Árvore AVL

Estruturas de Dados

Bruno Prado

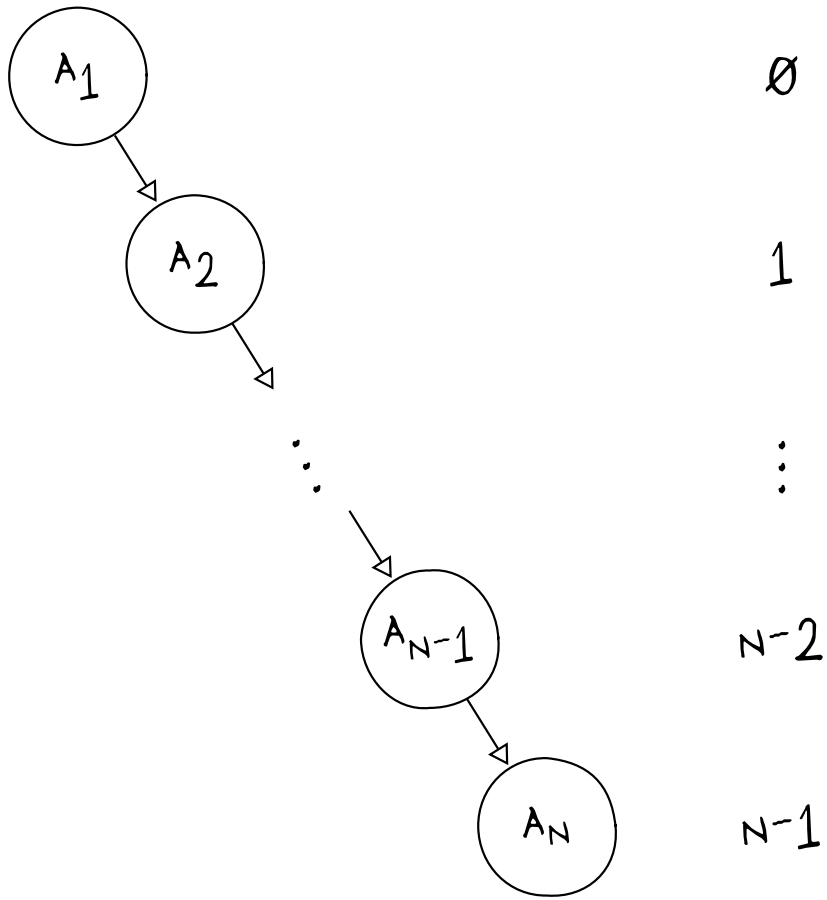
Departamento de Computação / UFS

Introdução

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

Introdução

- Conceitos chave de árvore binária
  - 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

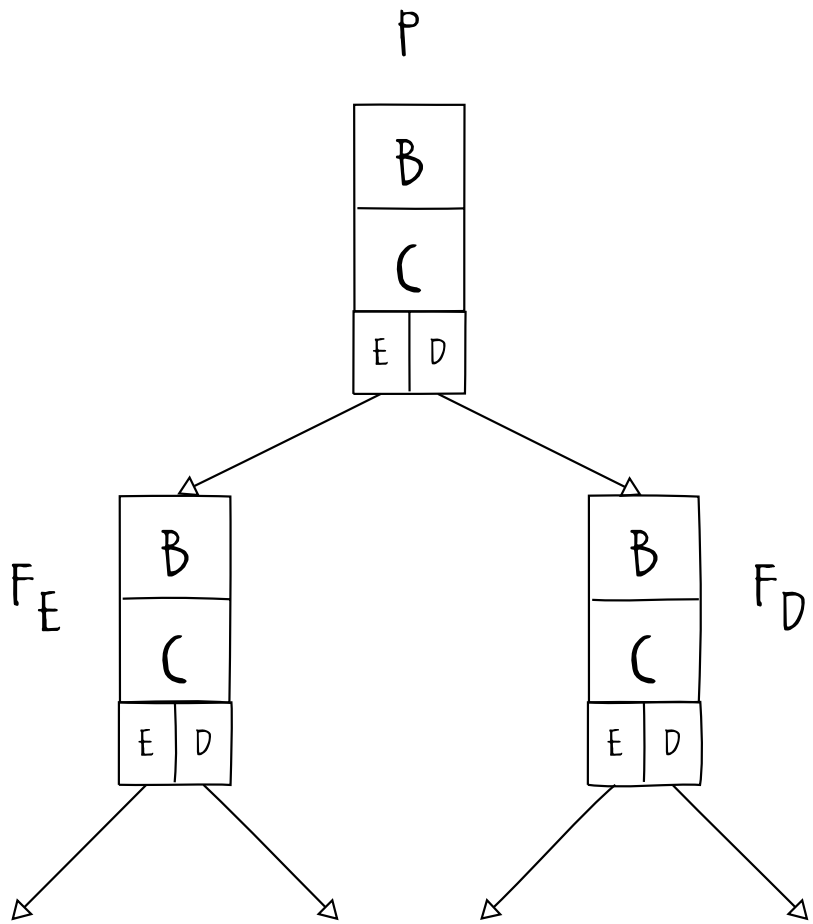
- O que é balanceamento de uma árvore binária?
  - É a aplicação de restrições estruturais na realização das operações para garantir que a árvore resultante possua uma altura logarítmica
  - Impede o processo de degeneração e garante a eficiência computacional da estrutura

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
  - Impede o processo de degeneração e garante a eficiência computacional da estrutura
- Árvore AVL: **A**delson-**V**elsky e **L**andis
  - É uma árvore binária balanceada criada em 1962
  - Seu funcionamento consiste em controlar a altura das subárvores para evitar o processo de degeneração

Árvore AVL

- Definição da estrutura
  - As alturas da direta e da esquerda tem valores positivos e negativos, respectivamente
  - É feito o cálculo da diferença de altura das subárvores de cada nó para obter o fator de balanceamento



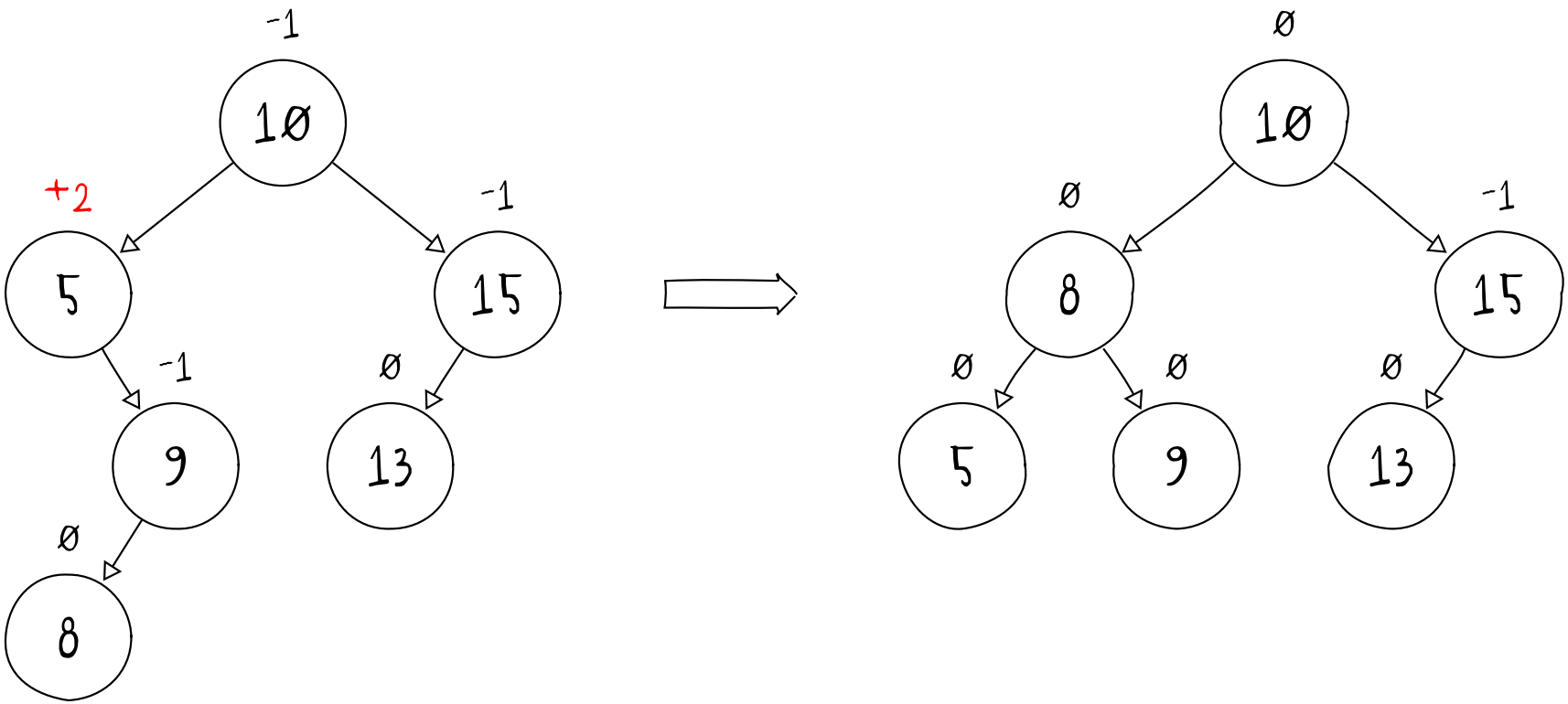
Árvore AVL

- Implementação em C
  - Estrutura e ponteiros

```c
// Padrão de tipos por tamanho
#include <stdint.h>
// Estrutura de nó
typedef struct no {
    // Fator de balanceamento
    int8_t B;
    // Chave do nó
    uint32_t C;
    // Filho da direita
    struct no* D;
    // Filho da esquerda
    struct no* E;
} no;
```

Árvore 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

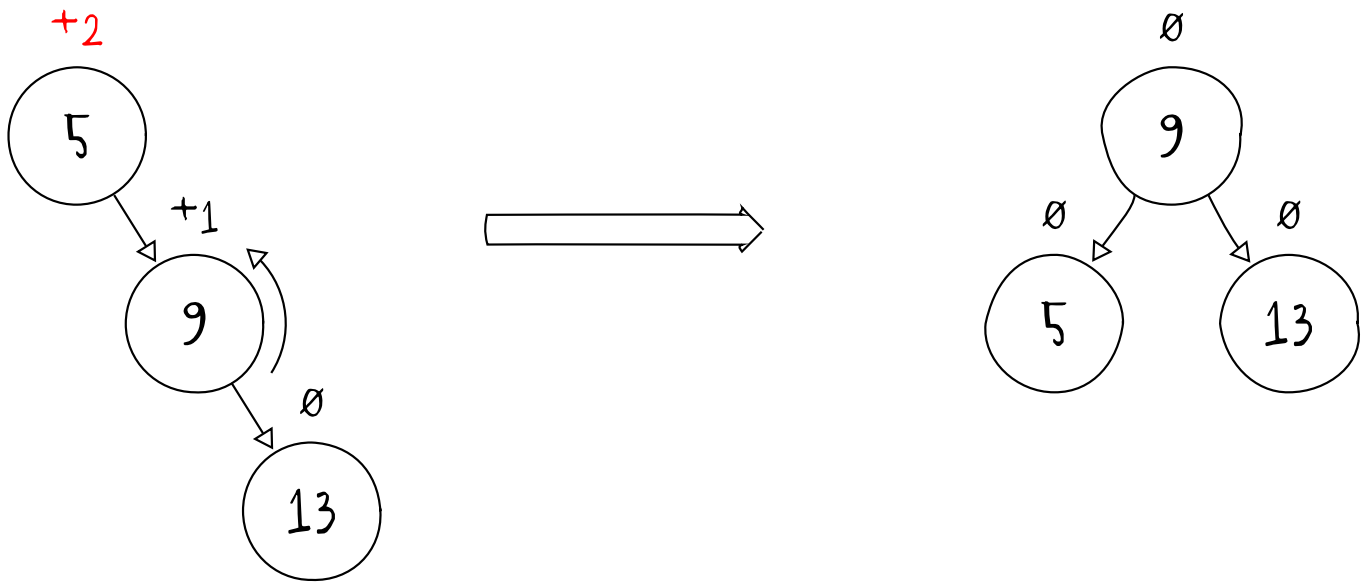


Árvore AVL

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

Árvore AVL

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



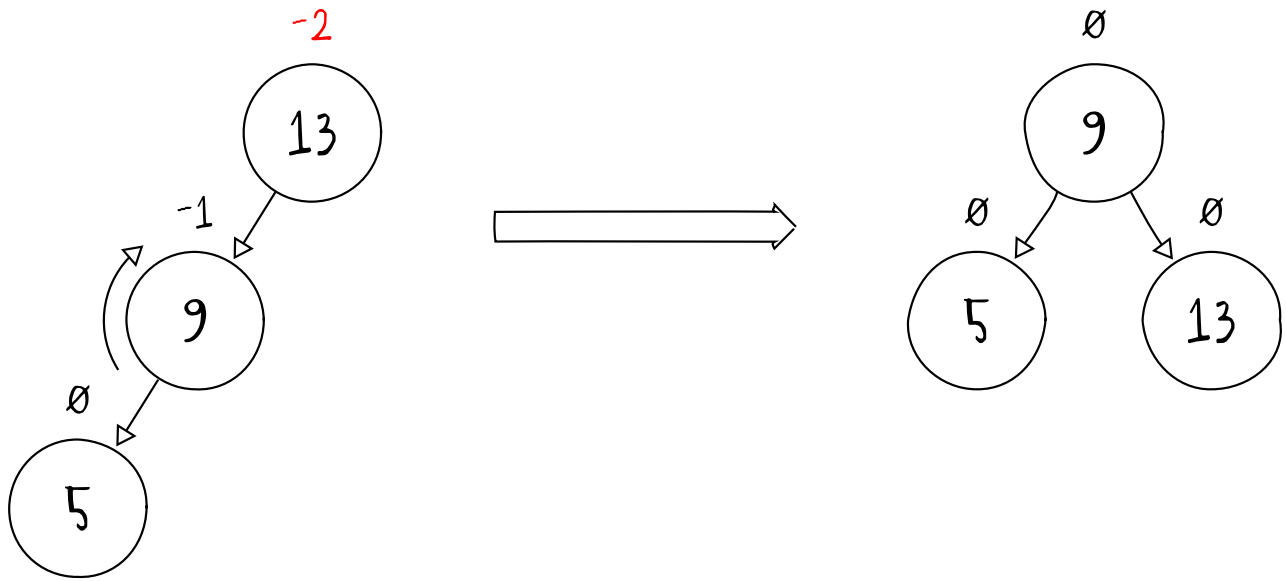
Árvore AVL

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

```
1 // Procedimento de rotação simples para esquerda
2 void rotacao_E(no* raiz) {
3     no* eixo = raiz->D;
4     raiz->D = eixo->E;
5     eixo->E = raiz;
6     raiz = eixo;
7     balanceamento(raiz);
8 }
```

Árvore AVL

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



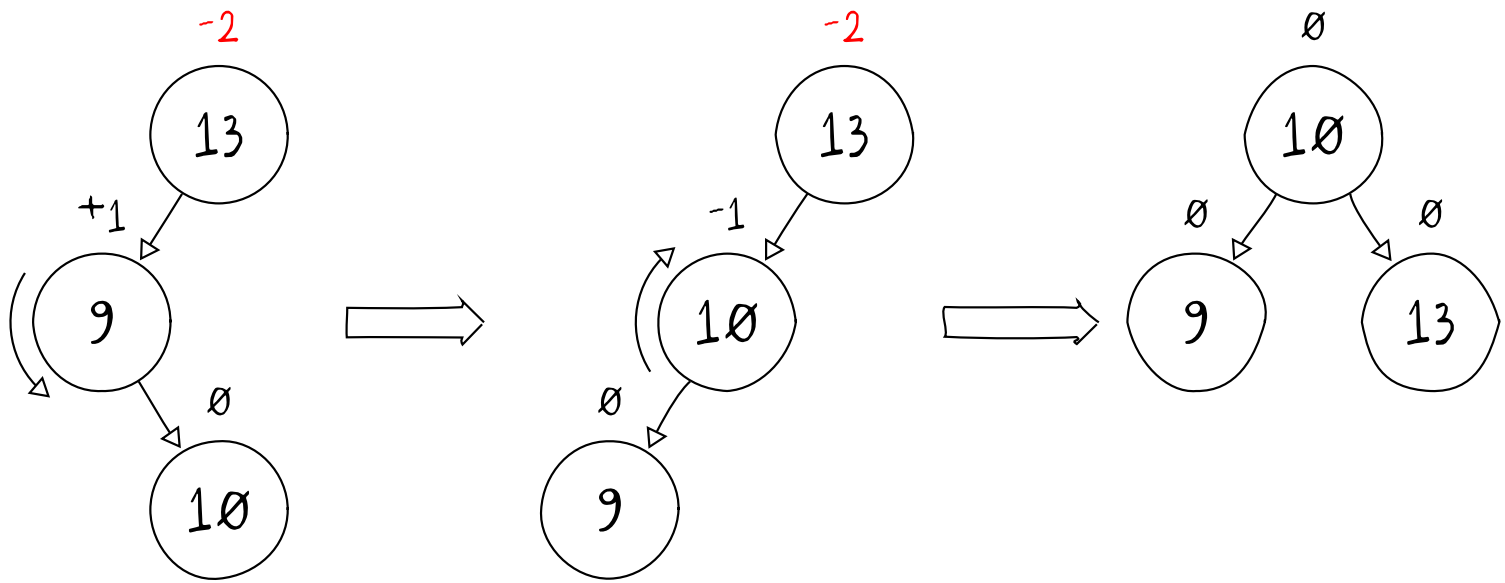
Árvore AVL

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

```
1 // Procedimento de rotação simples para direita
2 void rotacao_D(no* raiz) {
3     no* eixo = raiz->E;
4     raiz->E = eixo->D;
5     eixo->D = raiz;
6     raiz = eixo;
7     balanceamento(raiz);
8 }
```

Árvore 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



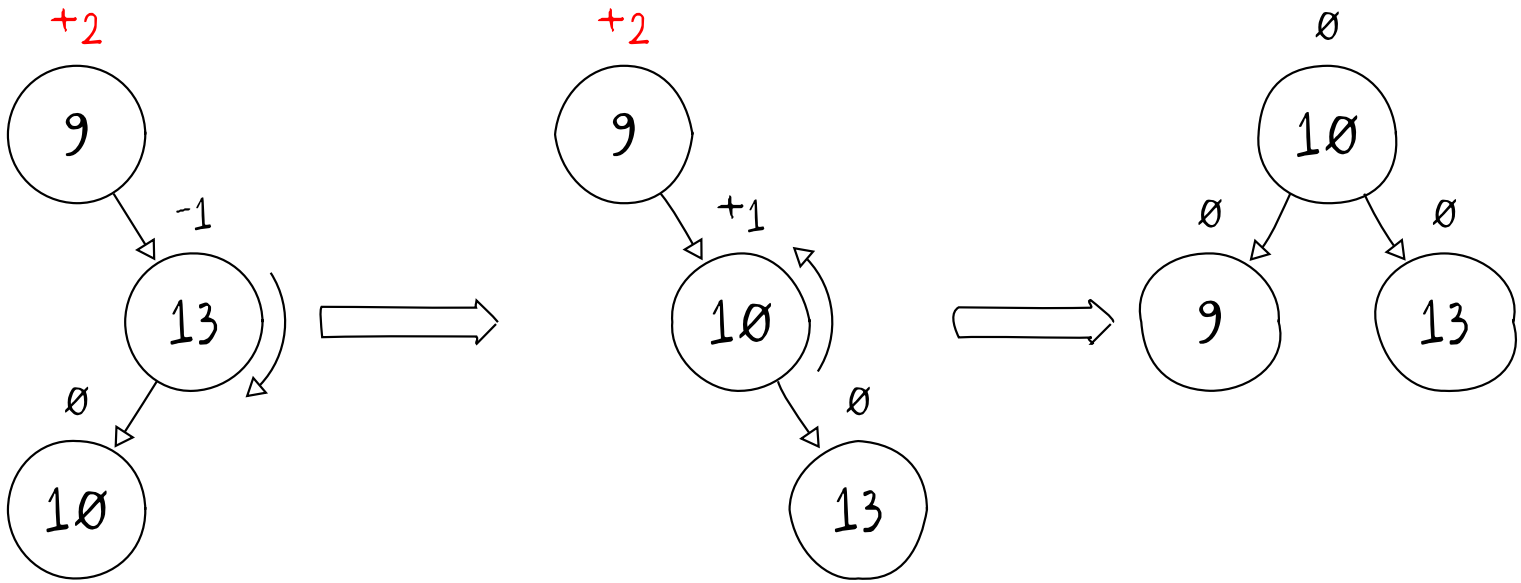
Árvore AVL

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

```
1 // Procedimento de rotação dupla esquerda-direita
2 void rotacao_E_D(no* raiz) {
3     rotacao_E(raiz->E);
4     rotacao_D(raiz);
5 }
```

Árvore 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



Árvore AVL

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

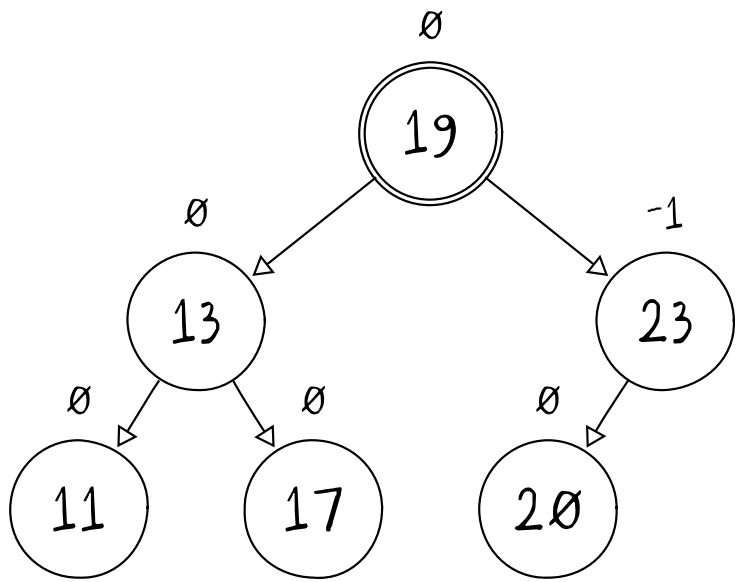
```
1 // Procedimento de rotação dupla direita-esquerda
2 void rotacao_D_E(no* raiz) {
3     rotacao_D(raiz->D);
4     rotacao_E(raiz);
5 }
```

Árvore AVL

- Operações básicas
  - Busca
  - Inserção
  - Remoção

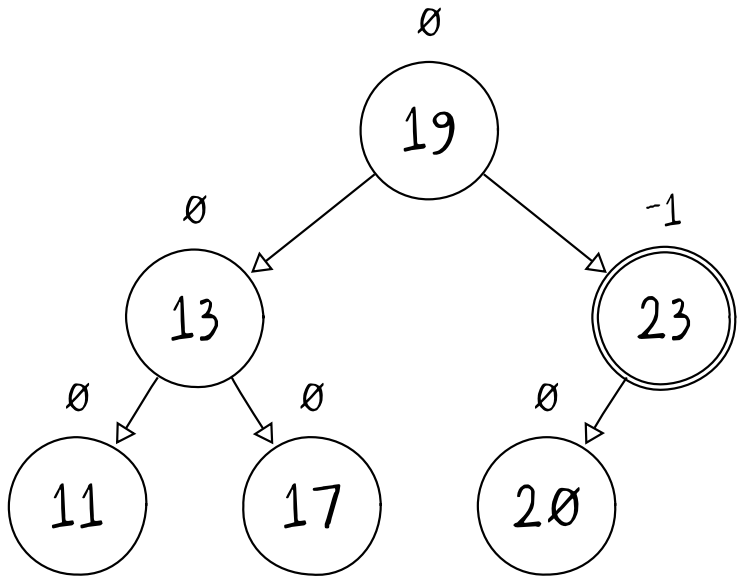
Árvore AVL

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



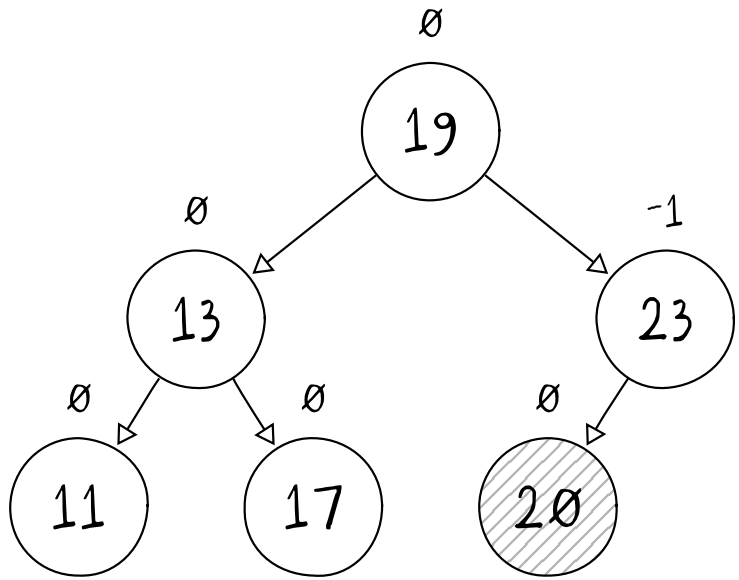
Árvore AVL

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



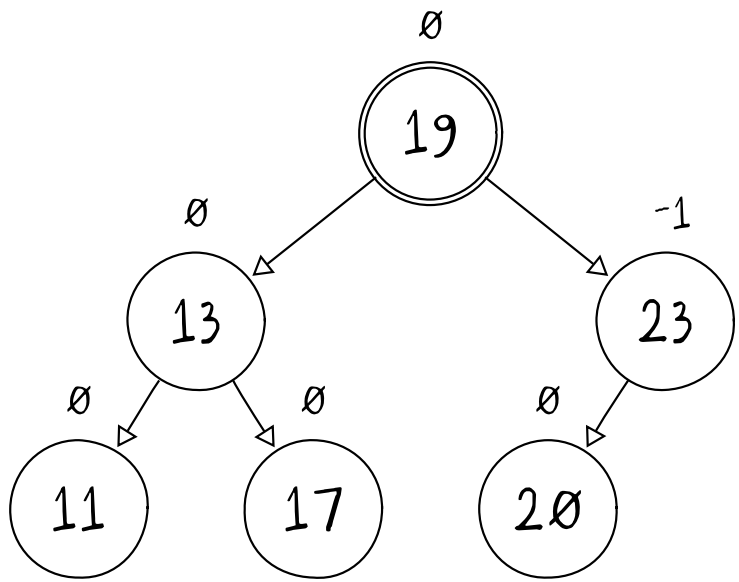
Árvore AVL

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



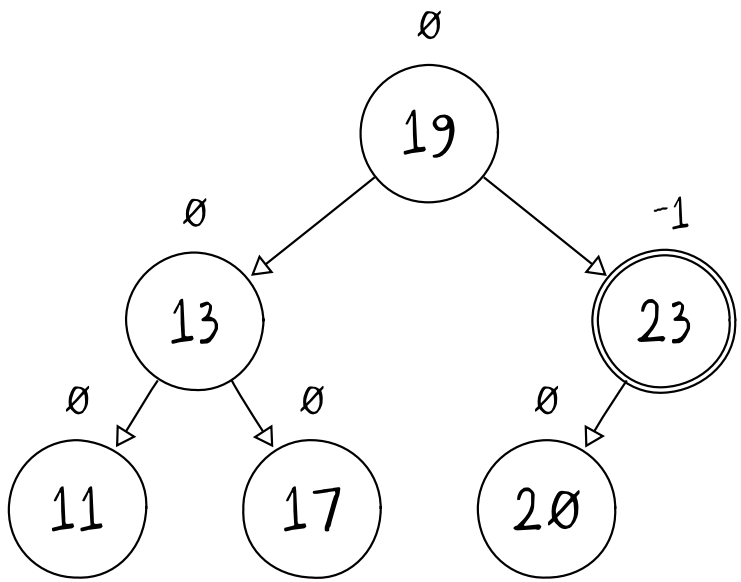
Árvore AVL

- Operação de inserção
  - Parâmetro de chave: 29
  - É feita uma busca pela chave do elemento que será inserido até encontrar uma referência nula



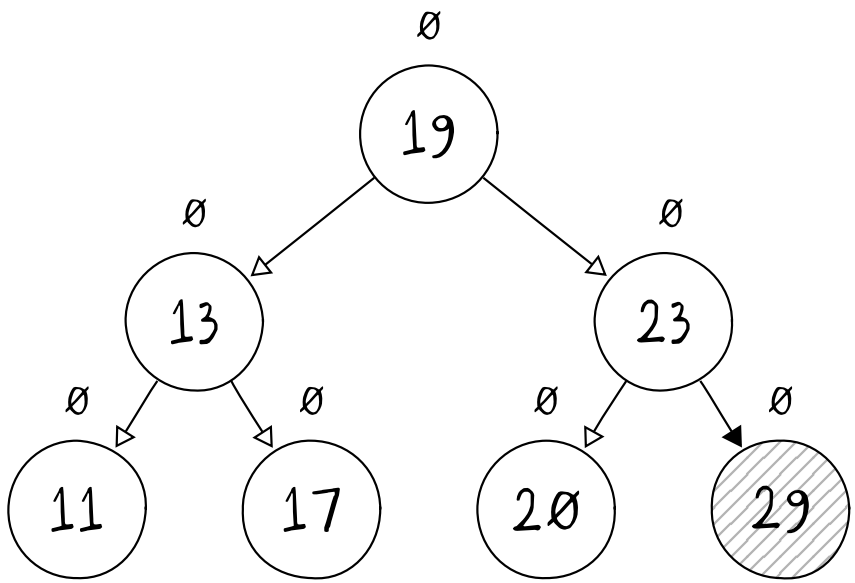
Árvore AVL

- Operação de inserção
  - Parâmetro de chave: 29
  - É feita uma busca pela chave do elemento que será inserido até encontrar uma referência nula



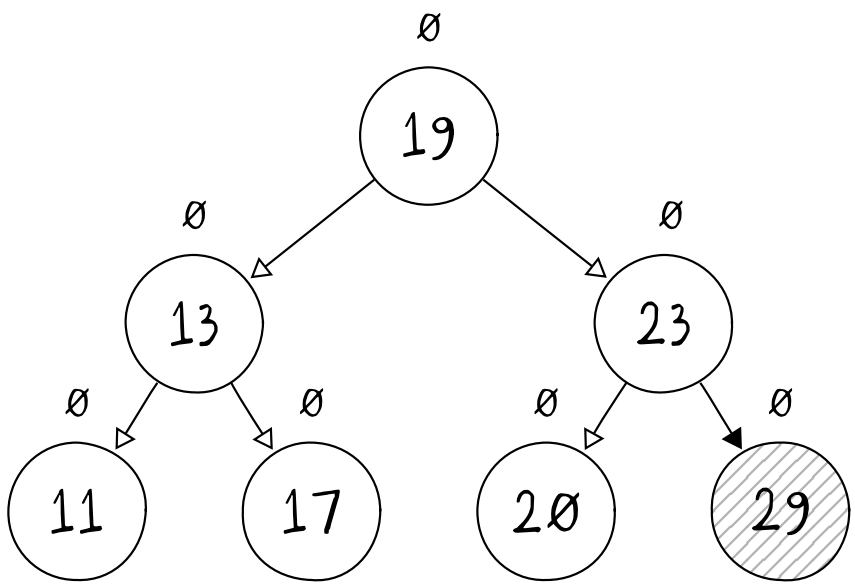
Árvore AVL

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



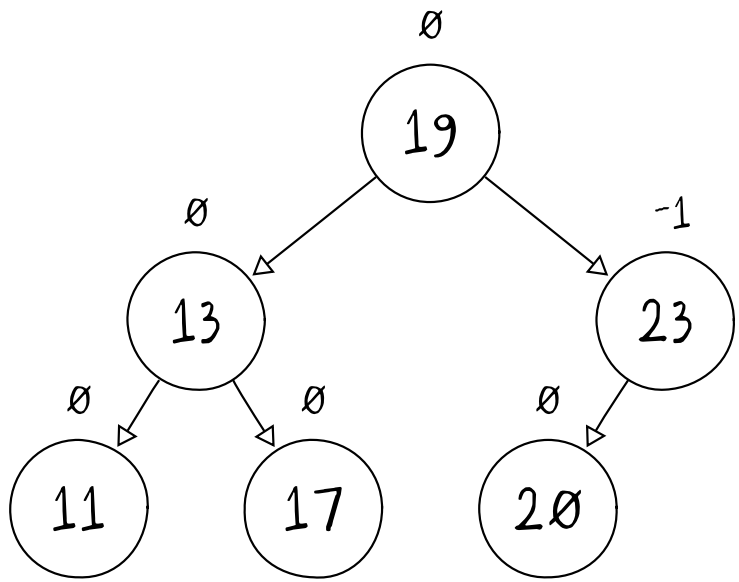
Árvore AVL

- 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



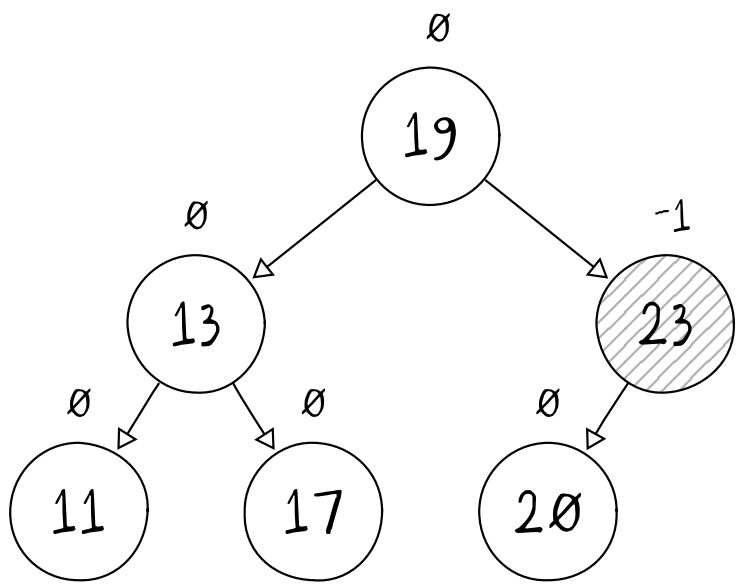
Árvore AVL

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



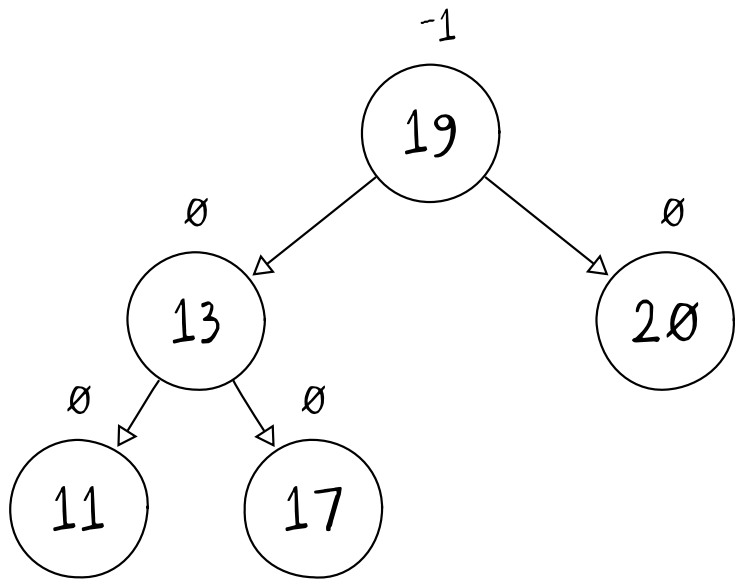
Árvore AVL

- 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



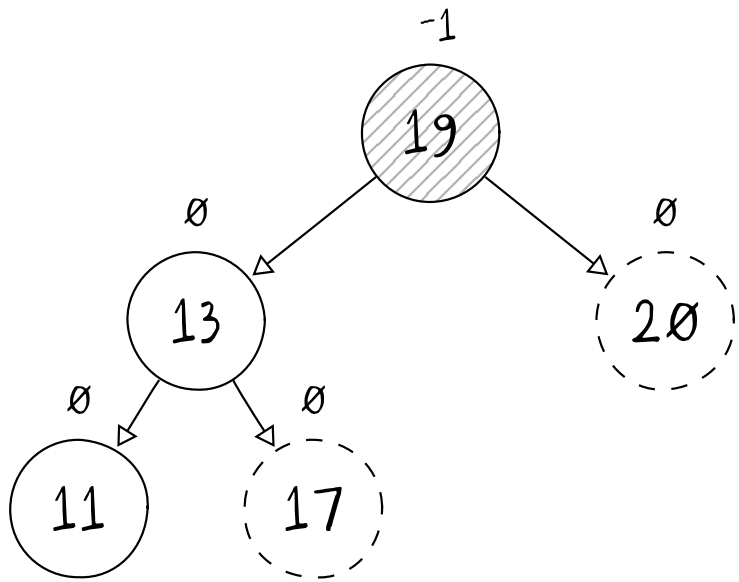
Árvore AVL

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



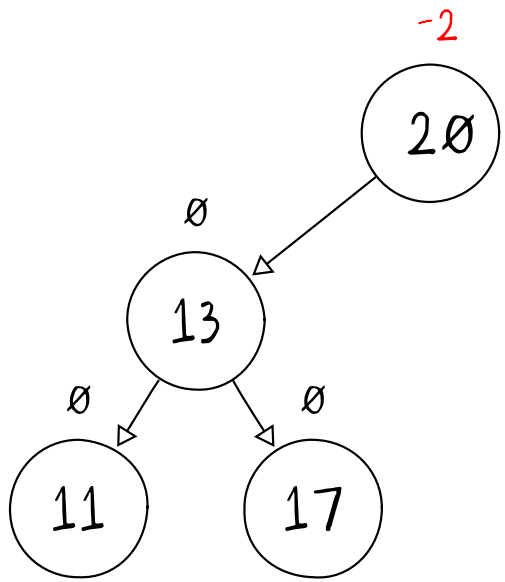
Árvore AVL

- 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



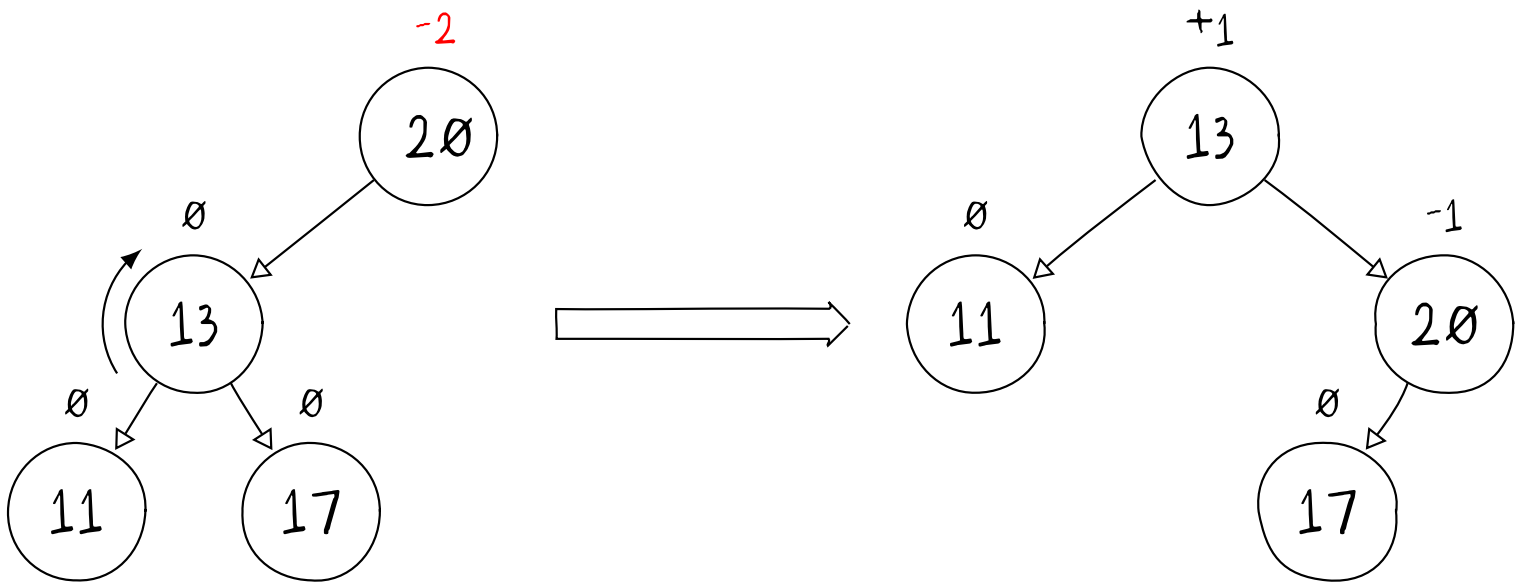
Árvore AVL

- 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



Árvore 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



Árvore 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: $\Omega(1)$ e $O(\log_2 n)$

Exemplo

- Construa uma árvore 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, quando comparada a uma árvore binária sem operações de balanceamento

Exercício

- A empresa de tecnologia Poxim Tech está desenvolvendo um sistema de dicionário de sinônimos com árvore 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*]
  - [$Consulta_1$]
  - ⋮
  - [$Consulta_m$]

Exercício

- Formato de arquivo de entrada

```
5
demais␣5␣bastante␣numeroso␣demasiado␣abundante␣excessivo
facil␣2␣simples␣ed
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 na árvore e a listagem de sinônimos da palavra pesquisada

```
1 [elegante->nada->facil]
2 simples,ed
3 [elegante->demais]
4 bastante,numeroso,demasiado,abundante,excessivo
5 [elegante->nada->trabalho->?]
6 -
```