# Árvores Binárias I

03/11/2025

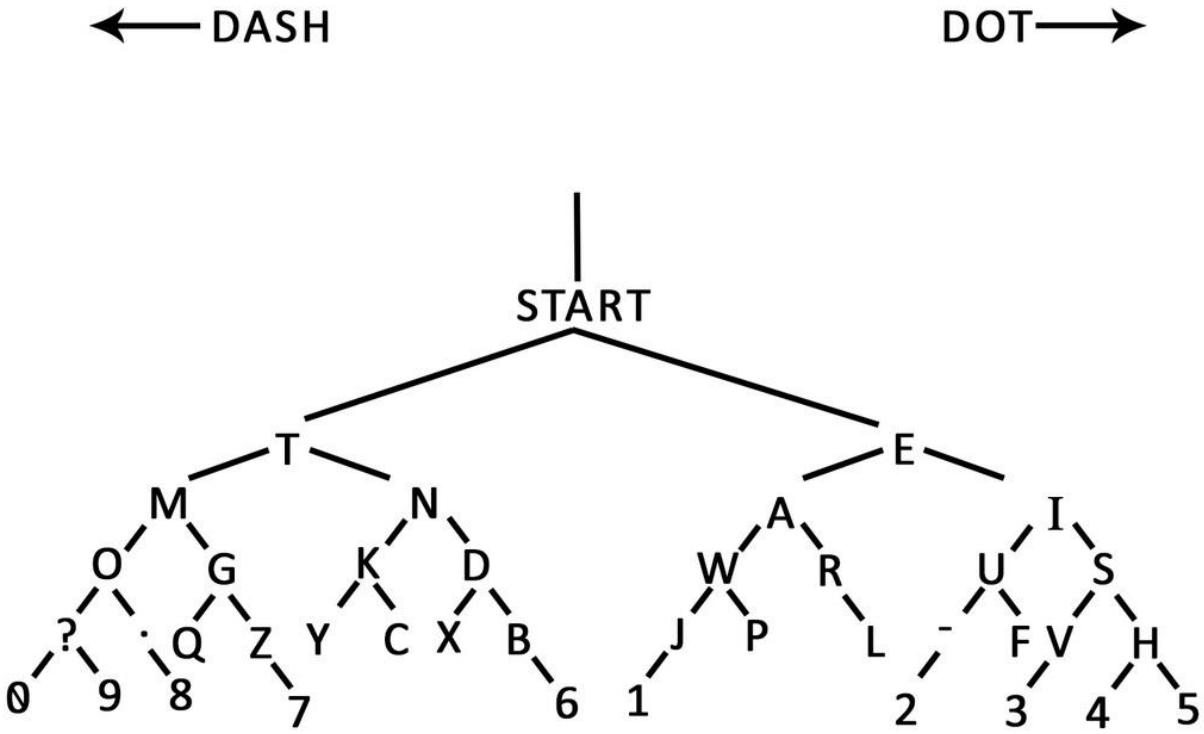
# Ficheiro ZIP

- Está disponível no Moodle um ficheiro ZIP de suporte aos tópicos de hoje
- 1ª versão do tipo abstrato Árvore Binária
- Funções incompletas, que permitem trabalho autónomo de desenvolvimento e teste

# Sumário

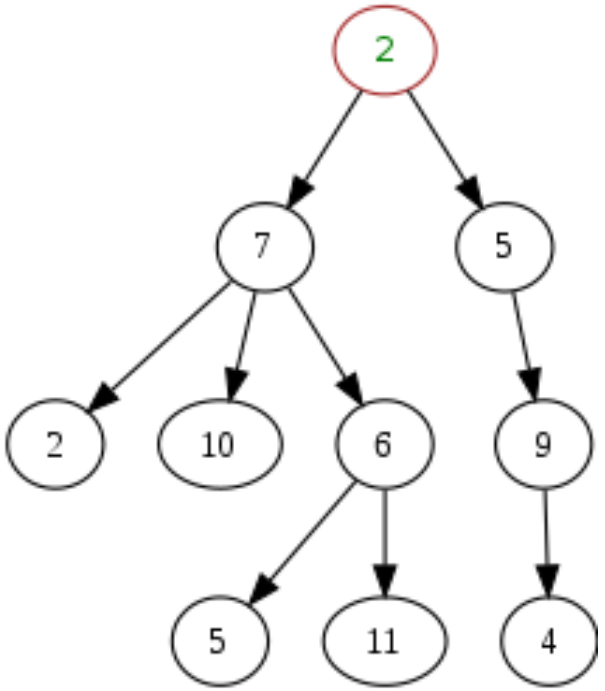
- Árvores binárias; terminologia e algumas propriedades
- O TAD Árvore Binária
- Possíveis estruturas de dados
- Algoritmos recursivos – exemplos simples
- Exercícios / Tarefas 

# Árvores

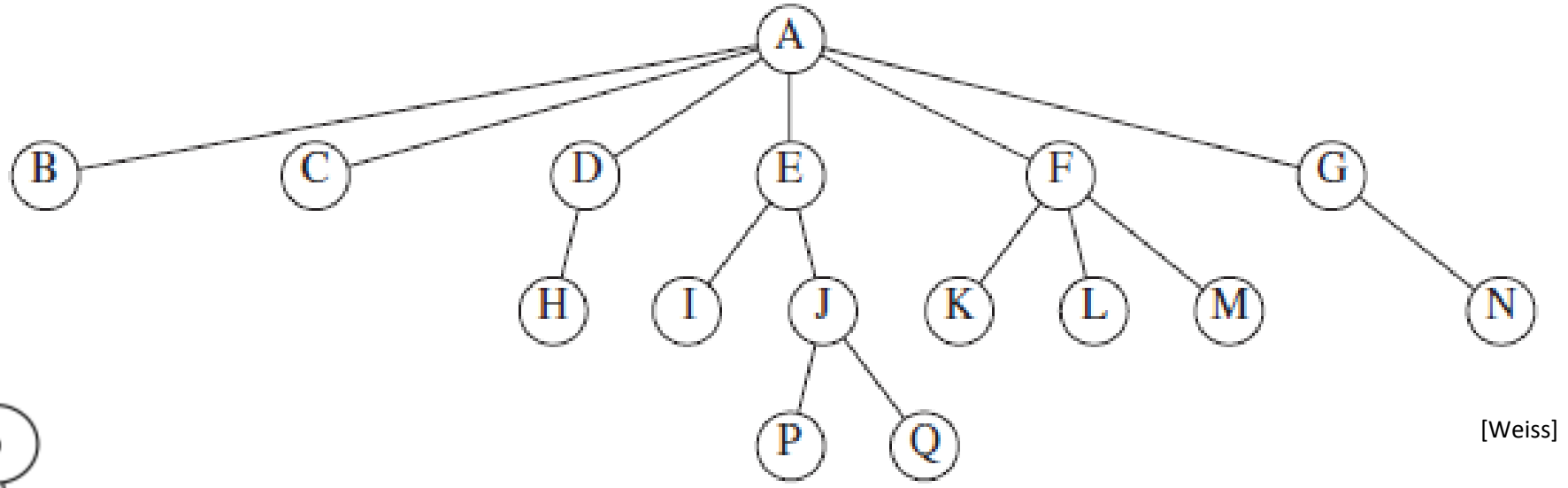



[weebly.com]

# Árvores – Arcos ? – Ordem ?




[Wikipedia]




[Weiss]

# Árvores vs Grafos – Definição ?

- Todas as árvores são grafos !
- MAS, nem todos os grafos são árvores !!

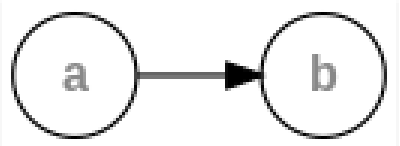
- Definição ?

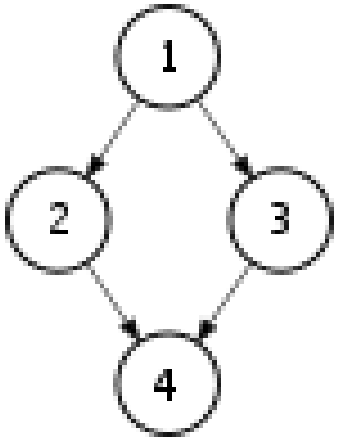
# Árvores vs Grafos – Definições equivalentes

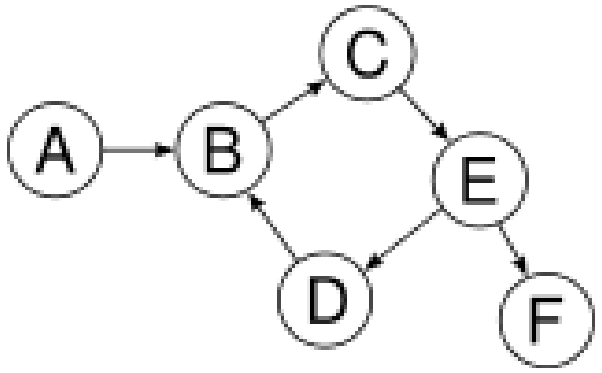
- Todas as árvores são grafos !
- MAS, nem todos os grafos são árvores !!

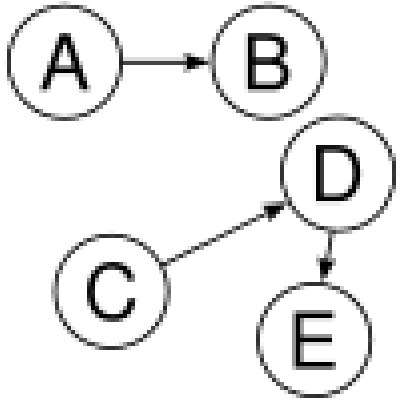
- Árvore: existe um só caminho entre qualquer par de nós distintos
- Árvore: apagar um qualquer arco origina duas árvores desconexas
- Árvore com n nós tem (n – 1) arcos e não contém nenhum ciclo
- Árvore com n nós tem (n – 1) arcos e é conexa

# São árvores orientadas ou não ?




[Wikipedia]

# Árvores orientadas vs Grafos orientados

- Todas as árvores orientadas são grafos orientados !
- MAS, nem todos os grafos orientados são árvores orientadas !!

- Definição ?

# Árvores orientadas vs Grafos orientados

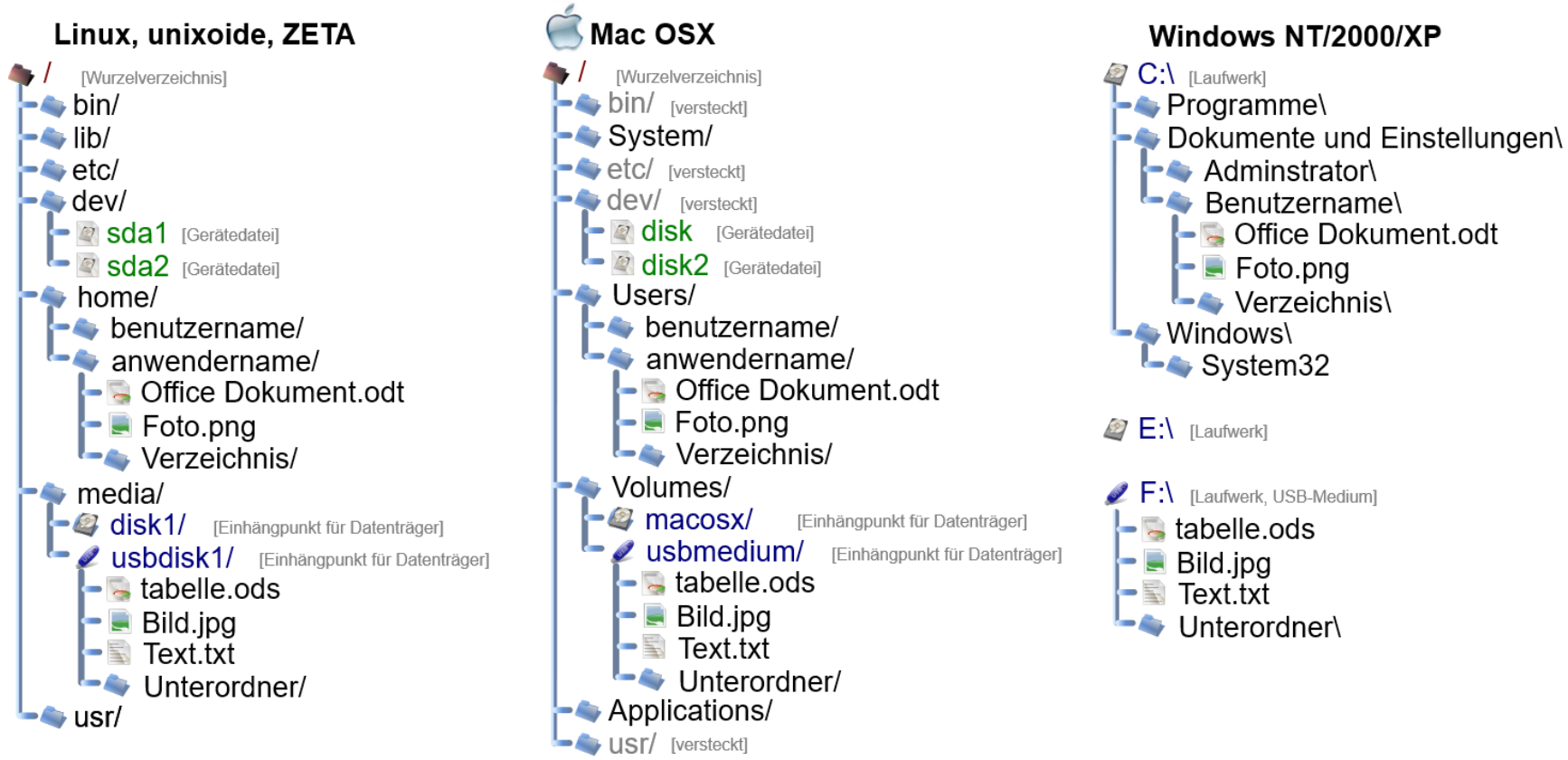
- Todas as árvores orientadas são grafos orientados !
- MAS, nem todos os grafos orientados são árvores orientadas !!

- Árvore orientada:
- O nó raiz não tem qualquer arco incidente
- Cada um dos outros nós tem um só arco incidente
- Existe um só caminho orientado entre a raiz e cada um dos outros nós

# Exemplos de aplicação

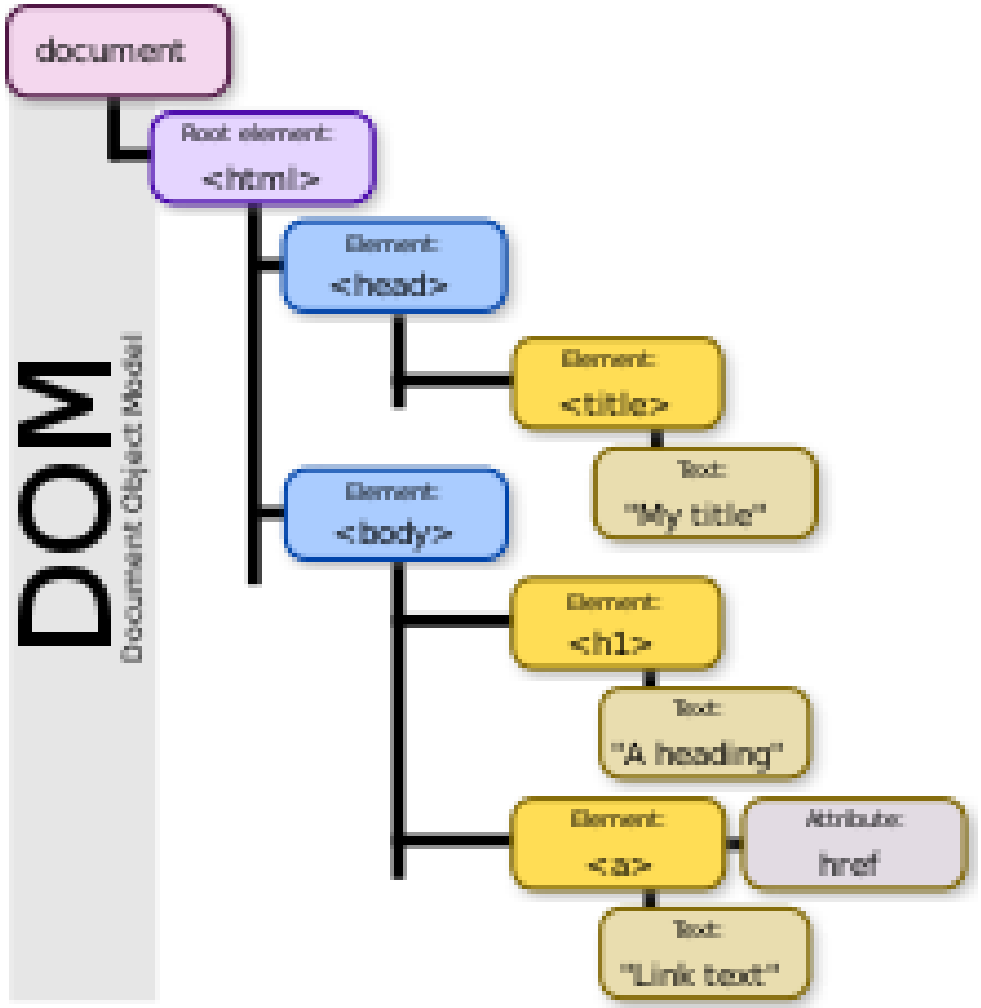
- Árvores genealógicas
- Árvores de pedigree
- Torneios
- Hierarquia de uma organização
- Estrutura de um livro
- Taxonomias hierárquicas
- …

# Organização do sistema de ficheiros




[Wikipedia]

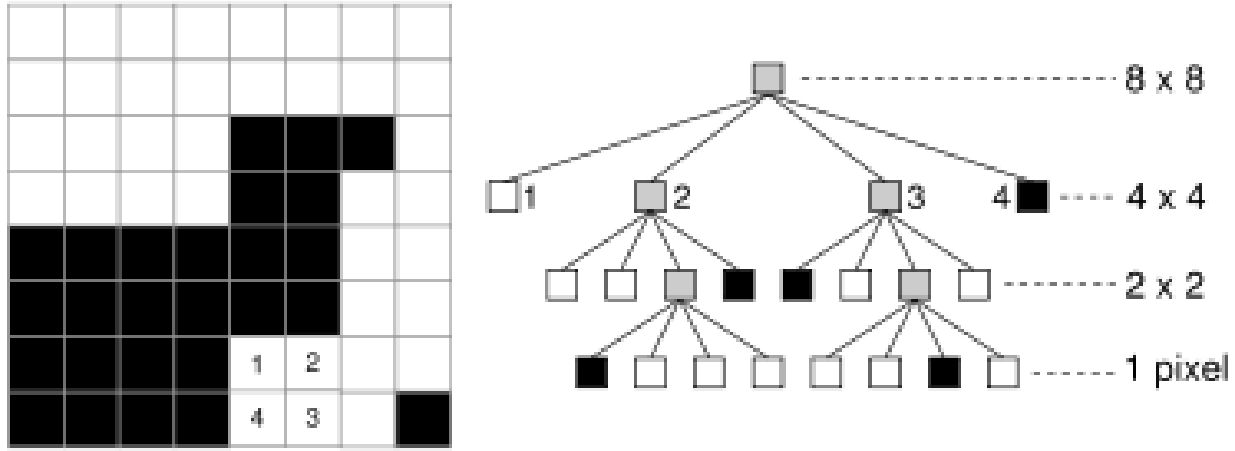
# DOM tree




[Wikipedia]

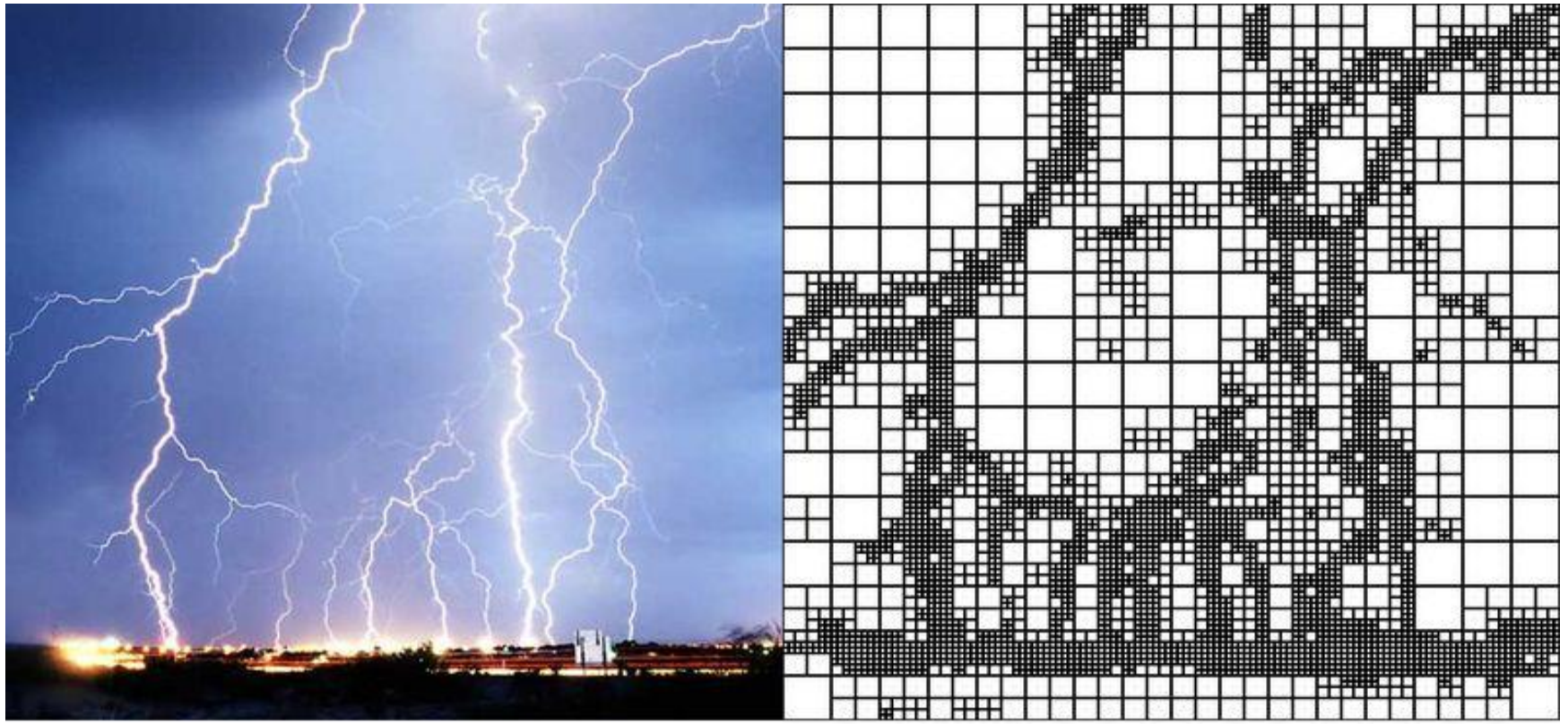
# Quadtrees – Árvores quaternárias

- Representação de imagens binárias
  - Subdivisão recursiva em 4 quadrantes
  - Nós pretos, brancos e cinzentos




[Wikipedia]

# Quadtrees



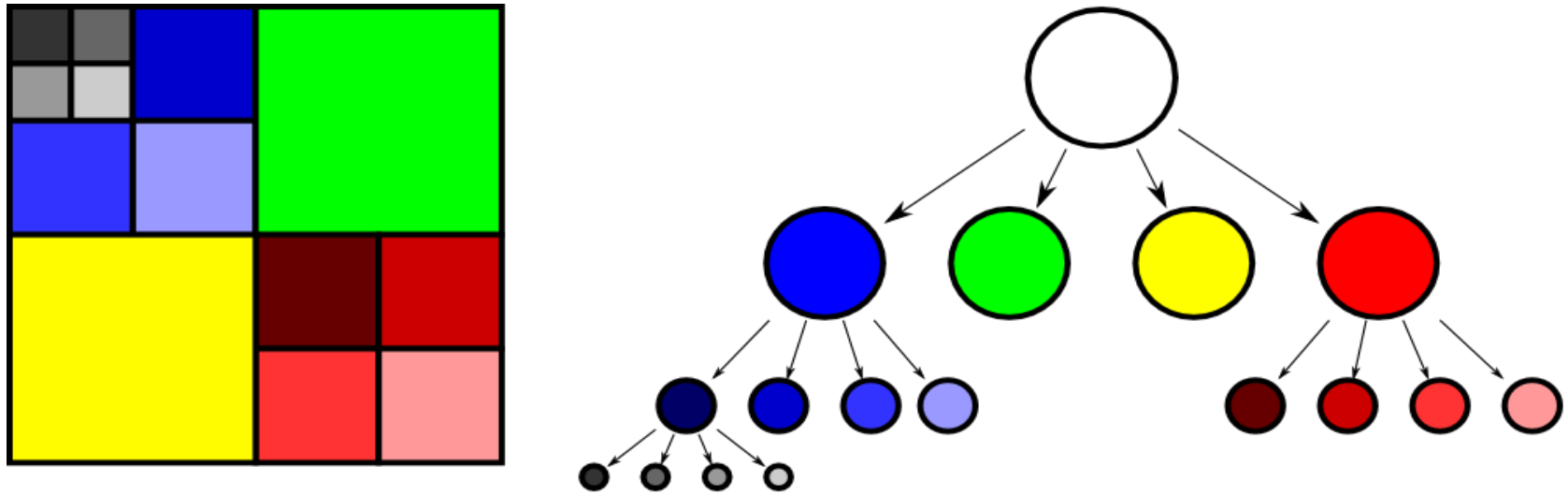
Originalbild

Aus Konturwerten erstellter Quadtree
(Größe: 1229 Byte)

[Wikipedia]

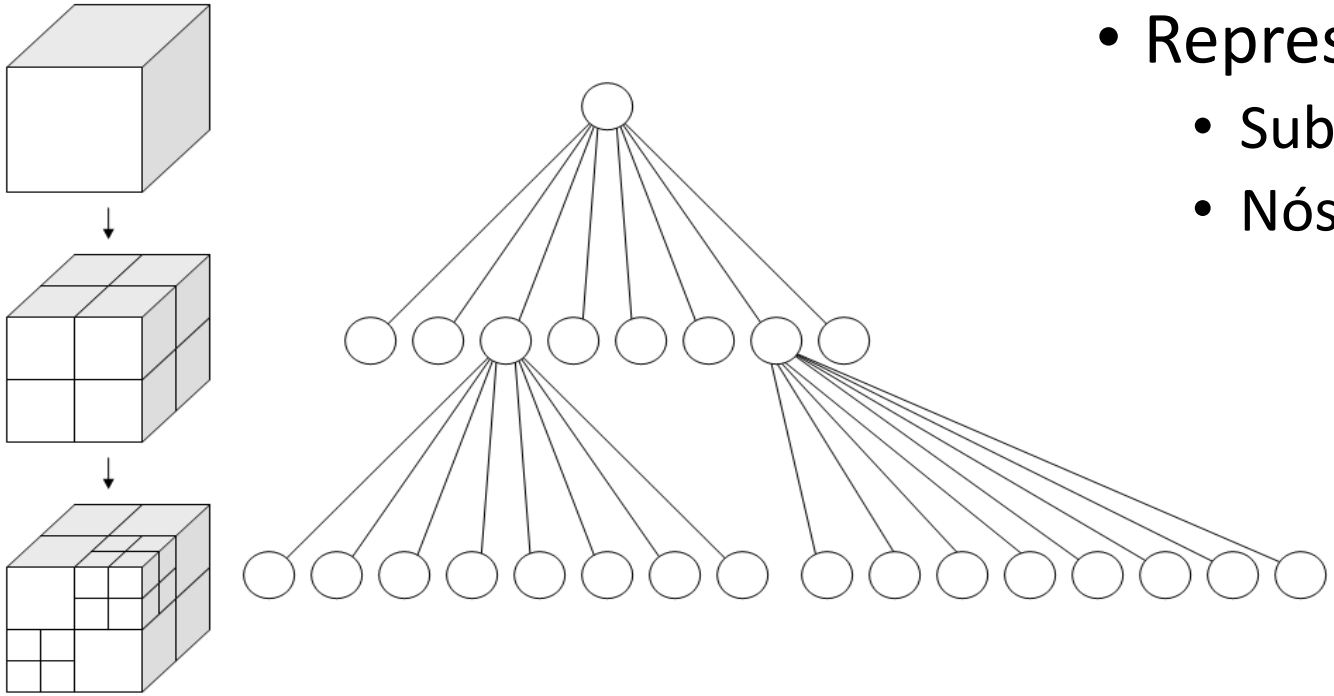
# Quadtrees

- Representação de imagens a cores



[Wikipedia]

# Octrees – Árvores octais



[Wikipedia]

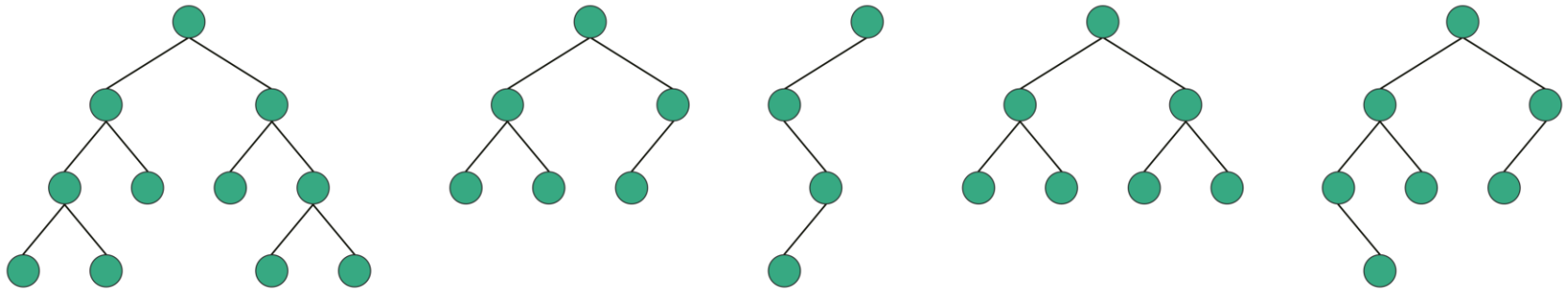
- Representação de  volumes
  - Subdivisão recursiva em 8 octantes
  - Nós pretos, brancos e cinzentos

# Mais exemplos de aplicação

- Herança simples em POO
- Comportamento dinâmico de algoritmos “divide-and-conquer”
- …

# Resumo – Tipos de árvores

- Árvores orientadas vs não orientadas
- Árvores binárias, ternárias, quaternárias, … , m-árias
- Árvores binárias ordenadas – Como ?
- Árvores binárias equilibradas em altura – Como ?

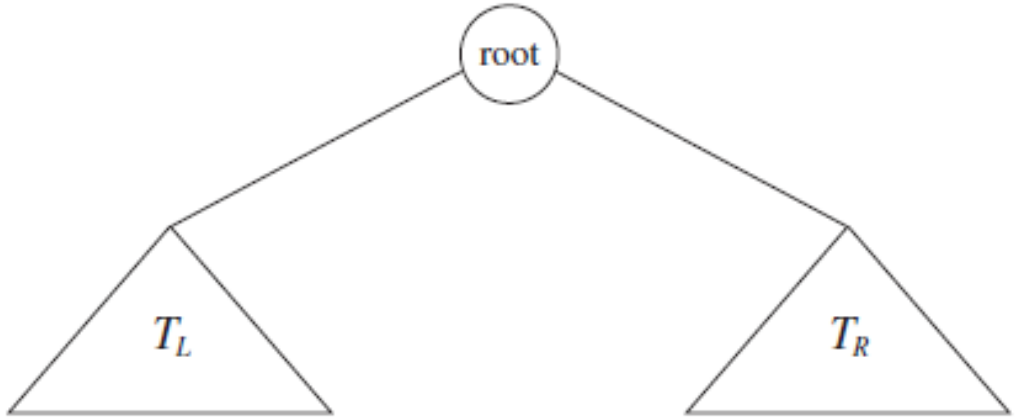


[towardsdatascience.com]

# Árvores Binárias

# Árvore Binária – Definição recursiva

- Uma árvore binária é formada por um conjunto finito de nós (n >= 0)
- Uma árvore binária é vazia
- OU é constituída por um nó raiz que referencia duas (sub-)árvores binárias disjuntas (SAEsq e SADir)
  - Arcos orientados para a SAEsq e para a SADir

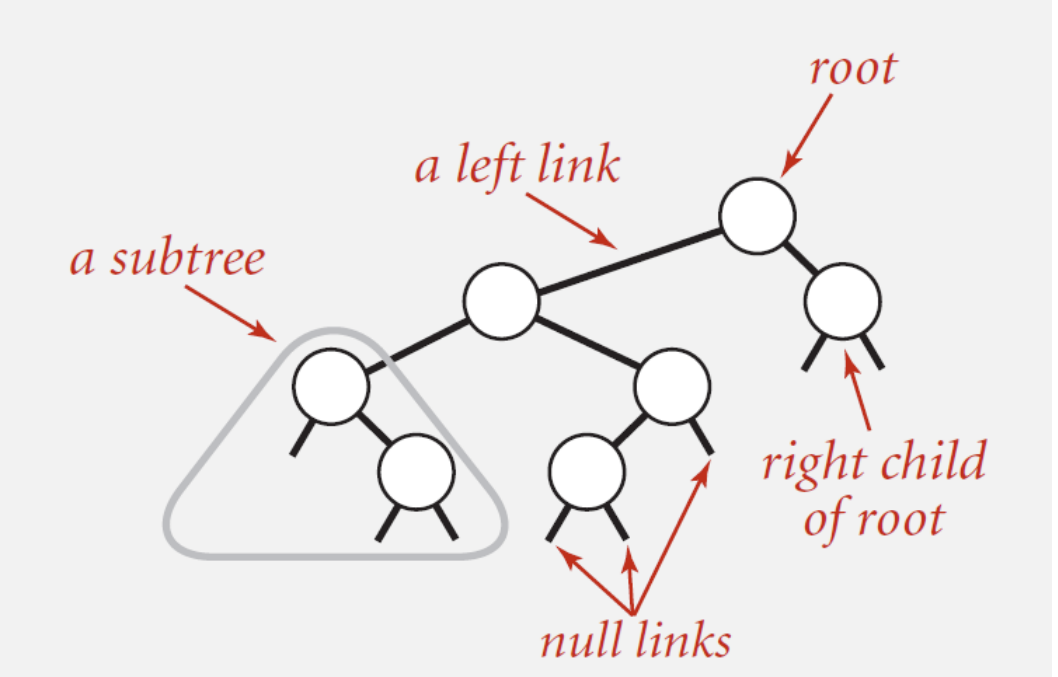



[Weiss]

# Árvore Binária – Terminologia

- Grau de um nó: nº das suas subárvores não-vazias
- Grau de uma árvore ?

- Nós não-terminais vs Folhas
- Pai, filhos, irmãos
- Antepassados, descendentes

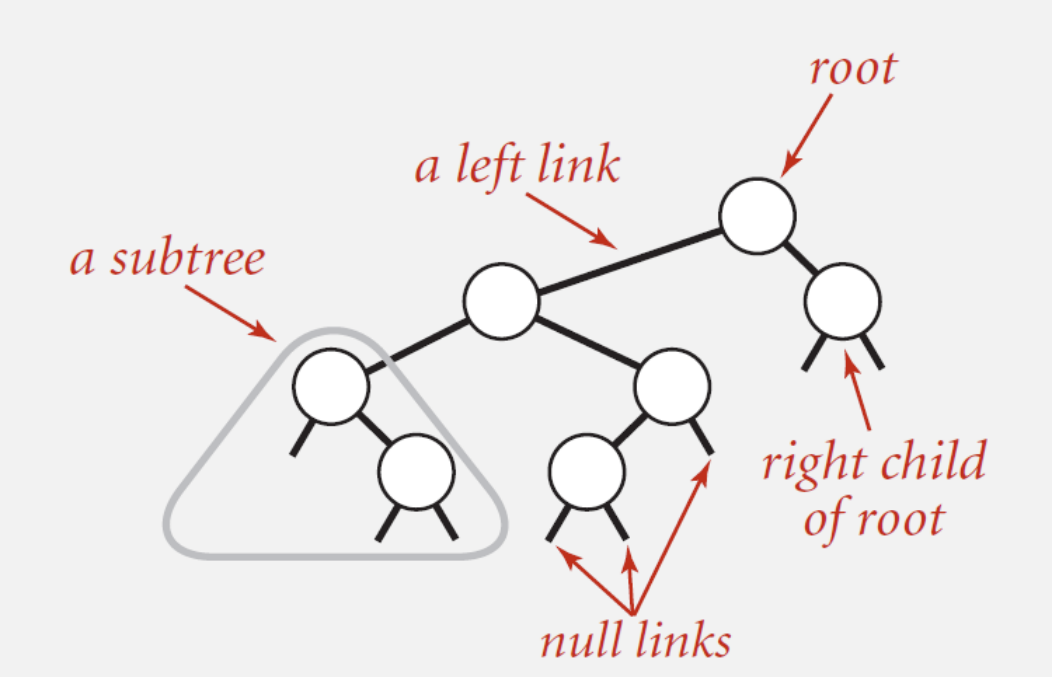



[Sedgewick & Wayne]

# Árvore Binária – Terminologia

- Nível de um nó ?
- A raiz está no nível 0
- Qual é a definição recursiva ?

- Altura de uma árvore ?
- Nº de arcos do caminho mais longo da raiz da árvore para uma das suas folhas
- Índice do último nível




[Sedgewick & Wayne]

# Algumas propriedades das árvores binárias

- Questões simples !!
- Nº máximo de nós no nível i ?
- Nº máximo de nós numa árvore de altura h ? Quando ?
- Nº mínimo de nós numa árvore de altura h ? Quando ?
- Façam exemplos !

# Algumas propriedades das árvores binárias

- Questões simples !!
- Nº máximo de nós no nível i ? $\mathbf{2^i}$
- Nº máximo de nós numa árvore de altura h ? Quando ? $\mathbf{2^{h+1} - 1}$
- Nº mínimo de nós numa árvore de altura h ? Quando ? $\mathbf{h + 1}$

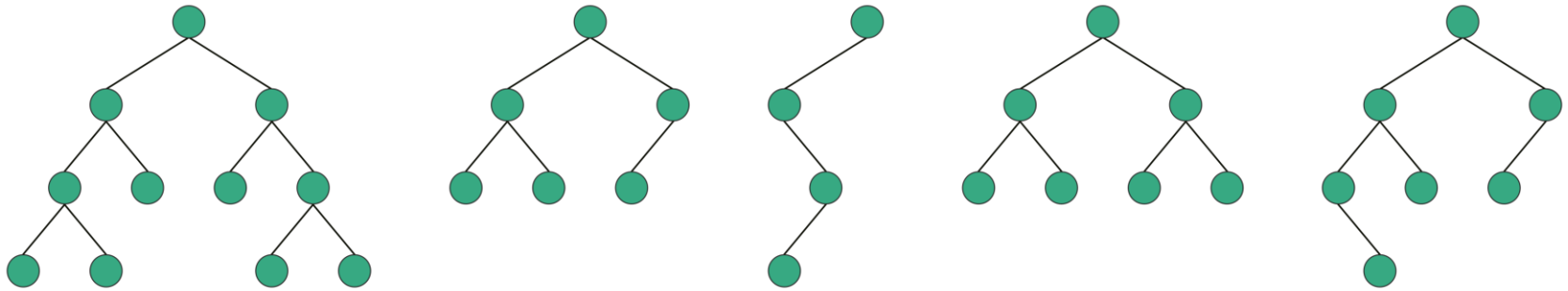
$$\mathbf{h + 1 <= n <= 2^{h+1} - 1}$$

# Algumas propriedades das árvores binárias

- Completar a frase:

- A altura de uma árvore binária com n nós é pelo menos … e quando muito …

- Façam exemplos !!



[towardsdatascience.com]

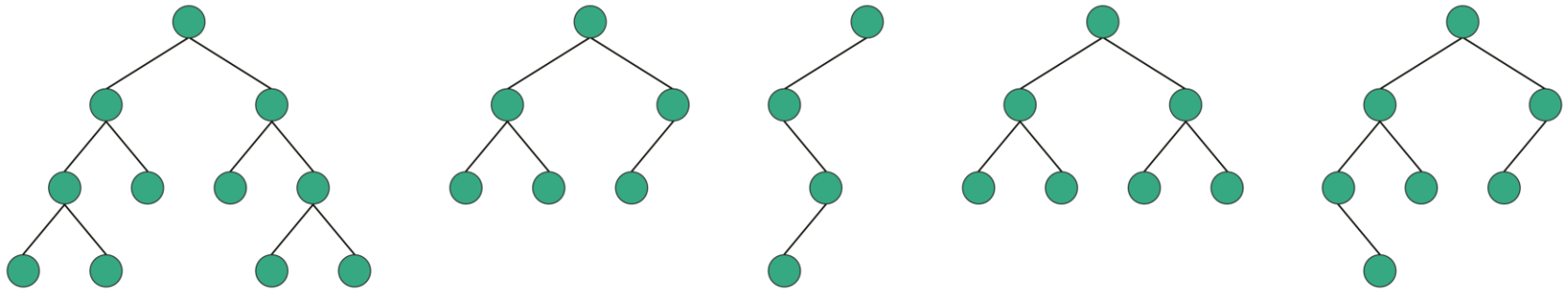
# O TAD Árvore Binária

# TAD Árvore Binária – Funcionalidades

- Conjunto de elementos do mesmo tipo
- Armazenados sem qualquer ordem particular

- Procura / inserção / remoção / substituição
- Pertença

- search() / insert() / remove() / replace()
- contains()
- size() / isEmpty()
- create() / destroy()



[towardsdatascience.com]

# O TAD Árvore Binária de Inteiros

# Árvore Binária de Inteiros – Funcionalidades

```c
#ifndef _INTEGERS_BINTREE_
#define _INTEGERS_BINTREE_

// JUST storing integers
typedef int ItemType;

typedef struct _TreeNode Tree;

Tree* TreeCreate(void);

void TreeDestroy(Tree** pRoot);
```

```c
// Tree properties

int TreeIsEmpty(const Tree* root);

int TreeEquals(const Tree* root1, const Tree* root2);

int TreeMirrors(const Tree* root1, const Tree* root2);

// ...
```

# Árvore Binária de Inteiros – Funcionalidades

```
// Getters

int TreeGetNumberOfNodes(const Tree* root);

int TreeGetHeight(const Tree* root);

ItemType TreeGetMin(const Tree* root);

ItemType TreeGetMax(const Tree* root);

Tree* TreeGetPointerToMinNode(const Tree* root);

Tree* TreeGetPointerToMaxNode(const Tree* root);

// ...
```

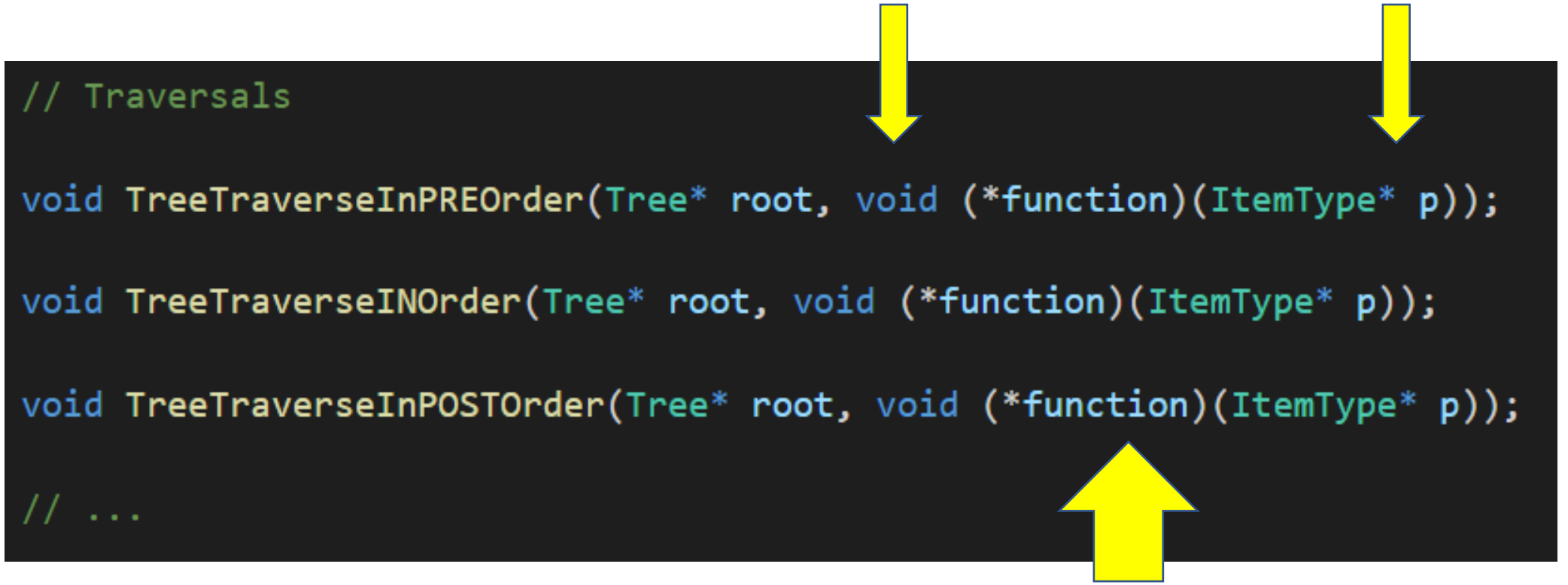
- A estrutura e os elementos da árvore não são alterados

# Árvore Binária de Inteiros – Funcionalidades

```
// Operations with items

int TreeContains(const Tree* root, const ItemType item);

int TreeAdd(Tree** pRoot, const ItemType item);

int TreeRemove(Tree** pRoot, const ItemType item);

// ...
```

- TreeAdd()
- Se vazia, criar o nó raiz e atualizar o ponteiro
- TreeRemove()
- Se o nó raiz original é apagado, atualizar o ponteiro

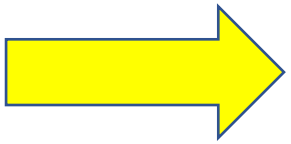
# Árvore Binária de Inteiros – Travessias



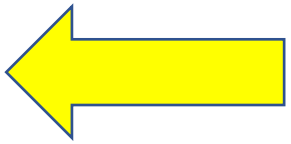
```
// Traversals

void TreeTraverseInPREOrder(Tree* root, void (*function)(ItemType* p));

void TreeTraverseINOrder(Tree* root, void (*function)(ItemType* p));

void TreeTraverseInPOSTOrder(Tree* root, void (*function)(ItemType* p));

// ...
```

- 2º argumento é um ponteiro para uma função genérica
- Que pode consultar ou alterar a informação dos nós da árvore

# Árvore Binária – Funções para as travessias

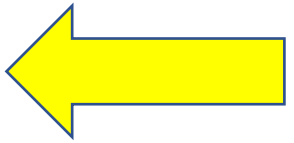


```c
#include "IntegersBinTree.h"

void printInteger(int* p) { printf("%d ", *p); }

void multiplyIntegerBy2(int* p) { *p *= 2; }

int main(void) {
  Tree* tree = createExampleTree();

  printf("Created an example tree\n");

  if (TreeIsEmpty(tree)) {
    printf("The created tree is EMPTY\n");
  } else {
    printf("The created tree is OK\n");
  }
```

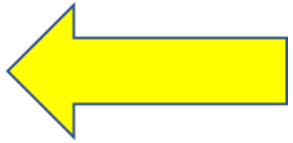


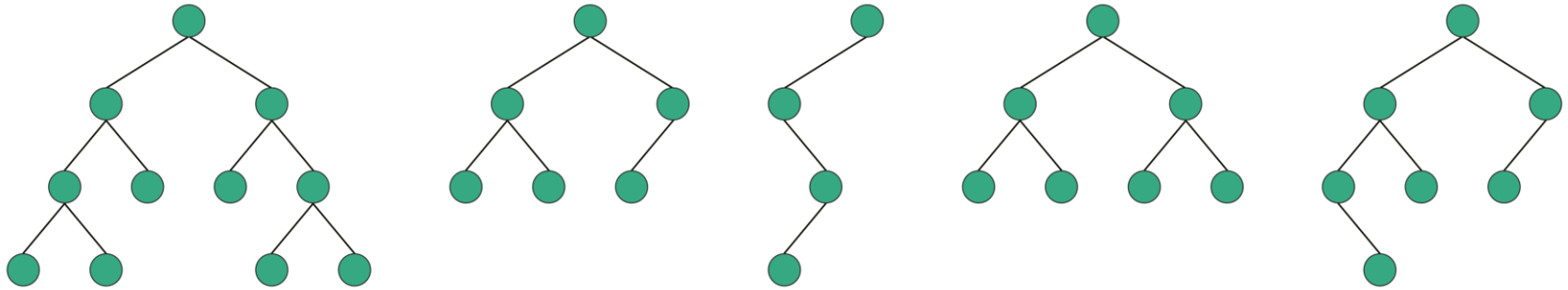
# Árvore Binária – Funções para as travessias

```
printf("PRE-Order traversal : ");

TreeTraverseInPREOrder(tree, printInteger);

printf("\n");
```



```
printf("Multiply each value by 2\n");

TreeTraverseInPREOrder(tree, multiplyIntegerBy2);
```





[towardsdatascience.com]

# O TAD Árvore Binária – Possíveis estruturas de dados

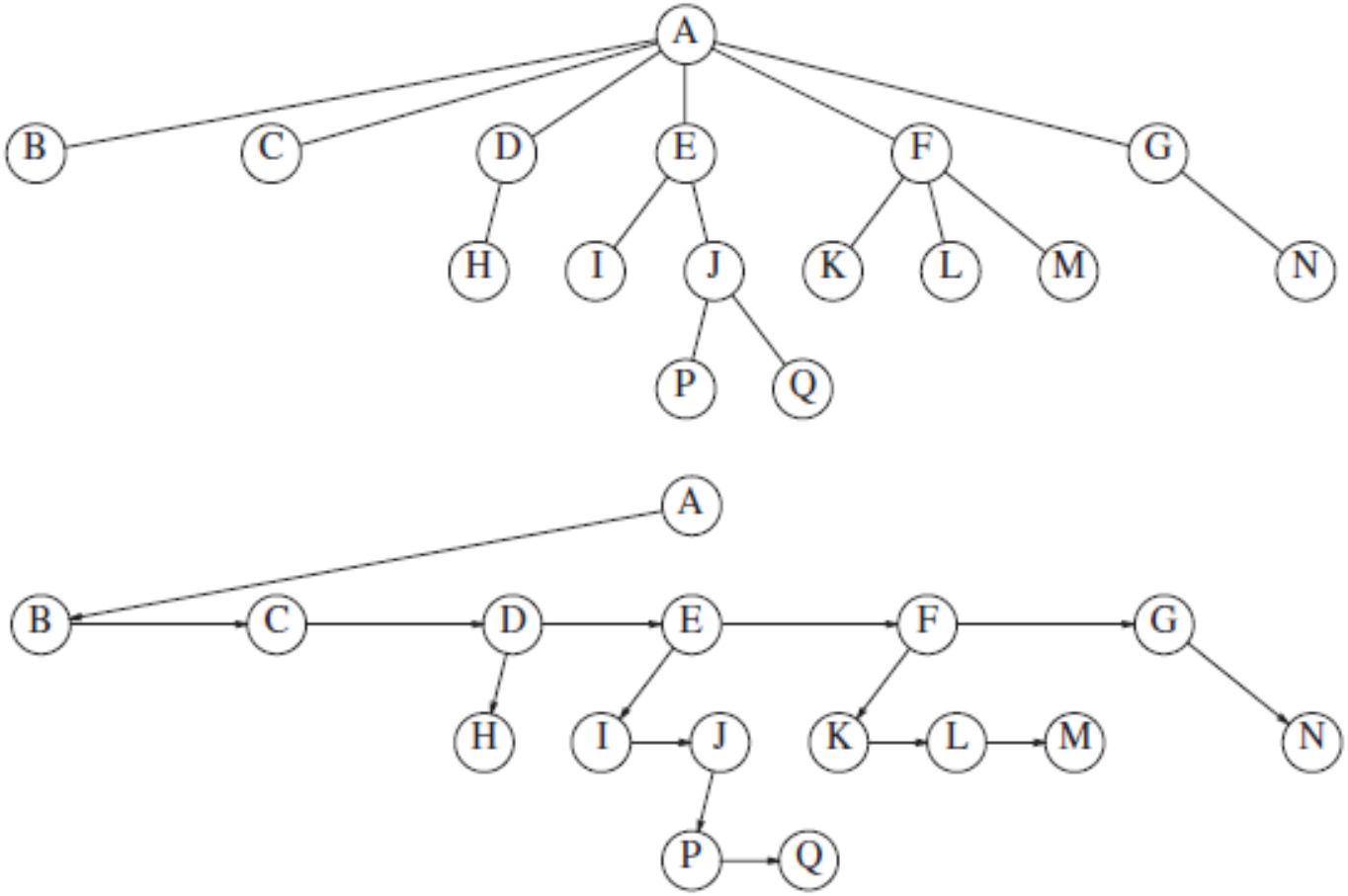
# Possíveis representações internas

- Array – representação sequencial por níveis ✔
- Lista de listas – 1º filho e irmãos
- Nó com 2 ponteiros para as subárvores esquerda e direita
- Nó com mais 1 ponteiro para o progenitor
- …

- Que operações são mais fáceis / difíceis com uma dada representação interna ?

# Lista de listas – 1º filho e irmãos

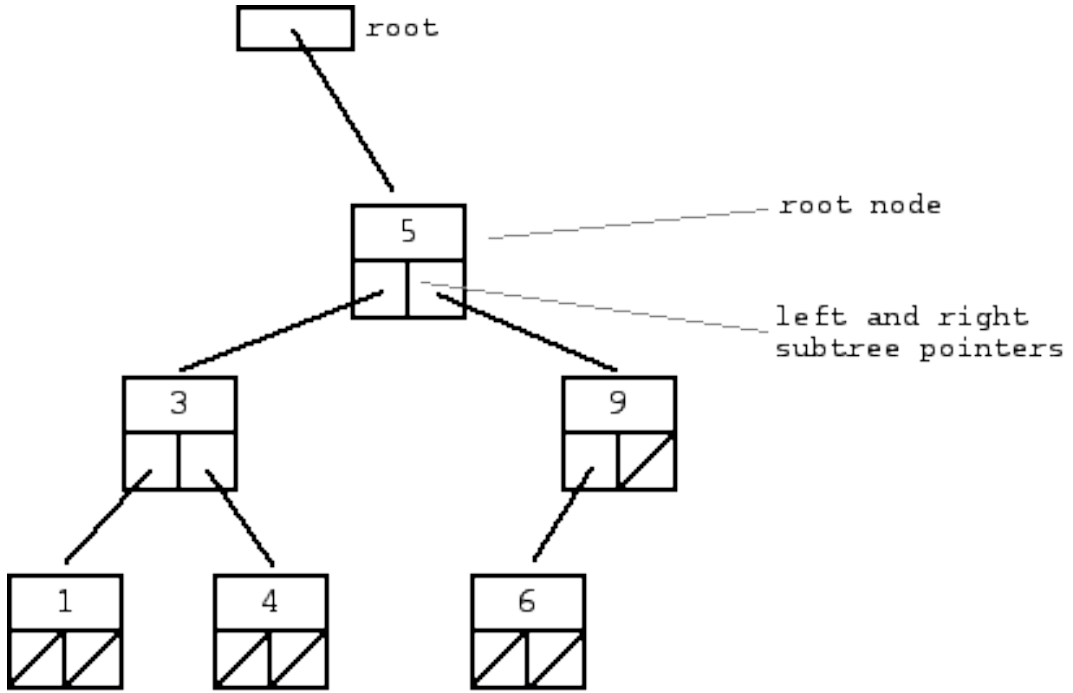
[Weiss]




```
struct TreeNode
{
    Object    element;
    TreeNode *firstChild;
    TreeNode *nextSibling;
};
```

# Árvore Binária – Nó com 2 ponteiros

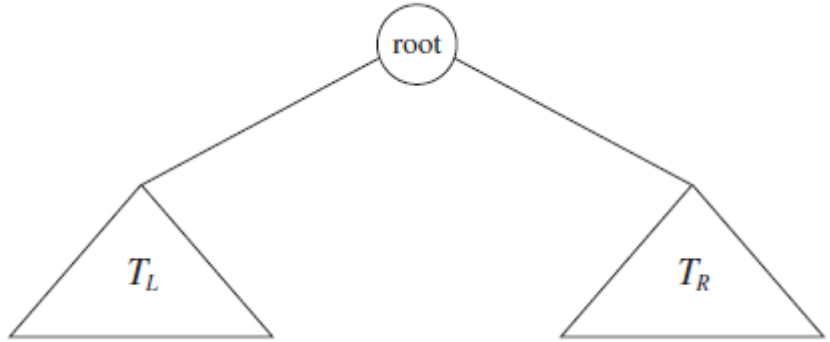
```
struct _TreeNode {
  ItemType item;
  struct _TreeNode* left;
  struct _TreeNode* right;
};
```




[stanford.edu]

# Possíveis representações internas

- Não alterar as funcionalidades do TAD !!
- Que operações são mais fáceis / difíceis com uma dada representação interna?
- P.ex., listar por níveis
- Ou encontrar antepassados de um dado nó




[Weiss]

# Algoritmos Recursivos – Exemplos simples

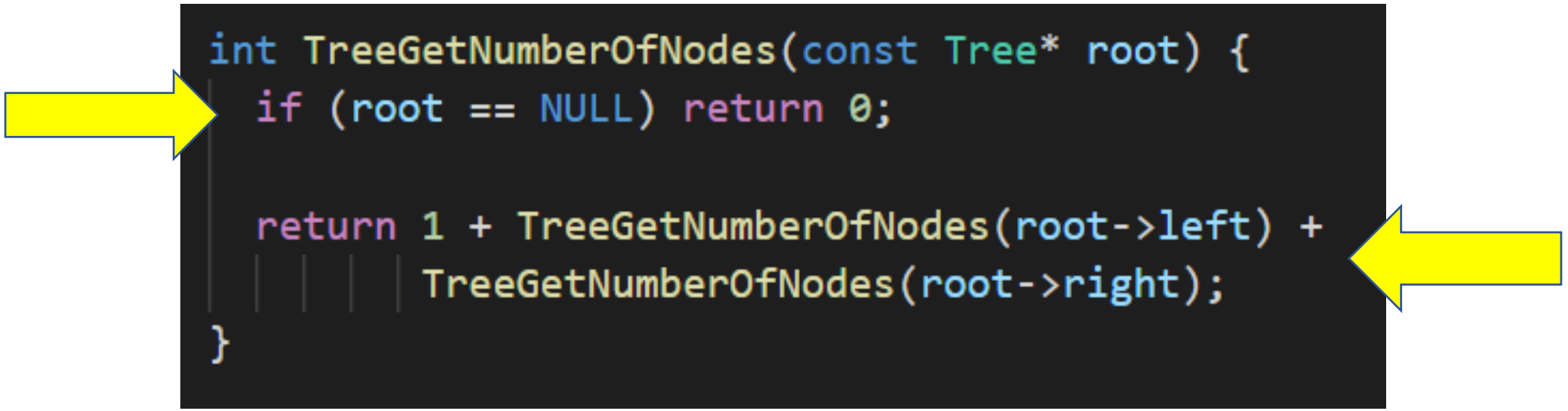
# Árvore Binária – Algoritmos recursivos

- Contar o nº de nós de uma árvore
- Determinar a altura de uma árvore
- Destruir uma árvore
- Verificar se duas árvores são iguais

- Estas operações dependem da **ordem de visita dos nós** das subárvores ?

- Quais são os casos de base ?
- Representar graficamente !!

# Contar o nº de nós de uma árvore binária



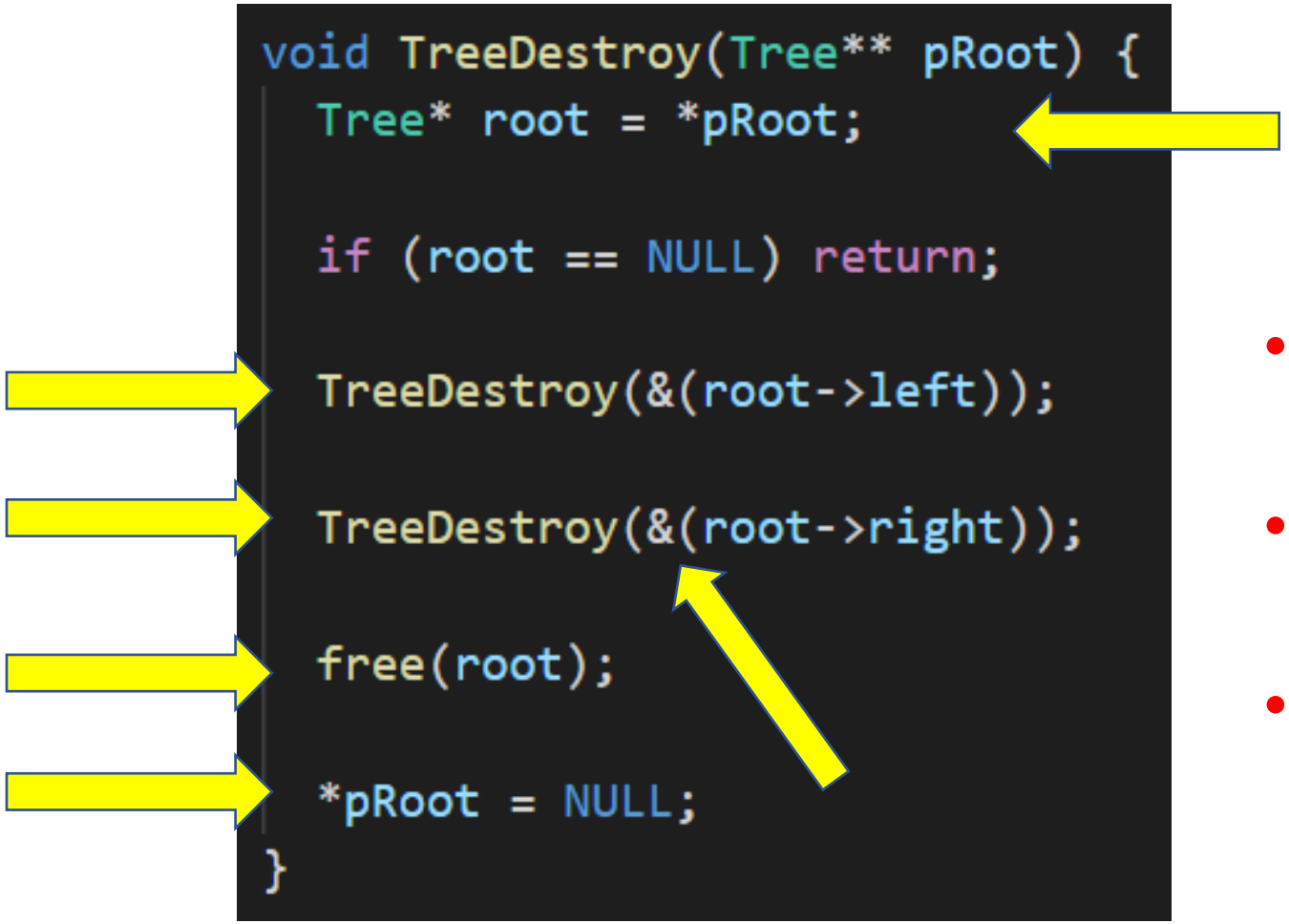
```
int TreeGetNumberOfNodes(const Tree* root) {
  if (root == NULL) return 0;

  return 1 + TreeGetNumberOfNodes(root->left) +
             TreeGetNumberOfNodes(root->right);
}
```

# Determinar a altura de uma árvore binária

```
int TreeGetHeight(const Tree* root) {
  if (root == NULL) return -1;

  int heightLeftSubTree = TreeGetHeight(root->left);

  int heightRightSubTree = TreeGetHeight(root->right);

  if (heightLeftSubTree > heightRightSubTree) {
    return 1 + heightLeftSubTree;
  }

  return 1 + heightRightSubTree;
}
```

- Árvore vazia tem altura -1
- Número de arcos da raiz até à folha mais longínqua

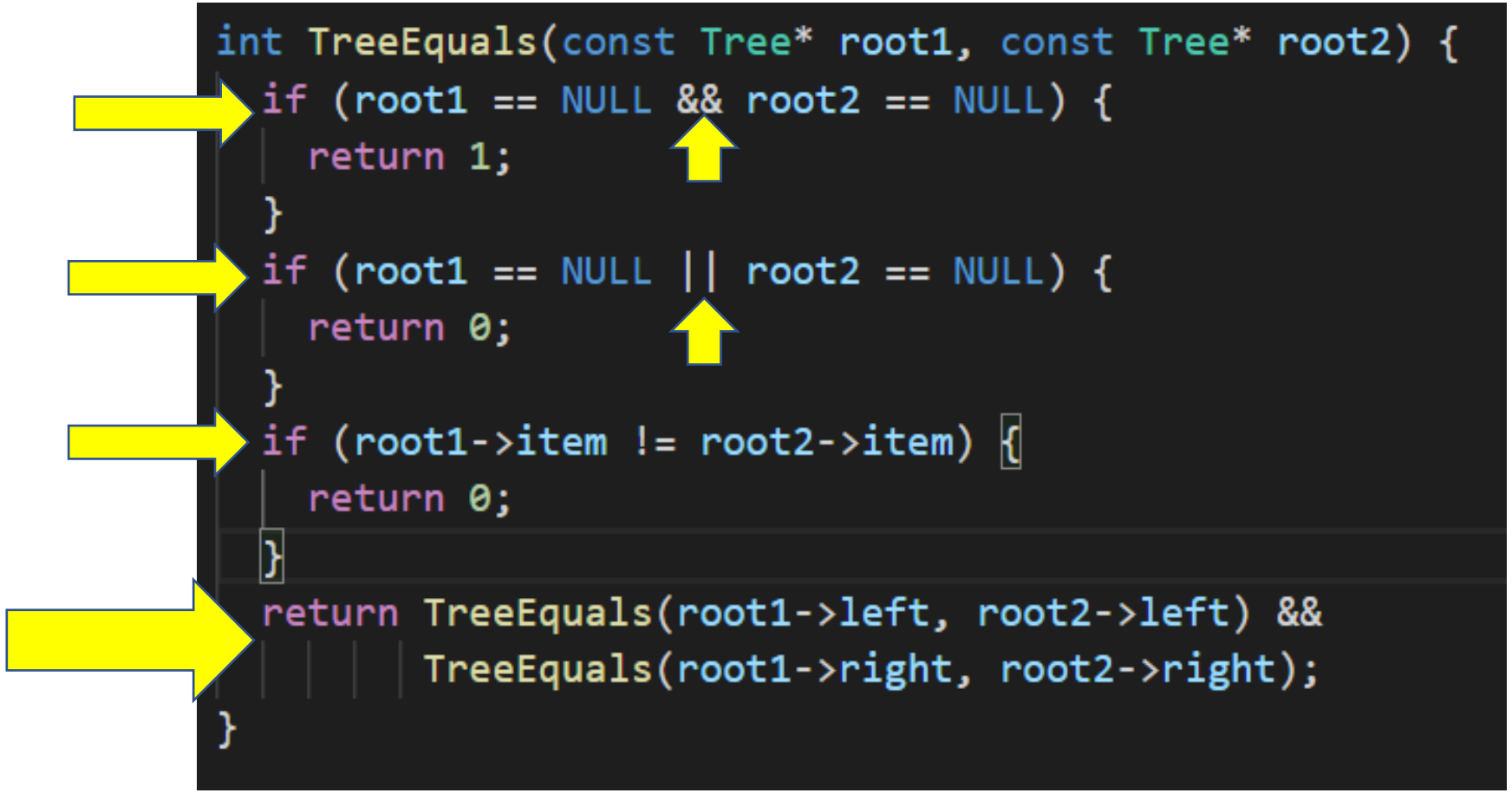
# Destruir uma árvore binária



```
void TreeDestroy(Tree** pRoot) {
  Tree* root = *pRoot;

  if (root == NULL) return;

  TreeDestroy(&(root->left));

  TreeDestroy(&(root->right));

  free(root);

  *pRoot = NULL;
}
```

- Destruir todos os nós da subárvore esquerda
- Destruir todos os nós da subárvore direita
- Libertar a memória alocada ao nó raiz

# Verificar se duas árvores são iguais



```
int TreeEquals(const Tree* root1, const Tree* root2) {
    if (root1 == NULL && root2 == NULL) {
        return 1;
    }
    if (root1 == NULL || root2 == NULL) {
        return 0;
    }
    if (root1->item != root2->item) {
        return 0;
    }
    return TreeEquals(root1->left, root2->left) &&
           TreeEquals(root1->right, root2->right);
}
```

- Casos de base
- Comparar as subárvores


# Exercícios / Tarefas

# Exercício 1 – Escolha-múlttipla

Seja dada uma **árvore binária completa**, i.e., em que todos os níveis da árvore estão completamente preenchidos, com ***n*** **níveis**.

a) A árvore tem um número ímpar de nós.

b) O número de **nós intermédios** (i.e., que **não são folhas** da árvore) é igual a $\mathbf{(2^{n-1} - 1)}$.

c) O número de **folhas** da árvore é igual a $\mathbf{2^{n-1}}$.

d) Todas estão corretas.

# Exercício 2 – Escolha-múltipla

O *"array"* seguinte armazena, **por níveis**, os elementos de uma **árvore ternária**.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 6 | 9 | 2 | 5 | 7 | 8 | 0 | 1 | 3 |

a) A árvore tem 7 folhas.

b) O elemento de valor 0 é filho do elemento de valor 9.

c) Ambas estão corretas.

d) Nenhuma está correta.

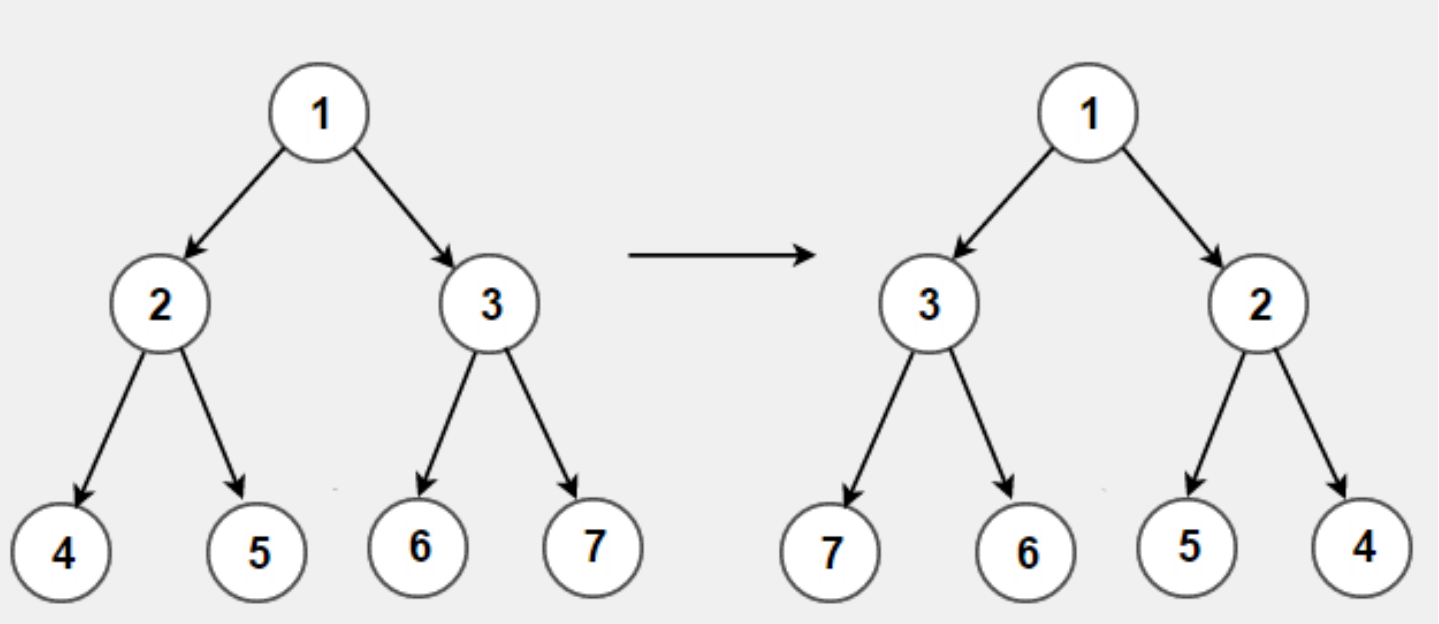
# Tarefa 1 – Número de nós que não são folhas

- Dado o ponteiro para o nó raiz de uma árvore binária
- Desenvolver uma função recursiva que devolva o número dos nós da árvore que não são folhas

- Não use quaisquer variáveis globais !!

# Tarefa 2 – Duas árvores são espelhadas ?

- Desenvolver uma função recursiva que devolva 1 ou 0, indicando se duas árvore dadas são ou não espelhadas




[techiedelight.com]

# Tarefa 3 – Item dado pertence a uma árvore ?

- Dado o ponteiro para o nó raiz de uma árvore binária e um dado item
- Desenvolver uma função recursiva que verifique se esse item pertence à árvore

- Não use quaisquer variáveis globais !!

# Tarefas Adicionais – Funções recursivas

- Determinar o valor do menor elemento
- Determinar o valor do maior elemento
- Devolver um ponteiro para o nó contendo o menor elemento
- Devolver um ponteiro para o nó contendo o maior elemento

- Não use quaisquer variáveis globais !!