As pessoas são como as árvores. Se você acertá-las com um machado, elas morrem!

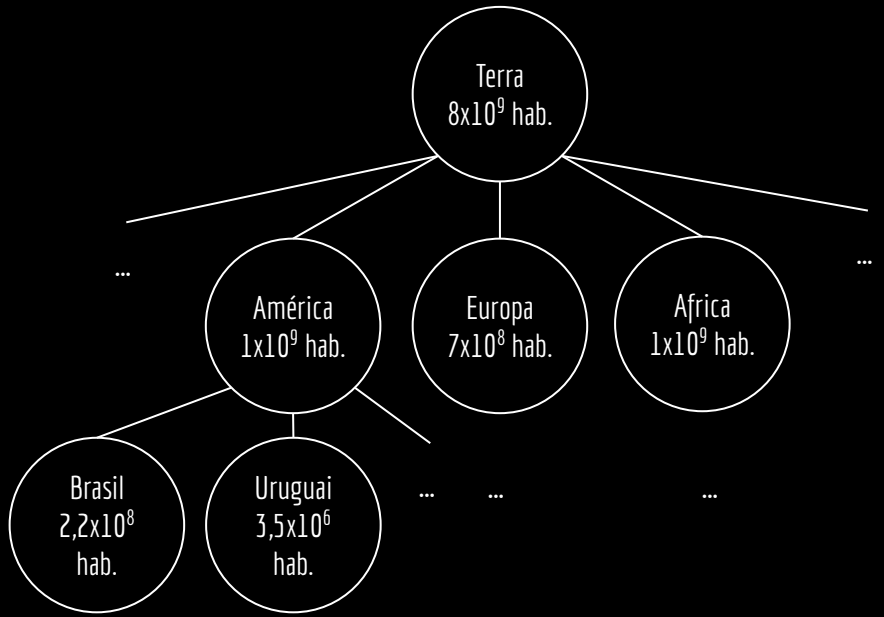
Árvores

Paulo Ricardo Lisboa de Almeida

Alguns conteúdos e exemplos foram baseados nas aulas do Professor Eduardo Almeida - www.inf.ufpr.br/eduardo

Árvores

Uma árvore é uma estrutura não linear (hierárquica) utilizada para armazenamento e busca eficiente de dados.

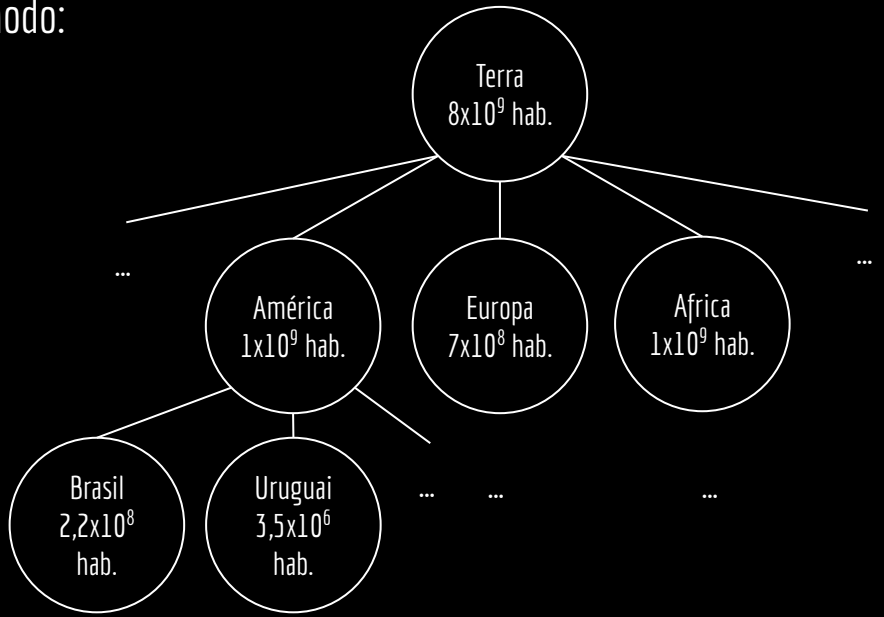


Árvores

Uma árvore é uma estrutura não linear (hierárquica) utilizada para armazenamento e busca eficiente de dados.

Possui **nodos, nós** ou **vértices**, onde cada nodo:

- Possui uma **chave**.
- Possui 0 ou mais filhos.

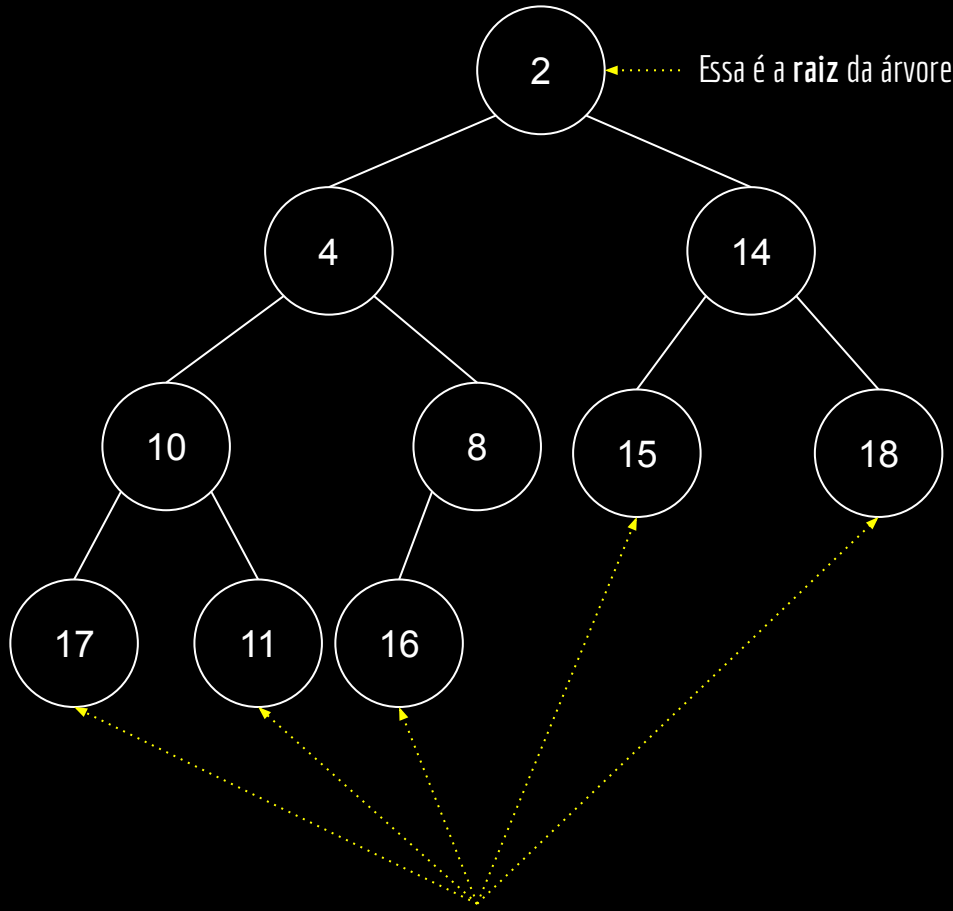


Árvore Binária

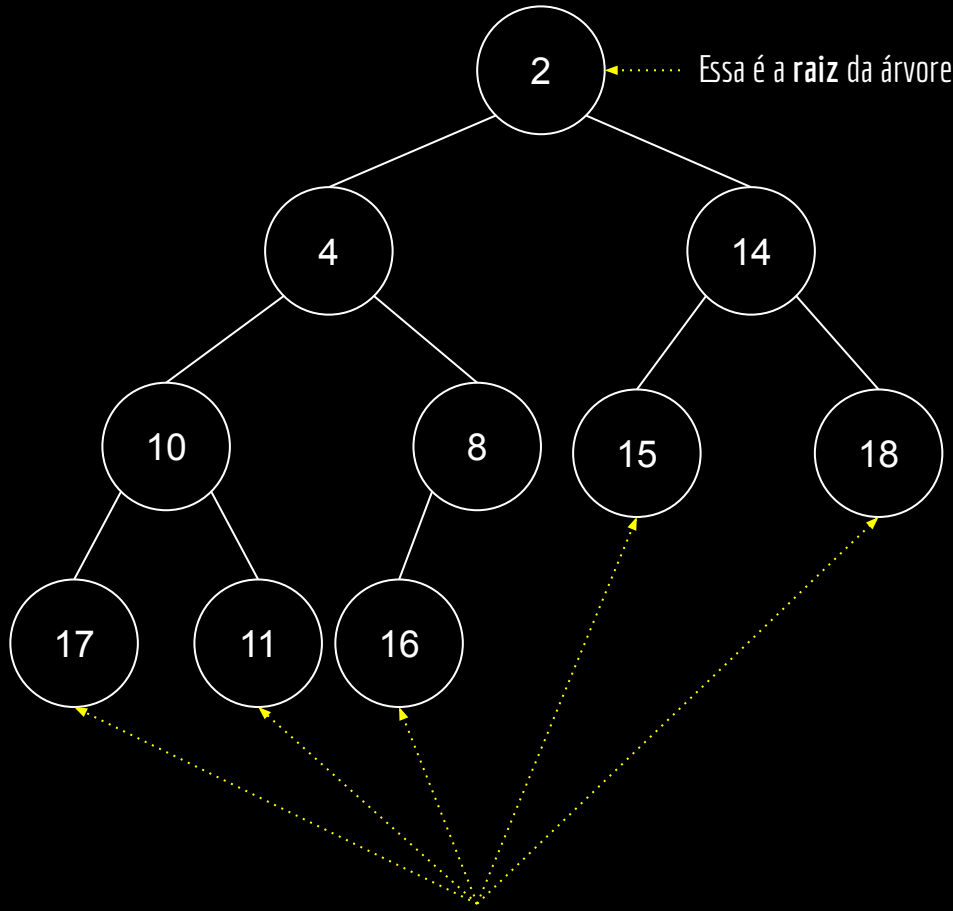
Vamos começar com árvores binárias, que são um subconjunto das árvores.

Em uma árvore binária, cada nodo ou nó possui **0, 1 ou 2 filhos**.

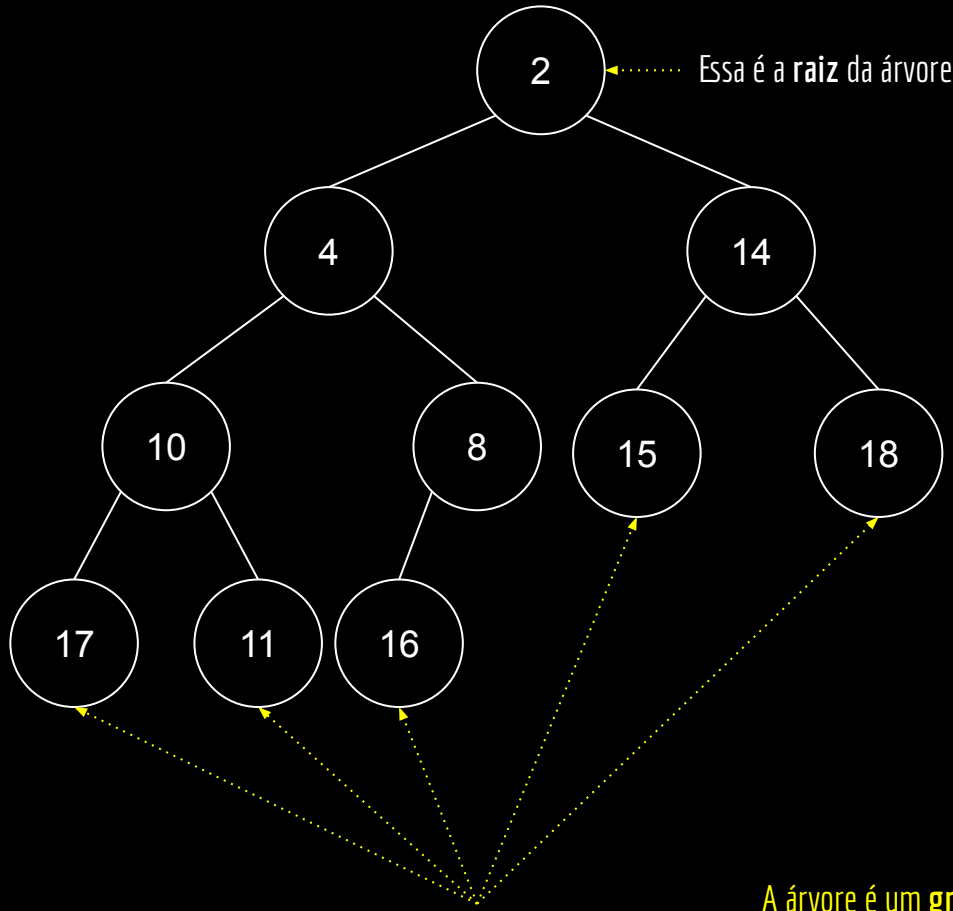
Exemplo


2
Essa é a **raiz** da árvore
4
14
10
8
15
18
17
11
16
Nós sem filhos são **folhas**.

Exemplo


2
Essa é a **raiz** da árvore
4
14
10
8
15
18
17
11
16
Nós sem filhos são **folhas**.

Exemplo



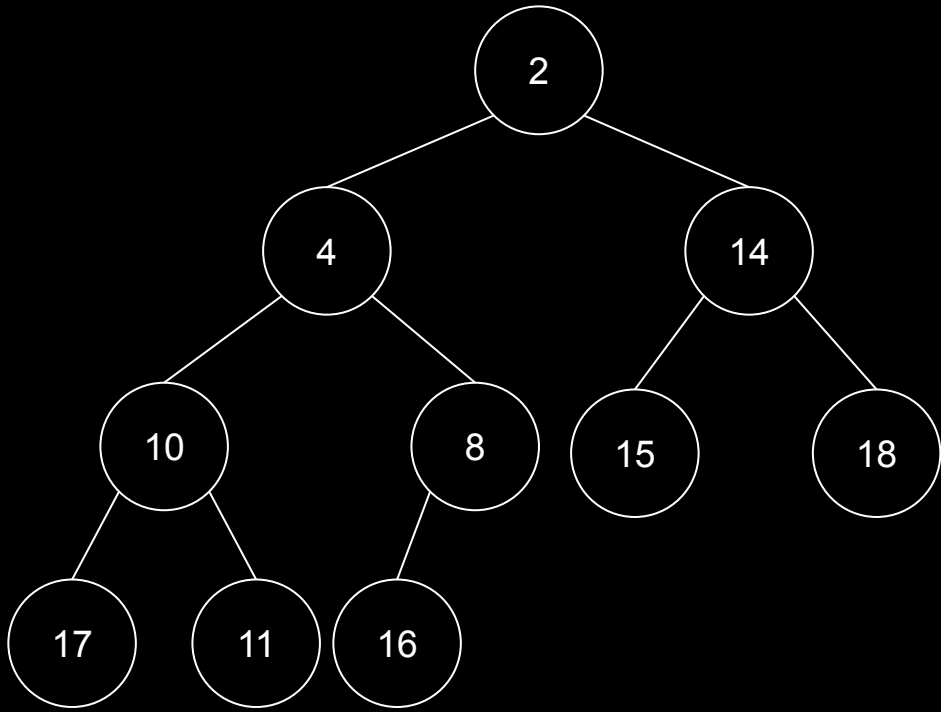
A árvore é um **grafo conexo e acíclico**, onde os **nodos** são os **vértices**, e as **ligações** são **arestas**.

Atenção

Vamos usar árvores binárias nos exemplos, mas as definições podem ser estendidas para outras árvores.

Representando

Como representar uma árvore binária em C?



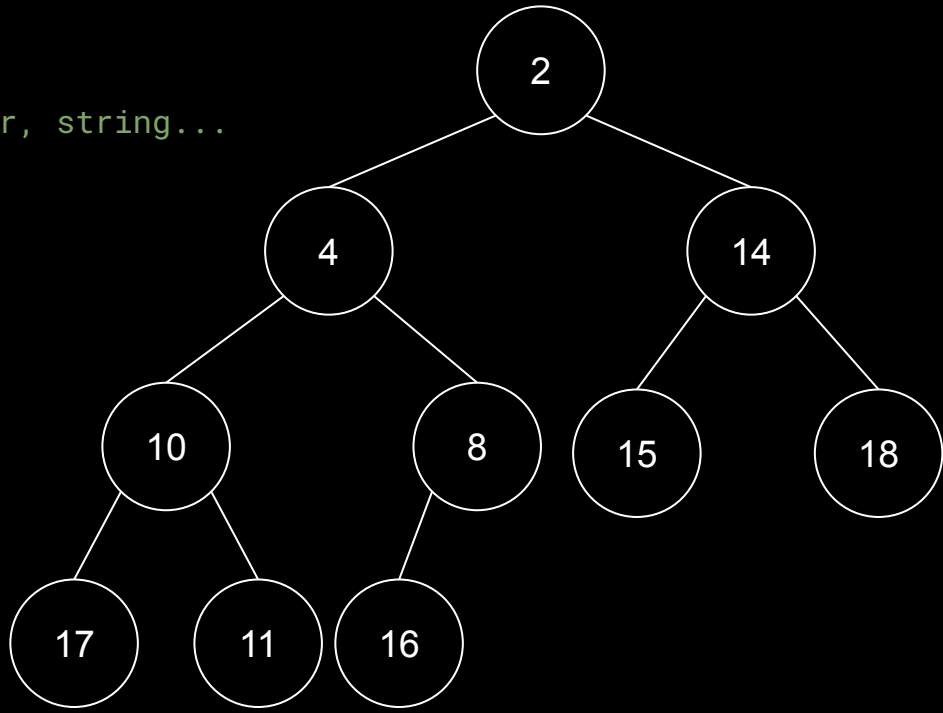
Representando

Como representar uma árvore binária em C?

```c
struct nodo{
   int chave;//poderia ser qualquer coisa: char, string...
   struct nodo* fe;
   struct nodo* fd;
};


int main(){
   struct nodo* raiz;
   //...

   return 0;
}
```



Definições

Uma Árvore T é um conjunto de nodos (ou nós/vértices).

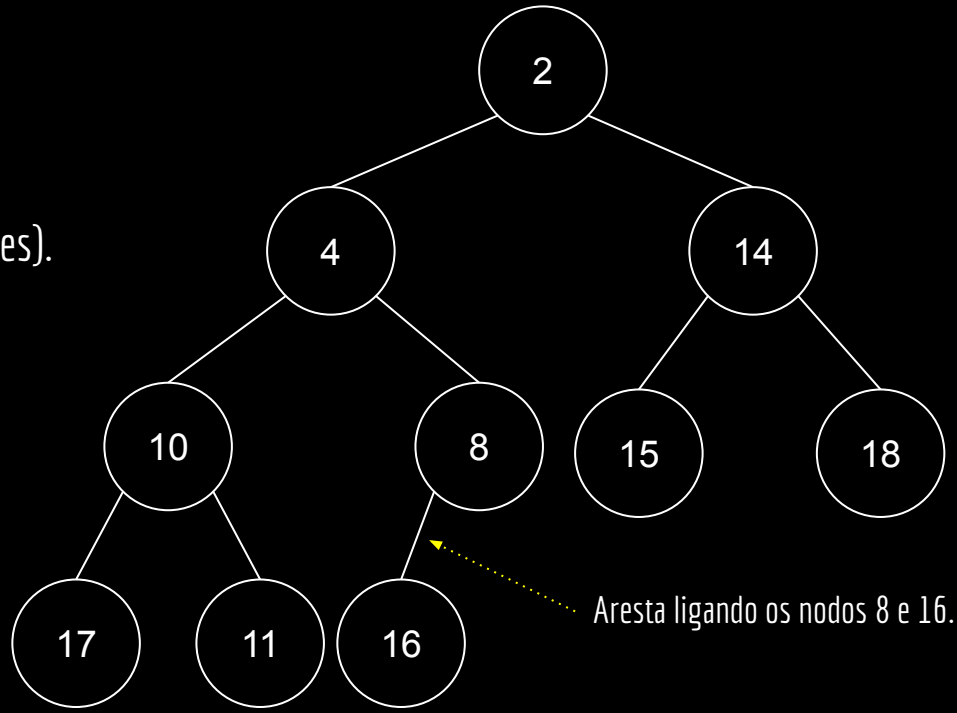
T = 0 é uma árvore vazia.

No exemplo T = {2, 4, 14, 10, 8, 15, 18, 17, 11, 16}.

Aresta é uma ligação entre dois nodos.

Representação de parênteses:

(2 (4 (10 (17)(11)) (8 (16))) (14 (15) (18)))



Definições

Nodo raiz é o único nodo sem pai.

Nodo pai.

Ex., nodo 4 é pai de 10.

Nodos filho.

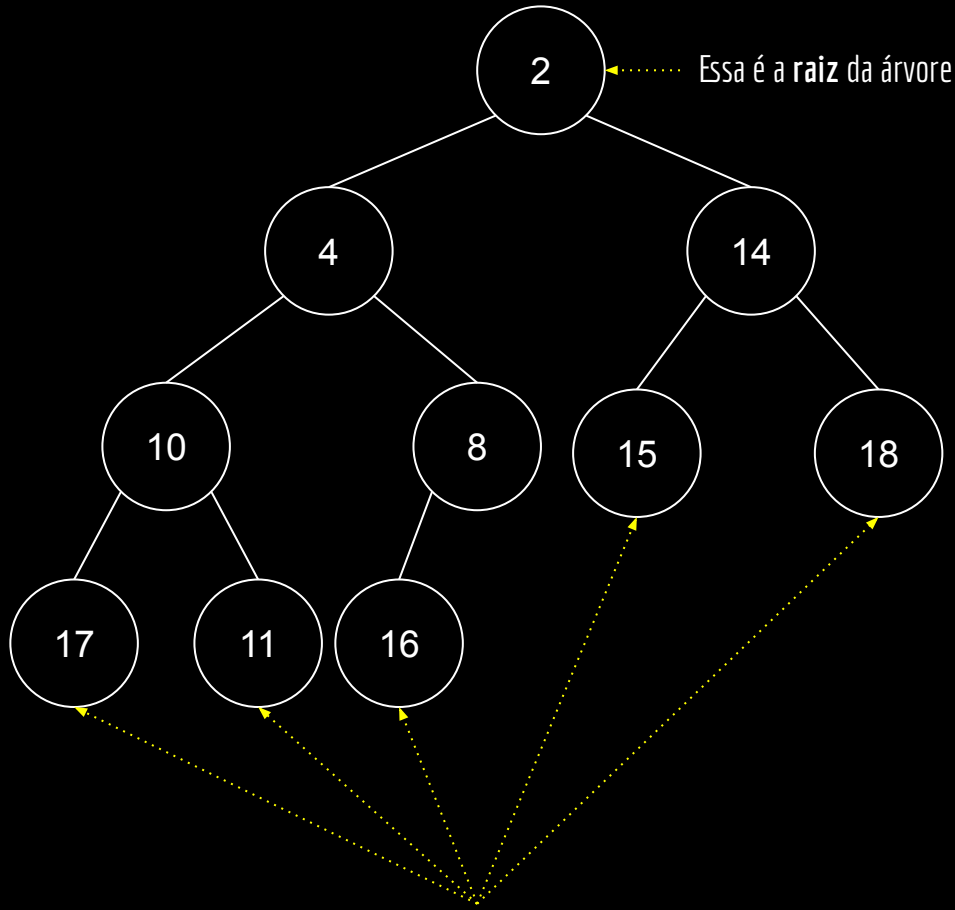
Ex., 4 e 14 são filhos do nodo 2.

Nodos externos ou folha são nodos sem filhos.

Ex. nodos 15, 18, 17, 11 e 16.

Nodos internos são nodos com filhos.

Ex.: nodos 2, 4, 14, 10 e 8.

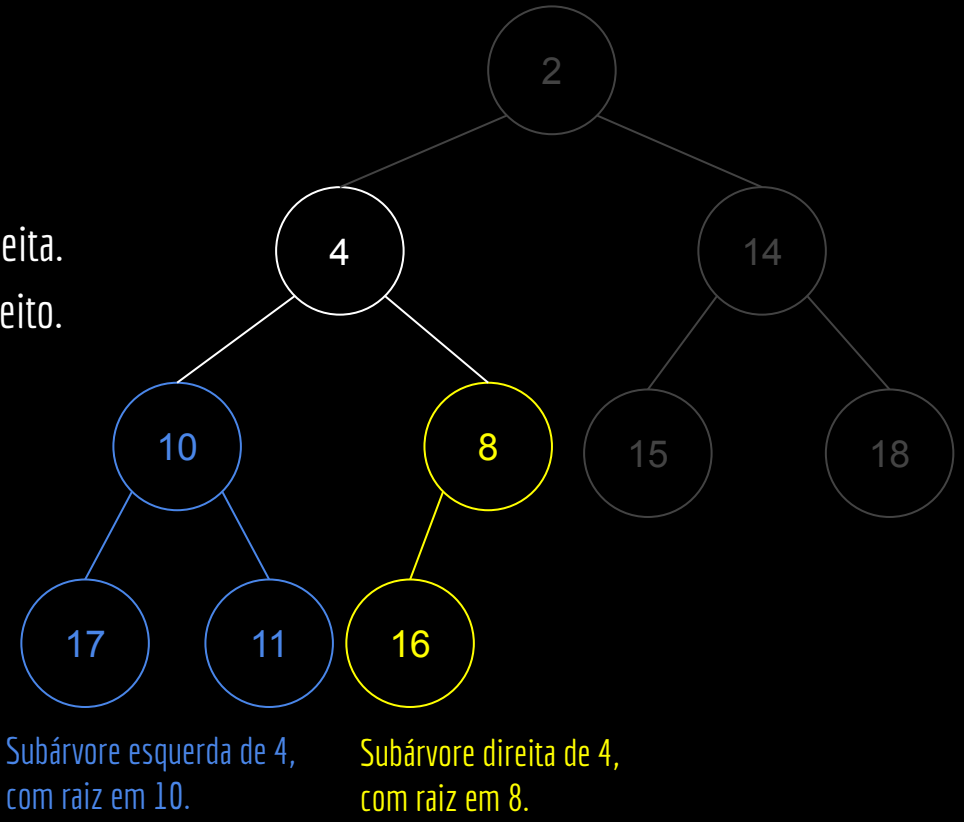


Definições

Um nodo pode possuir subárvores à esquerda e à direita.

Árvores que começam no filho esquerdo, e direito.

Cada subárvore possui sua raiz e subárvores.



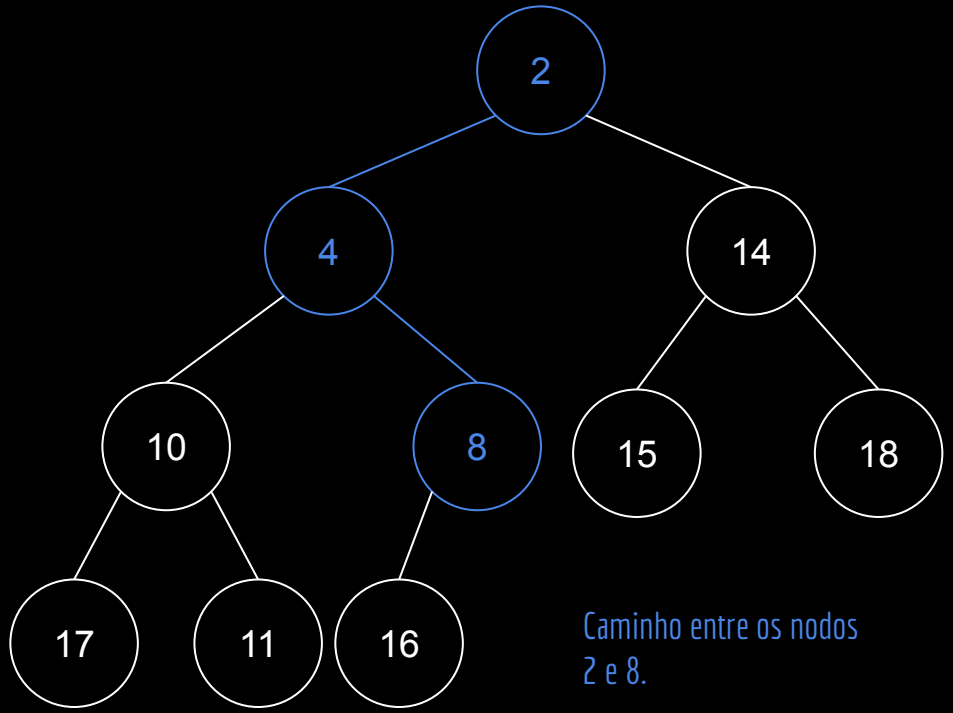
Caminho

Caminho: sequência de nodos relacionados.

Chegar de um nodo X a um nodo Y.

Existe somente 1 caminho entre um par de nodos.

No exemplo: C = {(2,4), (4,8)}.



Caminho entre os nodos 2 e 8.

Nível e Altura

Nível: Raiz = Nível 0.

Nível nodos = Nível Pai + 1.

Número de arestas entre a raiz e o nodo.

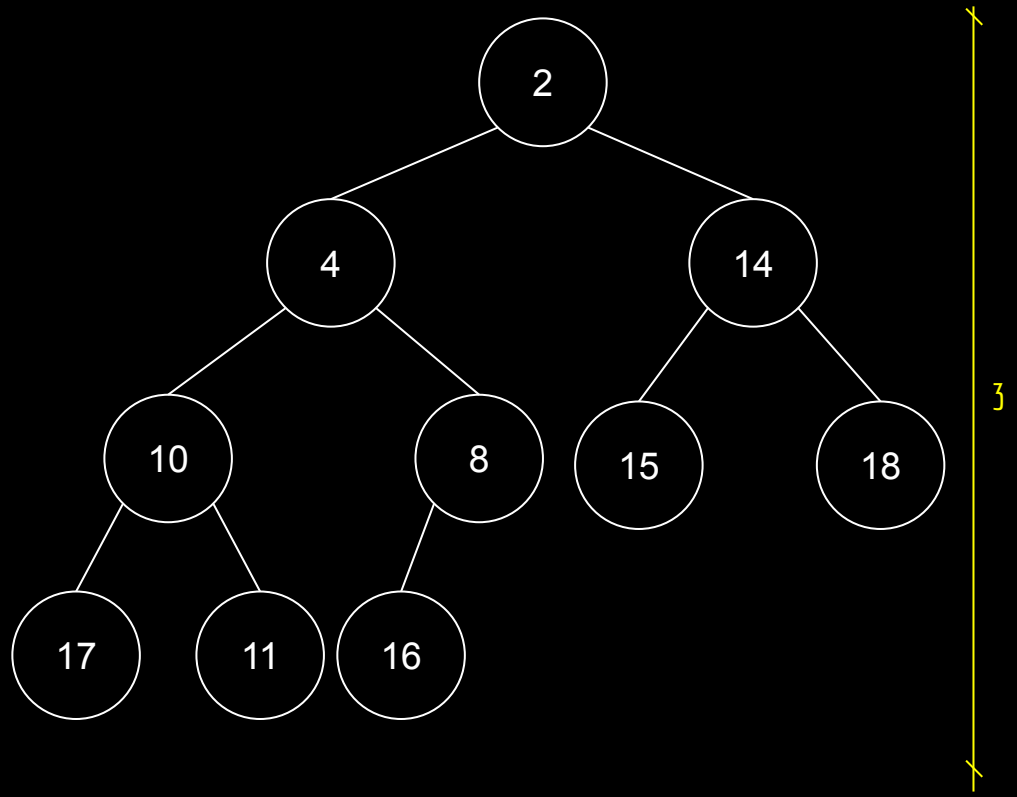
Altura: Nível do maior caminho que parte da Raiz

Folha mais profunda.

N-ária: número de filhos (2 filhos é binária).

Grau do nó: número de filhos do nó.

Grau da árvore = aridade = grau máximo.



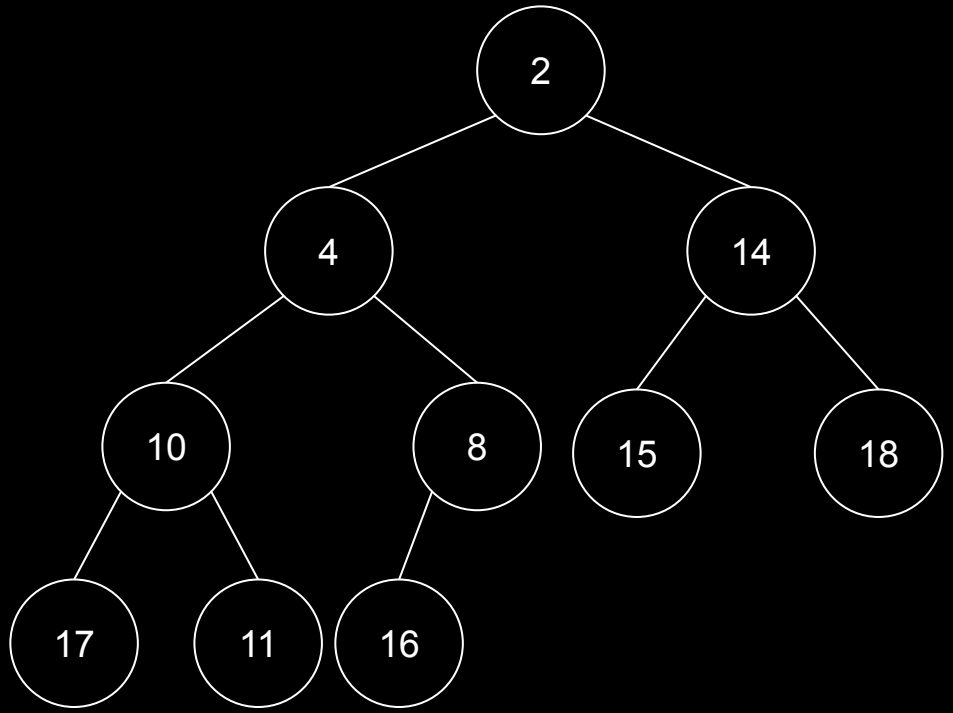
Calculando a altura

```
int calcularAltura(struct nodo* nodo){
    int altEsq, altDir;

    if (nodo == NULL)
        return -1;

    altEsq = calcularAltura(nodo->fe);
    altDir = calcularAltura(nodo->fd);

    if (altEsq > altDir)
        return altEsq+1;
    else
        return altDir+1;
}
```



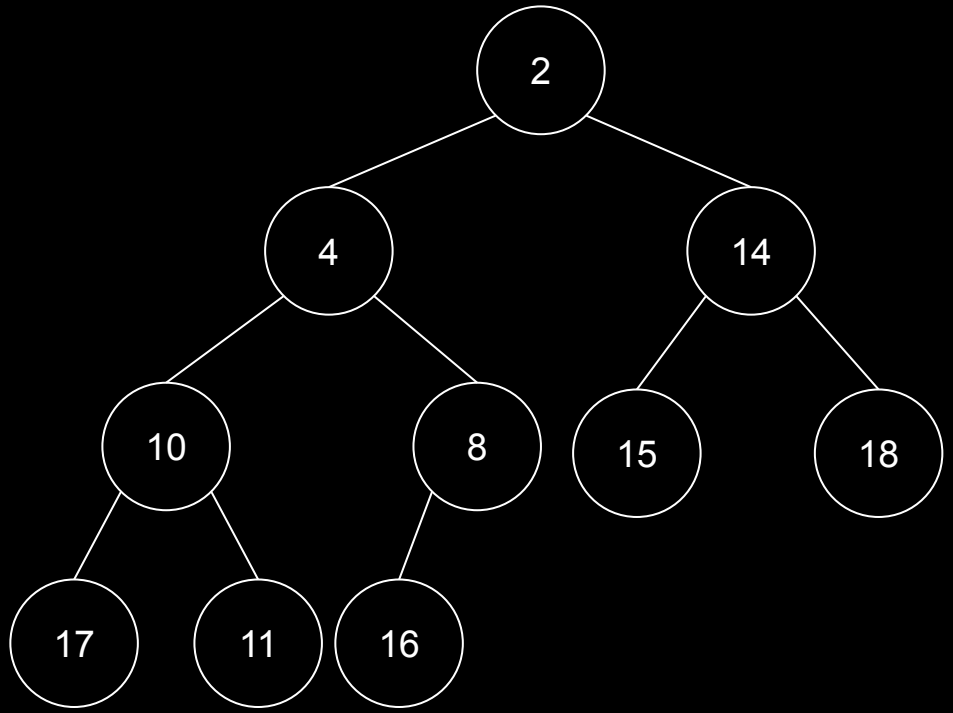
Travessia

3 formas:

- Pré-ordem;
- Em-ordem;
- Pós-ordem.

Pré-Ordem

- Visitar a raiz.
- Aplicar pré-ordem na subárvore esquerda.
- Aplicar pré-ordem na subárvore direita.

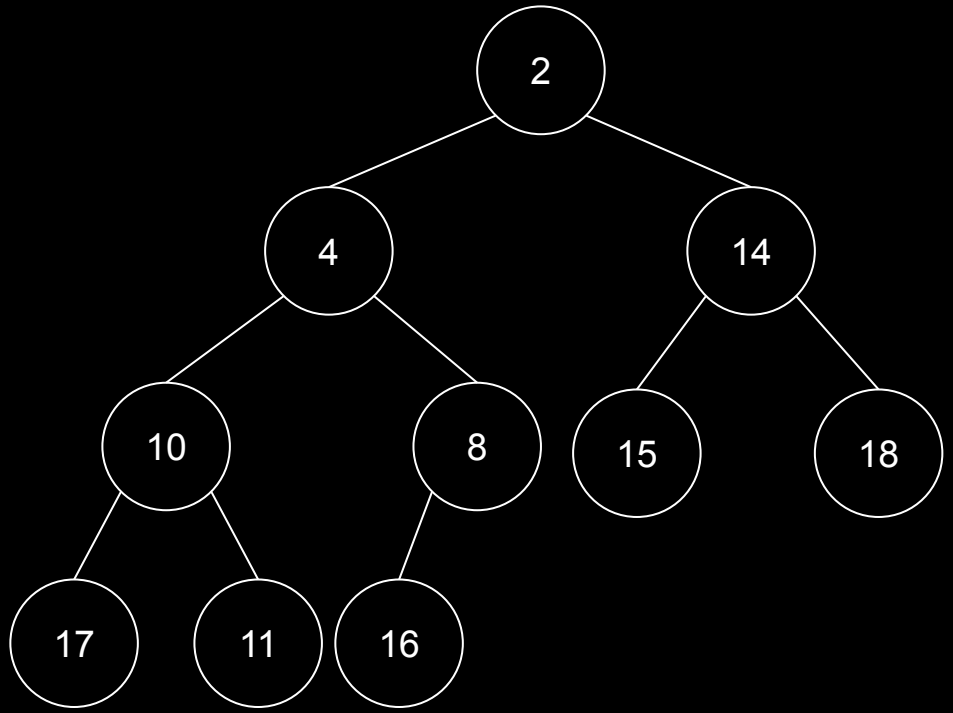


Visitar um nodo significa executar alguma ação no nodo.

Exemplo: imprimir.

Pré-Ordem

- Visitar a raiz.
- Aplicar pré-ordem na subárvore esquerda.
- Aplicar pré-ordem na subárvore direita.



Visitar um nodo significa executar alguma ação no nodo.

Exemplo: imprimir.

Visita em Pré-ordem: 2, 4, 10, 17, 11, 8, 16, 14, 15, 18.

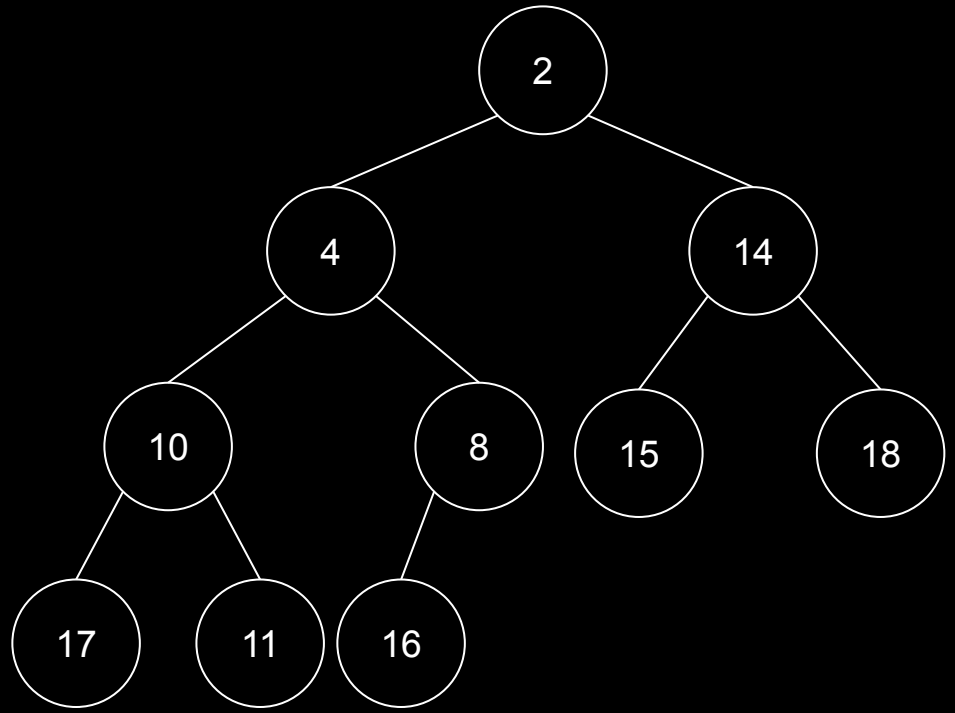
Pré-Ordem

Em pré-ordem buscamos os ramos mais profundos da esquerda para a direita.

Essa técnica para percorrer a árvore é conhecida por **percurso em profundidade** ou *depth-first*.

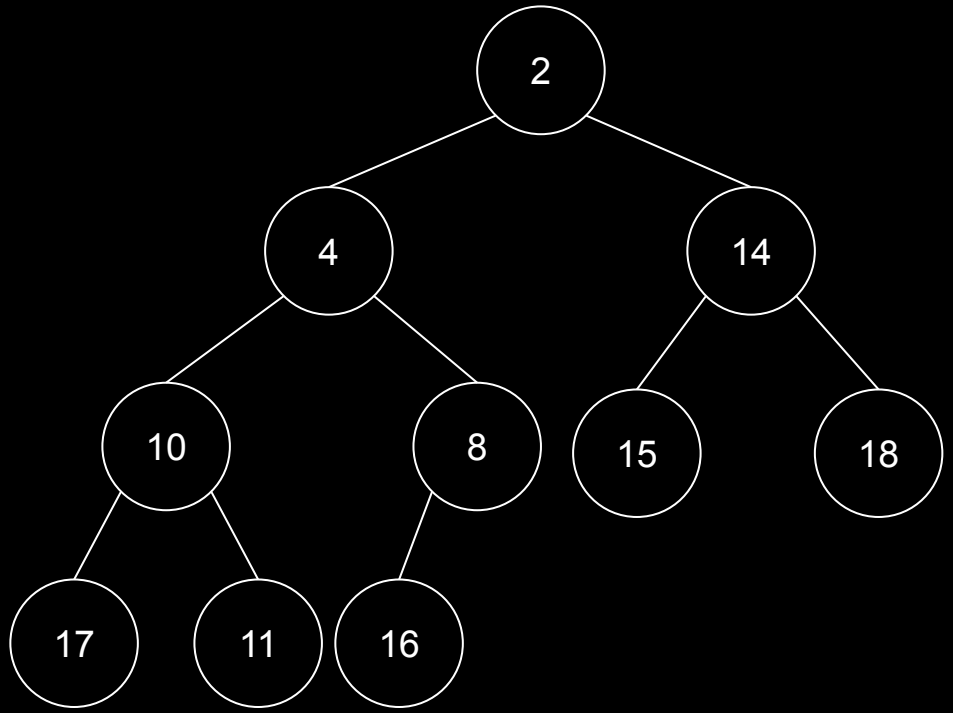
```
void preordem(struct nodo* no){
   if (no !=NULL){
       printf(no->valor);
       preordem(no->fe);
       preordem(no->fd);
   }
}
```

O algoritmo é recursivo!



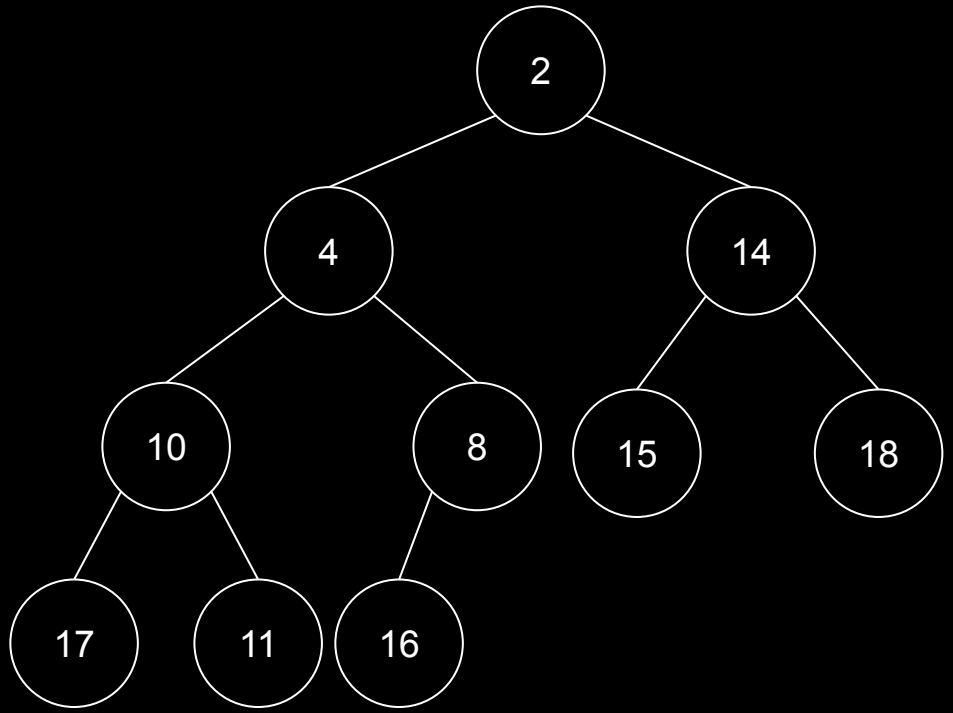
Em-ordem

- Aplicar em-ordem na subárvore esquerda.
- Visitar a raiz.
- Aplicar em-ordem na subárvore direita.



Em-ordem

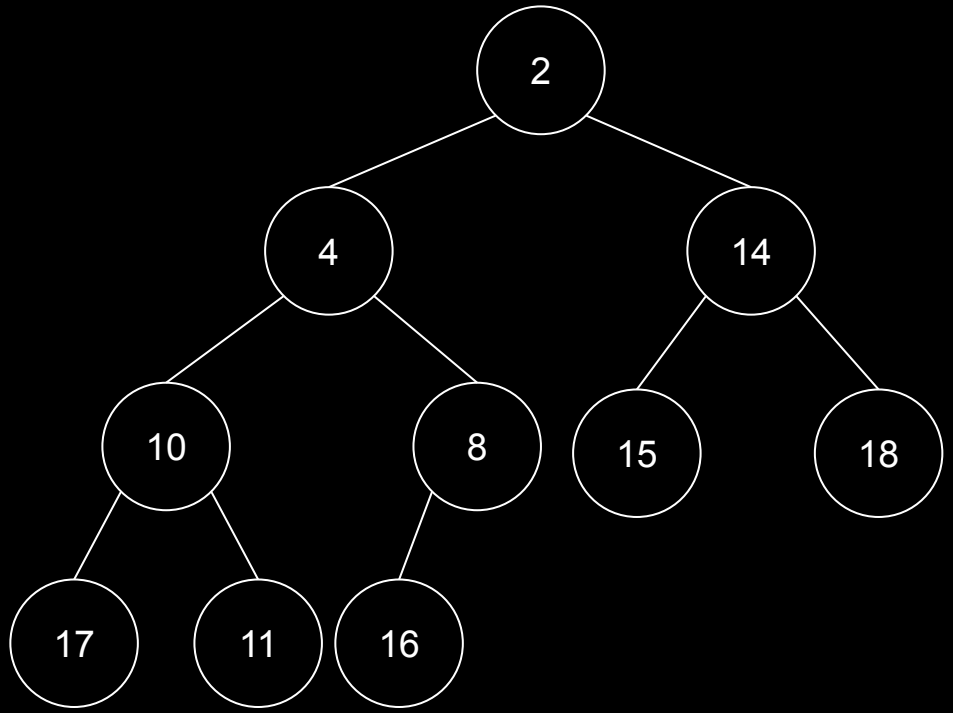
- Aplicar em-ordem na subárvore esquerda.
- Visitar a raiz.
- Aplicar em-ordem na subárvore direita.



Visita em em-ordem: 17, 10, 11, 4, 16, 8, 2, 15, 14, 18.

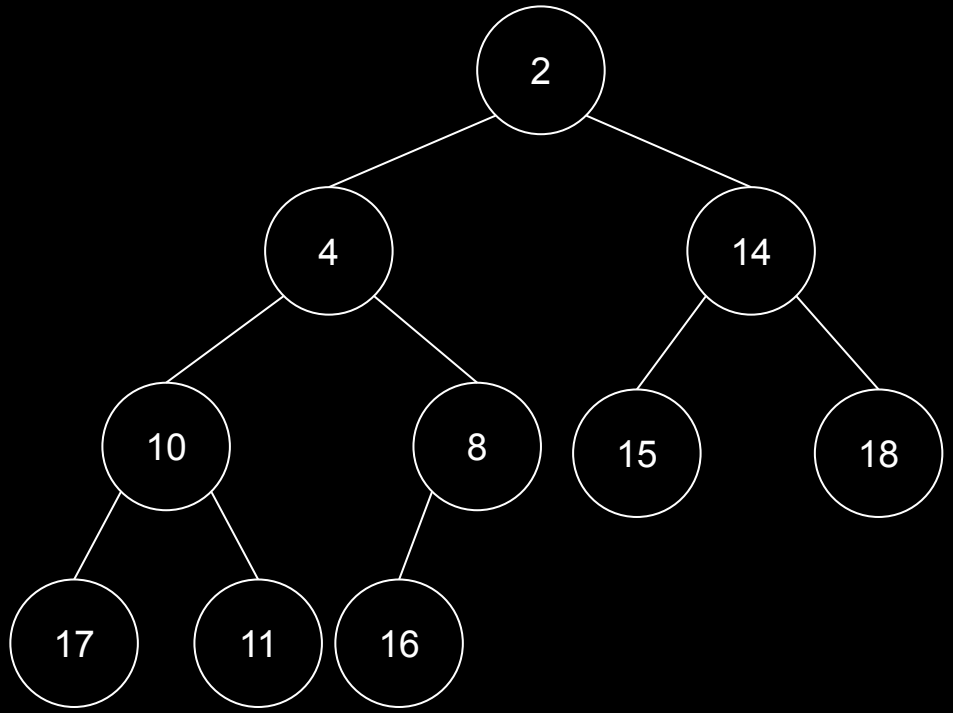
Pós-ordem

- Aplicar pós-ordem na subárvore esquerda.
- Aplicar pós-ordem na subárvore direita.
- Visitar a raiz.



Pós-ordem

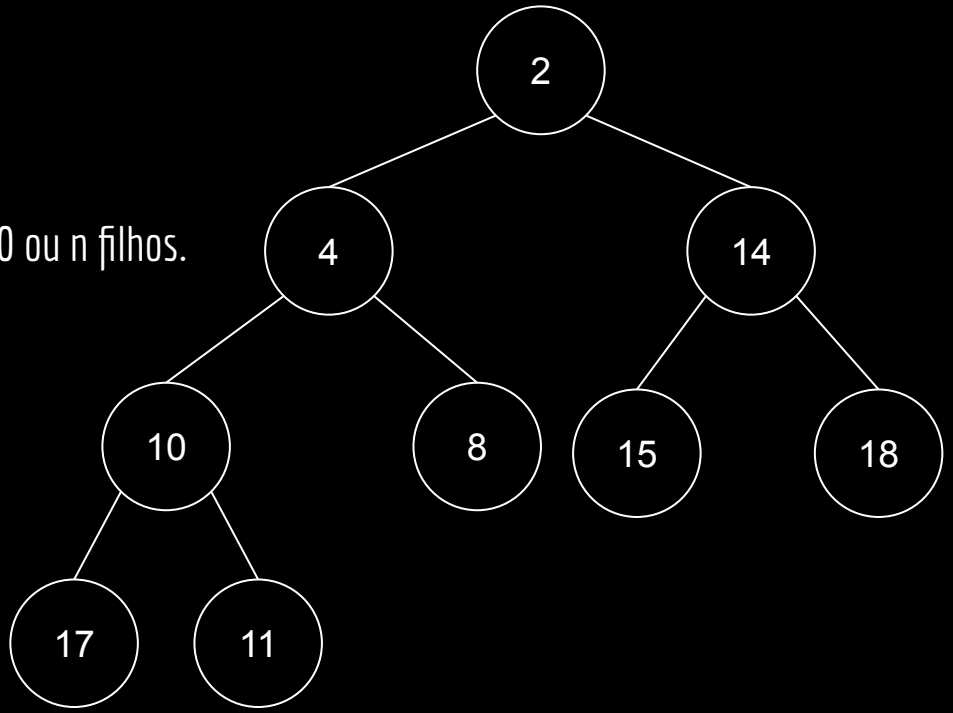
- Aplicar pós-ordem na subárvore esquerda.
- Aplicar pós-ordem na subárvore direita.
- Visitar a raiz.



Visita em pós-ordem: 17, 11, 10, 16, 8, 4, 15, 18, 14, 2.

Árvore Cheia

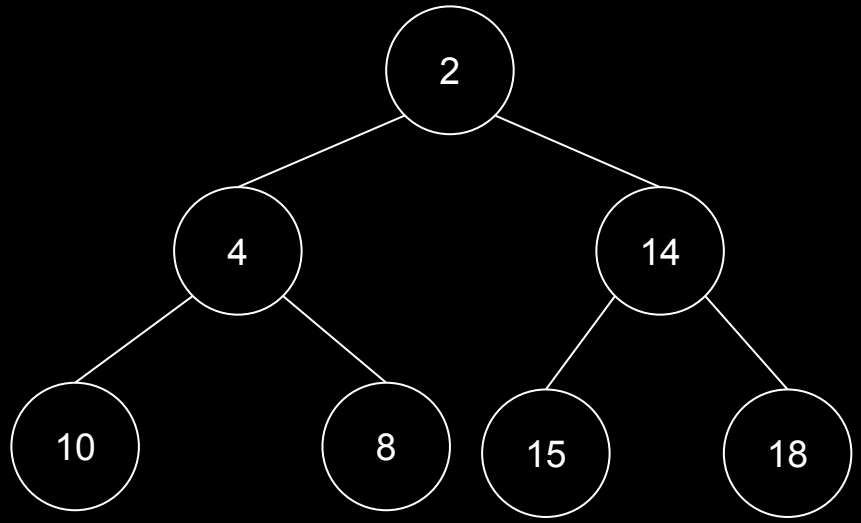
Os nodos de uma árvore n-Ária cheia possuem exatamente 0 ou n filhos.



Árvore Completa

Em uma árvore *n*-Ária completa, todos os níveis possuem *n* filhos, exceto o último nível, onde todas as folhas possuem 0 filhos.

Todas as folhas estão no mesmo nível.



Árvore Completa

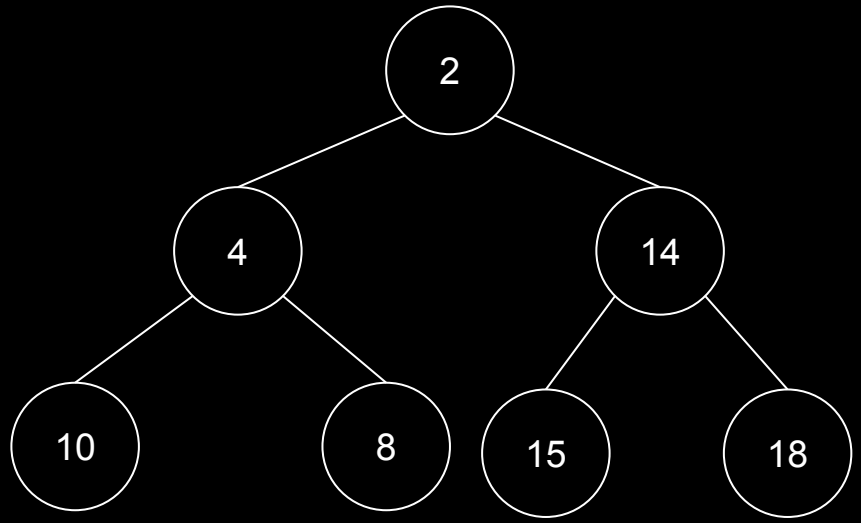
Em um árvore completa de altura *h* e *n* nodos:

Número de folhas (nodos externos): 2^h.

Número de nodos internos: $2^h - 1$.

Número total de nodos: $2^{h+1} - 1$.

Altura h: $\lfloor \log_2 n \rfloor$

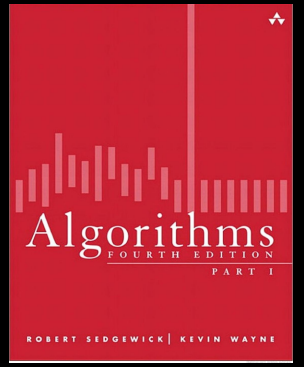


Exercícios

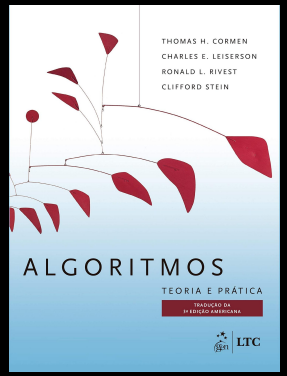
1. Crie uma árvore binária de pessoas, onde a chave é o cpf.
    a. Use a ideia da struct da aula passada, e combine com os conceitos da aula de hoje.
2. Insira elementos na árvore.
    a. Por enquanto, insira no “braço” sem se preocupar com nenhuma ordem entre os elementos da árvore.
3. Percorra a árvore em pré-ordem, em-ordem, e em pós-ordem, de forma recursiva.
4. Crie uma função para calcular a altura da árvore
5. Remova a recursão do item 3. Dica: você vai precisar de uma pilha.

Referências

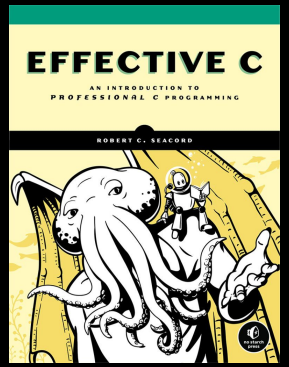
R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2011



T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012.



Seacord, R. C. Effective C: An ntroduction to Professional C Programming. 2020.



Licença