PRA – Projeto de Arquivos

Árvores AVL

Prof. Allan Rodrigo Leite

Árvores binárias

- Balanceamento de árvores
  - Árvore degenerada
    - Ocorre quando cada nível da árvore apresenta um único nó
    - Neste caso a árvore representa uma estrutura linear
    - Percorrer esta árvore corresponde a complexidade O(n)
  - Árvore cheia ou completa
    - Uma árvore cheia possui um balanceamento perfeito
    - Todos os nós possuem dois filhos, exceto os nós folhas
    - Percorrer nesta árvore corresponde a complexidade O($log\ n$)

Árvores binárias

- Benefícios de uma árvore balanceada
  - Minimizam o número de comparações efetuadas no pior caso

- Para garantir esta propriedade em aplicações dinâmicas é necessário
  - Reconstruir a árvore em seu estado ideal a cada operação
  - Operações de inclusão e exclusão
  - Cada operação realizada na árvore, pode ser necessário um balanceamento

Árvores AVL

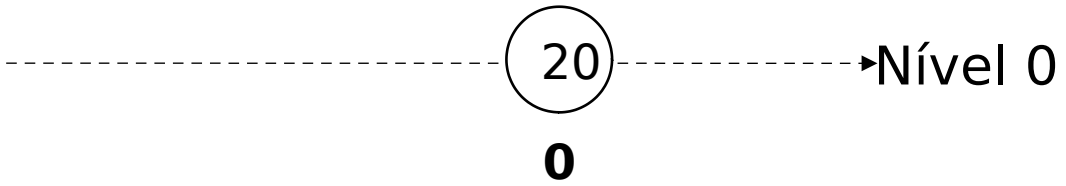
- Árvore AVL (Adelson-Velskii e Landis, 1962) é uma árvore binária altamente balanceada
  - A cada inserção ou exclusão, é executada uma rotina de balanceamento
  - Visam manter a altura da subárvores à esquerda e à direita equilibradas

- Formalmente
  - Alturas das subárvores à esquerda e direita podem diferir, no máximo, em uma unidade
  - Para calcular este equilíbrio é utilizado o fator de balanceamento (FB)

FB = **altura da subárvore esquerda** – **altura da subárvore direita**

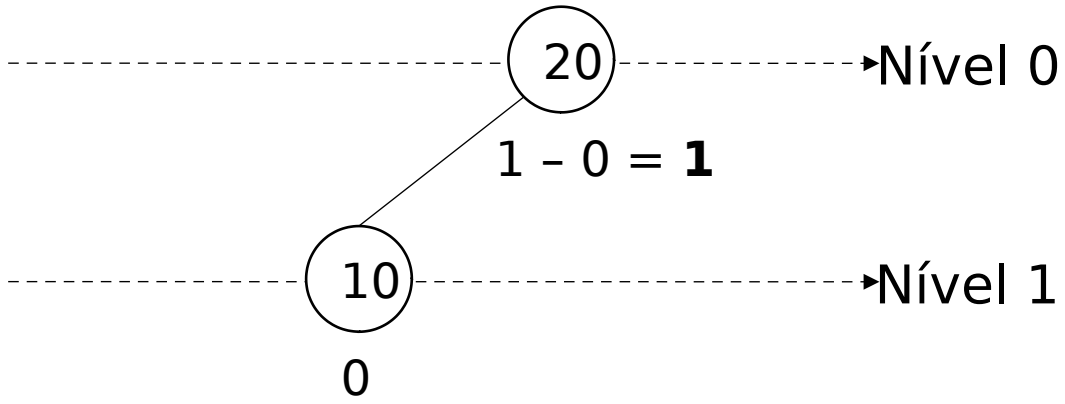
Árvores AVL

- Exemplo
  - 20



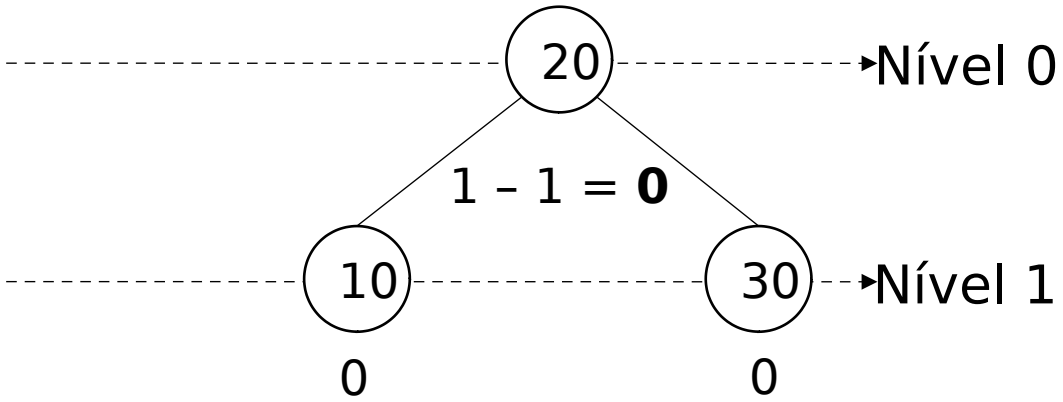
Árvores AVL

- Exemplo
  - 20
  - 10



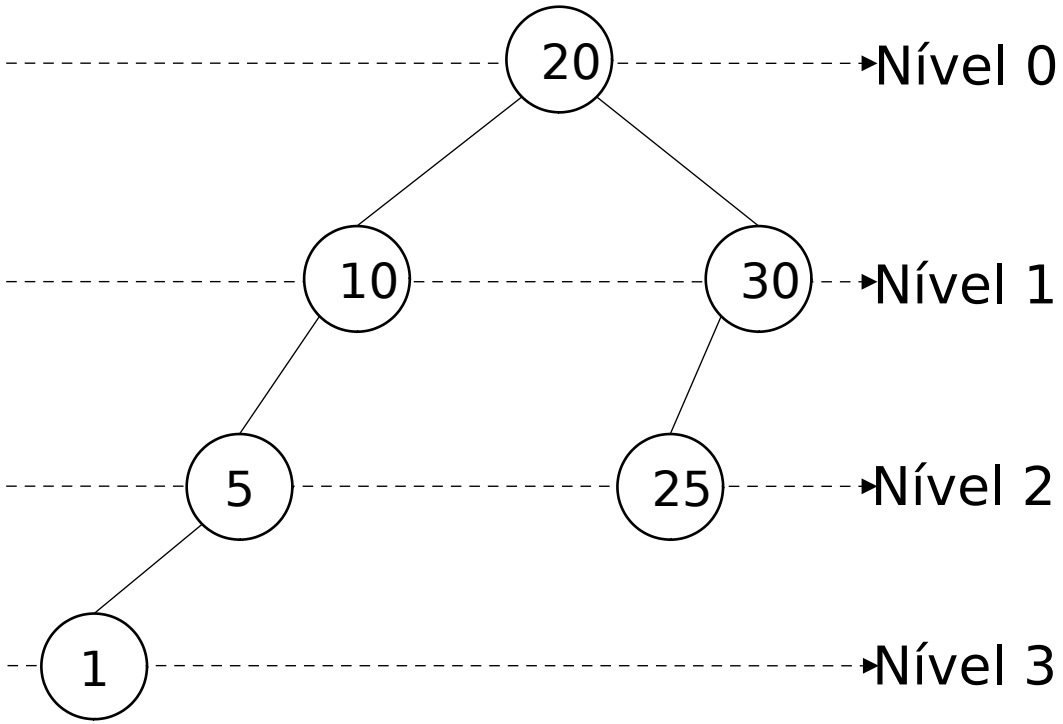
Árvores AVL

- Exemplo
  - 20
  - 10
  - 30



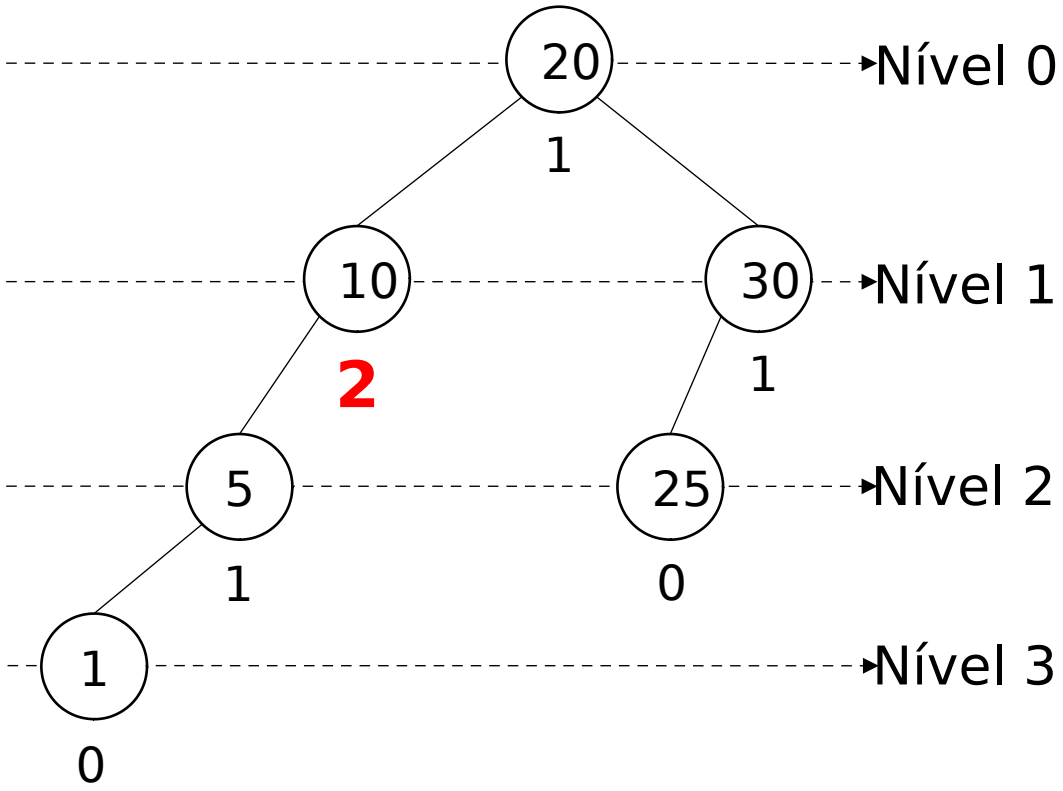
Árvores AVL

- Exemplo
  - 20
  - 10
  - 30
  - 5
  - 25
  - 1



Árvores AVL

- Exemplo
  - 20
  - 10
  - 30
  - 5
  - 25
  - 1



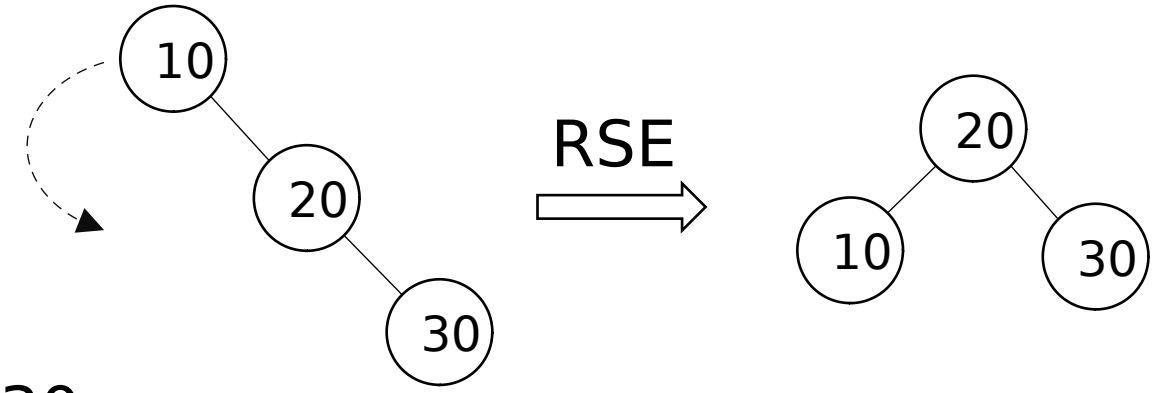
Árvores AVL

- Quando a árvore está desbalanceada, é possível realizar as seguintes operações para balanceá-la novamente
  - Rotação simples
    - Rotação simples à esquerda (**RSE**)
    - Rotação simples à direita (**RSD**)
  - Rotação dupla
    - Rotação dupla à esquerda (**RDE**)
    - Rotação dupla à direita (**RDD**)

- Quando o FB resultar em um valor diferente de `0`, `-1` ou `1`, deverá ser realizada uma das operações de balanceamento

Árvores AVL

- Rotação Simples a Esquerda
  - Neste caso a árvore está mais pesada para o lado direito
  - Regra: quando o FB do nó desbalanceado for negativo e o nó a direita também tiver FB negativo



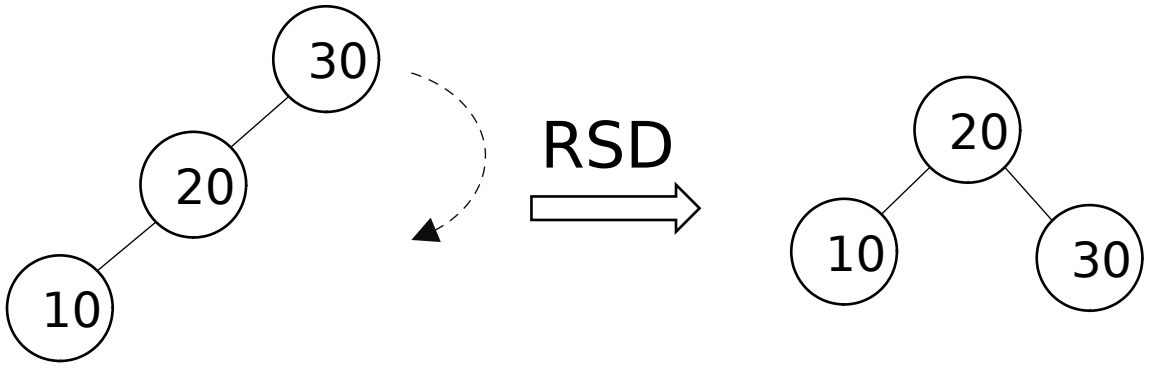
`RSE(10,20)`

Nó 10 passa a ser filho de 20

Nó 20 passa a ser pai de 10 e 30

Árvores AVL

- Rotação Simples a Direita
  - Neste caso a árvore está mais pesada para o lado esquerdo
  - Regra: quando o FB do nó desbalanceado for positivo e o nó a direita também tiver FB positivo



`RSD(30,20)`

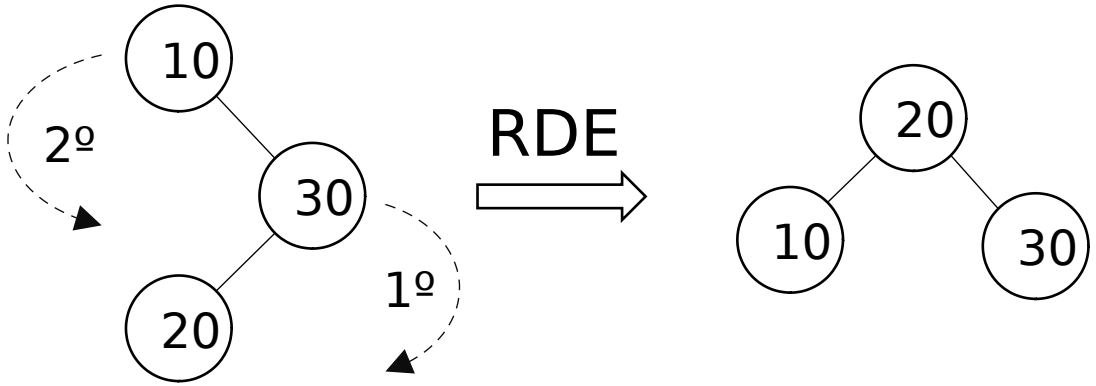
Nó 30 passa a ser filho de 20

Nó 20 passa a ser pai de 10 e 30

Árvores AVL

- Rotação Dupla a Esquerda
  - Neste caso a árvore está mais pesada para o lado direito
  - Regra: quando o FB do nó desbalanceado for negativo e o nó a esquerda tiver FB positivo

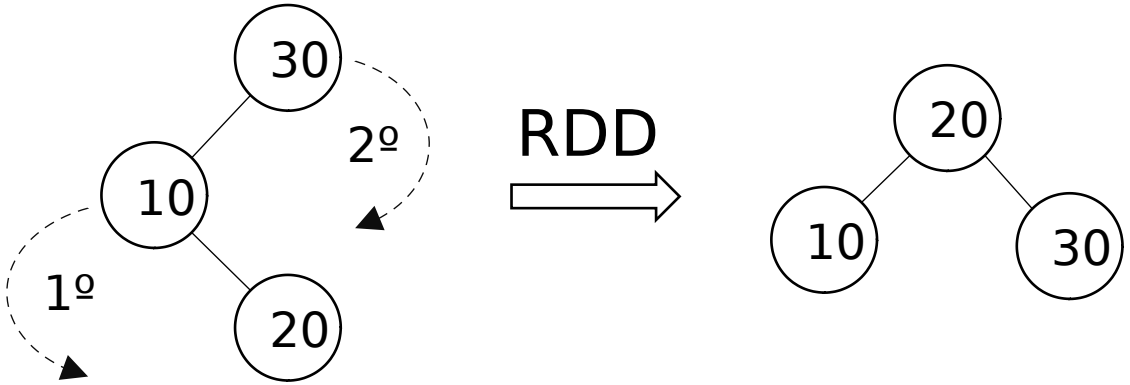
```
RDE(10,30)
    - RSD(30,20)
    - RSE(10,20)
```



Árvores AVL

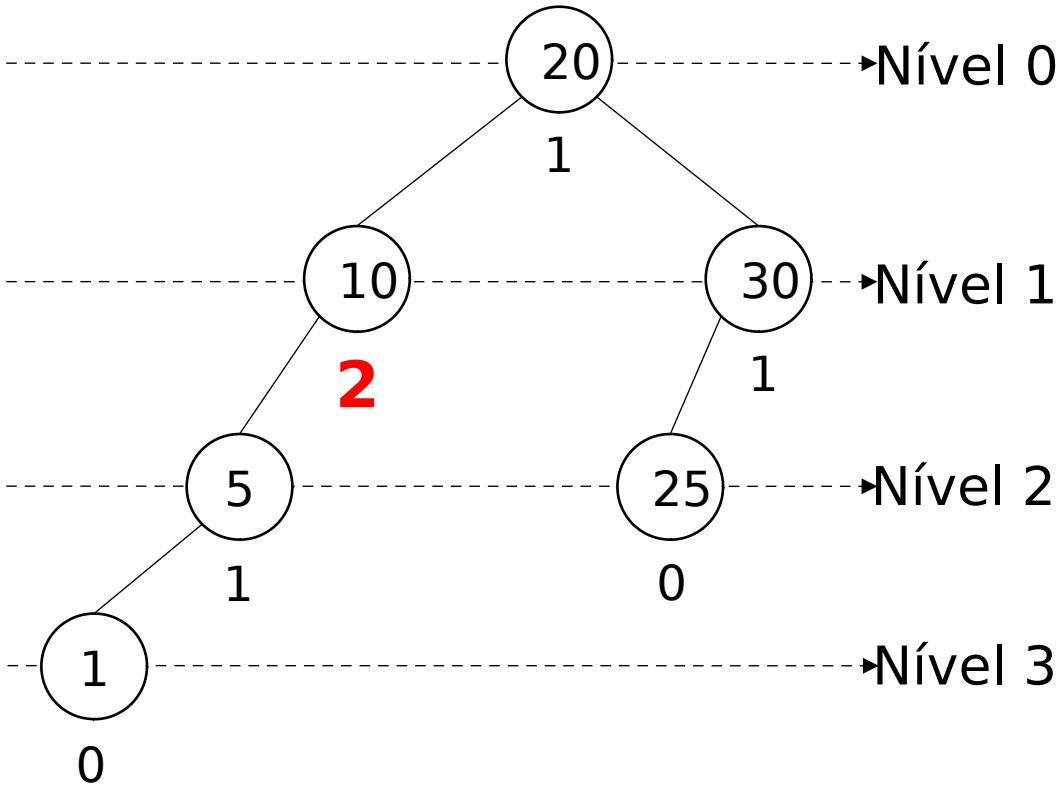
- Rotação Dupla a Direita
  - Neste caso a árvore está mais pesada para o lado esquerdo
  - Regra: quando o FB do nó desbalanceado for positivo e o nó a esquerda tiver FB negativo

```
RDD (30,10)
    - RSE(10,20)
    - RSD(30,20)
```



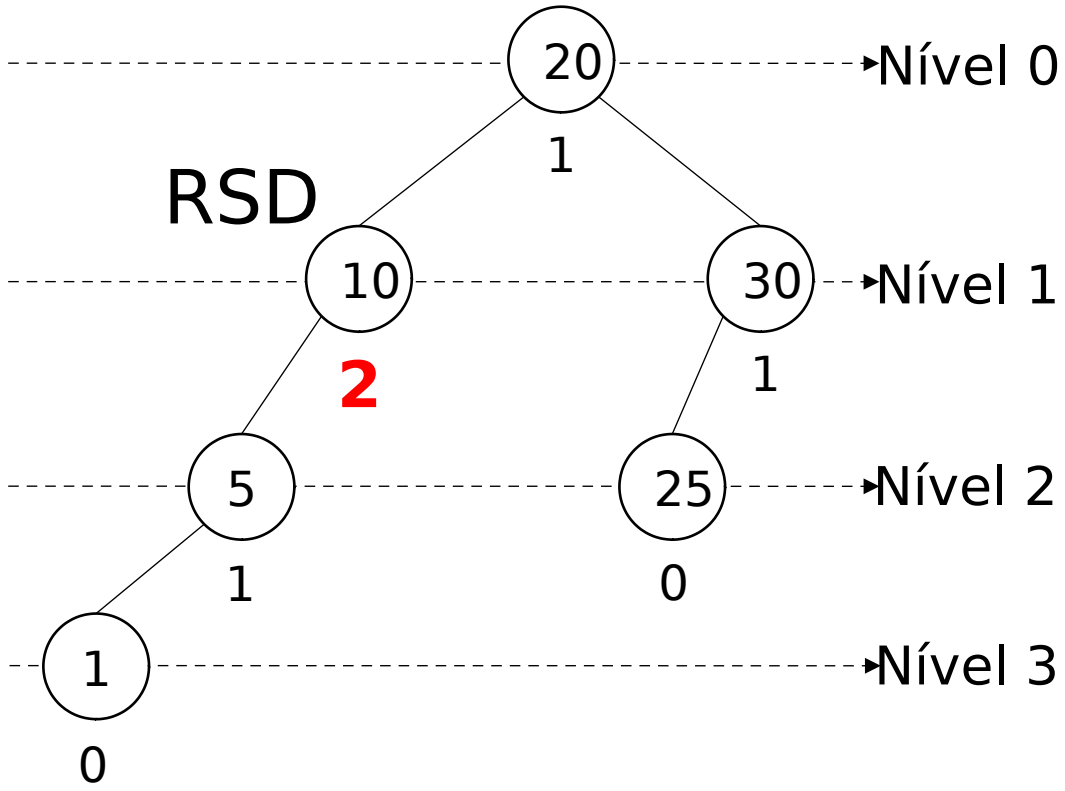
Árvores AVL

- Exemplo
  - 20
  - 10
  - 30
  - 5
  - 25
  - 1



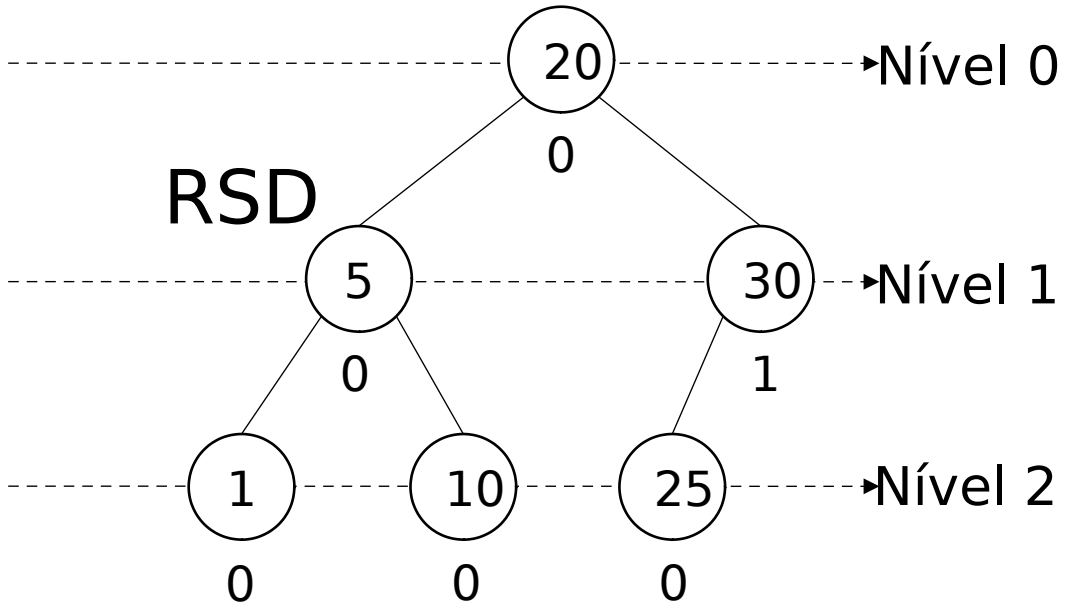
Árvores AVL

- Exemplo
  - 20
  - 10
  - 30
  - 5
  - 25
  - 1



Árvores AVL

- Exemplo
  - 20
  - 10
  - 30
  - 5
  - 15
  - 1



Árvores AVL

- FB – Fator de balanceamento

```
int fb(No* no) {
  int esquerda = 0,direita = 0;

  if (no->esquerda != NULL) esquerda = altura(no->esquerda);
  if (no->direita != NULL) direita = altura(no->direita);

  return esquerda - direita;
}

int altura(No* no){
  int esquerda = 0,direita = 0;

  if (no->esquerda != NULL) esquerda = altura(no->esquerda) + 1;
  if (no->direita != NULL) direita = altura(no->direita) + 1;

  return esquerda > direita ? esquerda : direita;
}
```

Árvores AVL

- RSE – Rotação Simples a Esquerda

```
No* rse(No* no) {
  No* pai = no->pai;
  No* direita = no->direita;

  no->direita = direita->esquerda;
  no->pai = direita;

  direita->esquerda = no;
  direita->pai = pai;

  return direita;
}
```

Árvores AVL

- RSD – Rotação Simples a Direita

```
No* rsd(No* no) {
  No* pai = no->pai;
  No* esquerda = no->esquerda;

  no->esquerda = esquerda->direita;
  no->pai = esquerda;

  esquerda->direita = no;
  esquerda->pai = pai;

  return esquerda;
}
```

Árvores AVL

- RDE – Rotação Dupla a Esquerda

```
No* rde(No* no) {
  no->direita = rsd(no->direita);
  return rse(no);
}
```

- RDD – Rotação Dupla a Direita

```
No* rdd(No* no) {
  no->esquerda = rse(no->esquerda);
  return rsd(no);
}
```

PRA – Projeto de Arquivos

Árvores AVL

Prof. Allan Rodrigo Leite