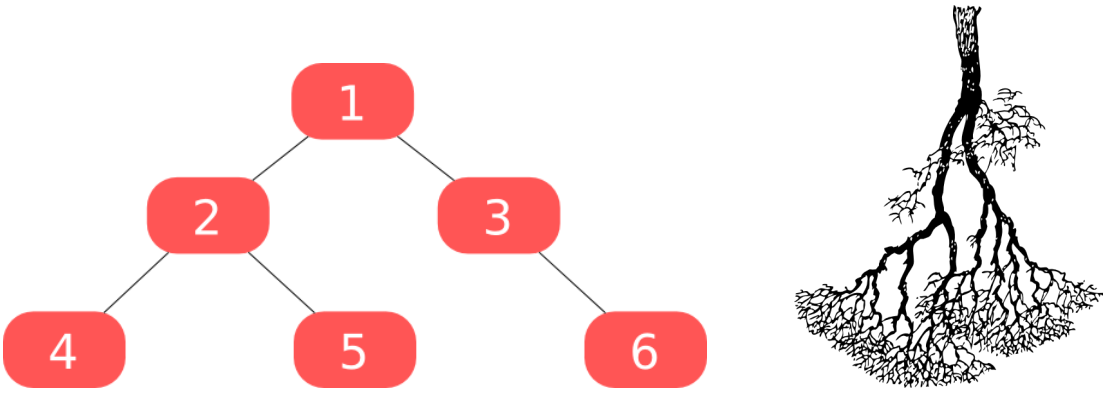
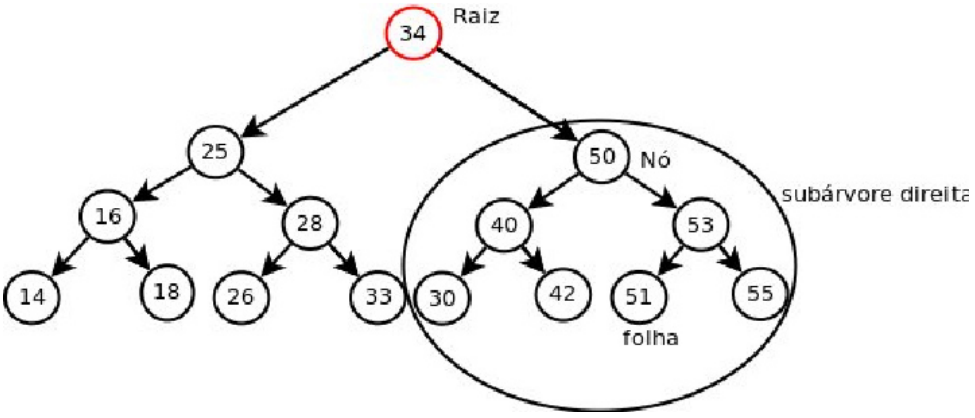
Estrutura de dados - Árvores



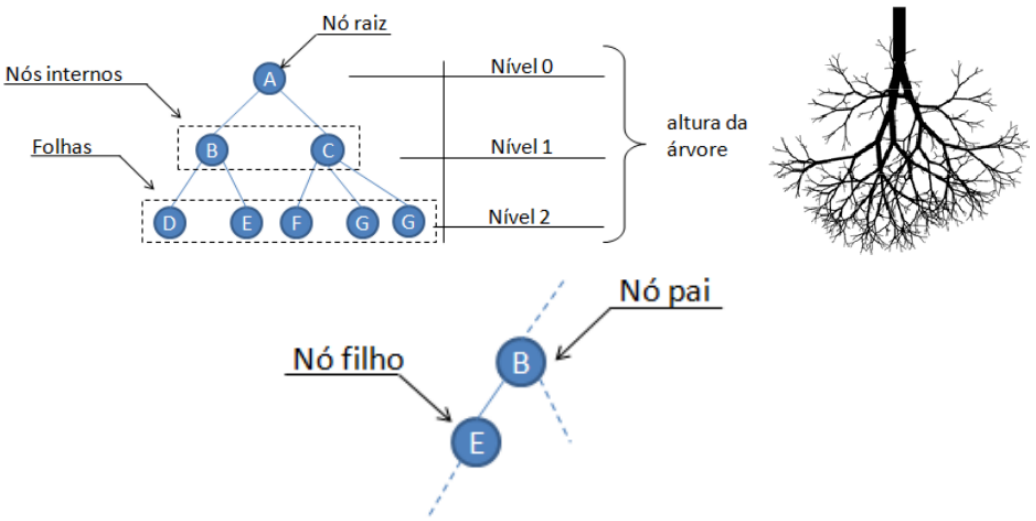
1 - Representação por imagens

1. Árvores
    - Um conjunto de nós.
    - Formada por um nó principal (raiz).
    - Abaixo de cada nó há as subárvores.
    - **Grau de uma árvore:** número de subárvores em cada nó.
    - Os nós de grau 0 são chamados de **folhas** ou nós externos**.** Os demais são chamados de **nós internos,** com exceção da raiz.
    - **Descendentes:** nós abaixos de um determinado nó.
    - **Nível:** O nível do nó raiz é 0, dos seus descendentes, ou filhos, têm nível 1, e assim sucessivamente.
    - **Altura:** é o comprimento do **caminho mais longo** entre ele e uma folha. Vale notar que a árvore é percorrida da **raiz às folhas**.
    - **Profundidade:** é a distância percorrida da raiz a esse nó.



2 - Estrutura de uma árvore [Nesse exemplo todo nó tem grau 2, exceto os da base, que têm grau 0].

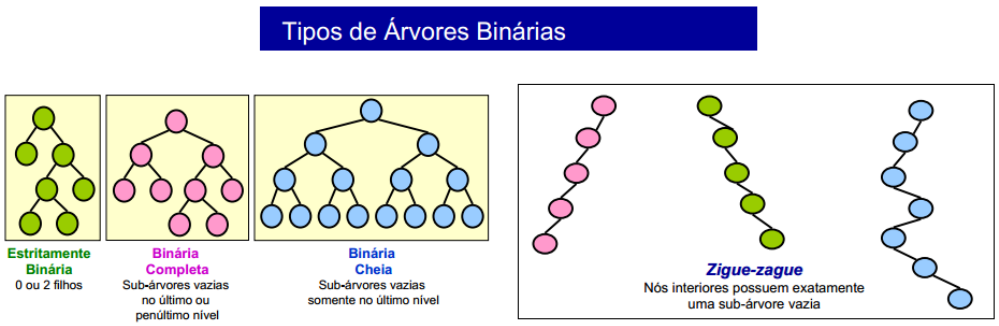
```c
#include <stdio.h>
#include <stdlib.h>
#define true 1
#define false -1

typedef int bool;
typedef int TIPOCHAVE;

typedef struct aux {
	TIPOCHAVE chave;
		/* Dados armazenados vão aqui */
	struct aux *esq, *dir;
} NO;

typedef NO * PONT;
```



3 - Nomenclaturas

2. Árvores Binárias

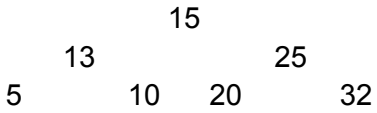
   É uma árvore em que os nós têm 0, 1 ou 2 descendentes.



4 - Tipos de árvores binárias

2.1. Árvores binárias de pesquisas

É uma árvore binária em que, a cada nó, todos os registros com chave **menores** que a deste nó estão na subárvore da **esquerda**, enquanto que os registros com chaves **maiores** estão na subárvore da **direita**.



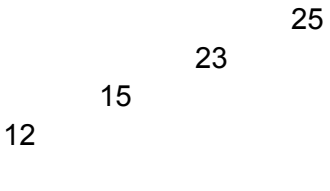
A ordem de inserção determina a forma da árvore. * Balanceamento *

- Exemplo 01: Resulta em árvore

23, 12, 25, 8, 15

- Exemplo 02: Se tornaria uma lista ligada, nesse caso a busca binária **não é eficiente** pois se torna uma busca linear que utiliza mais memória, pois as subárvoras a direita são NULL. Por isso é necessário o **balanceamento**.

25, 23, 15, 12, 8



Em suma, se os valores [chaves] forem **inseridos de maneira sequencial ordenada**, caíra no exemplo 2, e não será eficiente.

OBS: Se os elementos que compõem a árvore forem obtidos aleatoriamente, espera-se um desempenho apenas 39% pior do que a árvore completamente balanceada, ou seja, a árvore em que as folhas aparecem no mesmo nível ou, no máximo, em dois níveis adjacentes.

Fontes:
Playlist do YouTube - UNIVESP - Engenharia de Computação - Estrutura de Dados - 09º