem sinais trocados, neste caso a solução é uma rotação dupla.
- Primeiro rotaciona-se o nó com fator de balanceamento 1 ou –1 na direção apropriada e depois rotaciona-se o nó cujo fator de balanceamento seja 2 ou –2 na direção oposta.

Árvores AVL

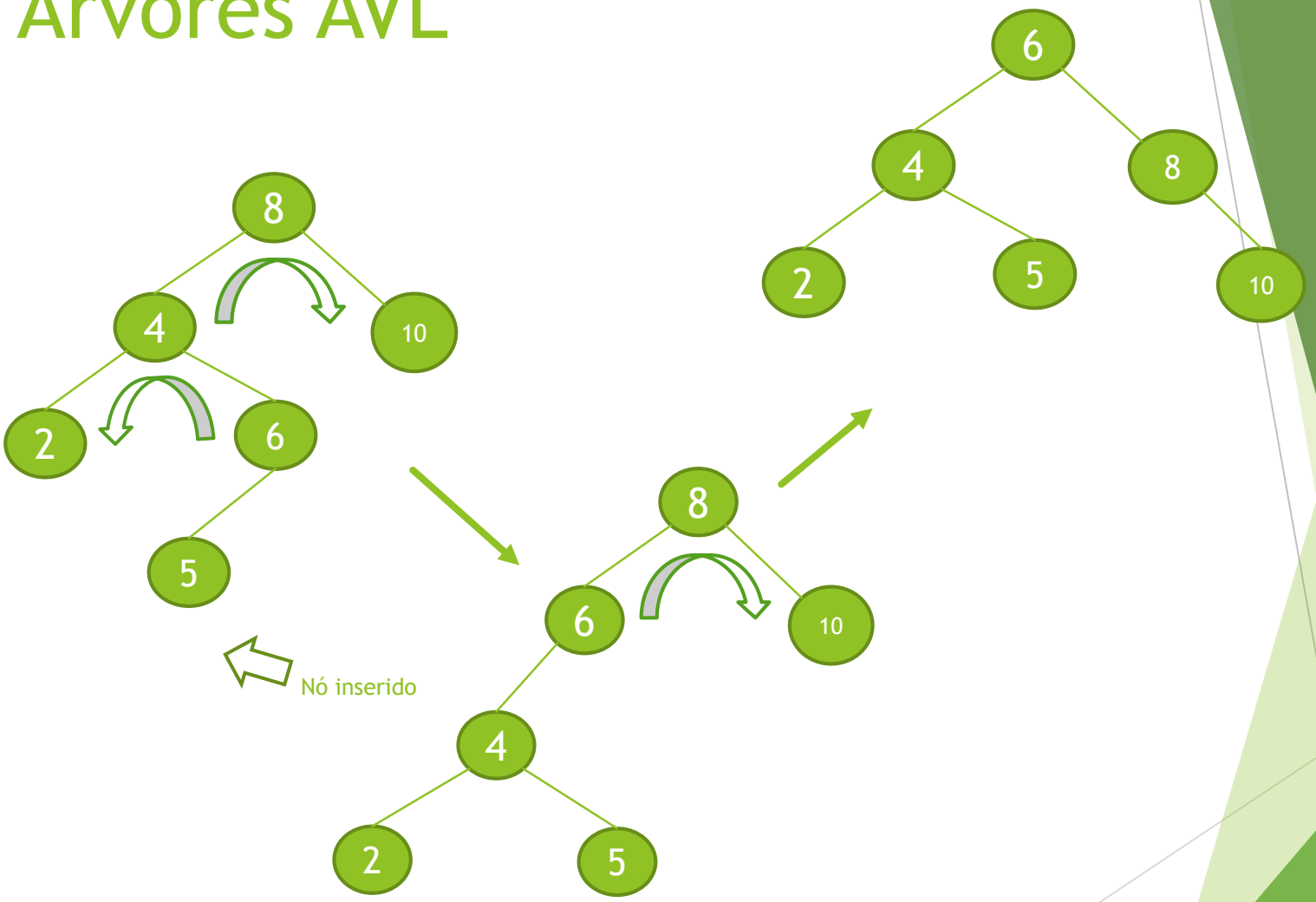


FB(raiz) = Hd – He
1 - 3 = -2

FB(4) = Hd – He
2 - 1 = 1

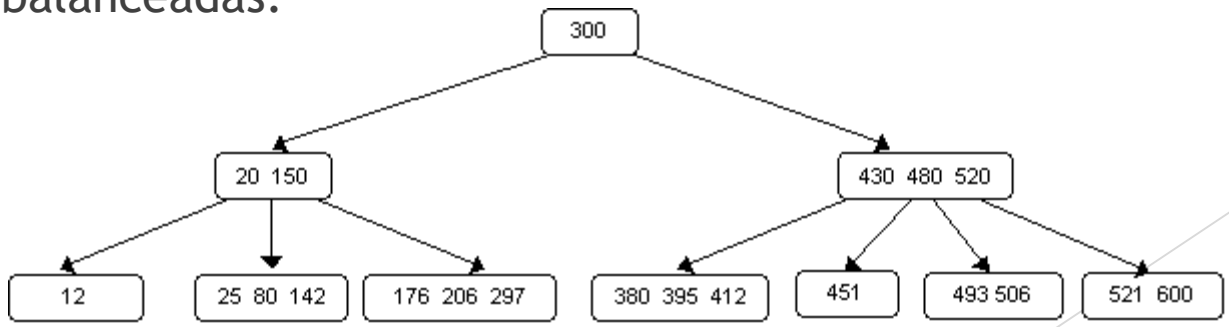
Solução: rotação do nó 4 à esquerda e rotação do nó 8 a direita.

Árvores AVL



Árvores B – Definição

- São árvores balanceadas, desenvolvidas para otimizar o acesso a armazenamento secundário
- Os nós da árvore B podem ter muitos filhos. Esse fator de ramificação é determinante para reduzir o número de acessos a disco.
- Árvores B são balanceadas.
- Arvores B são generalizações de árvores binárias balanceadas.



Árvores B – Definição

- Atualmente o armazenamento estável é feito em discos magnéticos, e o custo de cada acesso (da ordem de mili segundos) é muito alto quando comparado ao acesso `a memória RAM (ordem de nano segundos)
- Toda vez que um acesso é feito, deve-se aproveita-lo da melhor maneira possível, trazendo o máximo de informação relevante.
- A quantidade de dados utilizados numa árvore B obviamente não cabem na memória de uma só vez, por isso é necessário paginá-la

Árvores B – Definição

- Especializações são feitas de acordo com as necessidades da aplicação. O fator de ramificação, por exemplo, pode variar de 3 a 2048 por exemplo (dependendo do buffer dos discos e do tamanho das páginas de memória alocados pelo SO)
- Na grande maioria dos sistemas, o tempo de execução de um algoritmo de árvore-B é determinado pelas leituras e escritas no disco Um fator de ramificação alto reduz drasticamente a altura da arvore. Tomemos o exemplo:

Árvores B – Definição

- Seja T uma árvore-B com raiz (root[T]). Ela possuirá então as seguintes propriedades:
- 1. Todo o nó X tem os seguintes campos:
  - a. n[x], o número de chaves atualmente guardadas no nó x,
  - b. As n[x] chaves, guardadas em ordem crescente, tal que key1[x] ≤ key2[x] ≤ ⋯ ≤ keyn[x] [x],
  - c. leaf [x], Um valor booleano, TRUE se x é uma folha e FALSE se x é um nó interno.

Árvores B - Definição

- 2. Cada nó interno x também contém n[x] + 1 apontadores c1[x], c2[x], ...,cn[x]+1[x] para os filhos. As folhas tem seu apontador nulo
- 3. As chaves keyi [x] separam os intervalos de chaves guardadas em cada sub-árvore: se ki é uma chave guardada numa sub-árvore com raiz ci [x], então: k1 ≤ key1[x] ≤ k2 ≤ key2[x] ≤ ... ≤ keyn[x] [x] ≤ kn[x]+1
- 4. Todas as folhas têm a mesma profundidade, que é a altura da árvore: h

Árvores B - Definição

- 5. Existem limites superiores e inferiores para o número de chaves num nó. Estes limites podem ser expressados em termos de um inteiro fixo t ≥ 2 chamado grau mínimo:
  - a. Todo nó que não seja raiz deve ter pelo menos t −1 chaves. Todo nó interno, que não a raiz, tem portanto t filhos. Se a árvore for não vazia, a raiz deve ter pelo menos uma chave
  - b. Cada nó pode conter no máximo 2t − 1 chaves. Portanto, um nó interno, pode ter no máximo 2t filhos. O nó é considerado cheio quando ele contém exatamente 2t − 1 chaves

Árvores B - Operações

- ***Busca:*** Similar à uma árvore binária, só que ao invés de uma bifurcação em cada nó, temos vários caminhos a seguir de acordo com o número de filhos do nó.
- ***Inserção:*** É um pouco mais complicadas que em árvores binárias, pois, precisamos inserir a nova chave no nó correto da árvore, sem violar suas propriedades
  - Caso o nó esteja cheio, deve ser realizado um ***split*** (separação) o nó ao redor do elemento mediano.
- ***Remoção:*** Também é um pouco mais complexa, uma vez que devemos garantir que ele não torne-se pequeno demais, ou seja, deve sempre ter pelo menos t − 1 elementos.
  - Existem 6 casos possíveis para remoção em uma árvore B.

Árvores Preto Vermelho

- As árvores Vermelho-preto são árvores binárias de busca
- Também conhecidas como Rubro-negras ou Red-Black Trees
- Foram inventadas por Bayer sob o nome “Árvores Binárias Simétricas” em 1972, 10 anos depois das árvores AVL
- As árvores vermelho-preto possuem um bit extra para armazenar a cor de cada nó, que pode ser VERMELHO ou PRETO

Árvores Preto Vermelho

- Além deste, cada nodo será composto ainda pelos seguintes campos:
  - valor (os “dados” do nodo)
  - fe (filho esquerdo)
  - fd (filho direito)
  - pai
- De maneira simplificada, uma árvore rubro-negra é uma árvore de busca binária que insere e remove de forma inteligente, para assegurar que a árvore permaneça aproximadamente balanceada.

Árvores Preto Vermelho - Propriedades

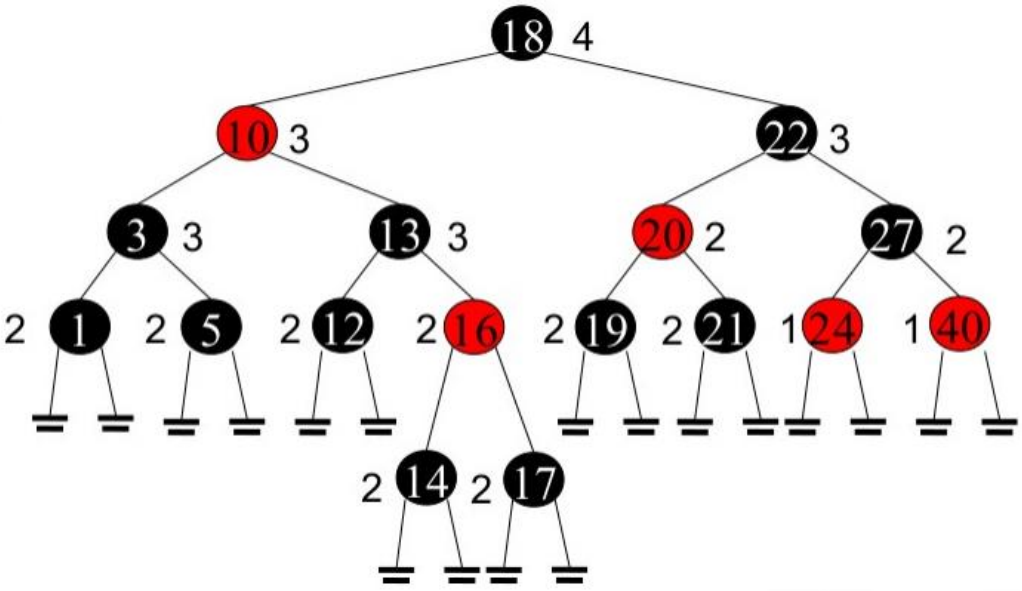
- 1. Todo nó é vermelho ou preto
- 2. A raiz é preta
- 3. Toda folha (Nil) é preta
- 4. Se um nó é vermelho, então os seus filhos são pretos
- 5. Para cada nó, todos os caminhos do nó para folhas descendentes contém o mesmo número de nós PRETOS.
- Quanto à raiz: Ser sempre preta é uma regra usada em algumas definições. Como a raiz pode sempre ser alterada de vermelho para preto, mas não sendo válido o oposto, esta regra tem pouco efeito na análise

Árvores Preto Vermelho - Propriedades

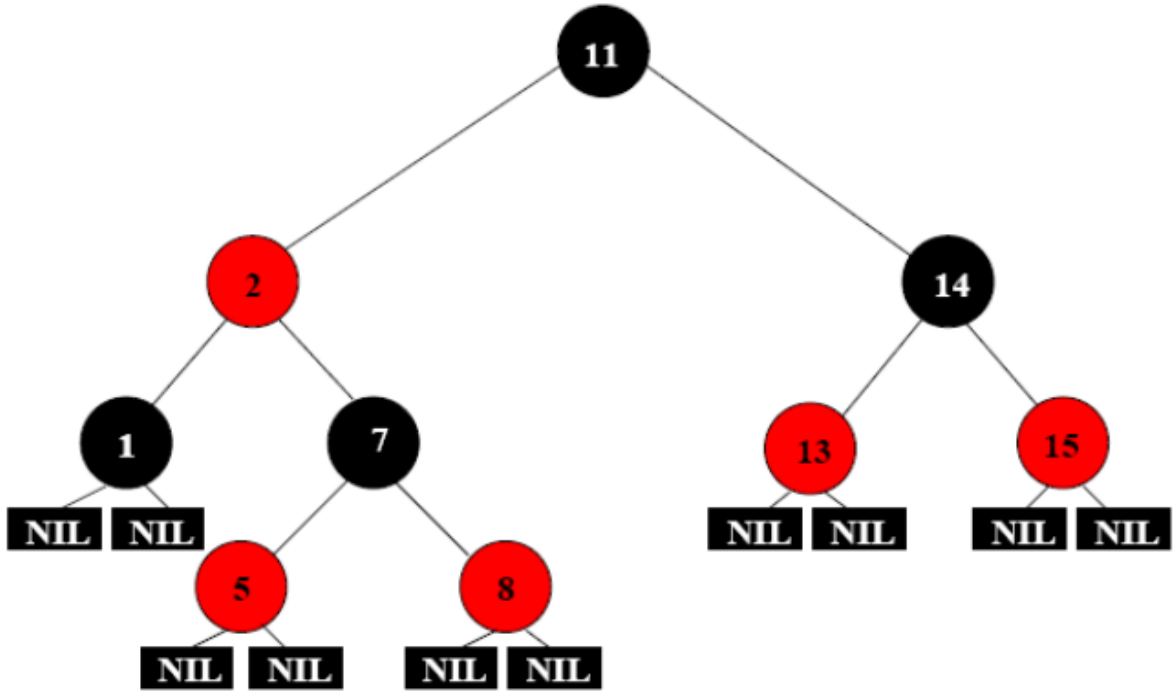
- Um nó que satisfaz os itens acima é denominado equilibrado, caso contrário é dito desequilibrado.
- Em uma árvore rubro-negra todos os nós estão equilibrados
- Uma condição óbvia obtida das propriedades é que num caminho da raiz até uma sub-árvore vazia não pode existir dois nós rubros consecutivos

Árvores Preto Vermelho - Propriedades

- Altura negra: é número de nós negros encontrados até qualquer nó folha descendente



Árvores Preto Vermelho - Propriedades



Árvores Preto Vermelho - Propriedades

- Cada vez que uma operação for realizada na árvore, o conjunto de propriedades é testado
- Caso alguma não seja satisfeita, são realizadas rotações e/ou ajustes de cores, de forma que a árvore permaneça balanceada

Árvores Preto Vermelho - Operações

- As operações Inserir e Remover são mais complicadas nas Árvores Rubro-Negras porque elas podem ferir alguma propriedade deste tipo de árvore.
- Como veremos, essas operações podem ser implementadas de forma bastante parecida com as respectivas operações nas Árvores Binárias de Busca, bastando apenas modificar as cores dos nós para que as propriedades de Árvores Rubro-Negras sejam satisfeitas.
- Como a inserção e a remoção propriamente ditas já foram vistas para Árvores Binárias de Busca, veremos apenas o que é necessário para acertar as cores da árvore.

Árvores PV - Inserção

- Um nó é inserido sempre na cor vermelha, assim, não altera a altura negra da árvore
- Se o nodo fosse inserido na cor preta, invalidaria a propriedade 5, pois haveria um nodo preto a mais em um dos caminhos

Dúvidas

Referências

- Aaron M. Tenenbaum, Yedidyah Langsam, Moshe J. Augenstein ***Estruturas de Dados Usando C***. Pearson (26 de junho de 1995)
- Backes A. ***Estrutura de Dados Descomplicada em Linguagem C***. Elsevier; Edição: 1ª (9 de agosto de 2016)
- Programar em C/Árvores Binárias. Disponível em: https://pt.wikibooks.org/wiki/Programar_em_C/%C3%81rvores_bin%C3%A1rias#Arvore_bin.C3.A1ria. Acesso em: 03/09/2017.
- Árvores Binárias. Disponível em: https://www.ime.usp.br/~pf/algoritmos/aulas/bint.html. Acesso em: 05/09/2017.