Estrutura de Dados

Árvore AVL

Prof. Dr. Daniel Vecchiato

Agenda

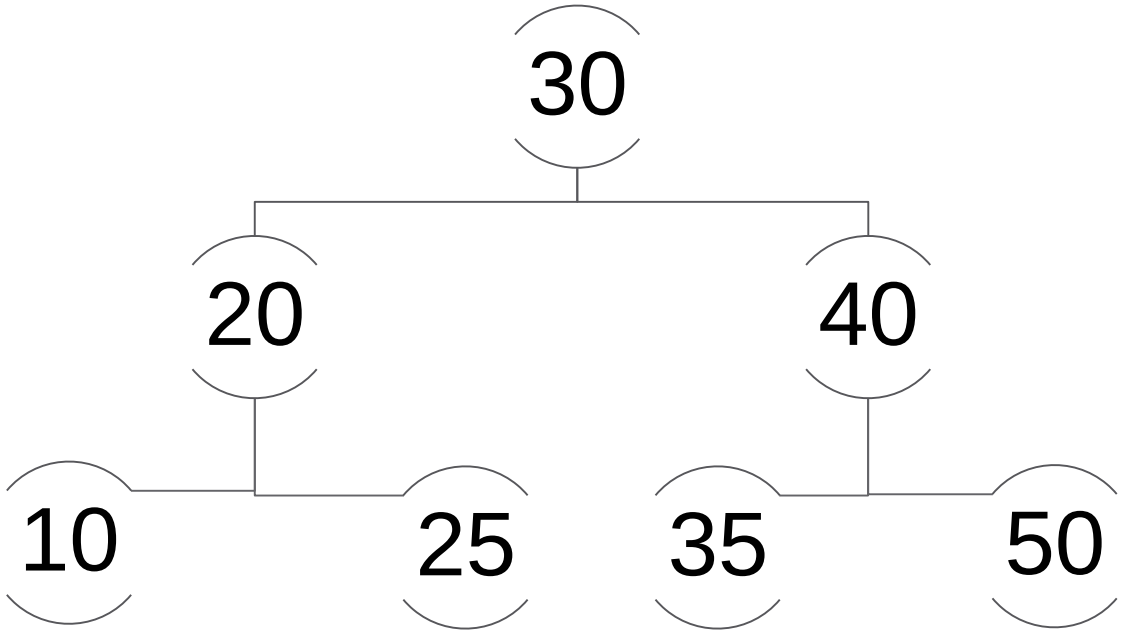
- Introdução
- Árvore AVL
  - Definição
  - Fator de balanceamento
  - Inserção/Remoção
  - Rotações
- Análise
- Exercícios

Introdução

- Como ficaria uma árvore binária se inserir os seguintes elementos:
  - 30, 20, 40, 10, 25, 35 e 50

Introdução

- Como ficaria uma árvore binária se inserir os seguintes elementos:
  - 30, 20, 40, 10, 25, 35 e 50

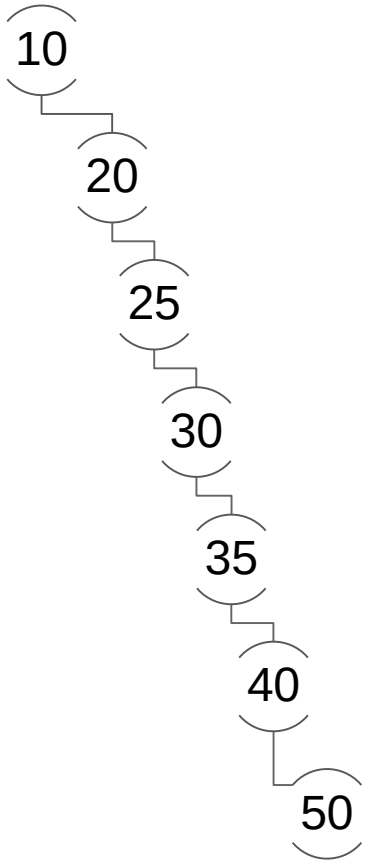


Introdução

- E se inserir os mesmos elementos, mas nesta ordem
  - 10, 20, 25, 30, 35, 40, 50

Introdução

- E se inserir os mesmos elementos, mas nesta ordem
  - 10, 20, 25, 30, 35, 40, 50



Introdução

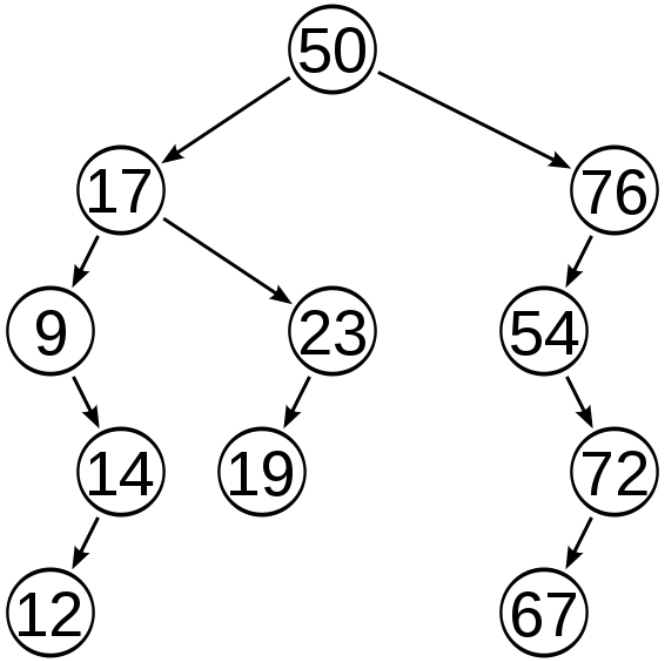
- Uma árvore binária não balanceada tem um desempenho ruim para as pesquisas
  - Com 10.000 nós, são necessárias em média 5.000 comparações se a árvore não estiver balanceada
  - E apenas 14 comparações na árvore balanceada
- O desempenho da árvore binária depende da ordem em que os elementos são inseridos
- Normalmente, não é possível prever e/ou alterar a ordem em que os elementos são inseridos
- Como alternativa, podemos balancear a árvore sempre que necessário

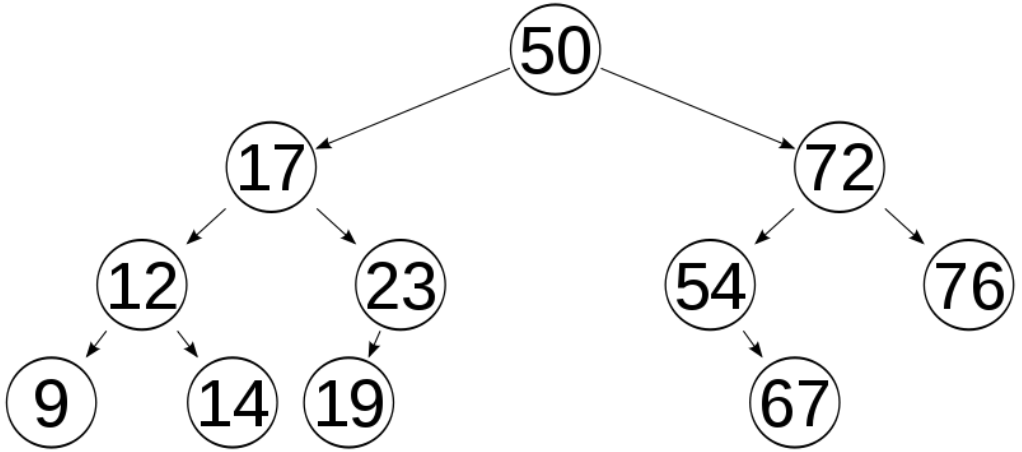
Árvore AVL

- O nome vem do nome dos seus dois criadores
  - Adel’son-Vel’skii e Landis
- Algoritmo de balanceamento de árvores binárias
- Definição
  - Árvore binária que, para qualquer nó interno, a diferença das alturas dos seus filhos é no máximo 1

Árvore AVL

- Definição
  - Árvore binária que, para qualquer nó interno, a diferença das alturas dos seus filhos é no máximo 1
  - Qual a altura de cada nó?
  - Qual árvore é AVL?





Árvore AVL

- Fator de balanceamento (FB)
  - O FB de um nó é a diferença entre a altura da sub-árvore esquerda em relação à sub-árvore direita

FB(p) = altura(sub-árvore esquerda de p) - altura(sub-árvore direita de p)

  - Em uma árvore binária balanceada todos os FB de todos os nós estão no intervalo -1 <= FB <= 1

Árvore AVL

- Fator de balanceamento (FB)
  - Altura

```
int altura (TNo* pRaiz) {
     int iEsq, iDir;

     if (pRaiz == NULL)
            return -1;

     iEsq = altura (pRaiz->pEsq);
     iDir = altura (pRaiz->pDir);

     if (iEsq > iDir)
            return iEsq + 1;
     else
            return iDir + 1;
}
```

Árvore AVL

- Fator de balanceamento (FB)
  - FB

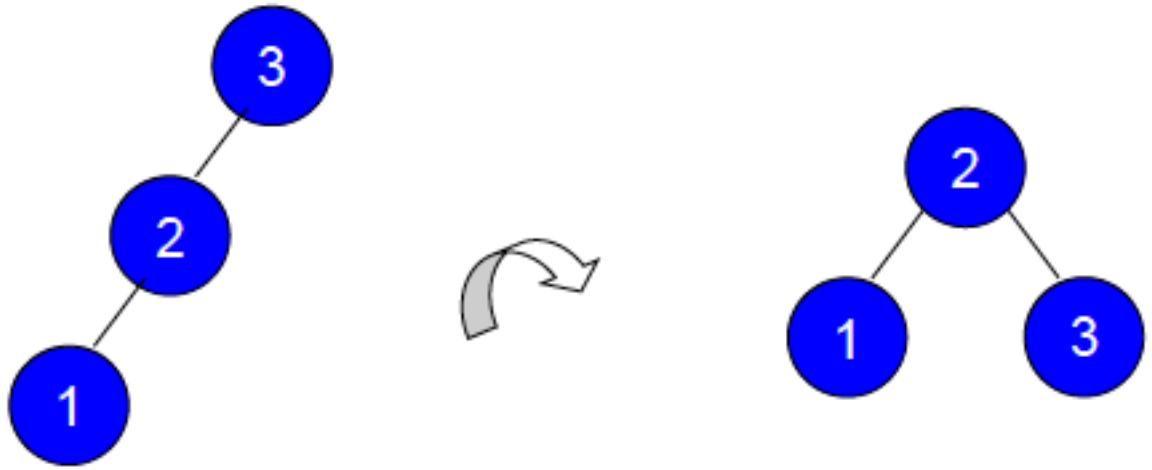
```
int fb (TNo* pRaiz) {
     if (pRaiz == NULL)
             return 0;

     return altura (pRaiz->pEsq) - altura (pRaiz->pDir);
}
```

Árvore AVL

- Inserção/Remoção
  - Realizar a inserção normalmente
  - Caso a árvore não fique balanceada, restaurar o balanceamento por meio de rotações no nó pivô
    - Nó pivô é aquele que possui FB fora do intervalo
    - Há quatro possíveis casos de rotações
      - rotação simples para a direita
      - rotação simples para a esquerda
      - rotação dupla para a direita
      - rotação dupla para a esquerda

Árvore AVL

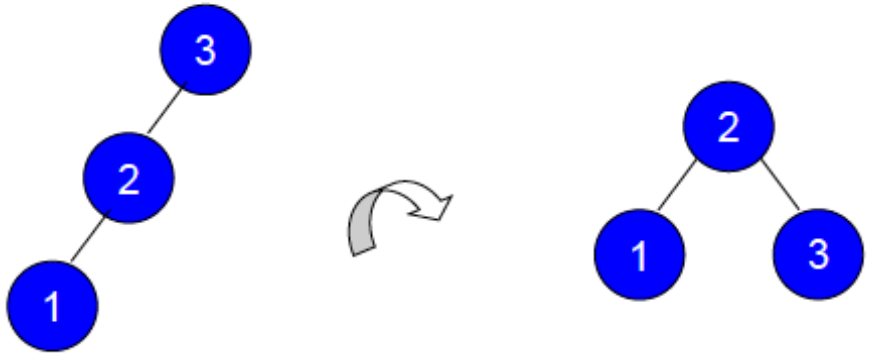
- Rotações
  - 1º caso – rotação simples para a direita
    - FB > 1 (subárvore da esquerda maior que subárvore da direita)
    - Subárvore da esquerda desta subárvore esquerda é maior ou igual que a subárvore da direita



Árvore AVL

- Rotações
  - 1º caso – rotação simples para a direita

```
TNo* rotacaoSimplesDireita (TNo* pR) {
     TNo *pAux = pR->pEsq;
     pR->pEsq = pAux->pDir;
     pAux->pDir = pR;
     return pAux;
}
```



Árvore AVL

- Rotações
  - 2º caso – rotação simples para a esquerda
    - FB < -1 (subárvore da esquerda menor que subárvore da direita)
    - Subárvore da direita desta subárvore direita é maior ou igual que a subárvore da esquerda

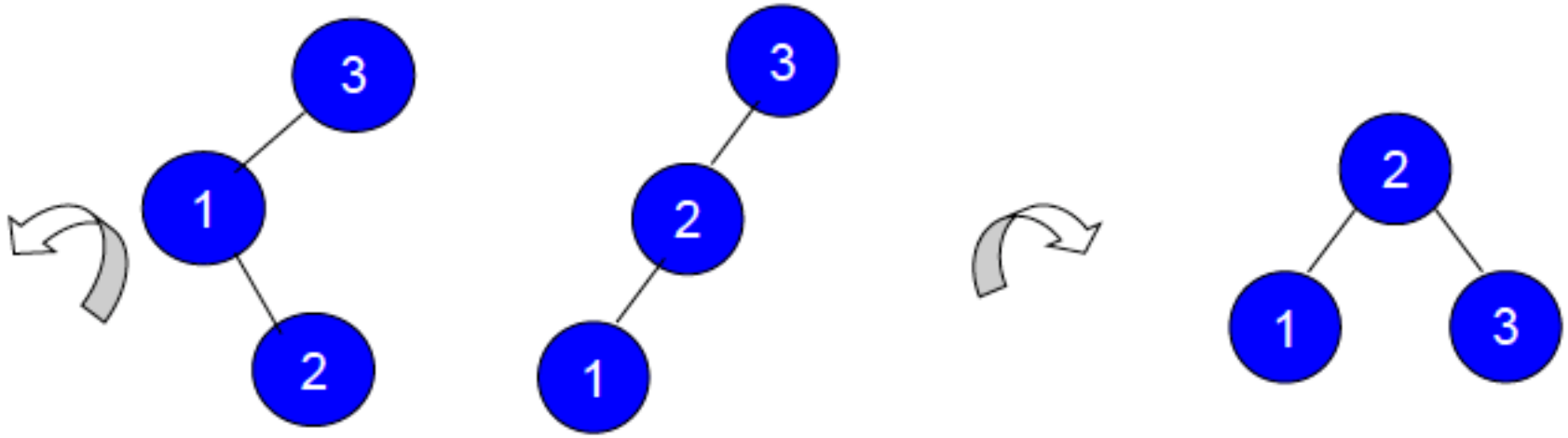
Árvore AVL

- Rotações
  - 2º caso – rotação simples para a esquerda

```
TNo* rotacaoSimplesEsquerda (TNo* pR) {
     TNo *pAux = pR->pDir;
     pR->pDir = pAux->pEsq;
     pAux->pEsq = pR;
     return pAux;
}
```

Árvore AVL

- Rotações
  - 3º caso – rotação dupla para a direita
    - FB > 1 (subárvore da esquerda maior que subárvore da direita)
    - Subárvore da esquerda desta subárvore esquerda é **menor** que a subárvore da direita

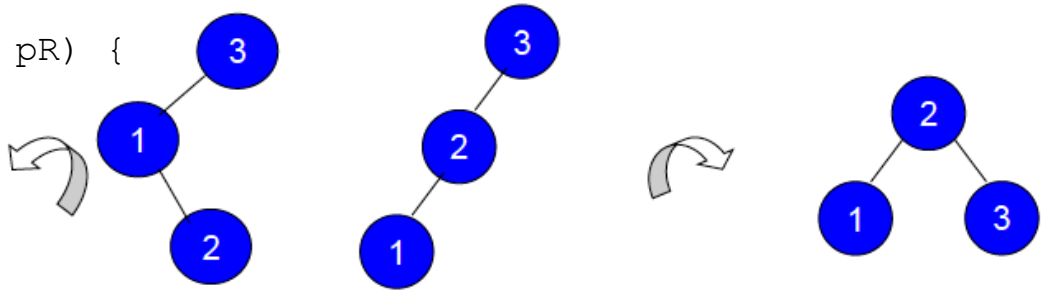


Árvore AVL

- Rotações
  - 3º caso – rotação dupla para a direita

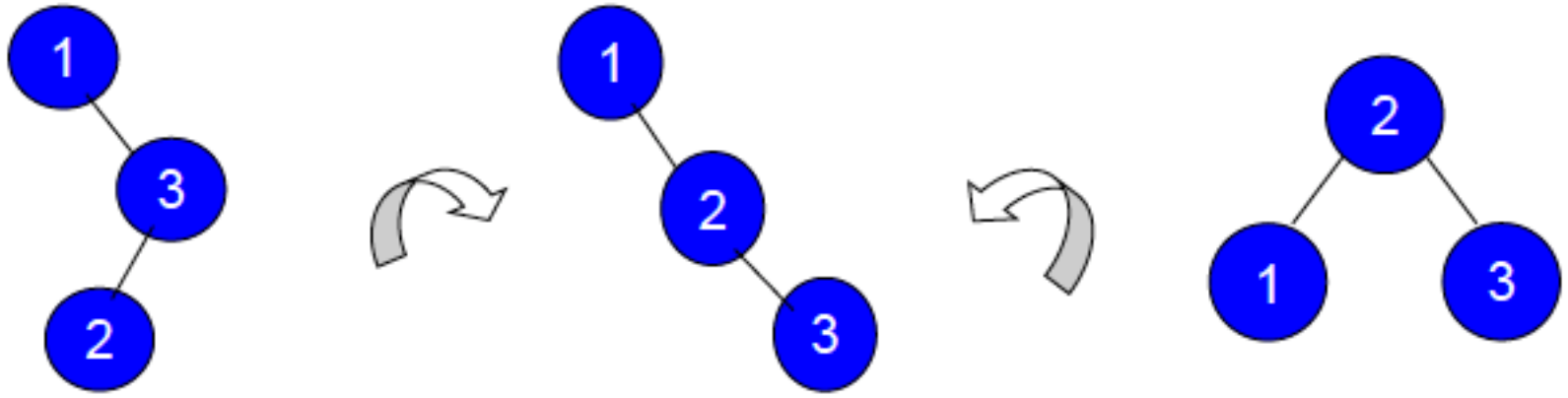
```
TNo* balancaEsquerda (TNo* pR) {
     int fb = fb (pR->pEsq);
     if (fb >= 0) {
             pR = rotacaoSimplesDireita(pR);
     } else {
             pR->pEsq = rotacaoSimplesEsquerda (pR->pEsq);
             pR = rotacaoSimplesDireita (pR);
     }
     return pR;
}
```

```
TNo* rotacaoSimplesEsquerda (TNo* pR) {
 TNo *pAux = pR->pDir;
 pR->pDir = pAux->pEsq;
 pAux->pEsq = pR;
 return pAux;
}
```



Árvore AVL

- Rotações
  - 4º caso – rotação dupla para a esquerda
    - FB < -1 (subárvore da esquerda menor que subárvore da direita)
    - Subárvore da direita desta subárvore direita é **menor** que a subárvore da esquerda

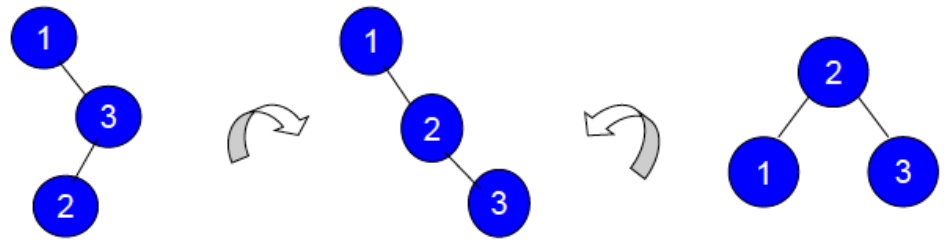


Árvore AVL

- Rotações
  - 4º caso – rotação dupla para a esquerda

```
TNo* balancaDireita (TNo* pR) {
     int fb = fb (pR->pDir);
     if (fb <= 0) {
             pR = rotacaoSimplesEsquerda (pR);
     } else {
             pR->pDir = rotacaoSimplesDireita (pR->pDir));
             pR = rotacaoSimplesEsquerda (pR);
     }
     return pR;
}
```

```
TNo* rotacaoSimplesDireita (TNo* pR) {
 TNo *pAux = pR->pEsq;
 pR->pEsq = pAux->pDir;
 pAux->pDir = pR;
 return pAux;
}
```



Árvore AVL

- Rotações
  - Balanceamento

```
TNo* balanceamento (TNo* pR) {
     int fb = fb (pR);
     if (fb > 1)
            pR = balancaEsquerda (pR);
     else if (fb < -1)
            pR = balancaDireita (pR);
     return pR;
}
```

Análise

- Uma única restruturação
  - O(1)
- Pesquisa
  - O(logn)
- Inserção
  - O(logn)
- Remoção
  - O(logn)

Exercícios

- Considere que estas chaves foram inseridas em uma árvore AVL
  - 10, 20, 5, 8, 12, 22, 23, 24, 11, 13, 18
  - Desenhe como ficará a árvore depois das inserções
- Considere agora que os itens 22, 11, 5 e 10 foram removidos
  - Como ficou a árvore?
- Implemente a árvore AVL
- Faça uma função para dizer se uma árvore é AVL

Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura
Material apresentado por:
Daniel Avila Vecchiato

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)
- Demaine, E., Devadas, S. Introduction to Algorithms (MIT OpenCourseWare), http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011
- http://www.ft.unicamp.br/liag/siteEd/