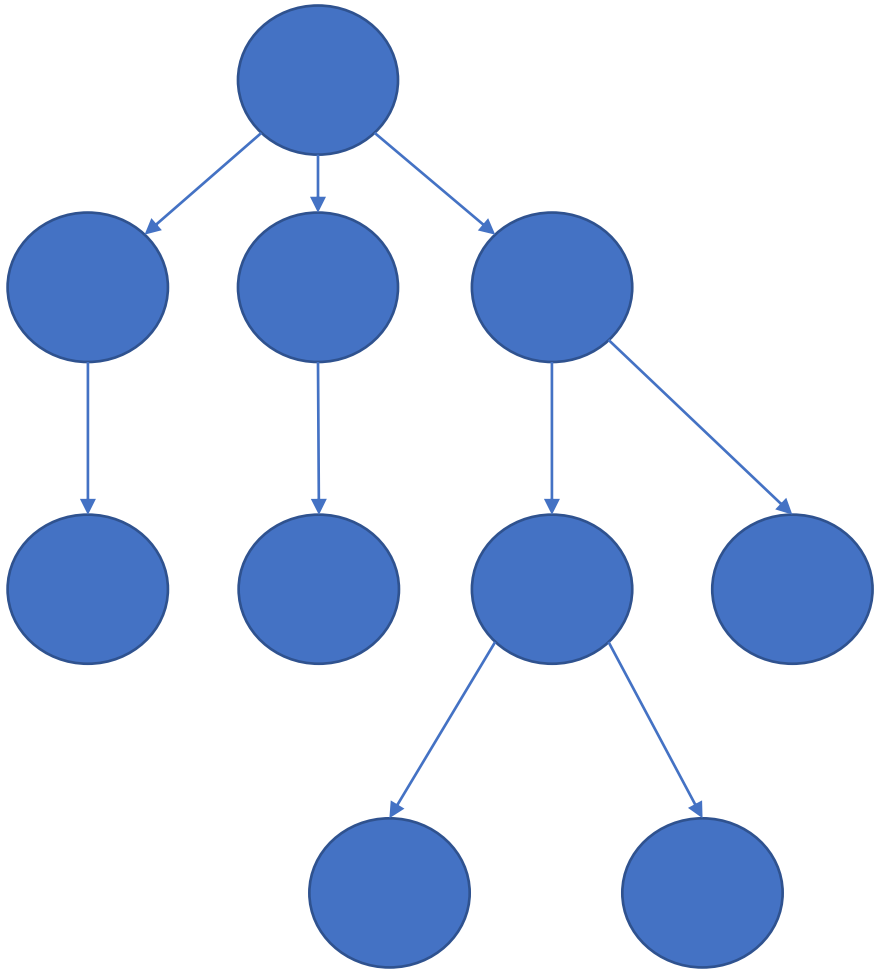
Árvores AVL

Prof. Me. Jonas Lopes de Vilas Boas

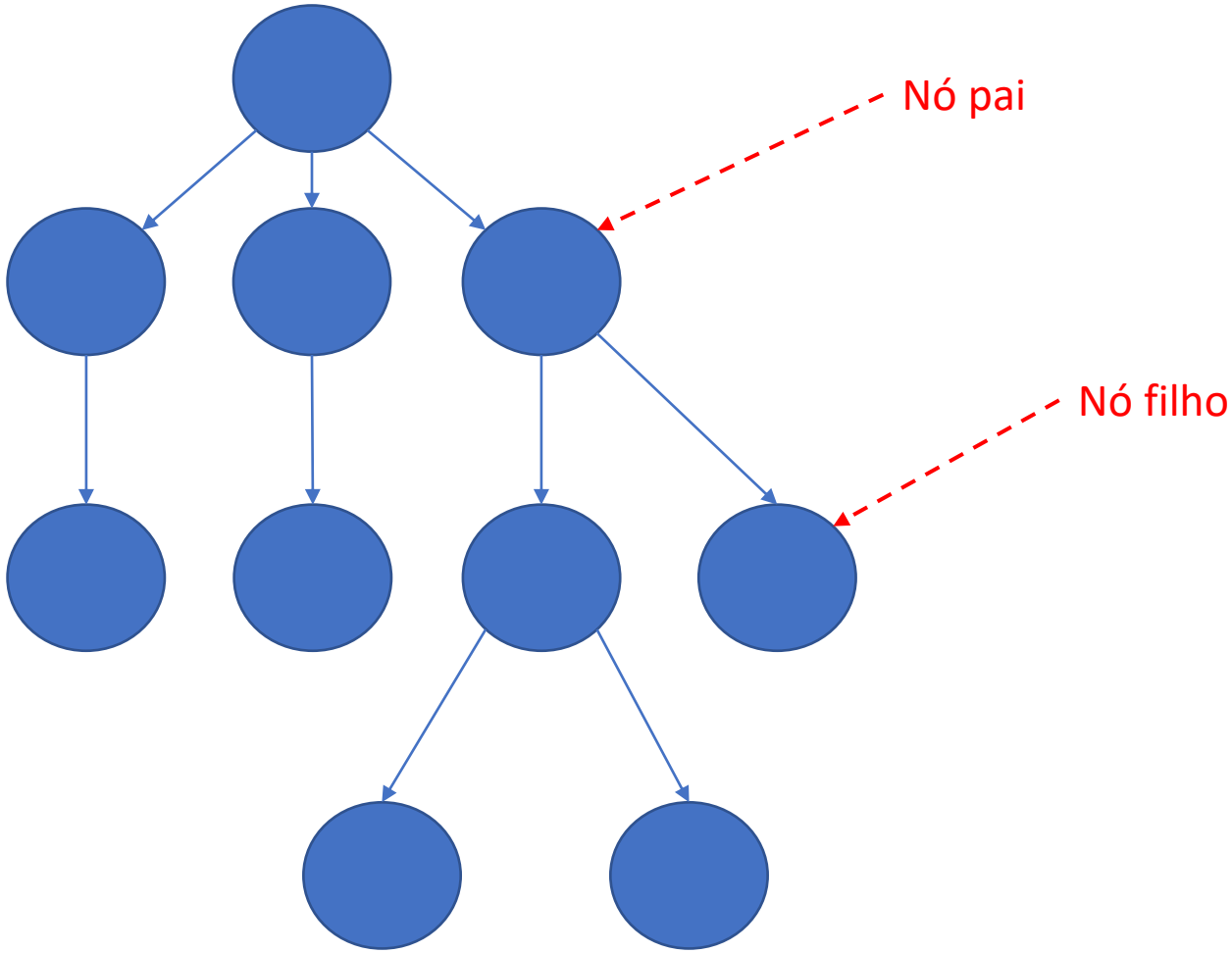
Árvore

- É um tipo de estrutura de dados não linear e dinâmica.
- É um Grafo Acíclico e Direcionado.
- Armazena os dados de forma hierárquica.

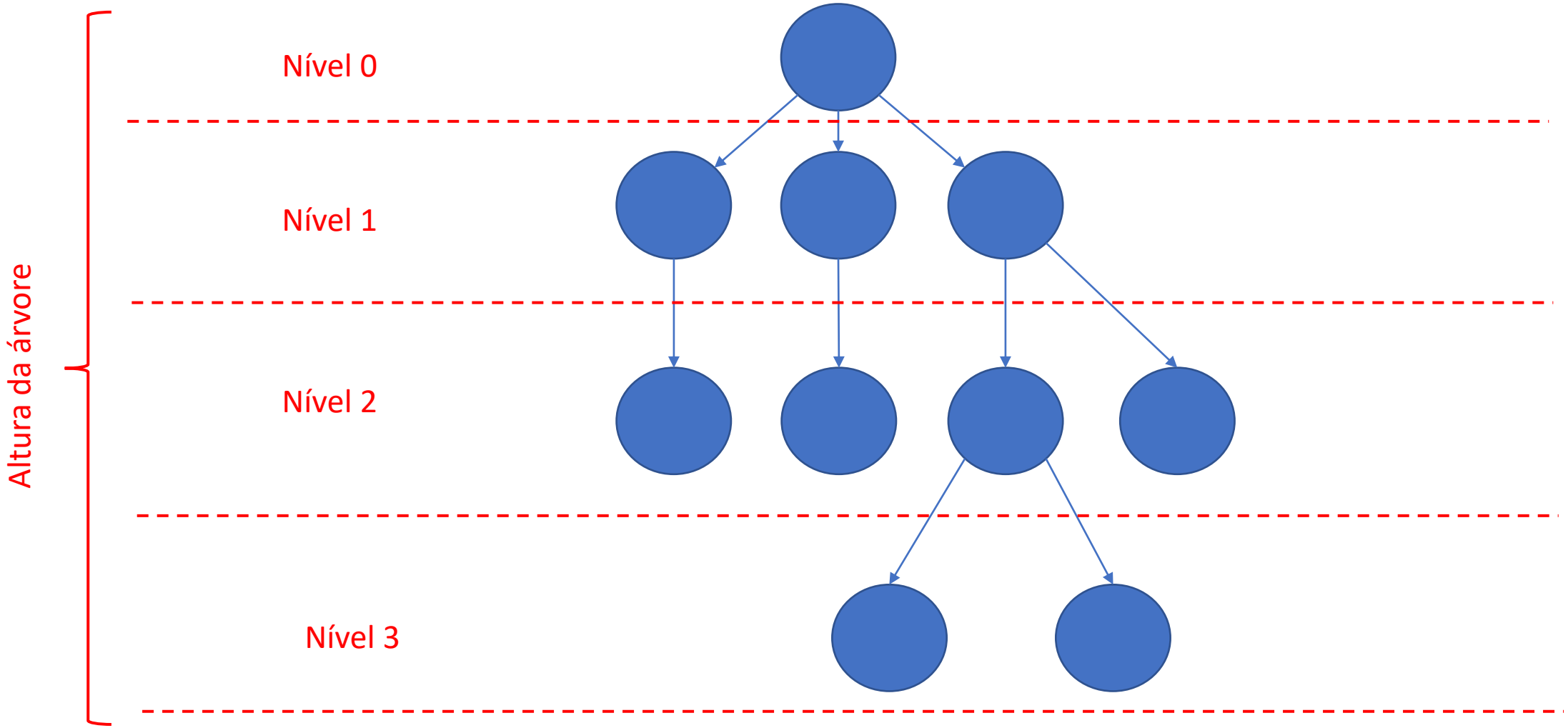
Exemplo de Árvore



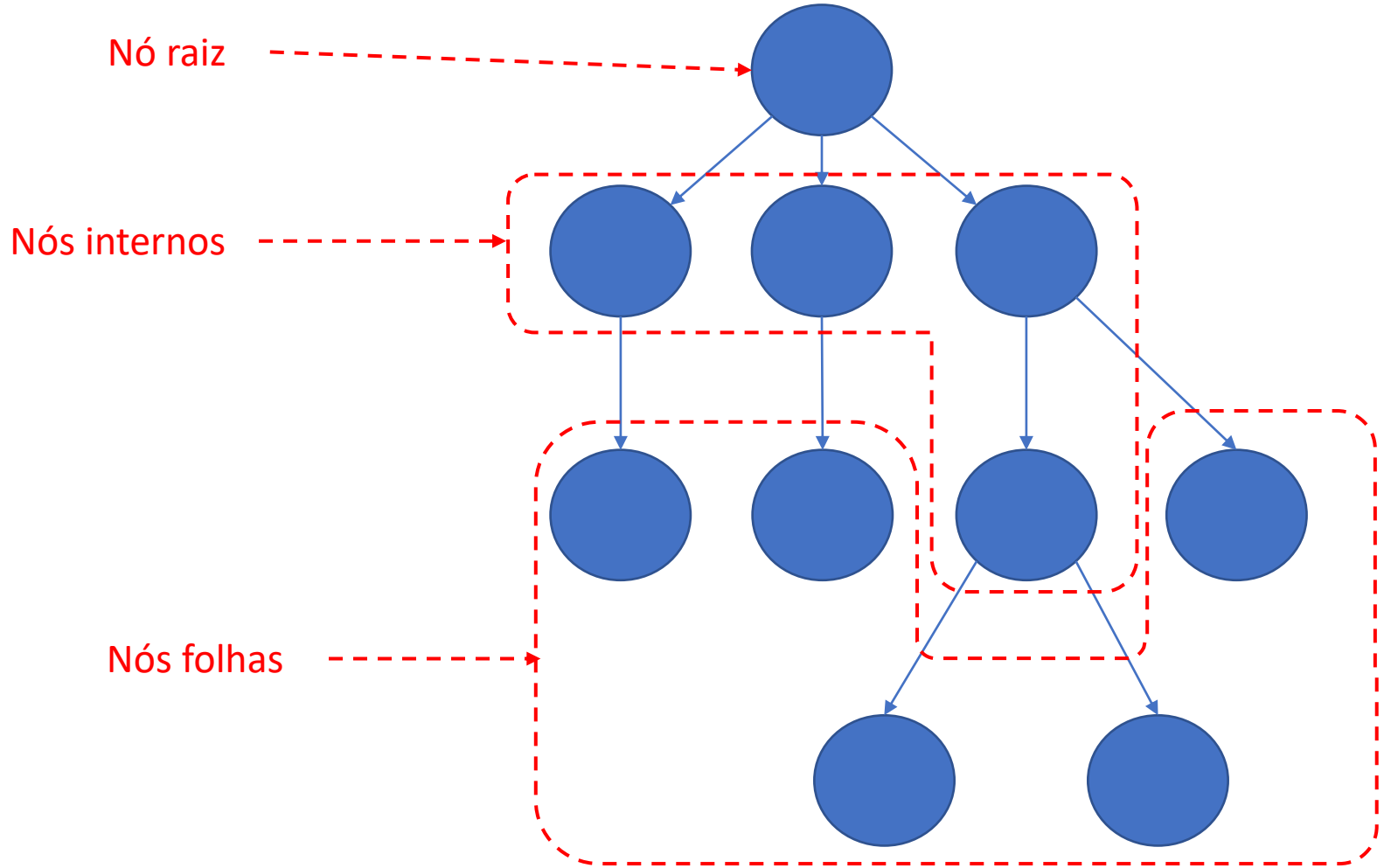
Exemplo de Árvore



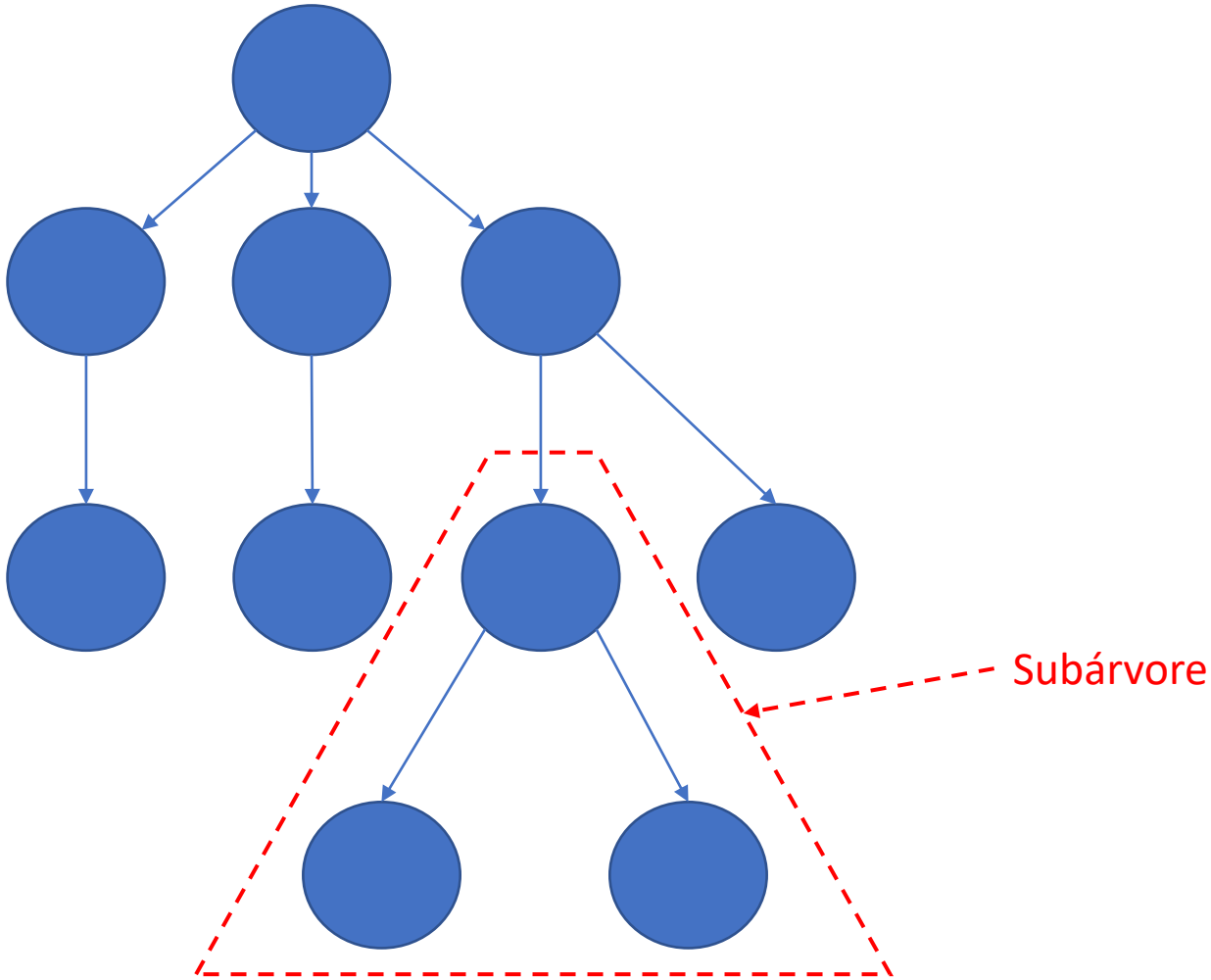
Exemplo de Árvore



Exemplo de Árvore



Exemplo de Árvore

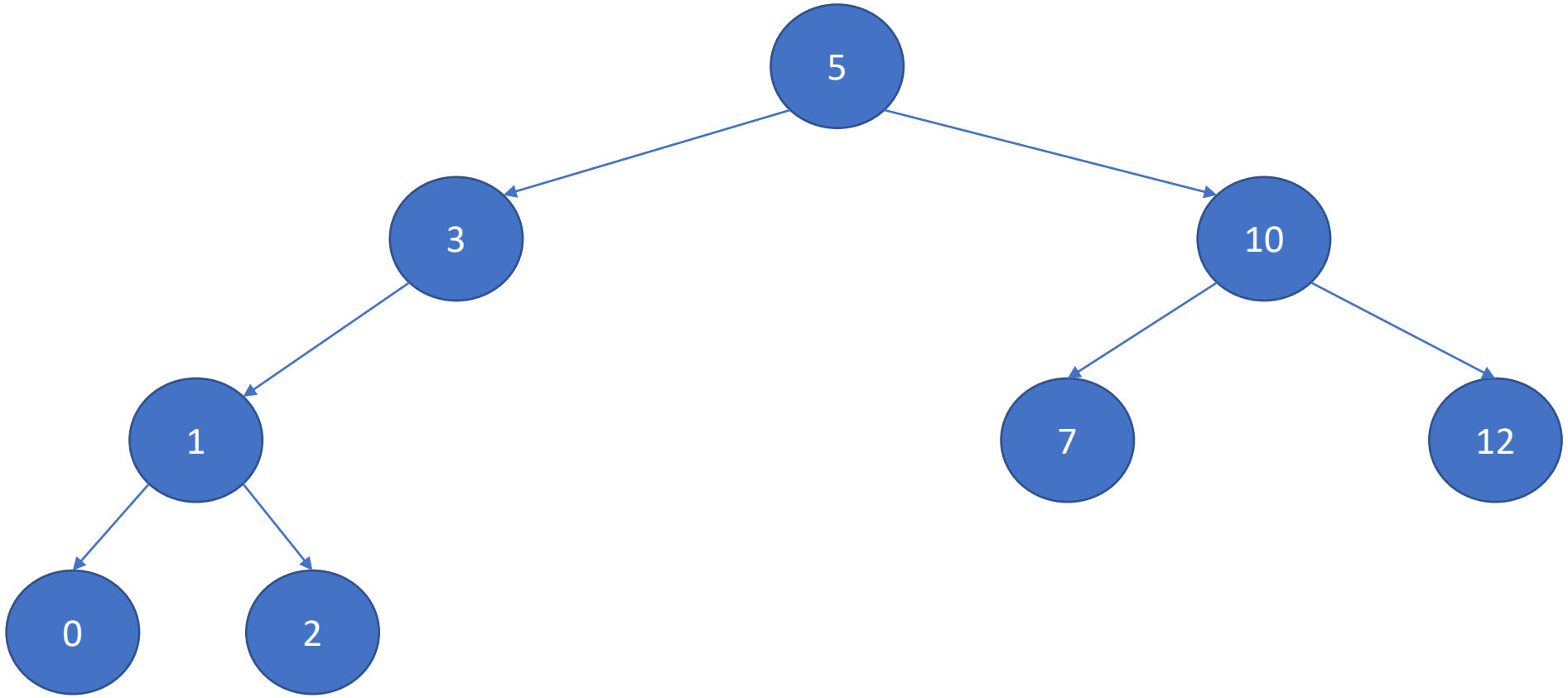


Árvore Binária

- É um tipo de árvore onde os nós podem ter no máximo dois filhos (filho da esquerda e filho da direita).

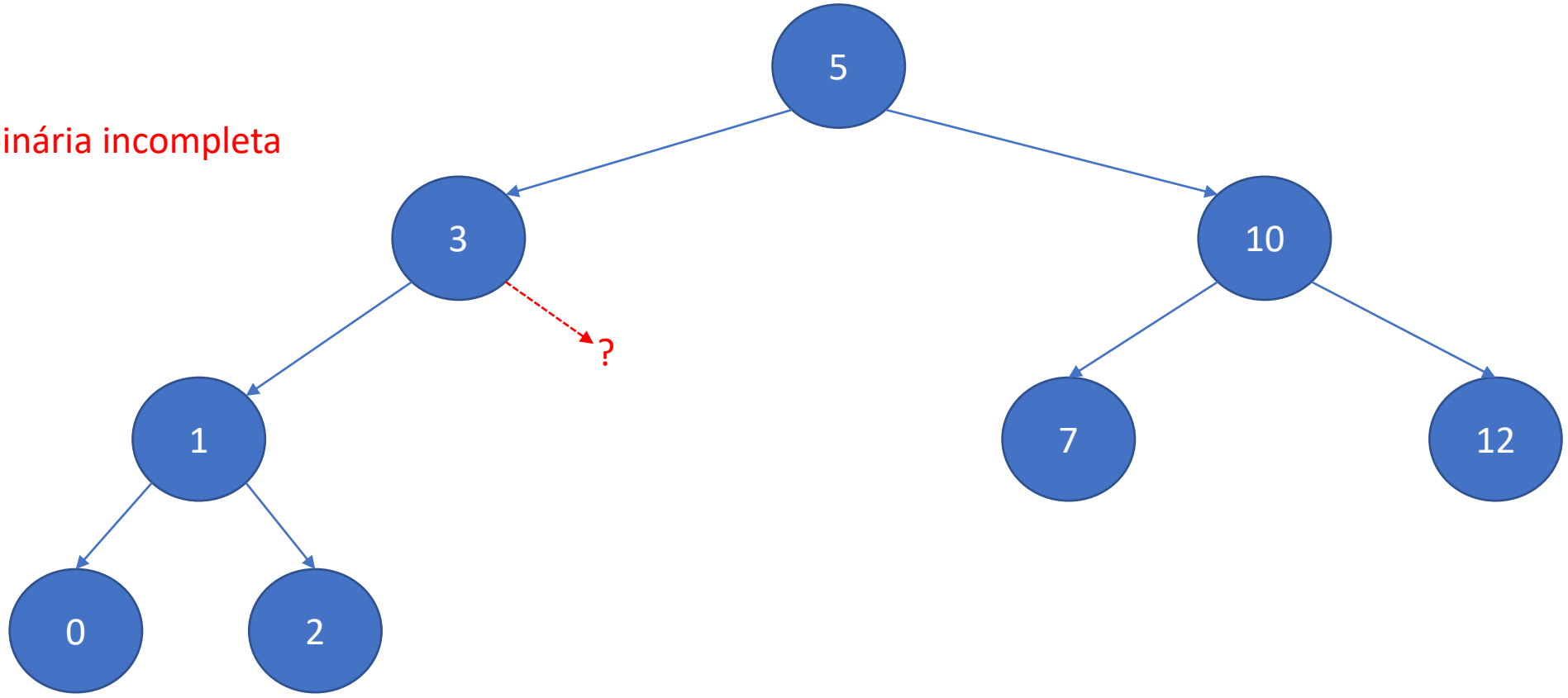
- É muito utilizada para otimização de buscas.

Exemplo de Árvore Binária de Busca



Exemplo de Árvore Binária de Busca

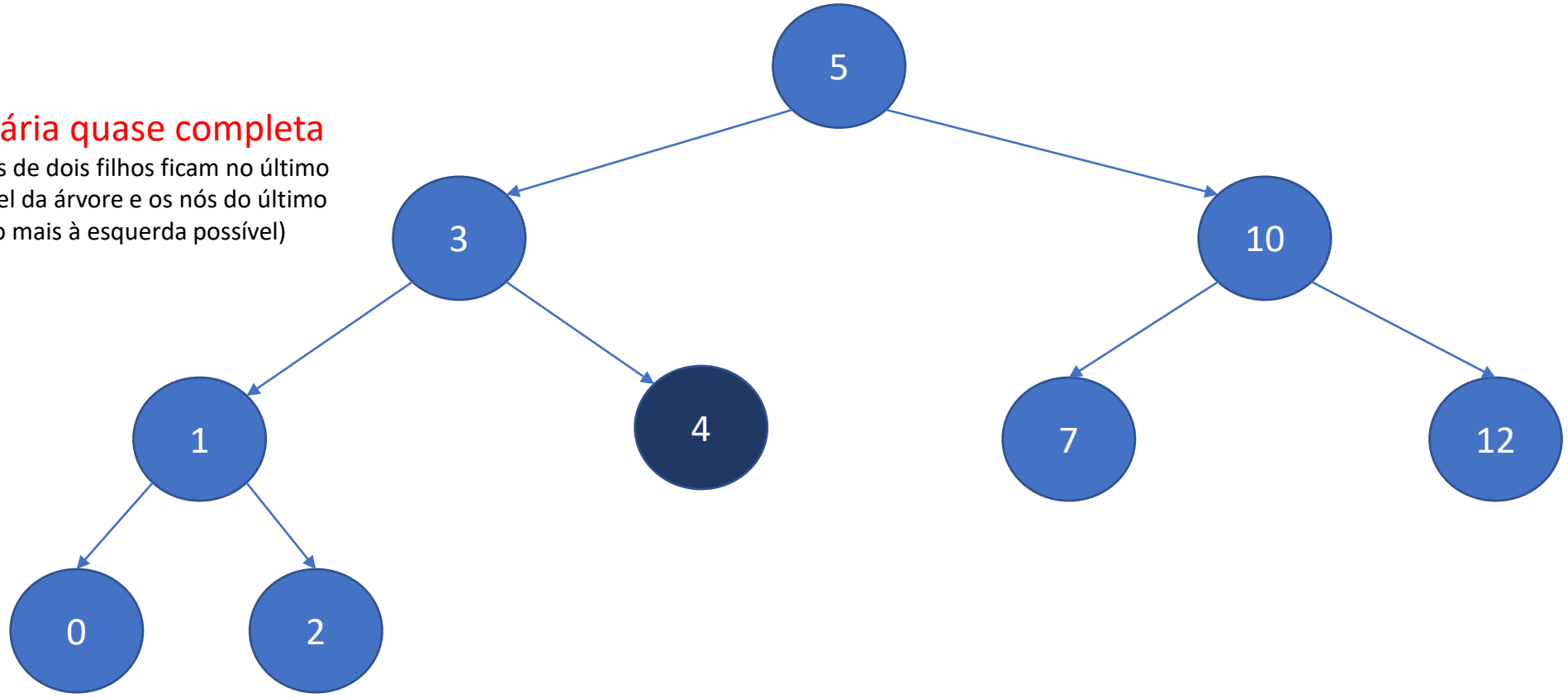
Árvore binária incompleta



Exemplo de Árvore Binária de Busca

Árvore binária quase completa

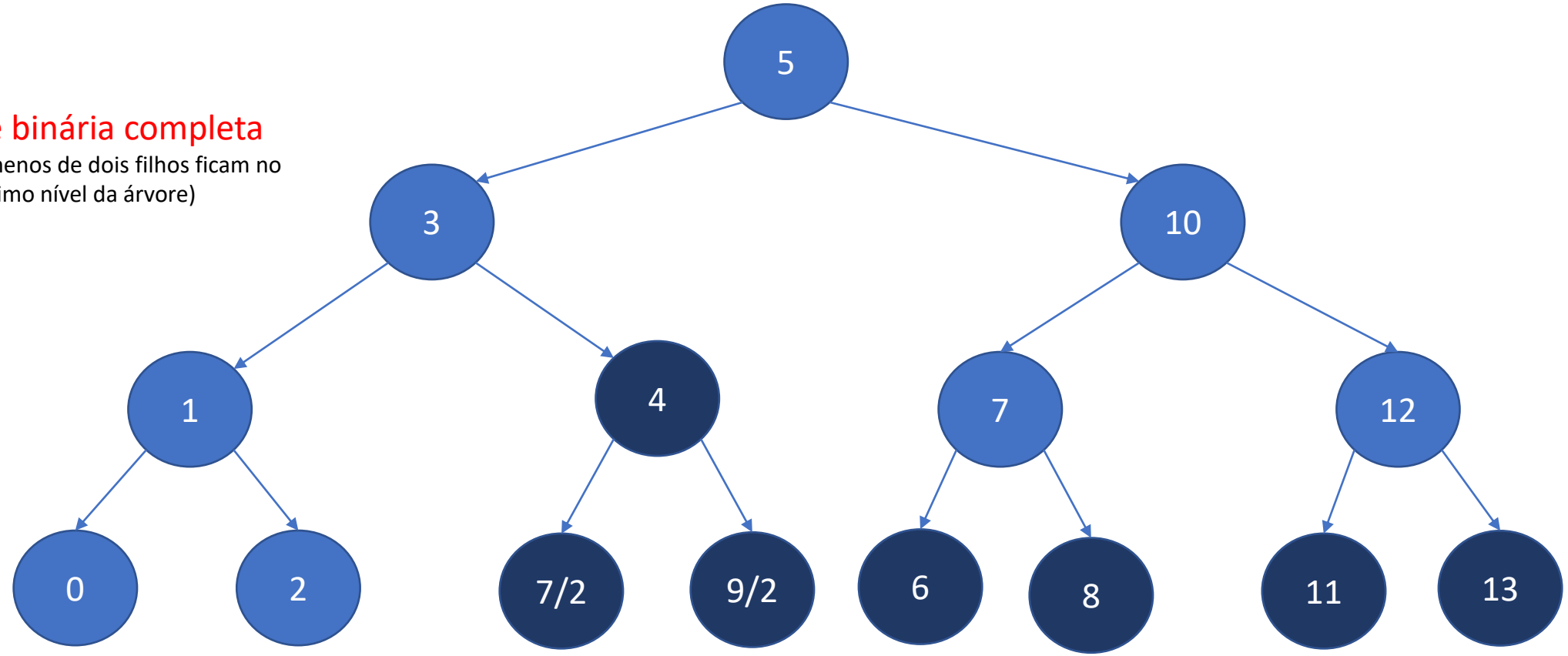
(Nós com menos de dois filhos ficam no último e penúltimo nível da árvore e os nós do último nível ficam o mais à esquerda possível)



Exemplo de Árvore Binária de Busca

Árvore binária completa

(Nós com menos de dois filhos ficam no último nível da árvore)



Árvore Binária de Busca

- É um tipo de árvore binária que é construída com uma estratégia para facilitara busca binária.

- A estratégia é já construir a estrutura de maneira ordenada:
  - Ao inserir um novo nó, ele é comparado com os nós já existentes;
  - Começando da raiz, se o nó a ser inserido tem valor menor que o nó visitado, ele continua a verificação para a esquerda, senão para a direita;
  - Se o nó visitado for um nó folha, o novo nó é inserido à esquerda se o valor for maior, ou à direita se for menor.

- Complexidade da busca: O(h), onde h é a altura da árvore

Árvore Binária de Busca

- Representação de um nó da árvore

```
struct treenode {
        int info;
        struct treenode * left;
        struct treenode * right;
};
```

Árvore Binária de Busca

- Algoritmo de inserção na árvore

```
void tInsert(treenodeptr &p, int x){
        if (p == NULL) {
                p = new treenode;
                p->info = x;
                p->left = NULL;
                p->right = NULL;
        } else if (x < p->info)
                tInsert(p->left, x);
        else
                tInsert(p->right, x);
}
```

Árvore Binária de Busca

- Algoritmo de busca na árvore em C

```
treenodeptr tSearch(treenode p, int x) {
        if (p == NULL)
                return NULL;
        else if (x == p->info)
                return p;
        else
                if (x < p->info)
                        return tSearch(p->left,x);
                else
                        return tSearch(p->right,x);
}
```

Complexidade da Árvore Binária de Busca

- Pode-se perceber que, no pior caso, a complexidade de busca vai ser igual ao número de elementos, ou seja, O(n).

- Dessa forma, é necessário um mecanismo para garantir a construção da árvore de maneira a se aproximar ao máximo da complexidade da busca binária, ou seja, $\mathrm{O}(\log_2 n)$.

- As árvores balanceadas são soluções para esse problema.

Árvore AVL

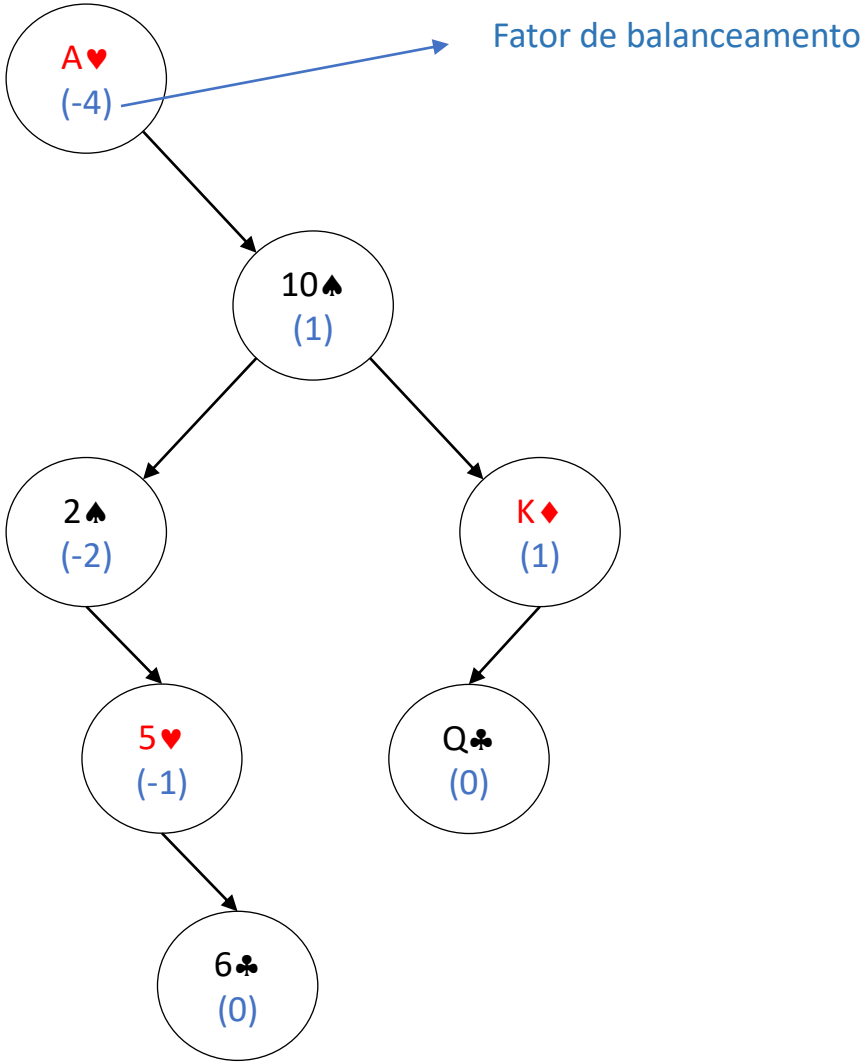
- É um tipo de árvore binária na qual as alturas das duas subárvores de todos os nós nunca difere em mais de um.

- O balanceamento de um nó é definido como a diferença da altura de sua subárvore esquerda pela altura da sua subárvore direita.

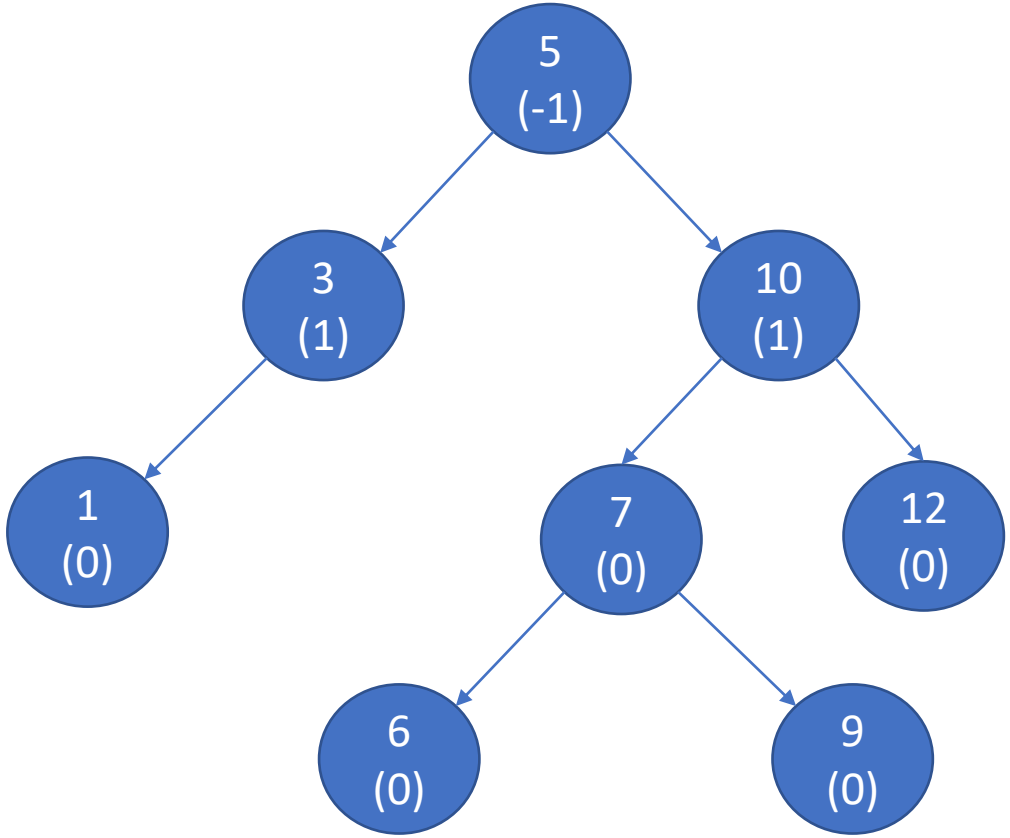
- Dessa forma, os possíveis valores de balanceamento são -1, 0 ou 1.

- Uma árvore binária balanceada garante buscas mais eficientes.

Exemplo de Árvore Binária desbalanceada



Exemplo de Árvore AVL



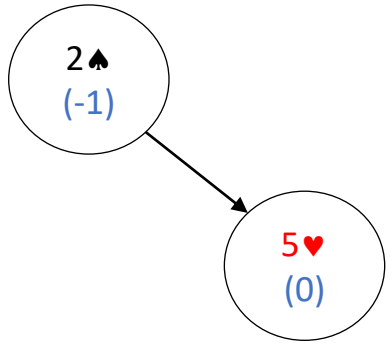
Rotação

- É um mecanismo para transformar a árvore binária para torna-la balanceada, sem mudar as características de navegação inicial.

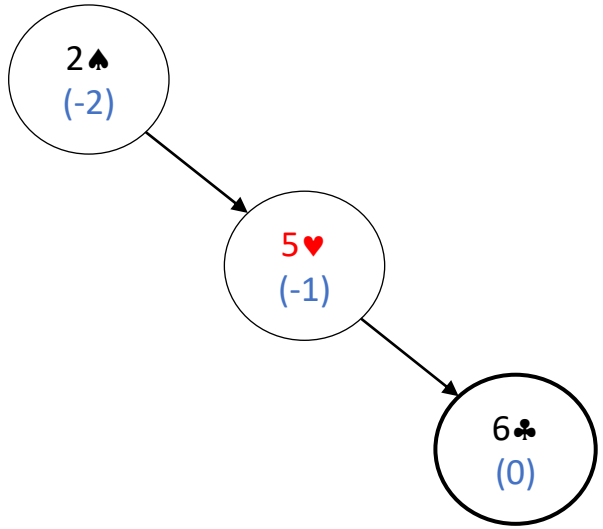
- O ideal é realizar a rotação sempre que uma árvore se torna desbalanceada.

- Existem rotações simples e rotações duplas, que podem ser feitas à esquerda ou à direita.

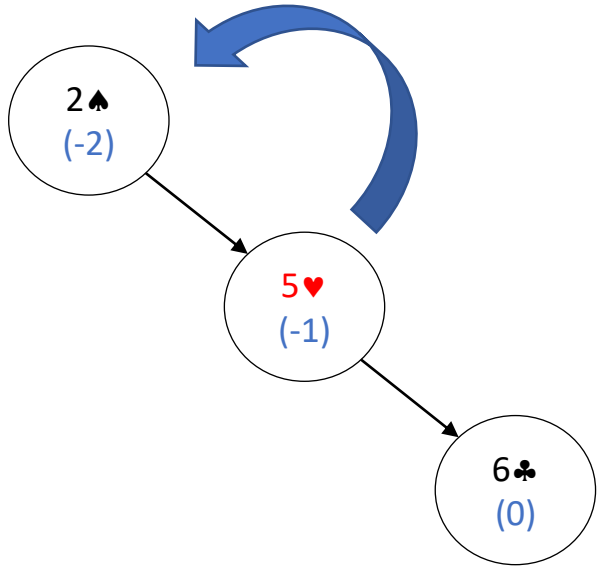
Rotação simples à esquerda (SE)



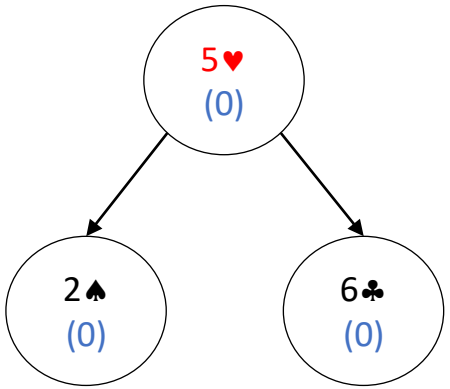
Rotação simples à esquerda (SE)



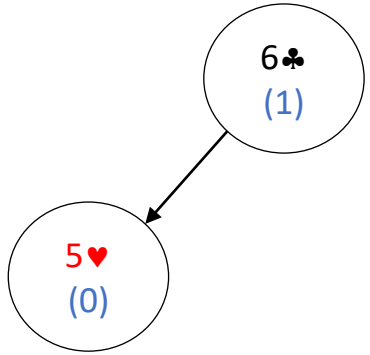
Rotação simples à esquerda (SE)



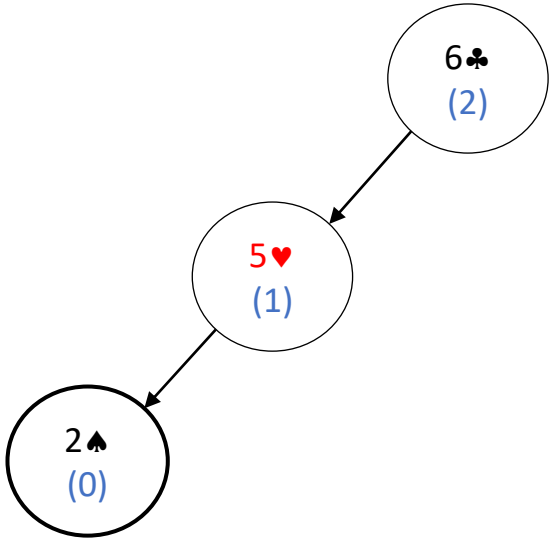
Rotação simples à esquerda (SE)



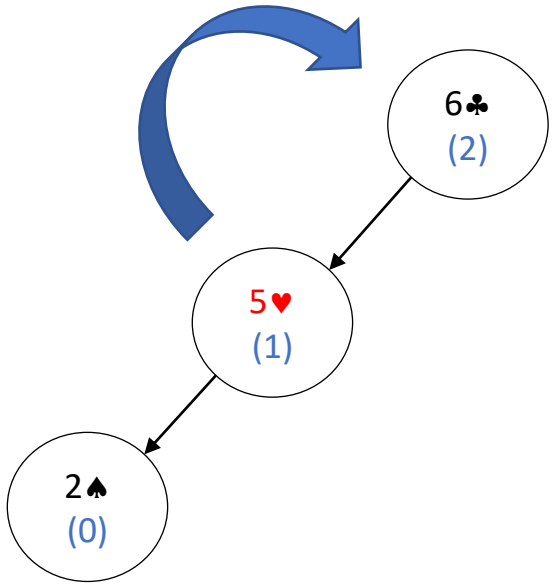
Rotação simples à direita (SD)



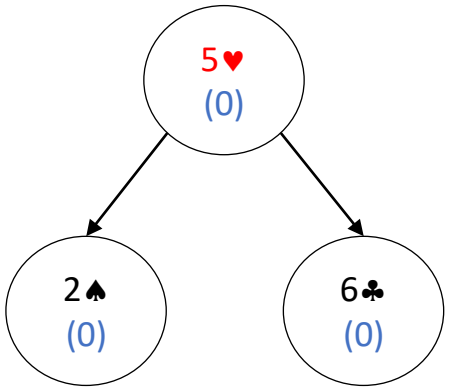
Rotação simples à direita (SD)



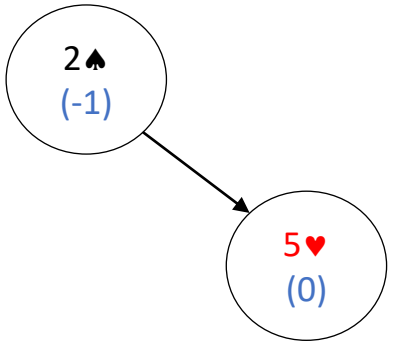
Rotação simples à direita (SD)



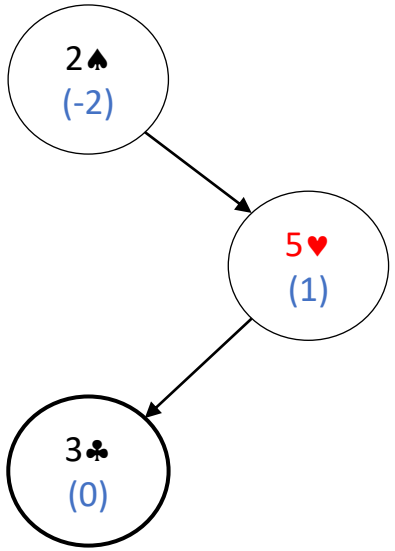
Rotação simples à direita (SD)



Rotação dupla à esquerda (DE)

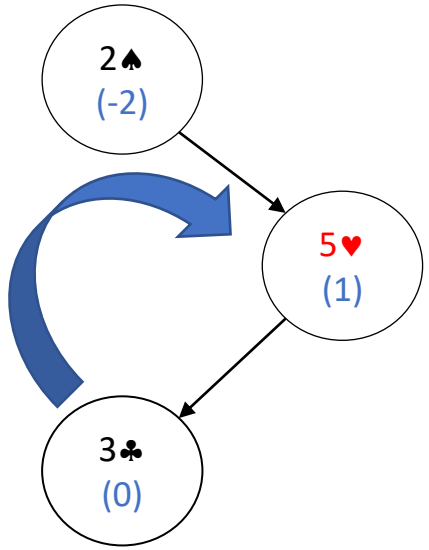


Rotação dupla à esquerda (DE)

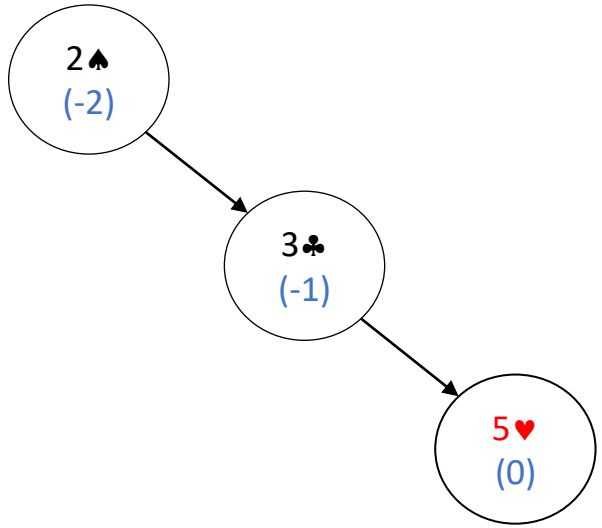


Rotação dupla à esquerda (DE)

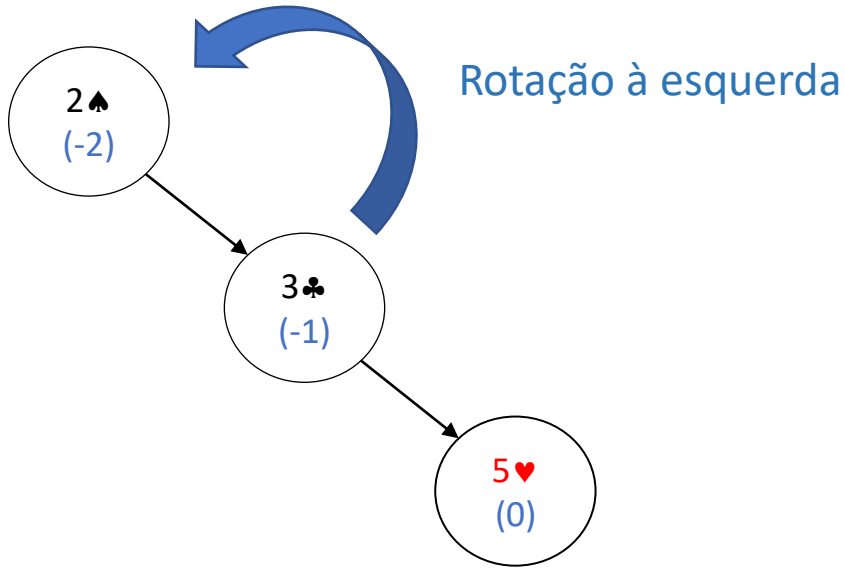
Rotação à direita



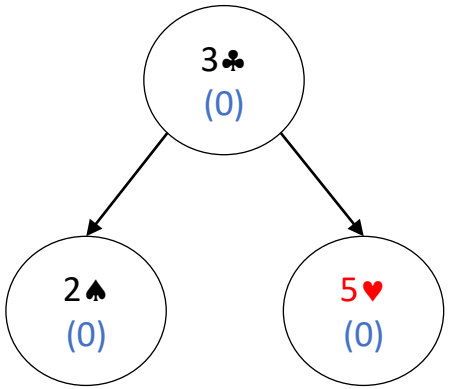
Rotação dupla à esquerda (DE)



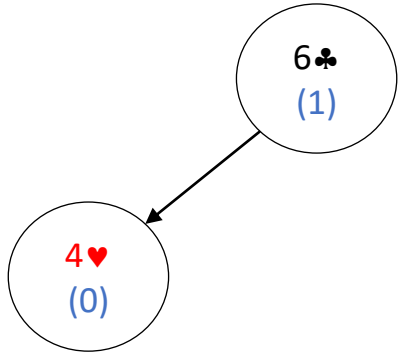
Rotação dupla à esquerda (DE)



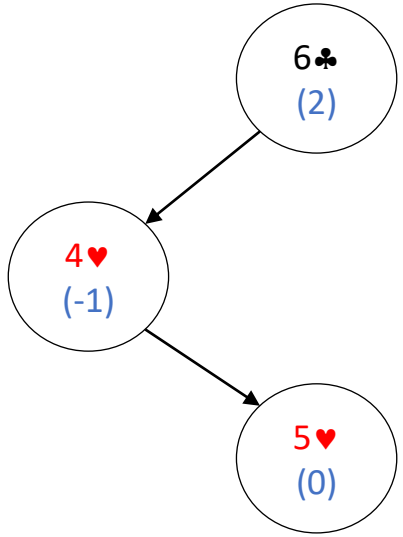
Rotação dupla à esquerda (DE)



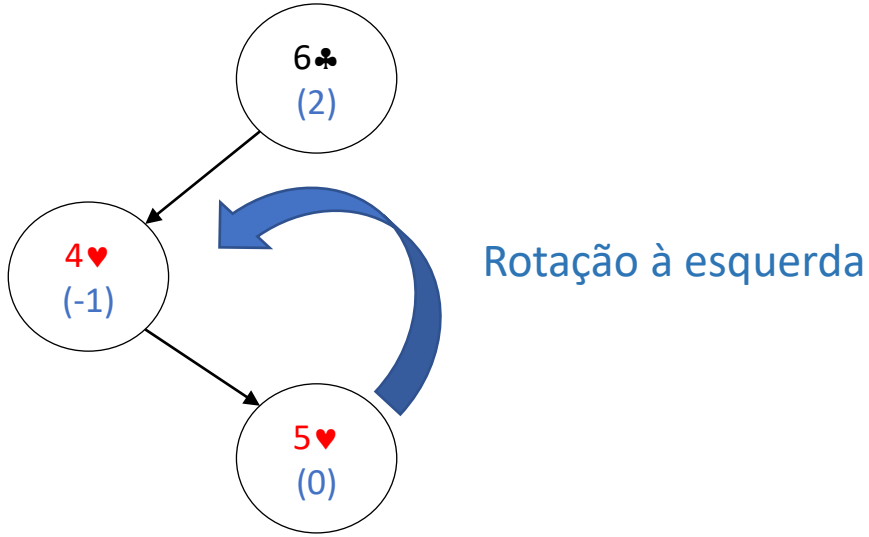
Rotação dupla à direita (DD)



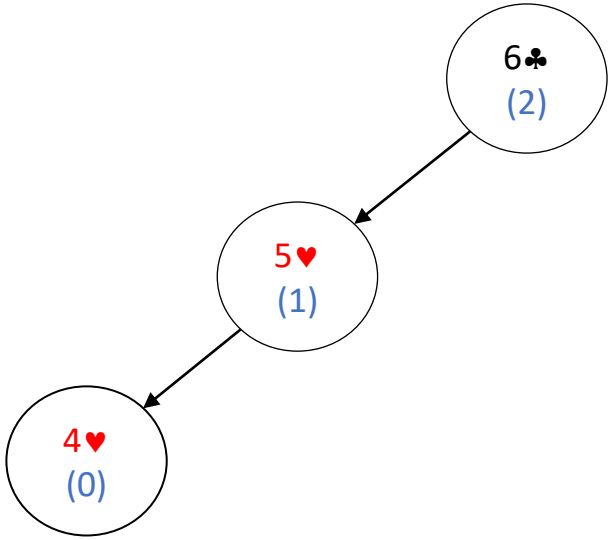
Rotação dupla à direita (DD)



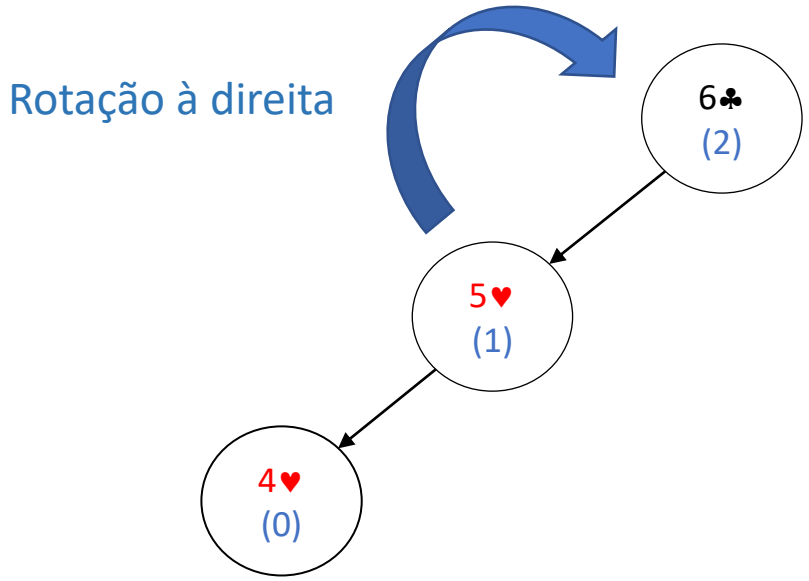
Rotação dupla à direita (DD)



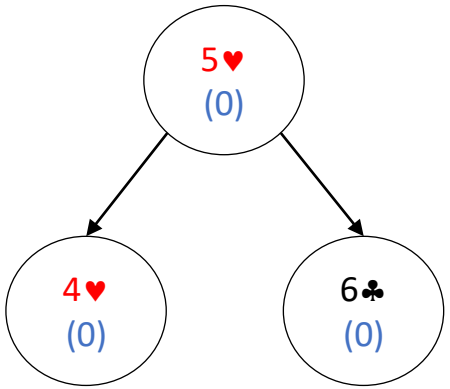
Rotação dupla à direita (DD)



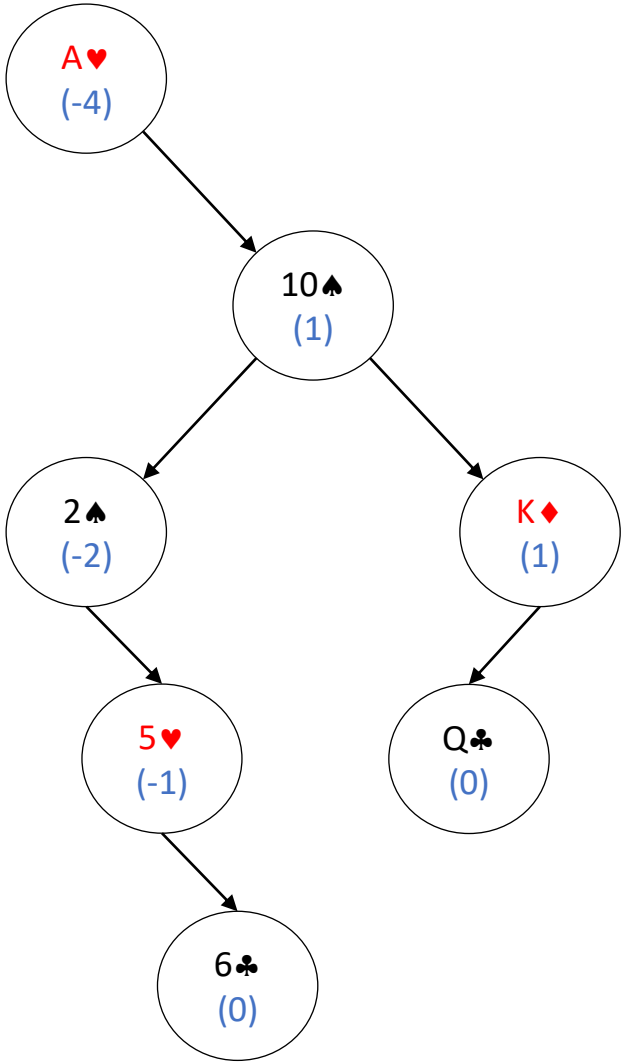
Rotação dupla à direita (DD)



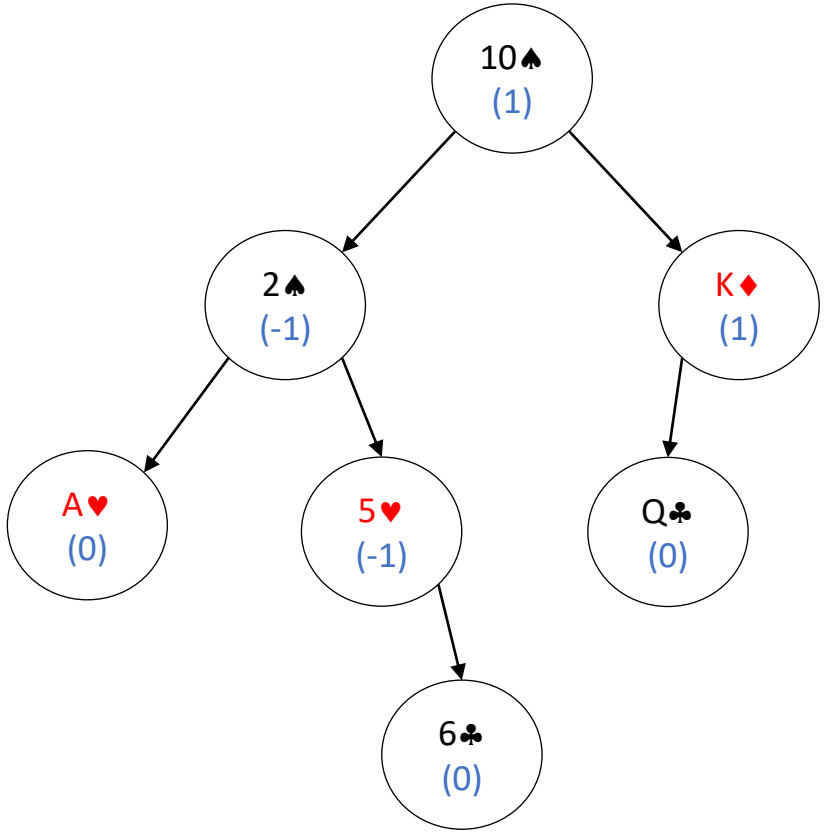
Rotação dupla à direita (DD)



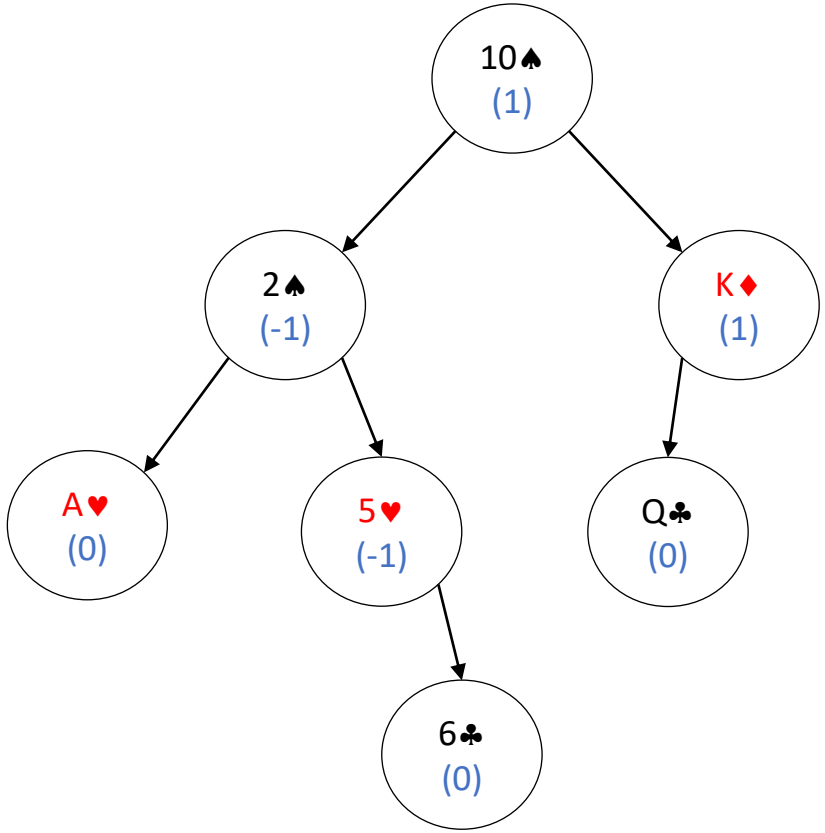
Exemplo de Árvore Binária desbalanceada



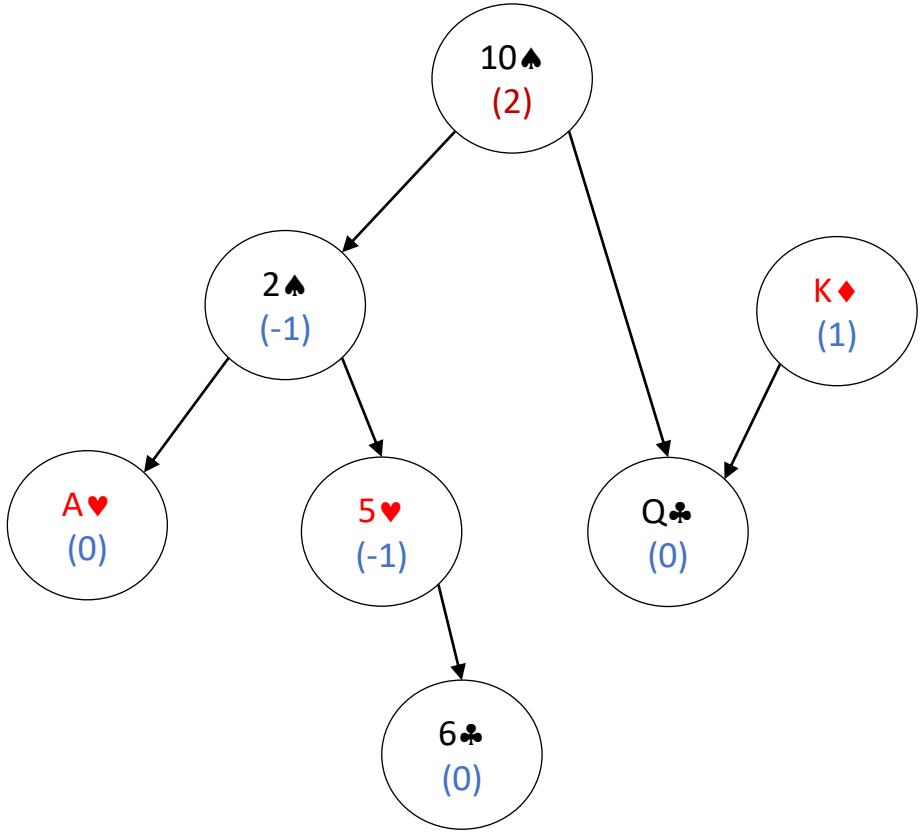
Exemplo de Árvore Binária balanceada



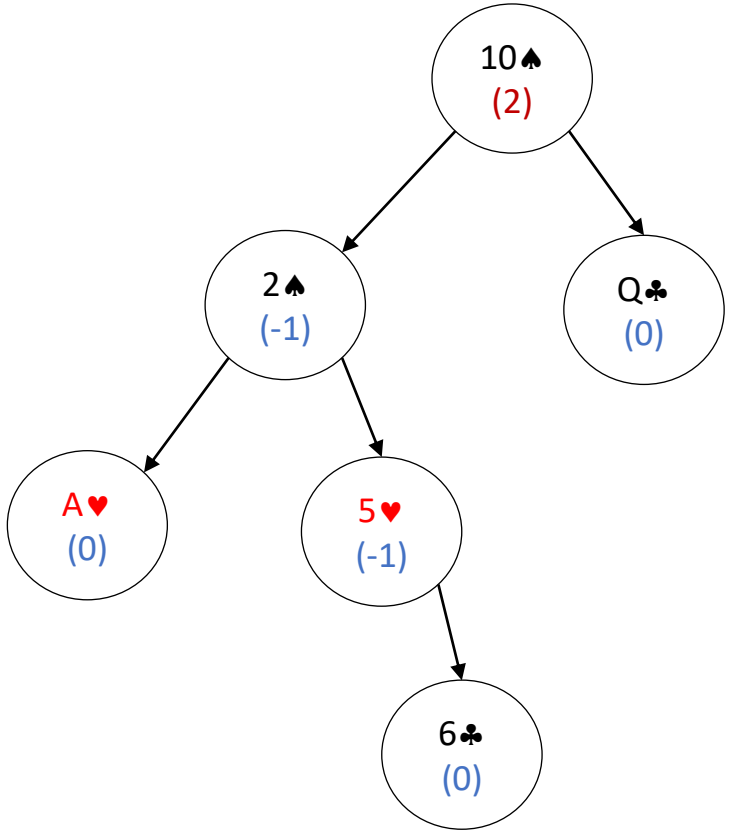
Removendo K♦ da árvore AVL



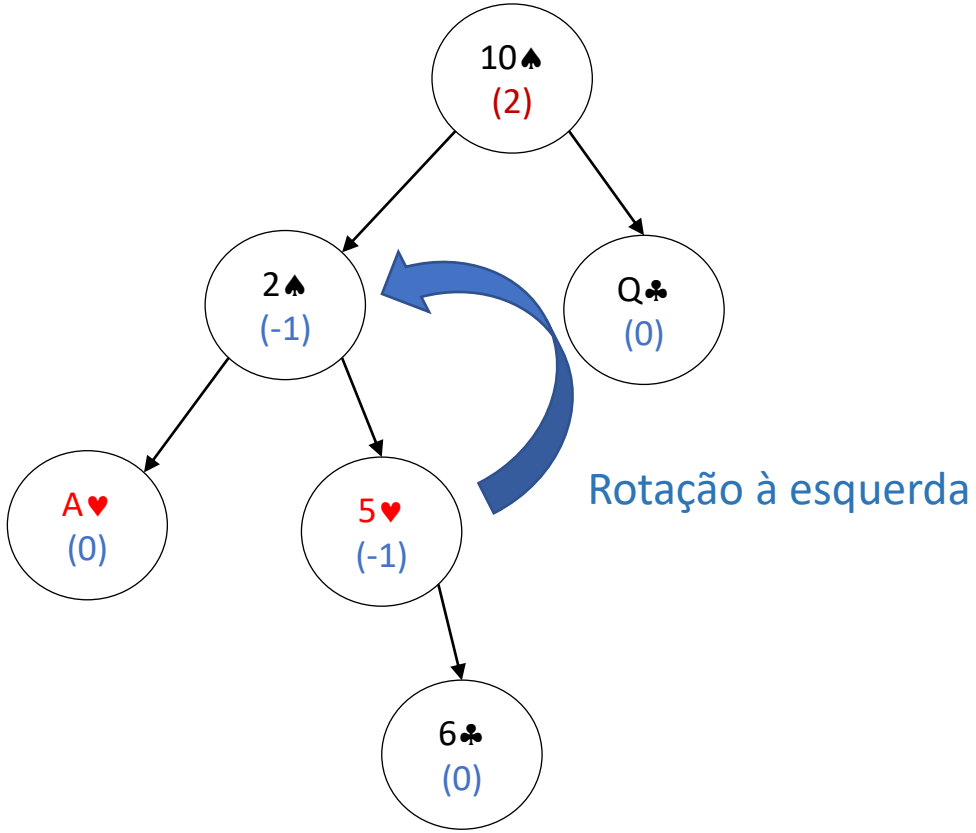
Removendo K♦ da árvore AVL



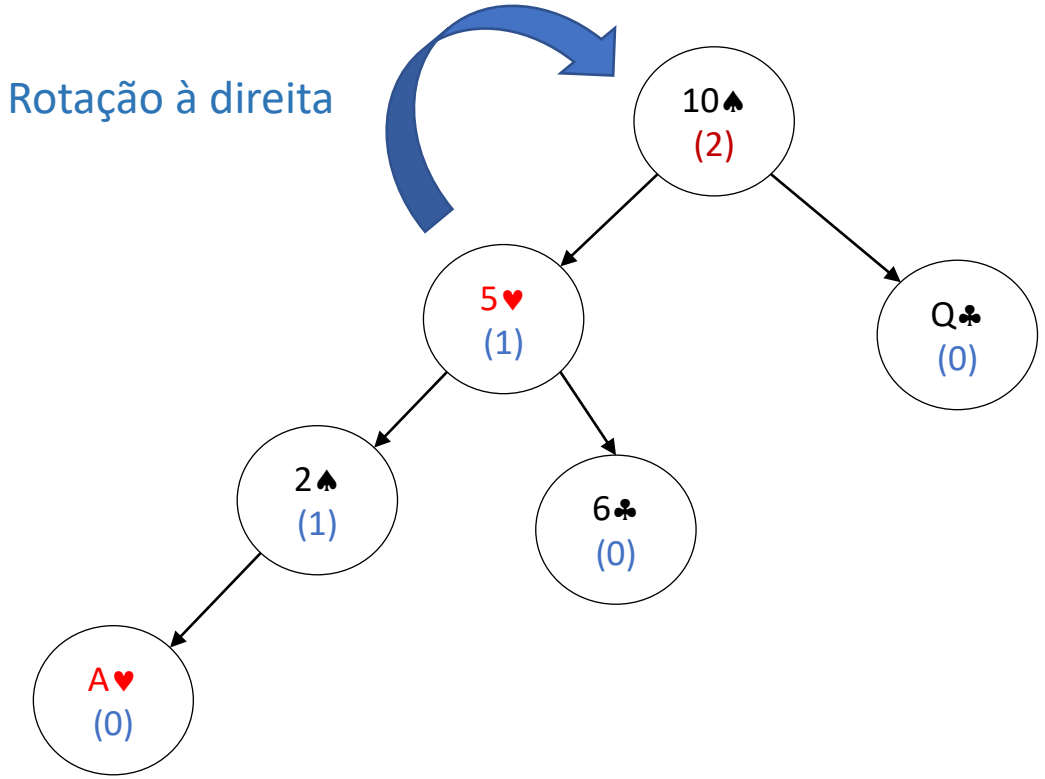
Removendo K♦ da árvore AVL



Removendo K♦ da árvore AVL



Removendo K♦ da árvore AVL



Removendo K♦ da árvore AVL