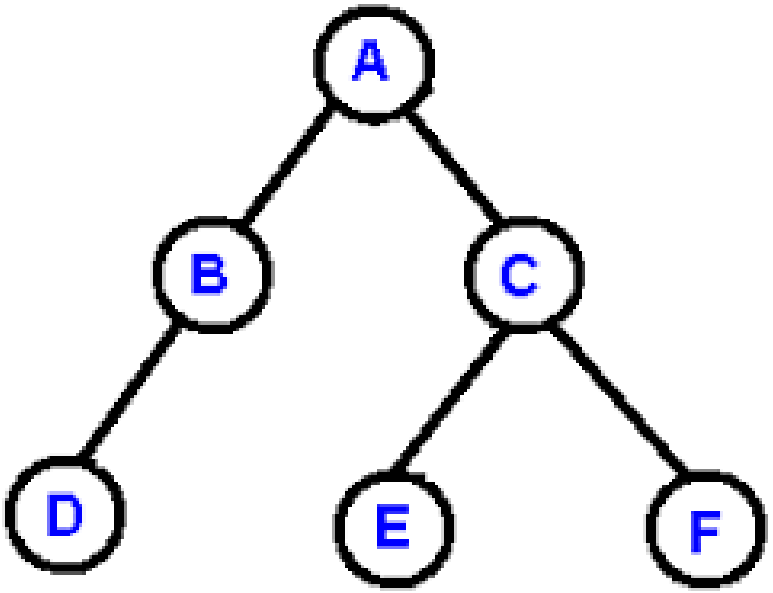
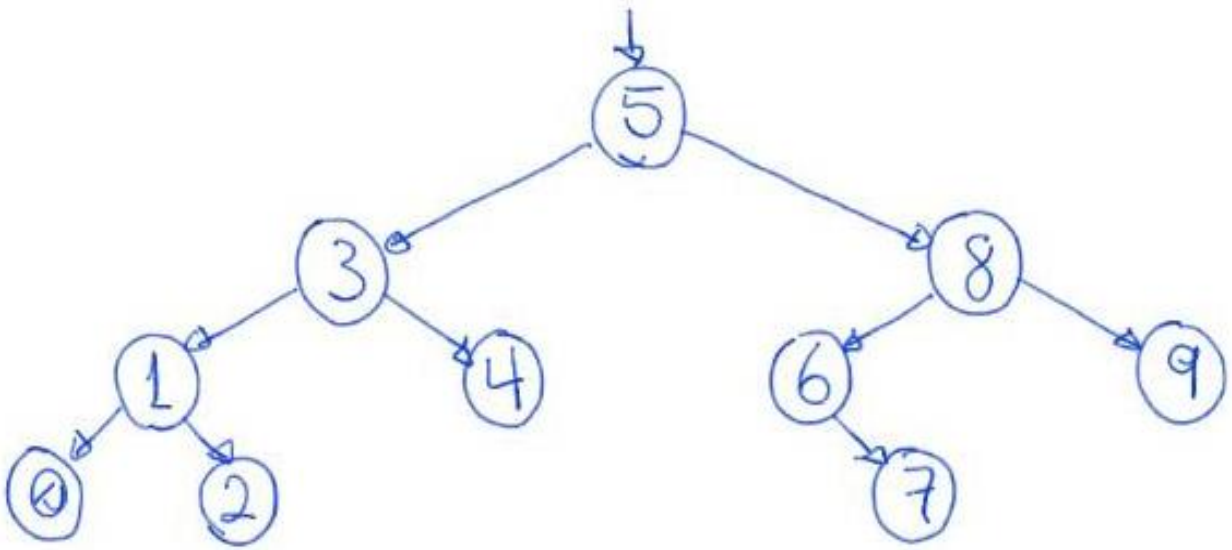
Árvore Binária



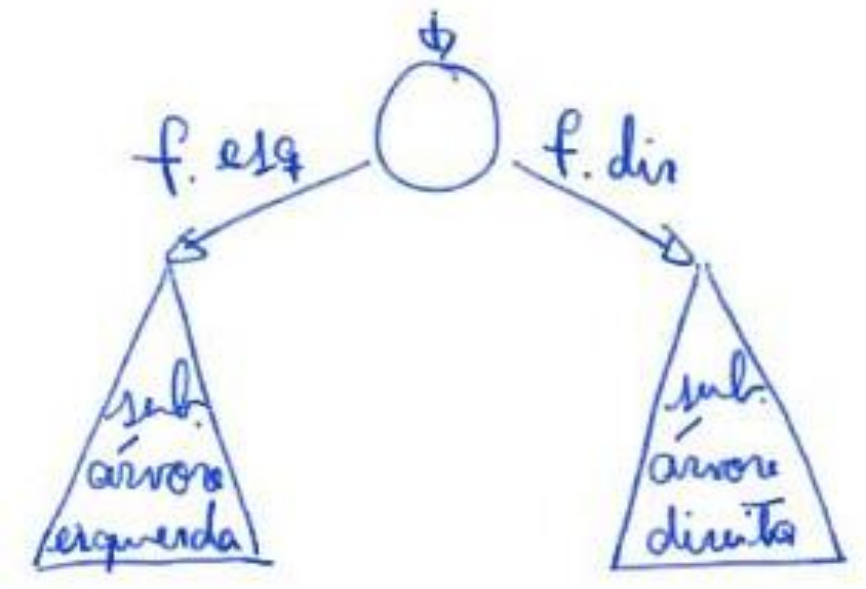
Árvore Binária de Busca



sem repetição

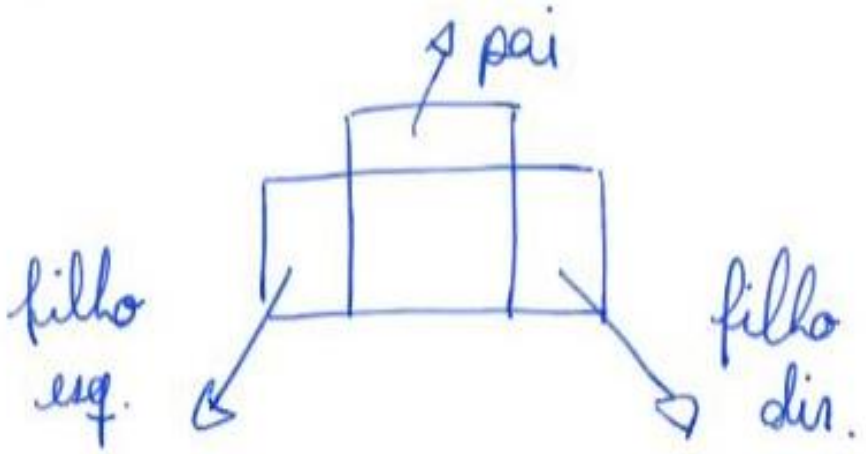
- info(esquerda) < info(pai)
- info(direita) > Info(pai)

Árvore Binária



- toda árvore binária
  - é um elemento com uma subárvore esquerda e uma subárvore direita
- Adicionamos à propriedade recursiva que
  1. cada elemento de uma árvore tem no máximo um pai, sendo que o único elemento sem pai é a raiz.
  a. os filhos esquerdo e direito de cada elemento são distintos.

Nó da árvore binária

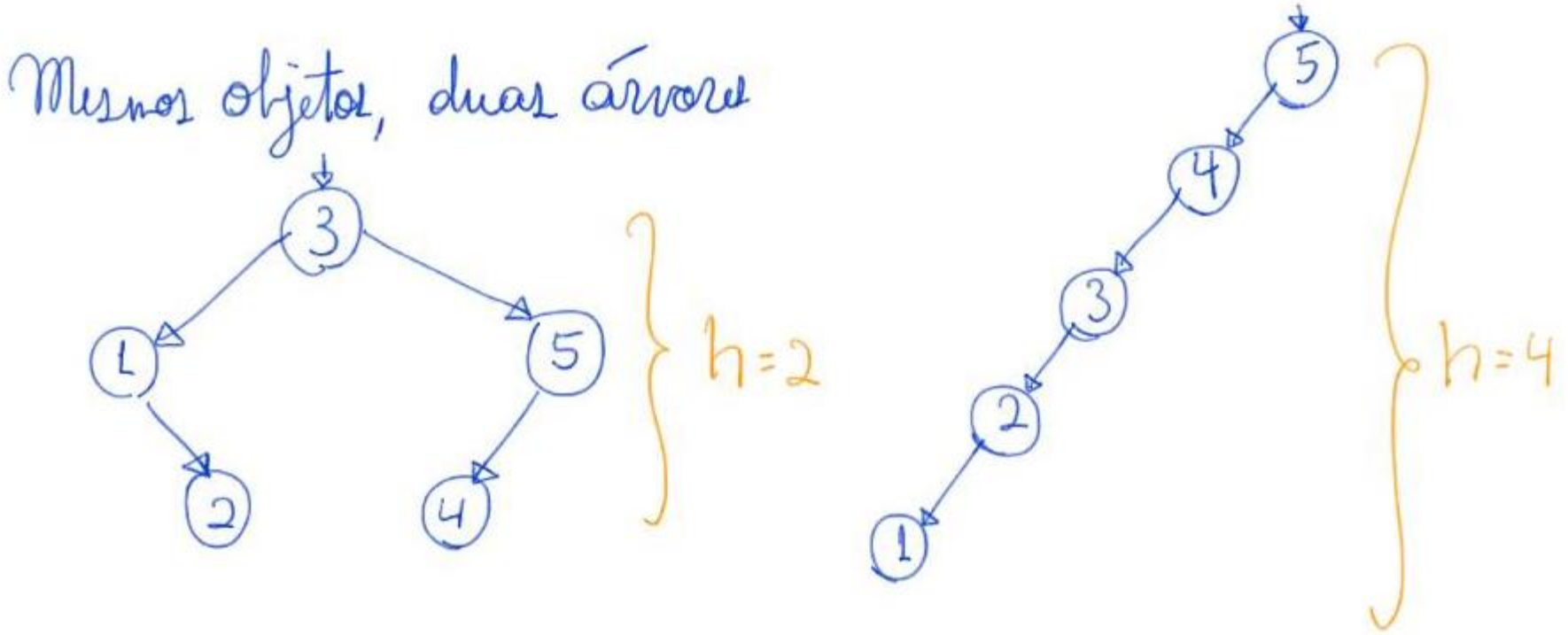



```c
typedef int Cont;

typedef struct noh {
    Cont conteudo;
    struct noh *pai; // opcional
    struct noh *esq;
    struct noh *dir;
} Noh;
```

Definições

- Definimos a **subárvore** de um nó x, como sendo x e seu conjunto de nós descendentes,
  - Também podemos dizer que trata-se da árvore enraizada em x.
- Chamamos de **folhas** os nós da árvore que não tem filhos,
- Definimos a **altura** de um nó x como sendo o comprimento do maior caminho de x até uma folha de sua subárvore,



Altura da árvore binária

- Qual a estrutura/formato da árvore binária com maior altura?
- E da árvore com menor altura?

No pior caso,

- a árvore terá altura n - 1.

No melhor caso,

- a altura será ~= lg n, caso seja completa ou quase completa, caso em que todos os níveis estão cheios, exceto talvez o último.

Cálculo da altura

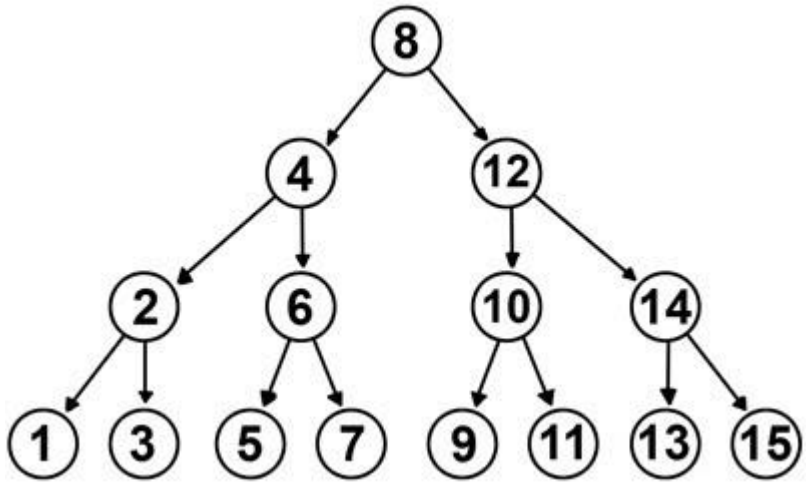
Altura

- se árvore corrente não for vazia
  - obtenha recursivamente a altura da subárvore esquerda,
  - obtenha recursivamente a altura da subárvore direita,
  - devolva 1 mais a altura da maior subárvore.

```
int altura(Arvore r)
{
    int hesq, hdir;
    // atenção para o valor devolvido no
caso base
    if (r == NULL)
        return -1;
    hesq = altura(r->esq);
    hdir = altura(r->dir);
    if (hesq > hdir)
        return hesq + 1;
    return hdir + 1;
}
```

Árvore Binária Balanceada

- Uma árvore binária é balanceada
  - se as subárvores esquerda e direita de cada nó
    - tiverem aproximadamente a mesma altura.
- Neste caso, a altura da árvore é da ordem de lg n, i.e., O(log n).



Percursos para percorrer

```
void inOrdem(Arvore r)
{
    if (r != NULL)
    {
        inOrdem(r->esq);
        printf("(%d) ", r->conteudo);
        inOrdem(r->dir);
    }
}
```