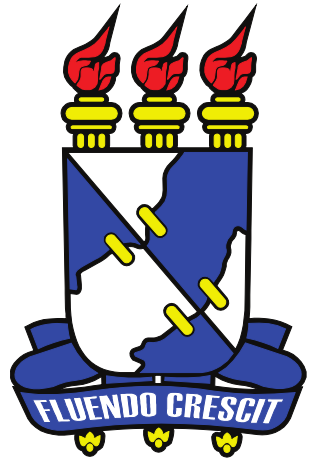


Estrutura de Dados I

Árvores AVL

Bruno Prado

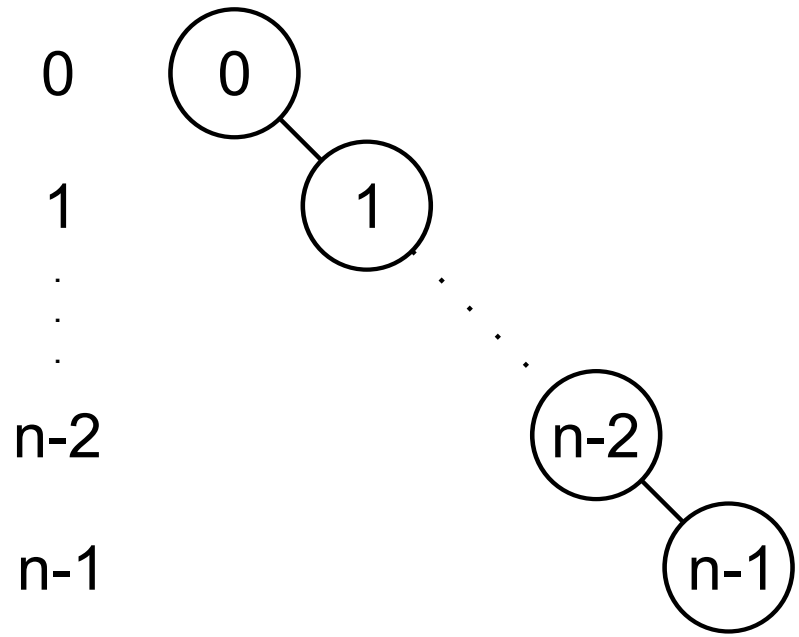
Departamento de Computação / UFS

Introdução

- Conceitos chave de árvores binárias
  - Cada nó possui até 2 filhos
  - Para uma árvore com altura h existem $\lceil 2^{h+1} - 1 \rceil$ nós
  - Com n nós possui altura entre $\log_2 n - 1 \leq h \leq n - 1$

Introdução

- Conceitos chave de árvores binárias
  - As operações na árvore tem custo $O(h)$
  - A degeneração da árvore binária leva a uma altura $\lceil h = n - 1 \rceil$ e ocorre devido ao desbalanceamento



Introdução

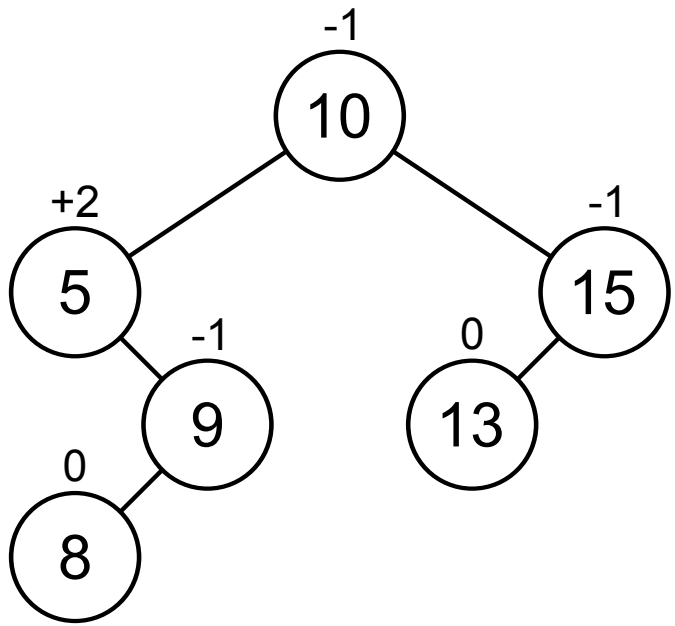
- O que é balanceamento de uma árvore binária?
  - É a aplicação de restrições estruturais na realização das operações para garantir que a árvore resultante possua uma altura logarítmica
  - Evita o processo de degeneração e garante a eficiência computacional da estrutura

Introdução

- O que é balanceamento de uma árvore binária?
  - É a aplicação de restrições estruturais na realização das operações para garantir que a árvore resultante possua uma altura logarítmica
  - Evita o processo de degeneração e garante a eficiência computacional da estrutura
- Árvore AVL: Adelson-Velsky e Landis
  - É uma árvore binária balanceada criada em 1962
  - Seu funcionamento é baseado na altura das subárvores para evitar o processo de degeneração

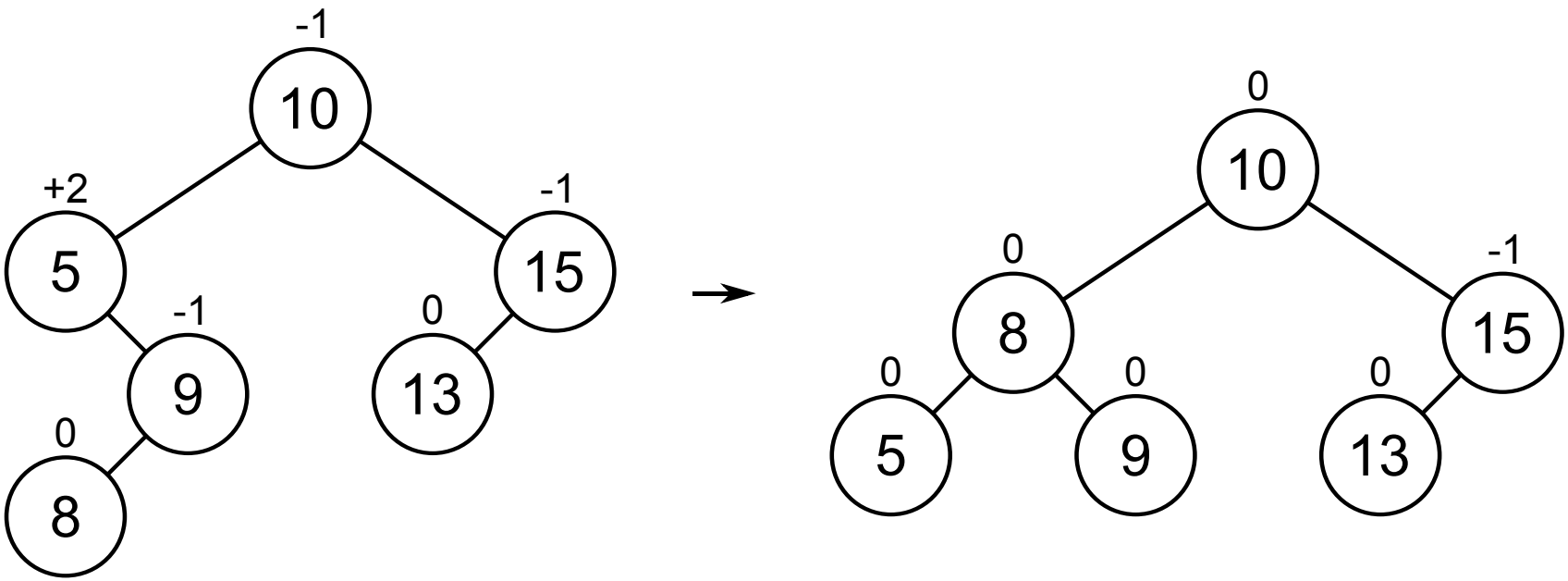
Árvores AVL

- Definição da estrutura
  - Cada nó possui um fator de balanceamento
  - É feito o cálculo da diferença de altura das subárvores de cada nó, considerando que a altura esquerda é negativa e a direita é positiva



Árvores AVL

- Operações de rotação
  - São necessárias para garantir o balanceamento
  - Sempre que o fator de balanceamento possui módulo superior a 1, é necessário rotacionar os nós para ajustar a altura das subárvores

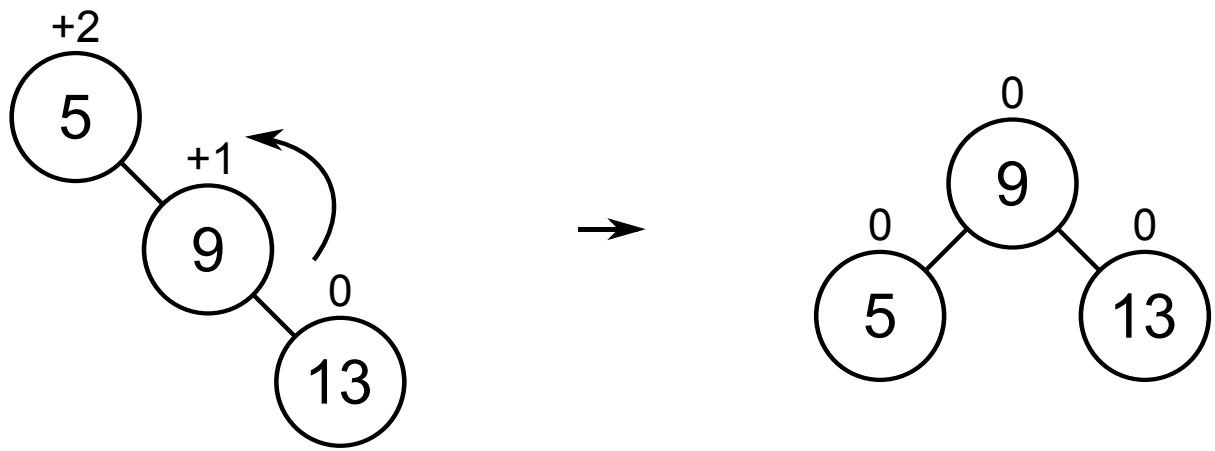


Árvores AVL

- Tipos de rotação
  - Simples para esquerda (L-rotation)
  - Simples para direita (R-rotation)
  - Dupla esquerda-direita (LR-rotation)
  - Dupla direita-esquerda (RL-rotation)

Árvores AVL

- Rotação simples para esquerda
  - É aplicada quando existe um desbalanceamento positivo com degeneração da subárvore direita



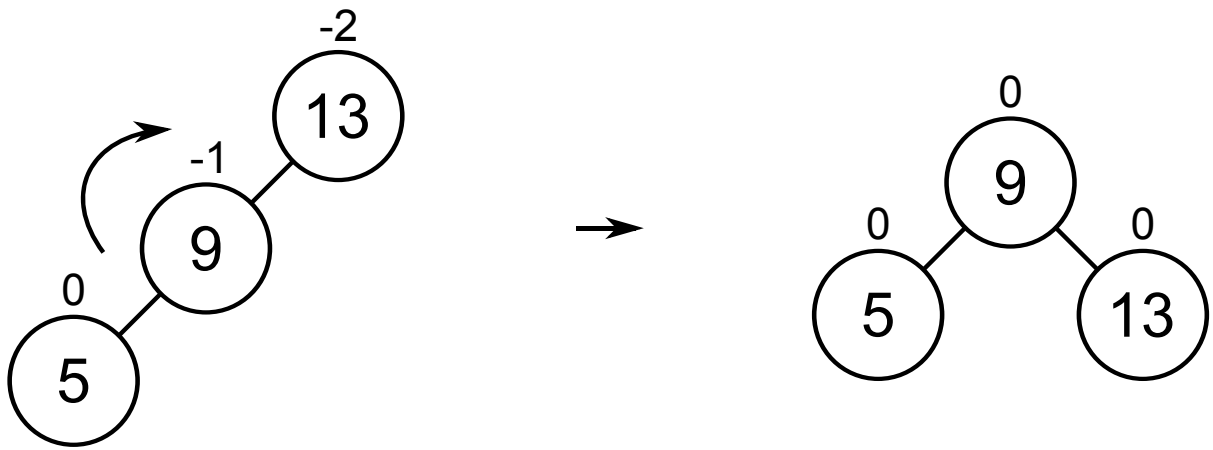
Árvores AVL

- Rotação simples para esquerda
  - É feito o ajuste dos ponteiros
  - O fator de balanceamento é recalculado

```
// Rotação simples para esquerda
void rotacao_esq(no* raiz) {
        no* eixo = raiz->dir;
        raiz->dir = eixo->esq;
        eixo->esq = raiz;
        raiz = eixo;
        fator_balanceamento(raiz);
}
```

Árvores AVL

- Rotação simples para direita
  - É aplicada quando existe um desbalanceamento negativo com degeneração da subárvore esquerda



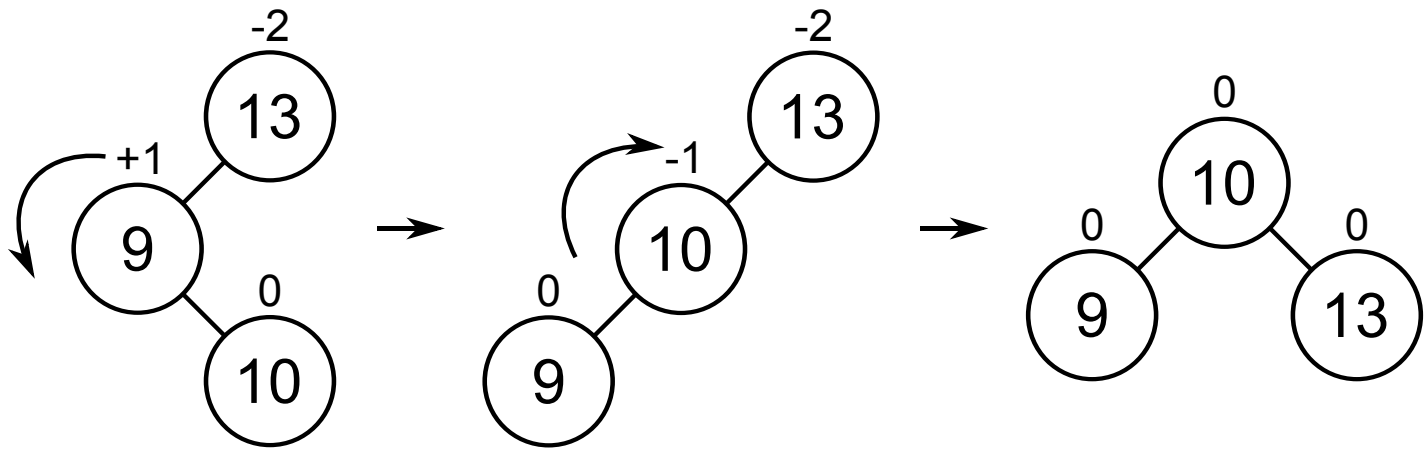
Árvores AVL

- Rotação simples para direita
  - É feito o ajuste dos ponteiros
  - O fator de balanceamento é recalculado

```
// Rotação simples para direita
void rotacao_dir(no* raiz) {
        no* eixo = raiz->esq;
        raiz->esq = eixo->dir;
        eixo->dir = raiz;
        raiz = eixo;
        fator_balanceamento(raiz);
}
```

Árvores AVL

- Rotação dupla esquerda-direita
  - É realizada quando existe um desbalanceamento negativo e positivo nas subárvores esquerda e direita
  - São aplicadas rotações para esquerda e para direita



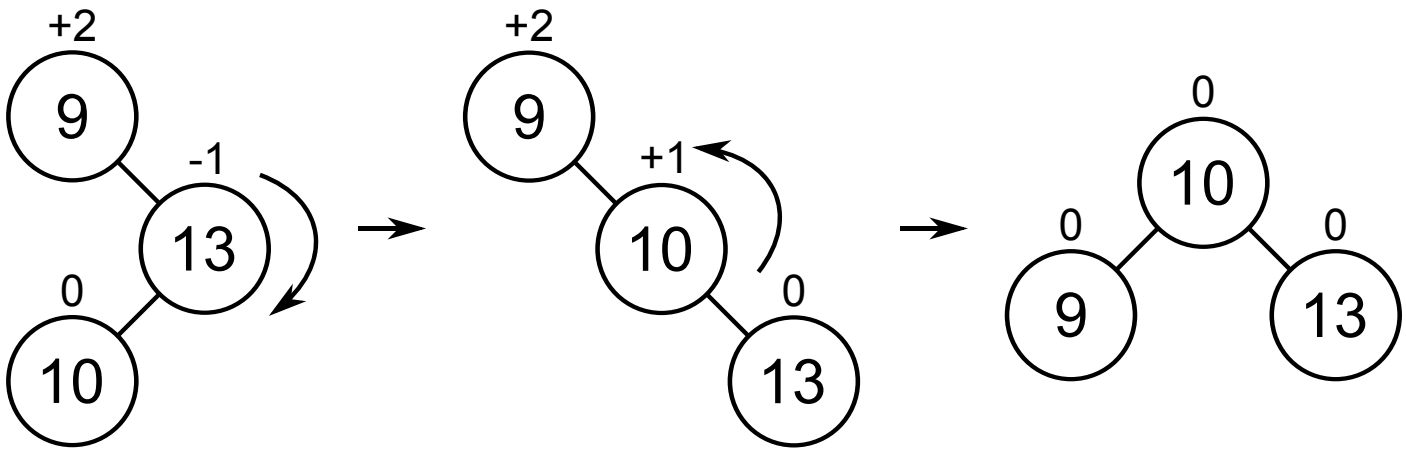
Árvores AVL

- Rotação dupla esquerda-direita
  - São realizadas rotações para esquerda e para direita
  - Os fatores de balanceamento são recalculados

```
// Rotação dupla esquerda-direita
void rotacao_esq_dir(no* raiz) {
        rotacao_esq(raiz->esq);
        rotacao_dir(raiz);
}
```

Árvores AVL

- Rotação dupla direita-esquerda
  - É realizada quando existe um desbalanceamento positivo e negativo nas subárvores direita e esquerda
  - São aplicadas rotações para direita e para esquerda



Árvores AVL

- Rotação dupla direita-esquerda
  - São realizadas rotações para direita e para esquerda
  - Os fatores de balanceamento são recalculados

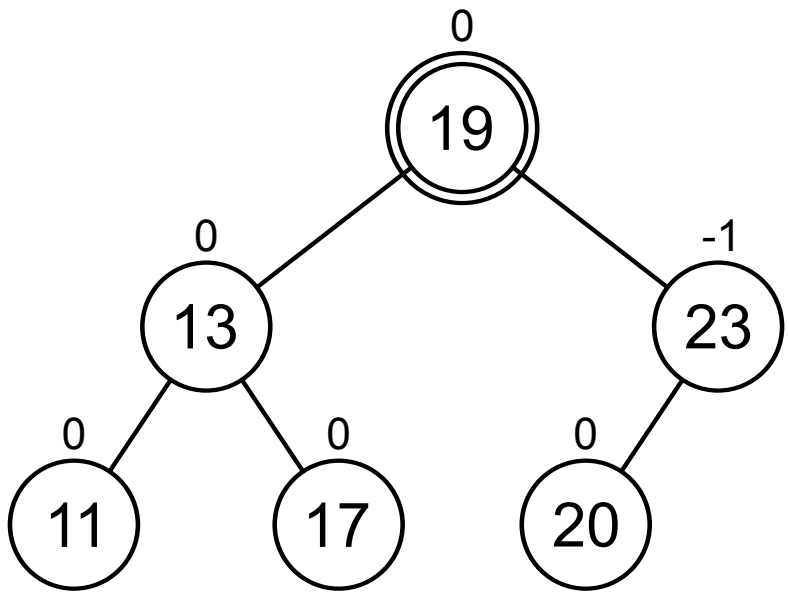
```
// Rotação dupla direita-esquerda
void rotacao_dir_esq(no* raiz) {
        rotacao_dir(raiz->dir);
        rotacao_esq(raiz);
}
```

Árvores AVL

- Operações básicas
  - Busca
  - Inserção
  - Remoção

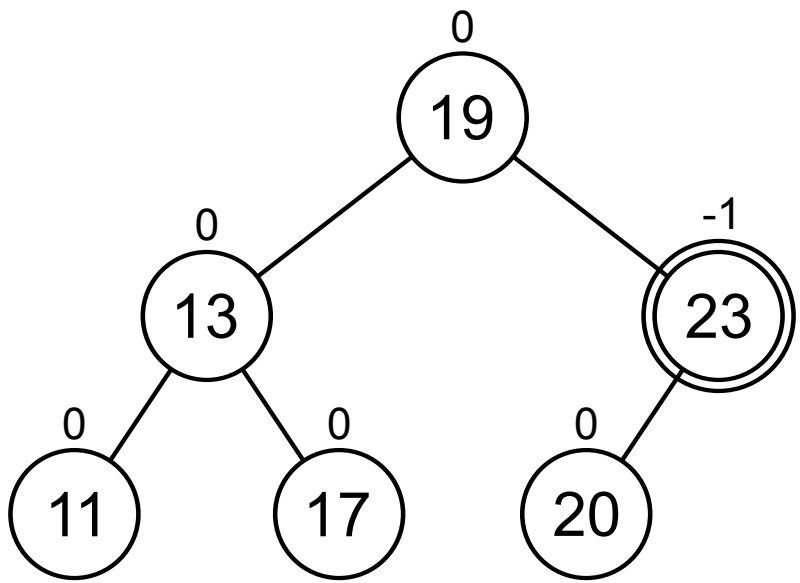
Árvores AVL

- Operação de busca
  - Parâmetro de chave: 20
  - A busca tem início pelo elemento raiz da árvore, comparando o valor de sua chave com o parâmetro de busca



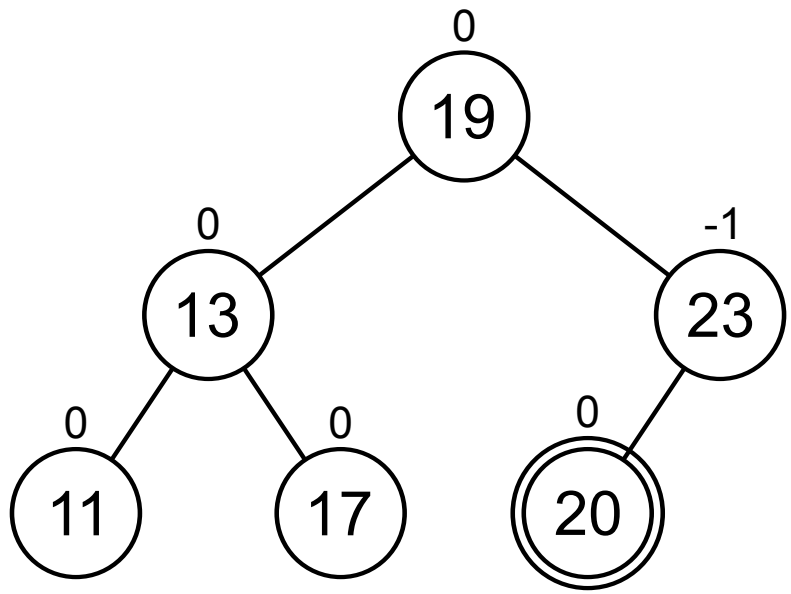
Árvores AVL

- Operação de busca
  - Parâmetro de chave: 20
  - Como o resultado da comparação indica que o valor é menor do que o procurado, a busca é aplicada na subárvore direita



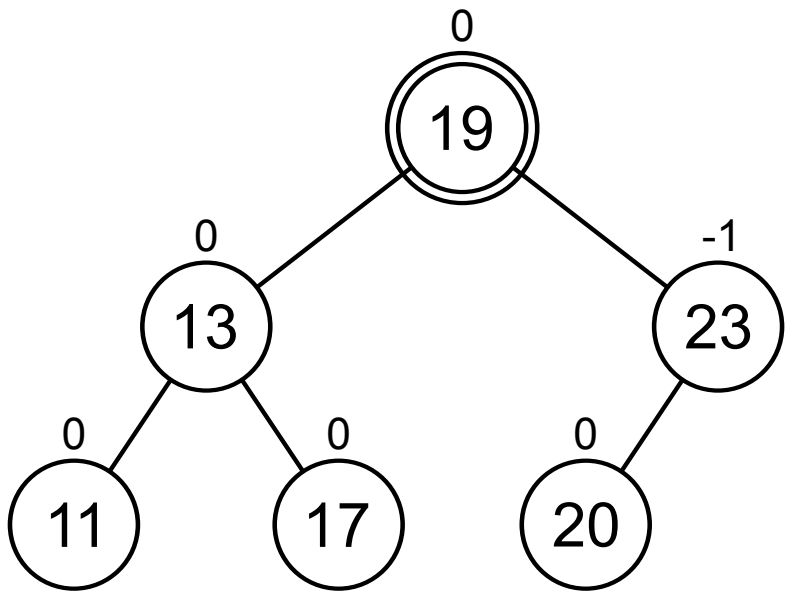
Árvores AVL

- Operação de busca
  - Parâmetro de chave: 20
  - A chave do nó é igual ao parâmetro de busca e sua referência é retornada



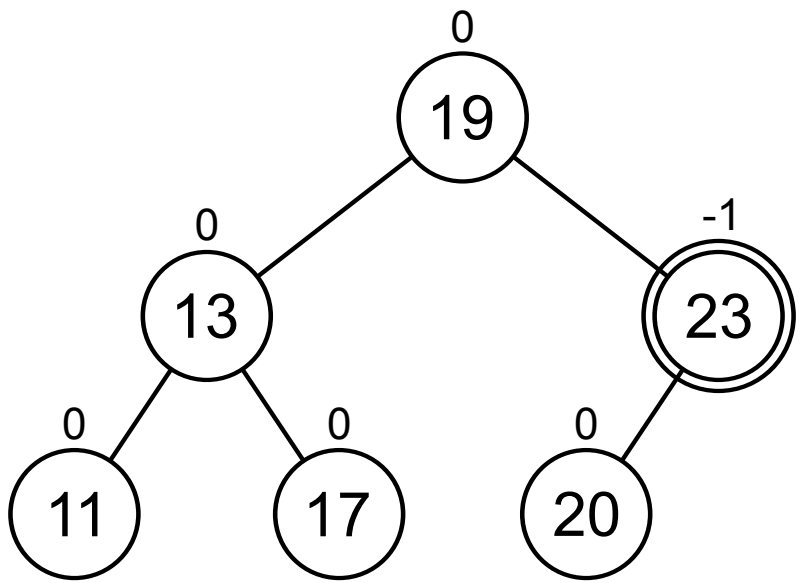
Árvores AVL

- Operação de inserção
  - Parâmetro de chave: 29
  - É realizada uma operação de busca utilizando a chave do elemento que será inserido até encontrar uma referência nula



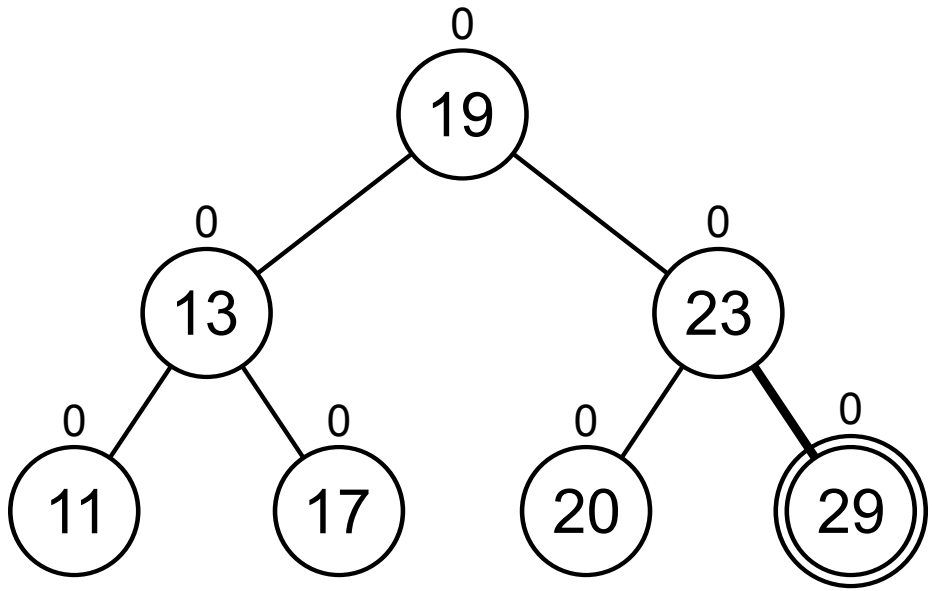
Árvores AVL

- Operação de inserção
  - Parâmetro de chave: 29
  - É realizada uma operação de busca utilizando a chave do elemento que será inserido até encontrar uma referência nula



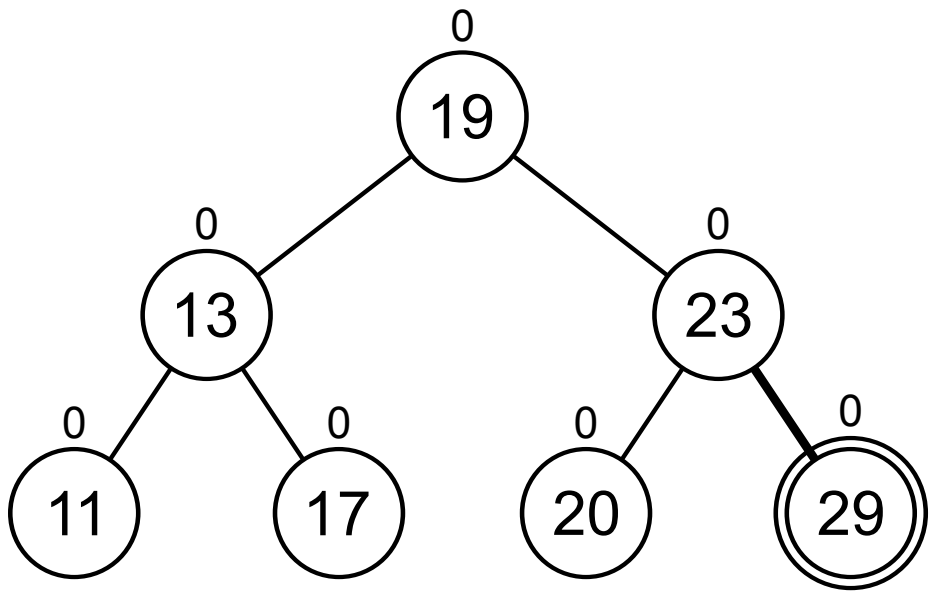
Árvores AVL

- Operação de inserção
  - Parâmetro de chave: 29
  - É feita a alocação do nó para inserção na árvore e verificação de balanceamento do nó pai



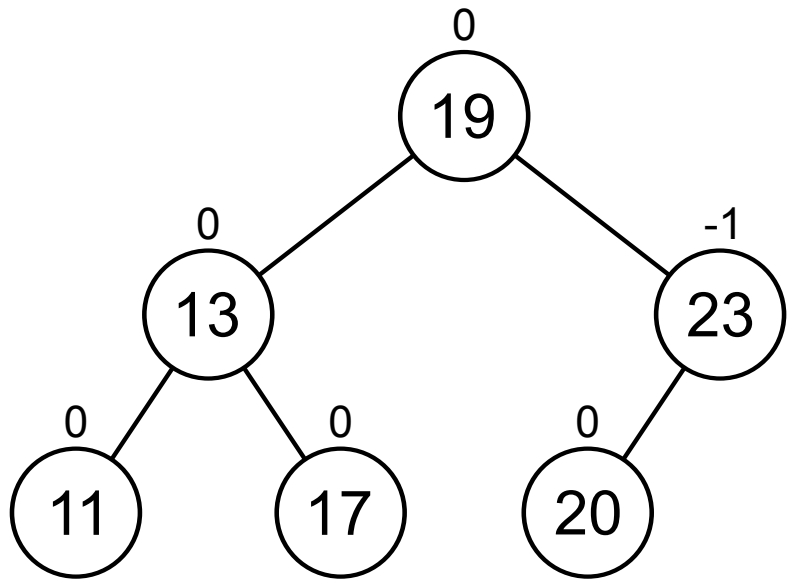
Árvores AVL

- Operação de remoção
  - Caso 1: o nó removido é uma folha
    - Parâmetro de chave: 29
    - É feita a busca e remoção pela chave do elemento, além da checagem do balanceamento do nó pai



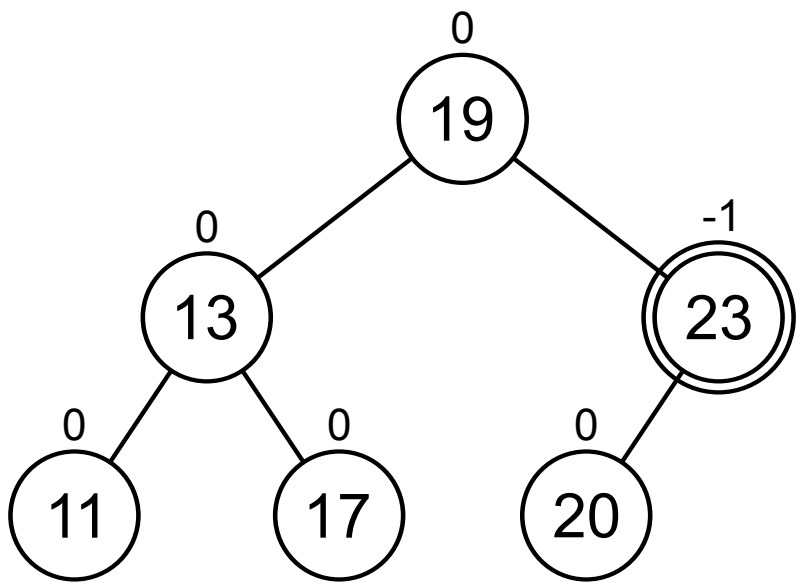
Árvores AVL

- Operação de remoção
  - Caso 1: o nó removido é uma folha
    - Parâmetro de chave: 29
    - A operação não desbalanceou a árvore



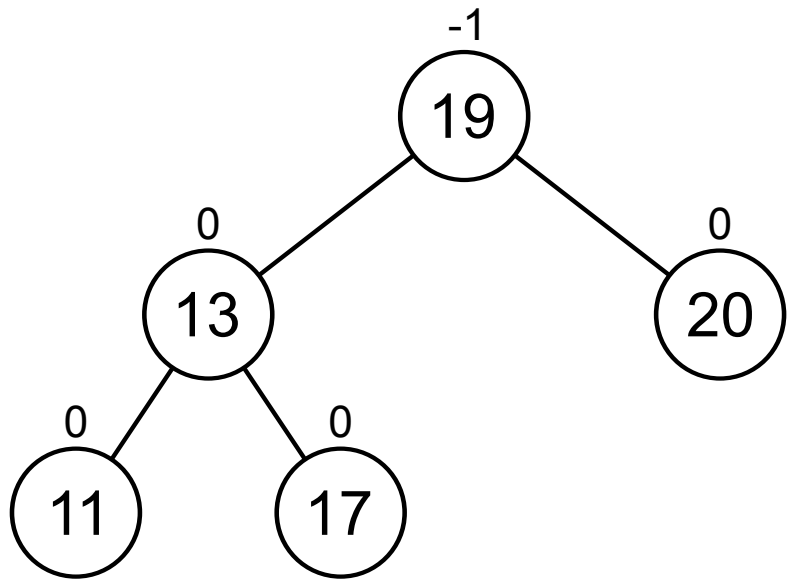
Árvores AVL

- Operação de remoção
  - Caso 2: o nó removido possui uma subárvore
    - Parâmetro de chave: 23
    - É feita a busca e remoção pela chave do elemento, além da checagem do balanceamento do nó pai



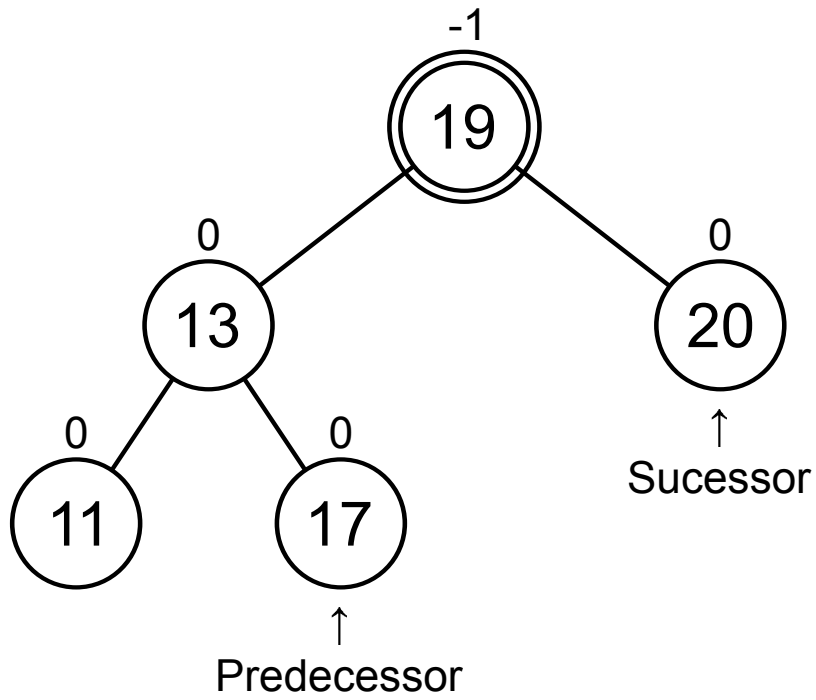
Árvores AVL

- Operação de remoção
  - Caso 2: o nó removido possui uma subárvore
    - Parâmetro de chave: 23
    - A operação não desbalanceou a árvore



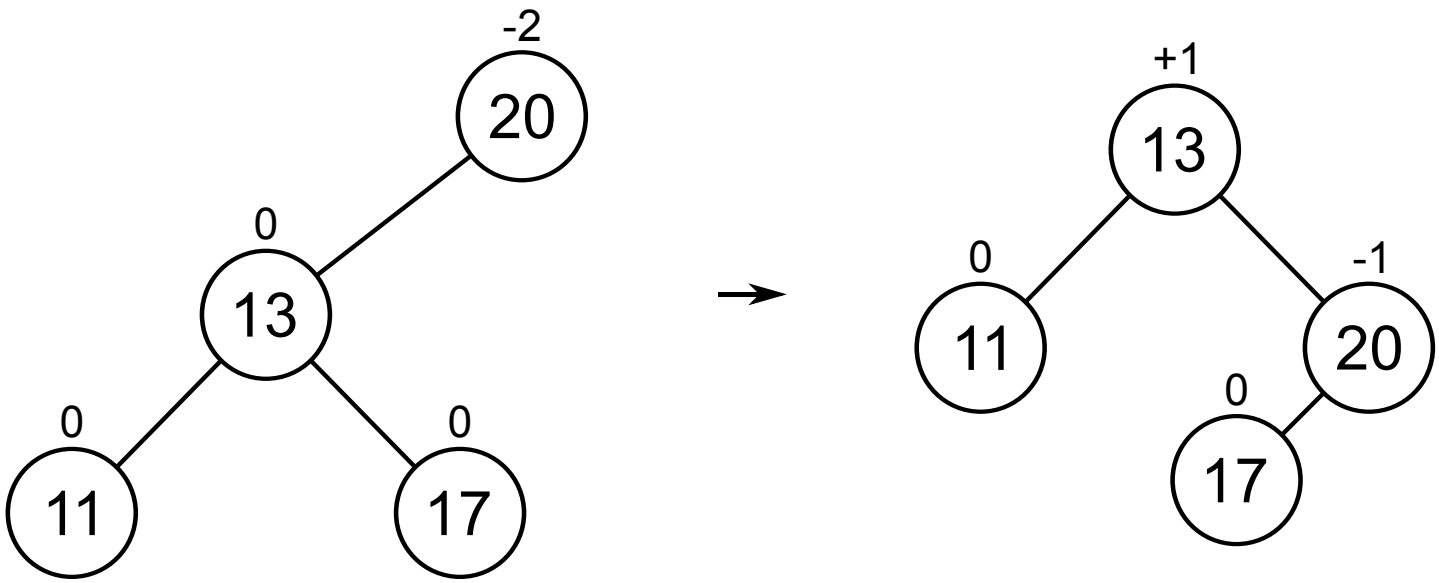
Árvores AVL

- Operação de remoção
  - Caso 3: o nó removido possui duas subárvores
    - Parâmetro de chave: 19
    - É feita a busca e remoção pela chave do elemento, além da checagem do balanceamento do nó pai



Árvores AVL

- Operação de remoção
  - Caso 3: o nó removido possui duas subárvores
    - Parâmetro de chave: 19
    - A operação desbalanceou a árvore, necessitando que seja feita uma rotação simples para direita



Árvores AVL

- Análise de complexidade
  - No pior caso, a busca percorre a altura h da árvore que possui n nós, entretanto a aplicação das técnicas de balanceamento garante que $h \approx \log_2 n$
  - Espaço $\Theta(n)$
  - Tempo $O(\log_2 n)$

Exemplo

- Construa uma árvore binária AVL
  - Insira os elementos com chaves 13, 2, 34, 11, 7, 43 e 9
  - Realize a remoção dos elementos de chave 7 e 9
  - Explique os princípios que garantem uma melhor eficiência desta estrutura com relação a uma árvore binária sem balanceamento

Exercício

- A empresa de tecnologia Poxim Tech está desenvolvendo um sistema de dicionário de sinônimos baseado em árvore binária AVL
  - As palavras do dicionário e a lista de sinônimos de cada palavra são compostas exclusivamente por letras minúsculas com até 30 caracteres
  - A listagem de sinônimos para cada palavra possui capacidade máxima de 10 palavras
  - Para demonstrar a eficiência da busca no dicionário de sinônimos é exibido o percurso realizado na árvore

Exercício

- Formato de arquivo de entrada
  - [#*Número de palavras*]
  - [$Palavra_1$] [i] [$Sinônimo_1$] ... [$Sinônimo_i$]
  - ⋮
  - [$Palavra_n$] [j] [$Sinônimo_1$] ... [$Sinônimo_j$]
  - [#*Número de consultas*]
  - [$Consulta_1$]
  - ⋮
  - [$Consulta_m$]

```
5
demais 5 bastante numeroso demasiado abundante excessivo
facil 2 simples ed1
elegante 3 natural descomplicado trivial
nada 4 zero vazio osso nulo
trabalho 3 atividade tarefa missao
3
facil
demais
zero
```

Exercício

- Formato de arquivo de saída
  - Percurso realizado pela busca e a listagem de sinônimos da palavra pesquisada

```
elegante->nada->facil:
[simples, ed1]
elegante->demais:
[bastante, numeroso, demasiado, abundante, excessivo]
elegante->nada->trabalho->?:
[-]
```