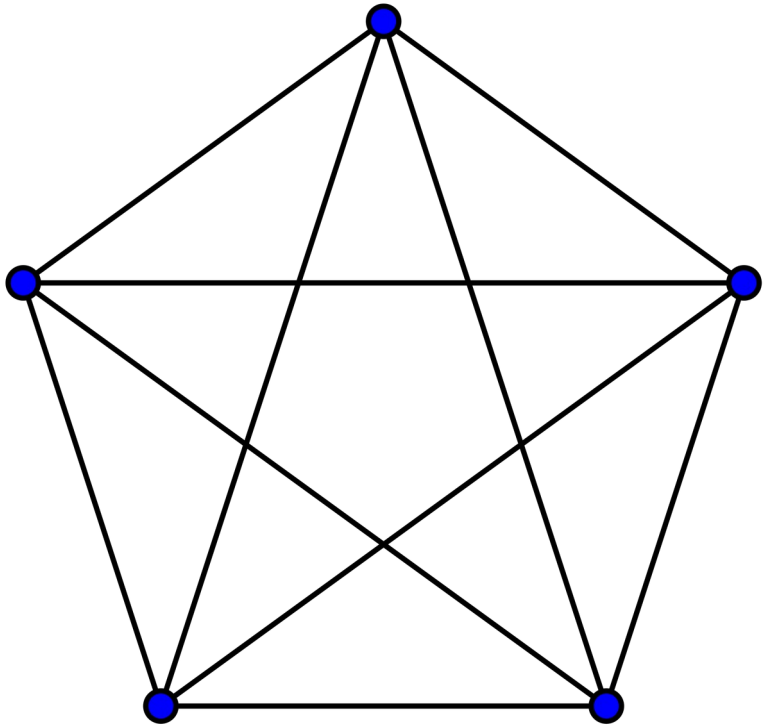
Árvores

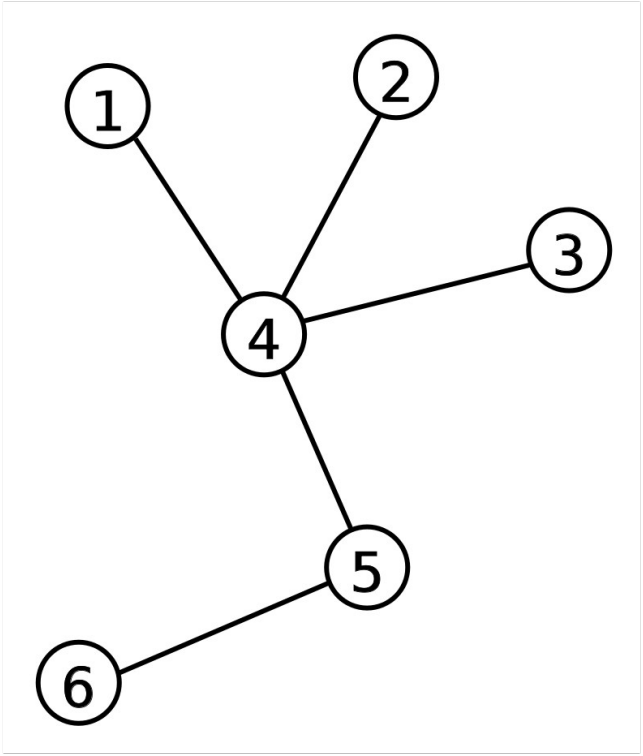
Recaptulando Grafos

- Estrutura que relaciona nós entre si
- Representado visualmente por circulos conectados
- Muito usado
  - Porém potencialmente muito complexo



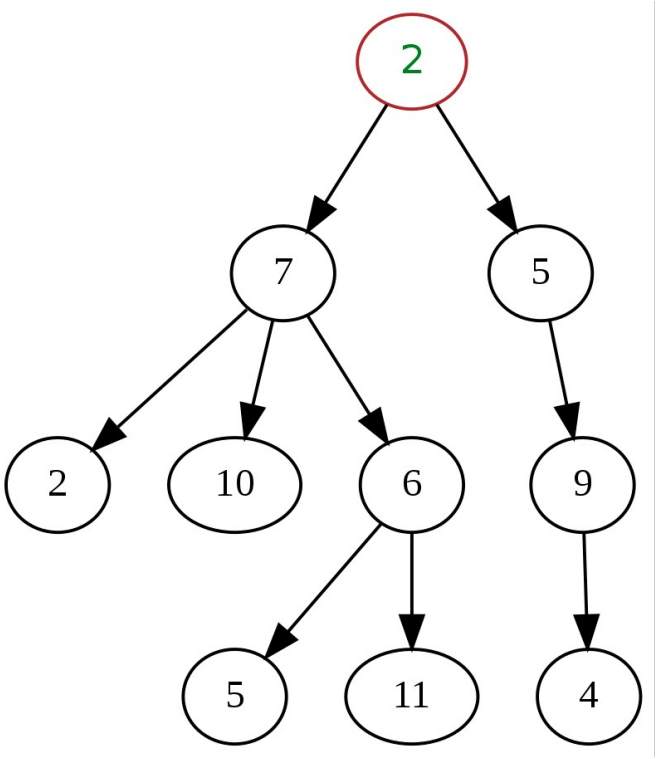
Árvores

- São grafos que são:
  - Acíclicos
  - Simples
  - Conectados
- Geralmente têm “raíz”
- Mais fáceis de manipular



Porque Árvores?

- São hierárquicas
- Traversia eficiente
- Usadas em uma diversidade de aplicações
- Caem muito em ED

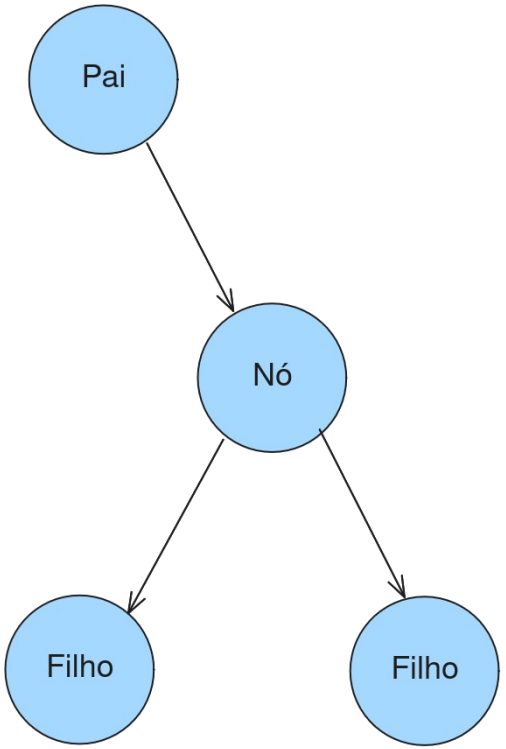


O que se modela com Árvores?

- Estruturas intrisicamente hierárquicas
  - Sistemas de arquivos
  - Cadeia de comando
  - Árvores de decisão
- Estruturas Ordenadas
  - Listas de nomes de usuários
- Estruturas Compactas
  - Dicionário com contagem de frequências

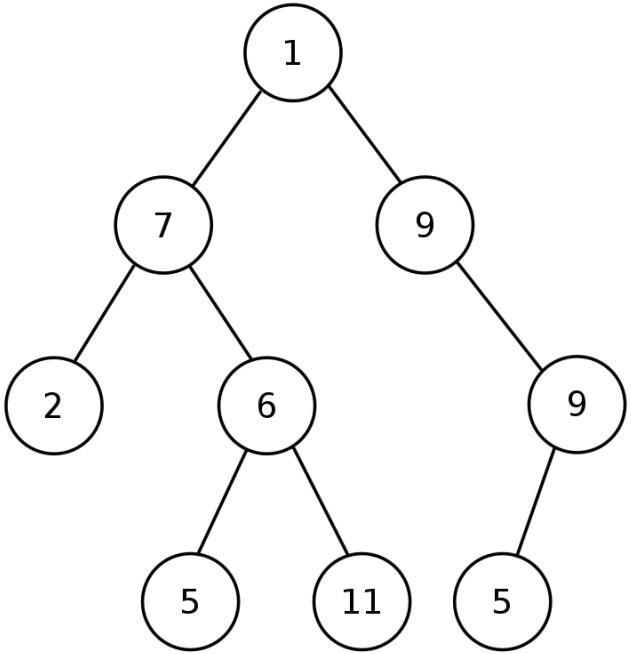
Nomenclatura

- Os vizinhos abaixo de um nó são chamados “filhos”
- Os vizinhos acima de um nó são chamados “pais”
- O primeiro nó da árvore é chamado de “raíz”



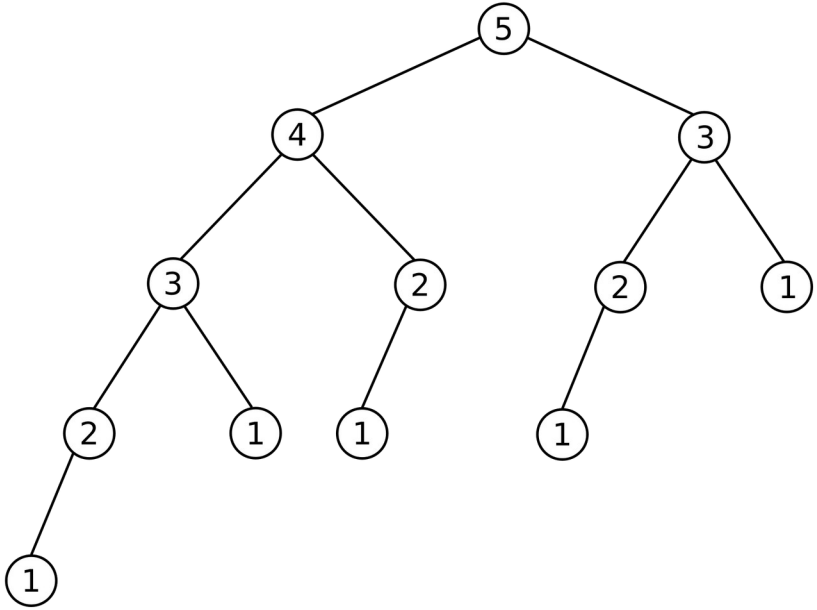
Algumas Propriedades: N-árias

- Nós podem ter números variáveis de filhos
- Mas se tiverem um máximo fixo N, são chamadas N-árias
- As árvore N-árias mais importantes são as binárias
  - No máximo 2 filhos



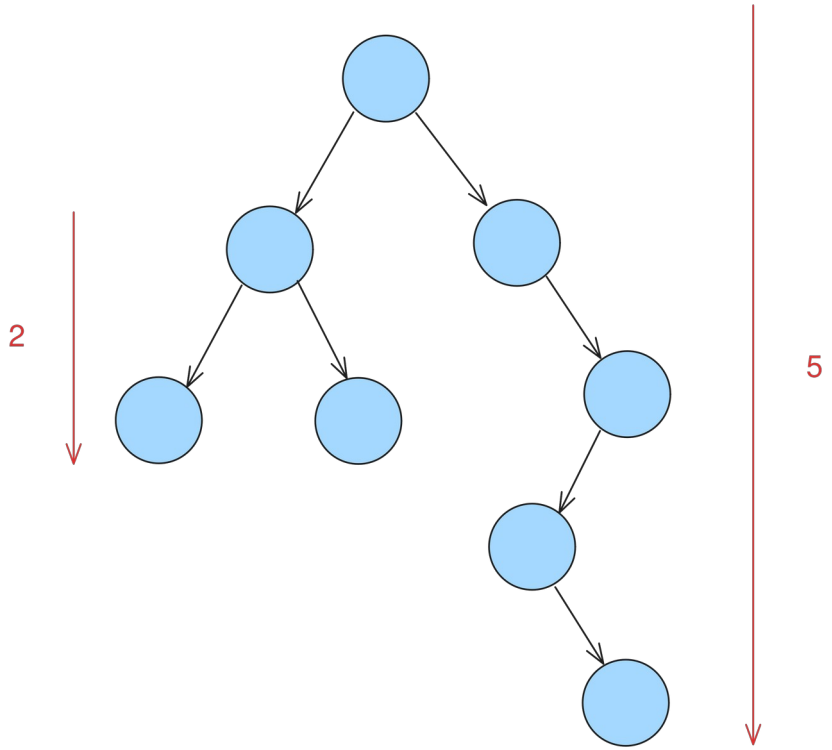
Algumas Propriedades: Recursividade

- Árvores são inerentemente recursivas:
  - Toda sub-árvore é uma árvore válida
- Geralmente são usadas para entender recursividade
- Ao lado: Árvore de Fibonacci



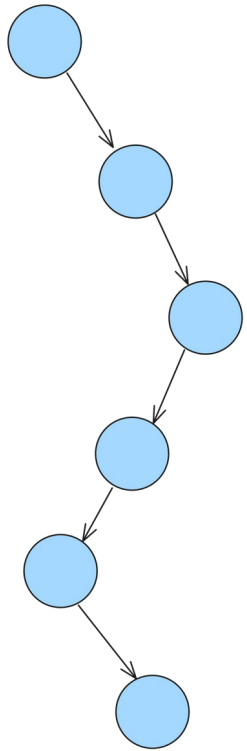
Propriedades: Altura

- A altura de uma árvore é a maior distância da raiz para uma folha
  - Folhas são os nós sem filhos
- Altura também pode ser definida para sub-árvores



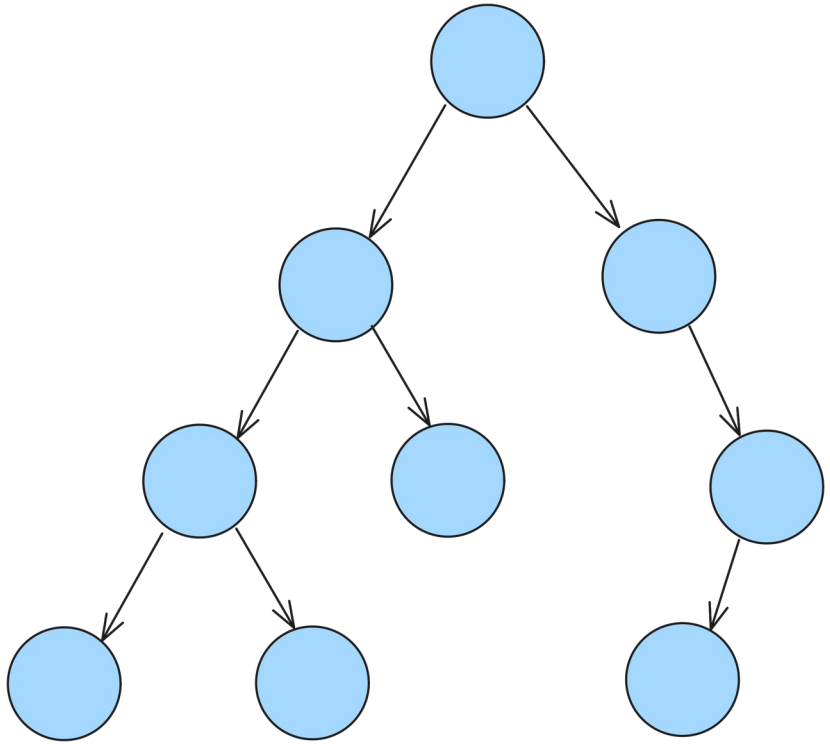
Propriedade: Degenerada

- Todo nó tem apenas 1 filho no máximo
  - Efetivamente é só uma sequência
- Evitar árvores degeneradas é foco de muitas EDs



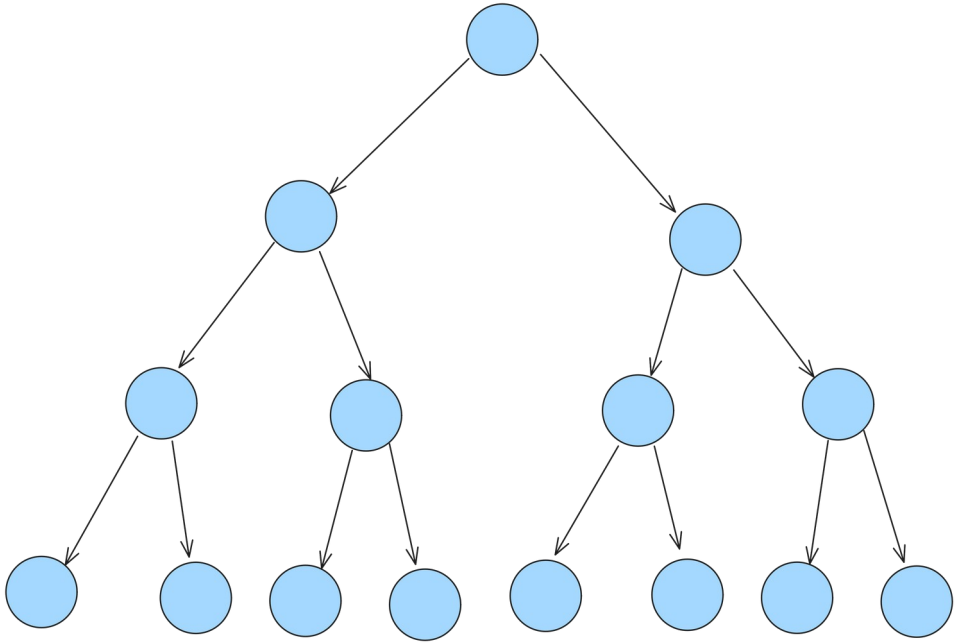
Propriedade: Completude

- Uma árvore n-ária é completa se
  - Todos os nós (exceto os ultimos não folhas) tem n filhos
- Por exemplo uma árvore completa binária ao lado
  - Mas ela não é “Perfeita”



Árvore perfeita

- **Todos** não-folhas tem o máximo de filhos
- Raro de acontecer na realidade
- Mas é um bom limite superior



Sobre Árvores Perfeitas

- Uma árvore n-ária perfeita de altura h contém o máximo de nós possíveis para essa altura

- Como uma árvore divide n vezes por nível, teremos no máximo:
  - $1 + n + n^2 + n^3 + \ldots + n^h$ nós
  - Para uma árvore binária isso é
    - $1 + 2 + 4 + \ldots + 2^h = 2^{h+1}-1$

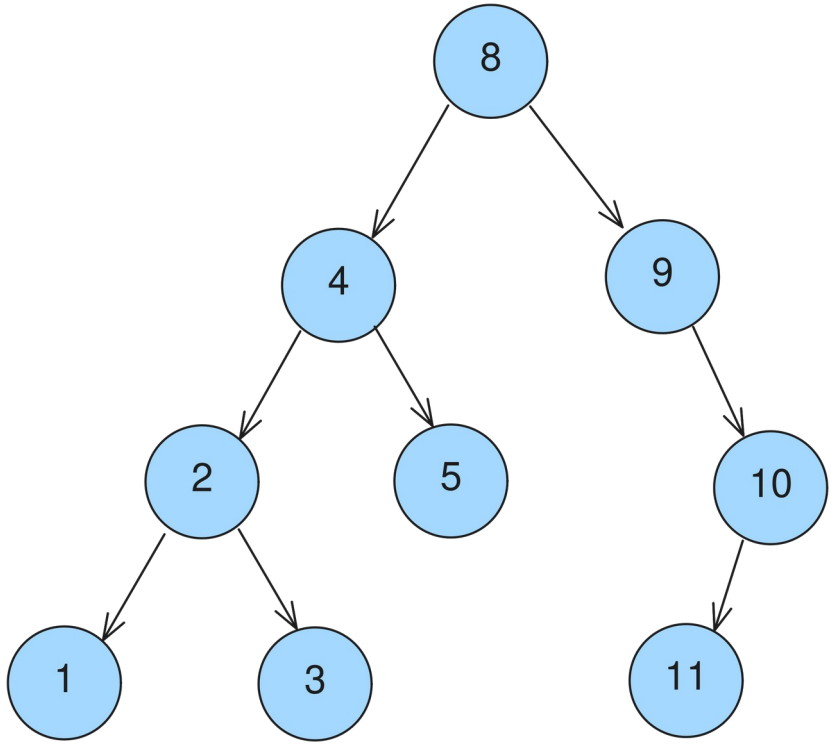
- Então árvores armazenam um número exponencial de nós proporcional à altura

Sobre Árvores Perfeitas

- Tomando uma árvore binária de altura h e n nós
- Se a sua complexidade do seu algoritmo depende apenas de h
  - Por exemplo, seu algoritmo tem f(h) = O(h)
- Então ela dependerá proporcinalmente de log(n)
  - Se f(h) = O(h), e $n < 2^{h+1}$
  - h = $O(\log_2(n))$, portanto f(n) = O(log(n))

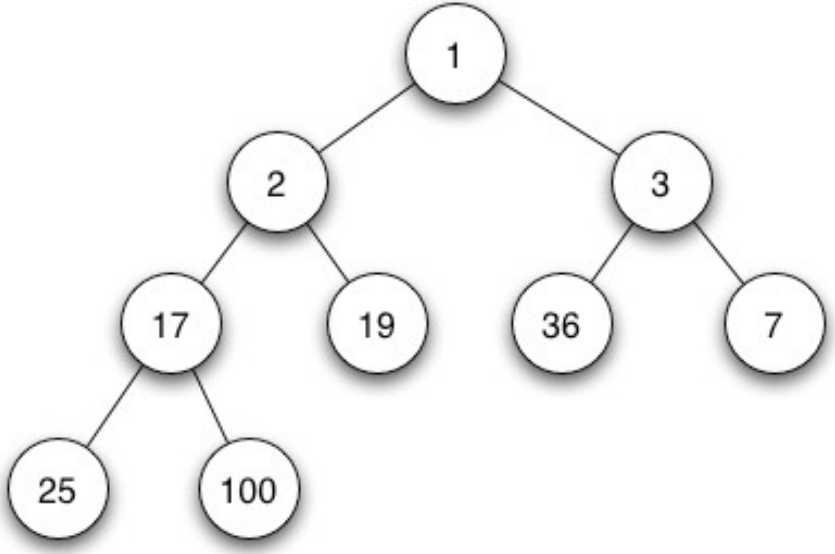
Exemplo: Árvore de Busca

- Nós à esquerda menores
- Nós à direita maiores
- Usada para facilitar busca binária
- Pode ficar degenerada
  - Aí que entra a “tal da AVL”



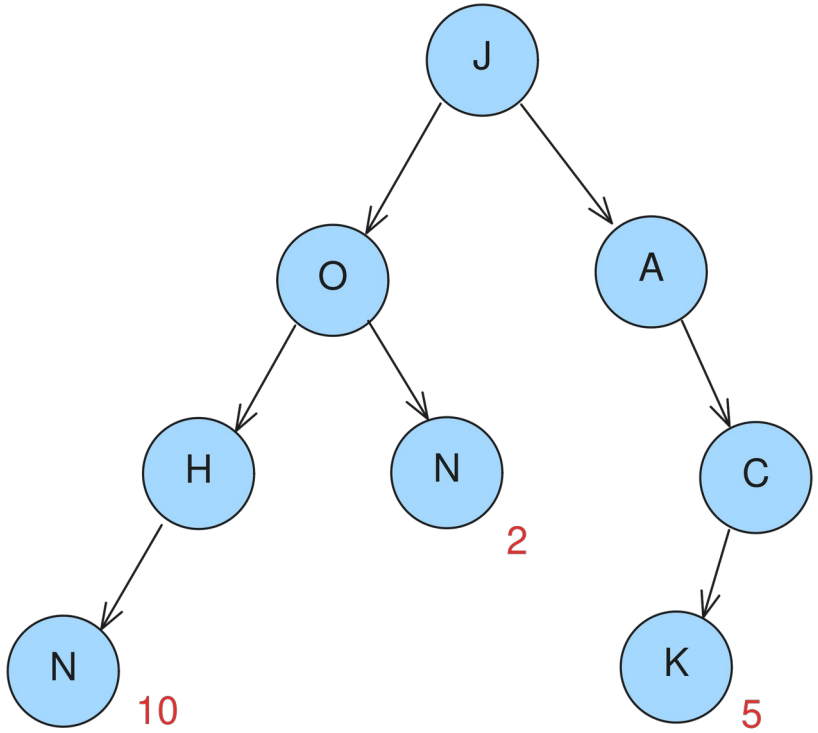
Exemplo: Heap Binário

- Ordenada de cima pra baixo
  - Filhos sempre menores que pais
- Usada pra filas de prioridades
  - Raíz é sempre menor número
  - Operações mantêm essa propriedade



Exemplo: Árvore de Prefixos

- Armazena strings de forma compacta
- Uma travessia com “contador” indica uma palavra encontrada
- Evita repetição de caracteres



Como se implementa uma Árvore?

- Em C, é mais fácil
  - Ponteiros são como arestas
- Sua estrutura base será o nó
  - Cada nó terá um ponteiro pro pai
  - Caso n-ária, n ponteiros pra filhos
  - Caso não, um array (ou vector)

```c
struct no {
    struct no* pai;
    struct no* filho_esquerdo;
    struct no* filho_direito;
};
```

```c
struct no {
    struct no* pai;
    struct no* filhos; //array
};
```

Fim

Agora só relatórios