

# Estrutura de dados

## Árvores

Prof<sup>a</sup>. Dr<sup>a</sup>. Alana Moraes

# Recapitulando - Estrutura de dados

## Dados simples:

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

## Dados estruturados:

- Estáticos:
  - arrays;
  - registros;
  - arquivos;
  - conjuntos;
  - cadeias.
- Dinâmicos:
  - filas;
  - pilhas;
  - listas encadeadas;
  - árvores;
  - grafos.

# Recapitulando - Estrutura de dados

## Dados simples:

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

## Dados estruturados:

- Estáticos:

- arrays;
- registros;
- arquivos;
- conjuntos;
- cadeias.



- Dinâmicos:

- filas;
- pilhas;
- listas encadeadas;
- árvores;
- grafos.



# Recapitulando - Estrutura de dados

## Dados simples:

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

## Dados estruturados:

- Estáticos:

- arrays;
- registros;
- arquivos;
- conjuntos;
- cadeias.



- Dinâmicos:

- filas;
- pilhas;
- listas encadeadas;
- árvores;
- grafos.



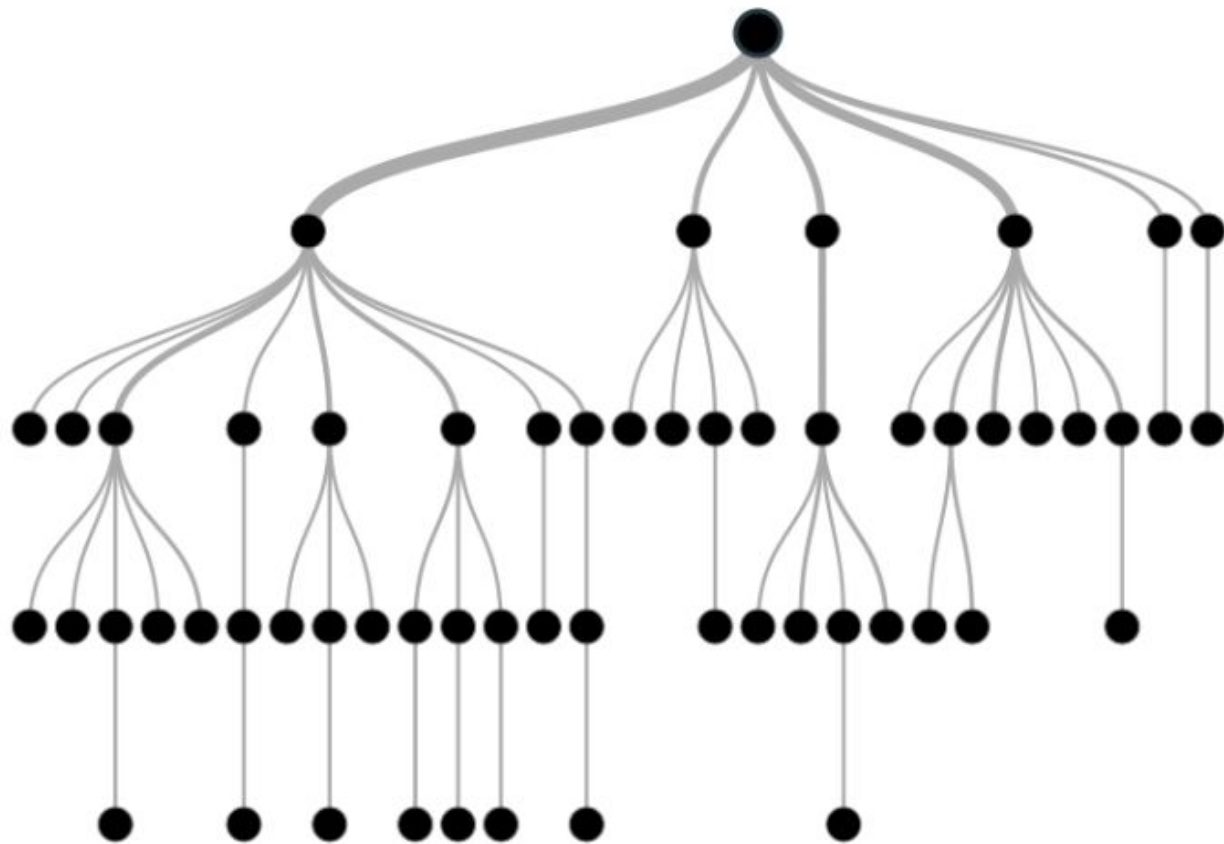
# Árvores

- As listas encadeadas usualmente fornecem maior flexibilidade que as matrizes, mas são estruturas lineares, e é difícil usá-las para organizar uma representação hierárquica de objetos.
- Embora pilhas e filas reflitam alguma hierarquia, são limitadas a somente uma dimensão.
- Para superar esta limitação, criamos um novo tipo de dados chamado **árvore**, que consiste em nós e arcos.

# Árvores



# Árvores



# Árvores

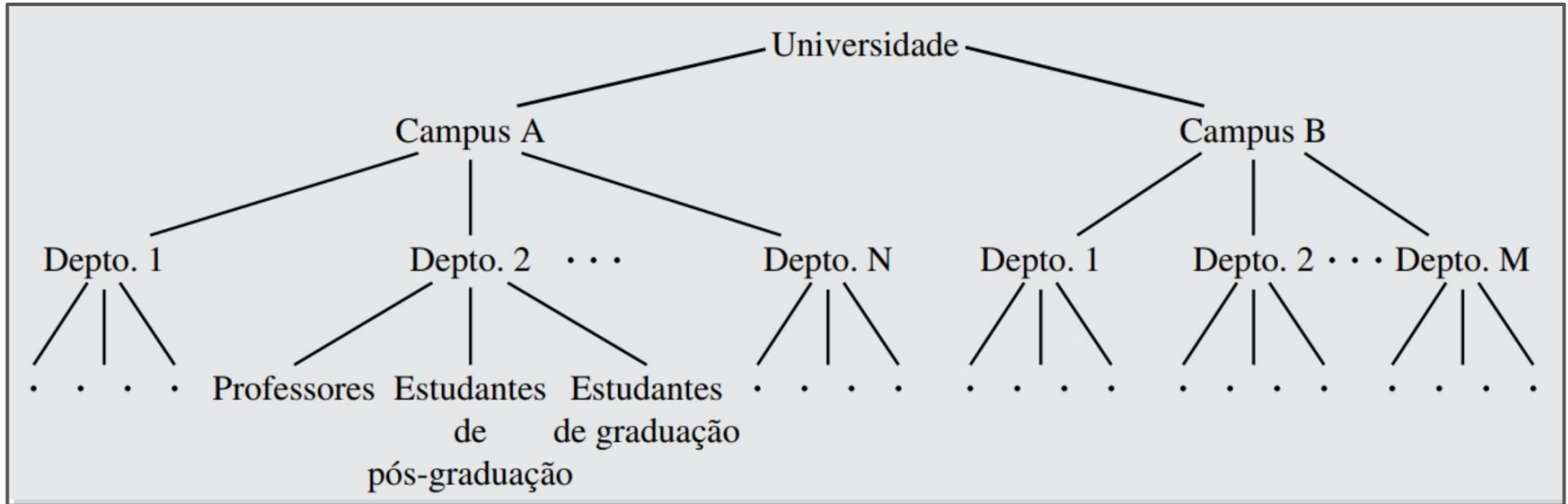
- Diferentemente de uma árvore natural, estas são representadas de cima para baixo, com a raiz no topo e as folhas na base (nodos terminais).
- A **raiz** é um nó que não tem ancestrais; só pode ter nós filhos.
- As **folhas**, por outro lado, não têm filhos, ou melhor, seus filhos são estruturas vazias.
- Uma árvore não impõe qualquer condição sobre o número de filhos de um nó.
- Há diversos tipos de árvores: binárias, AVL, hierárquicas, etc.



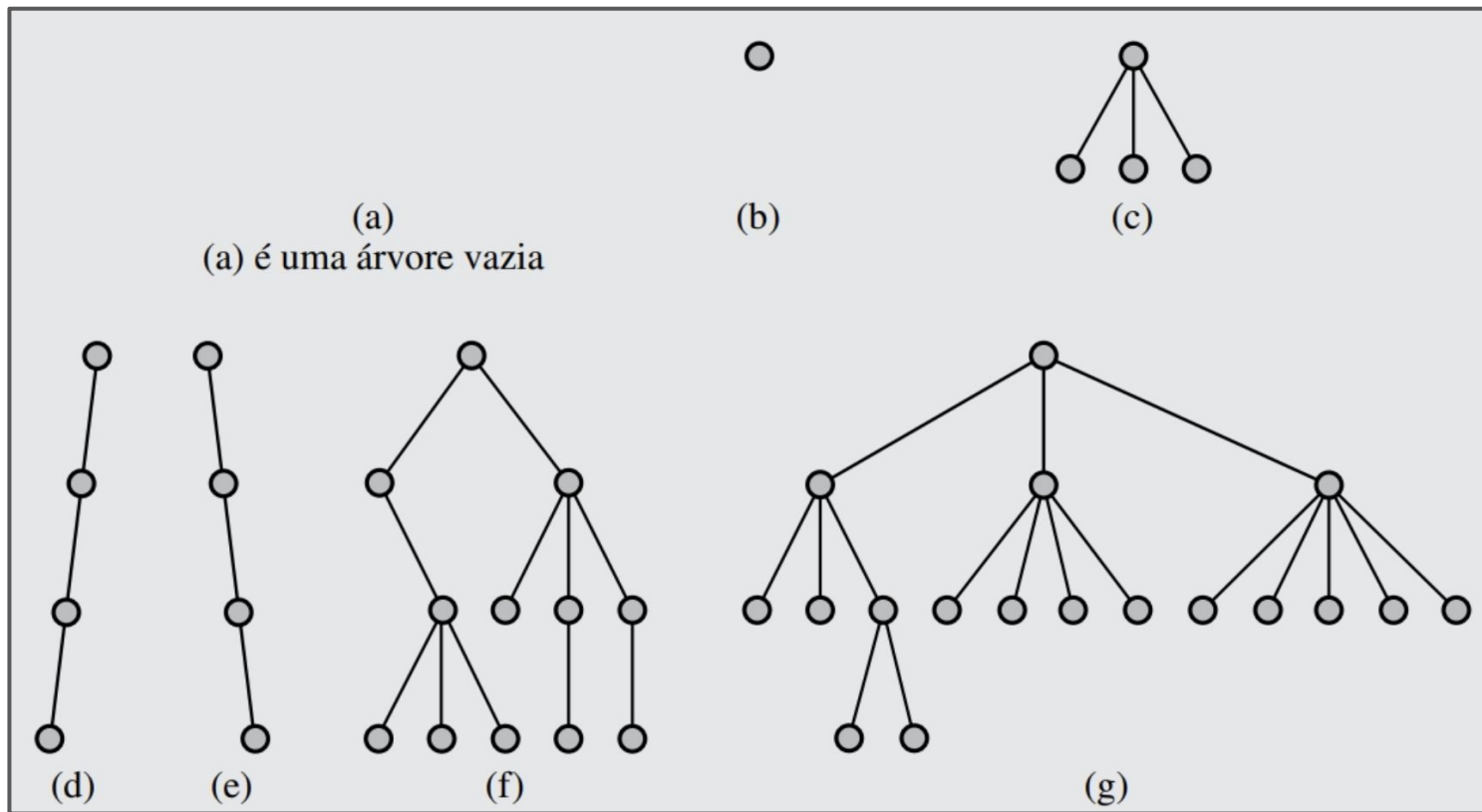
# Árvores

- Uma árvore pode ser definida recursivamente como:
  1. Uma estrutura vazia é uma árvore vazia.
  2. Se  $t_1, \dots, t_k$  são árvores disjuntas, então a estrutura cuja raiz tem como suas filhas as raízes de  $t_1, \dots, t_k$  também é uma árvore.
  3. Somente estruturas geradas pelas regras 1 e 2 são árvores.

# Árvores



# Árvore - Qual delas é uma árvore?

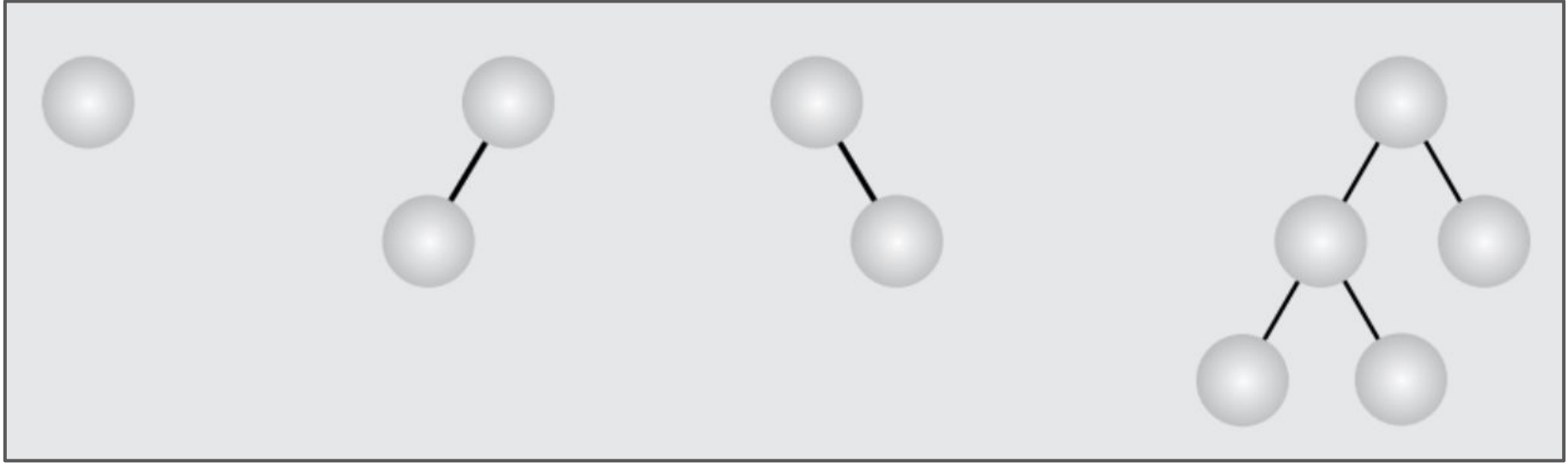


# Árvores Binárias

- Uma espécie comum de árvores é a árvore binária, em que cada nó contém referências a dois outros nós (possivelmente nulos).
- Tais referências são chamadas de subárvore esquerda e direita.
- Como os nós das listas ligadas, os nós das árvores também contém uma carga (valor de cada nó).

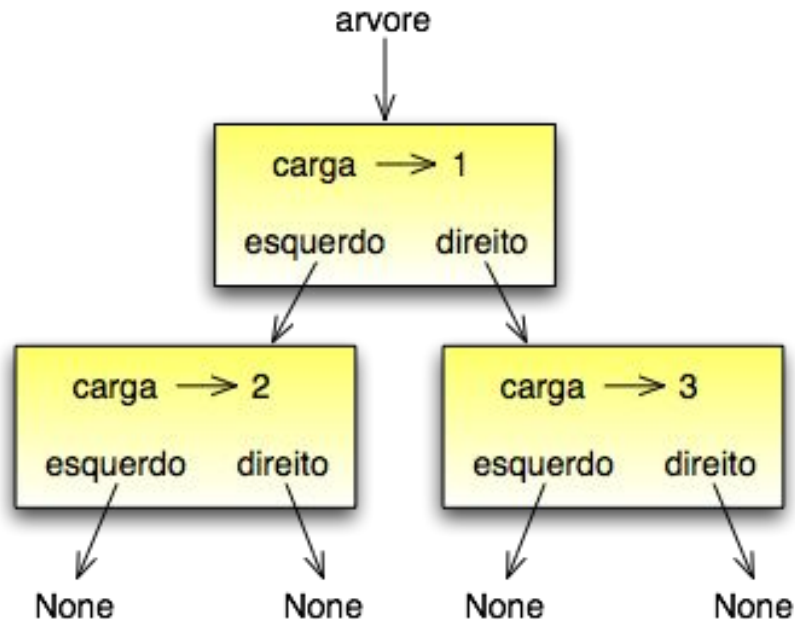
# Árvores Binárias

- Exemplos de Árvores Binárias



# Árvores Binárias

- Uma árvore binária pode aparecer assim em detalhes:



# Árvores Binárias

- O topo da árvore (nó ao qual o apontador *tree* se refere) é chamada de raiz.
- Seguindo a metáfora das árvores, os outros nós são chamados de galhos e os nós das pontas contendo as referências vazia são chamadas de folhas (ou elementos terminais).
- Pode parecer estranho que desenhamos a figura com a raiz em cima e as folhas em baixo, mas isto nem será a coisa mais estranha.

# Árvores Binárias de Busca

- Todos os nós pertencentes à subárvore esquerda de qualquer nó possuem valor menor que o valor do mesmo, e em que os nós da subárvore à sua direita possuem valor maior que o valor do nó em questão.
- Essa propriedade deve ser válida para todas as subárvores, possibilitando a realização de buscas mais eficientes, pois se pode comparar o valor procurado com o valor de um nó e decidir se continuar a busca somente na subárvore à esquerda ou à direita do nó, reduzindo assim a quantidade de nós a serem visitados na busca.



# Operações com árvores

- Construindo uma árvore;
- Inserindo ordenado;
- Buscando elementos na árvore;
- Calculando a altura de uma árvore;
- Percorrendo e Mostrando;

# Construindo uma Árvore Binária

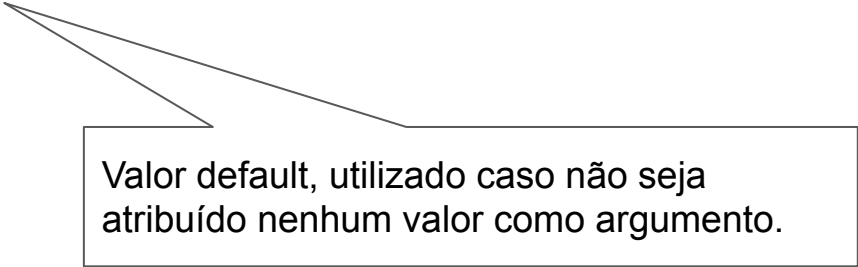
tree.py

```
class Arvore:
    def __init__(self, c, d = None, e = None):
        self.chave = c
        self.esquerda = e
        self.direita = d
```

# Construindo uma Árvore Binária

tree.py

```
class Arvore:
    def __init__(self, c, d = None, e = None):
        self.chave = c
        self.esquerda = e
        self.direita = d
```



Valor default, utilizado caso não seja atribuído nenhum valor como argumento.

# Construindo uma Árvore Binária

tree.py

```
class Arvore:
    def __init__(self, c, d = None, e = None):
        self.chave = c
        self.esquerda = e
        self.direita = d

noRaiz = Arvore(4) # Cria nó raiz da arvore (raiz)

noEsq = Arvore(2)
noDir = Arvore(6)

arvore = Arvore(4, noDir, noEsq)
```

# Construindo uma Árvore Binária

tree.py

```
class Arvore:
    def __init__(self, c, d = None, e = None):
        self.chave = c
        self.esquerda = e
        self.direita = d
```

```
arvore = Arvore(4, Arvore(6), Arvore(2))
```

Como está essa árvore?



# Exercício

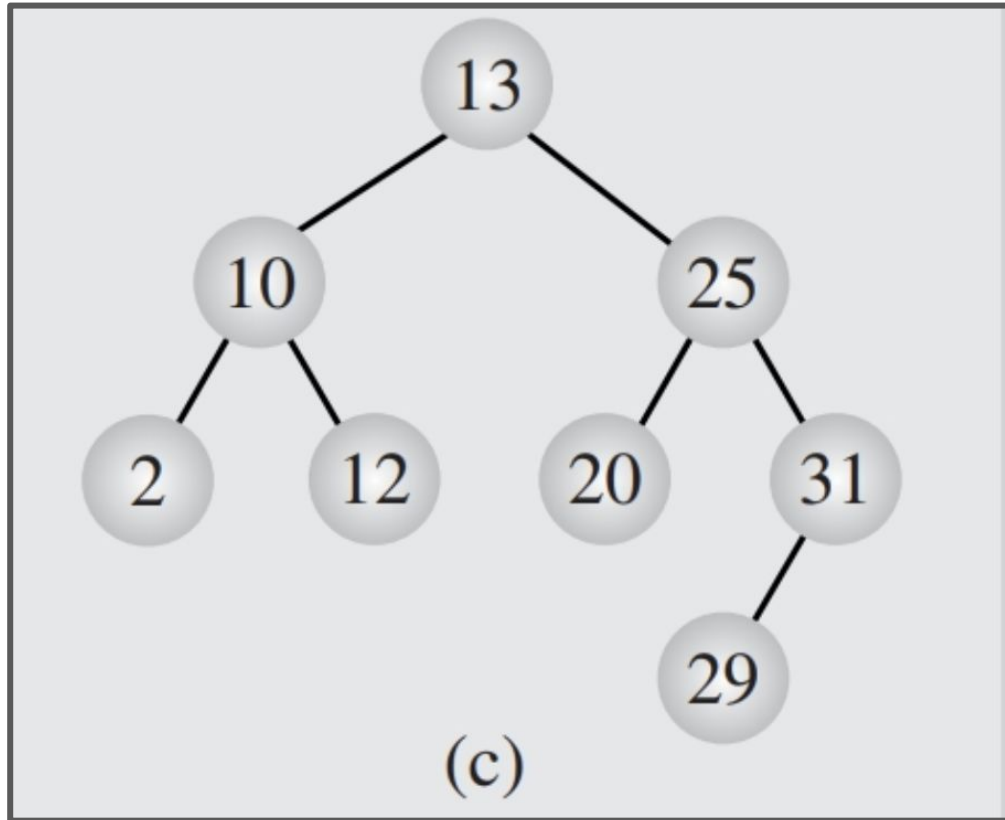
Crie uma árvore binária que armazene seu nome no nó raiz, no nó da direita sua matrícula e no nó da esquerda seu curso.

# Operações com árvores

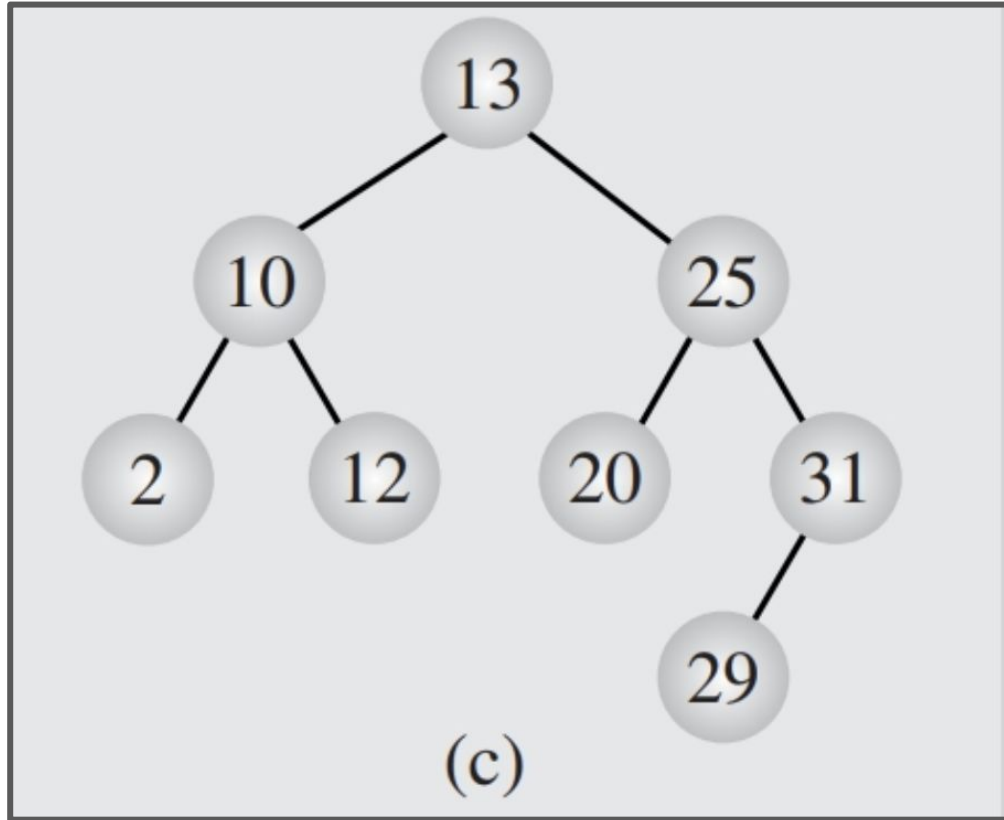
- Construindo uma árvore;
- **Inserindo ordenado;**
- Buscando elementos na árvore;
- Calculando a altura de uma árvore;
- Percorrendo e Mostrando;



# Inserindo ordenado



# Inserindo ordenado



E se eu  
quisesse  
adicionar o  
7? E o 13?

# Inserindo ordenado

tree.py

```
class Arvore:
    ...

def insere(no, chave):
    if no is None:
        no = Arvore(chave)
    else:
        if chave < no.chave:
            no.esquerda = insere(no.esquerda, chave)
        else:
            no.direita = insere(no.direita, chave)
    return no

...

arvore = Arvore(3, Arvore(1), Arvore(13) ) #Cria arvore (raiz)
arvore = insere(arvore, 2) #Insere Valores
```

# Exercício

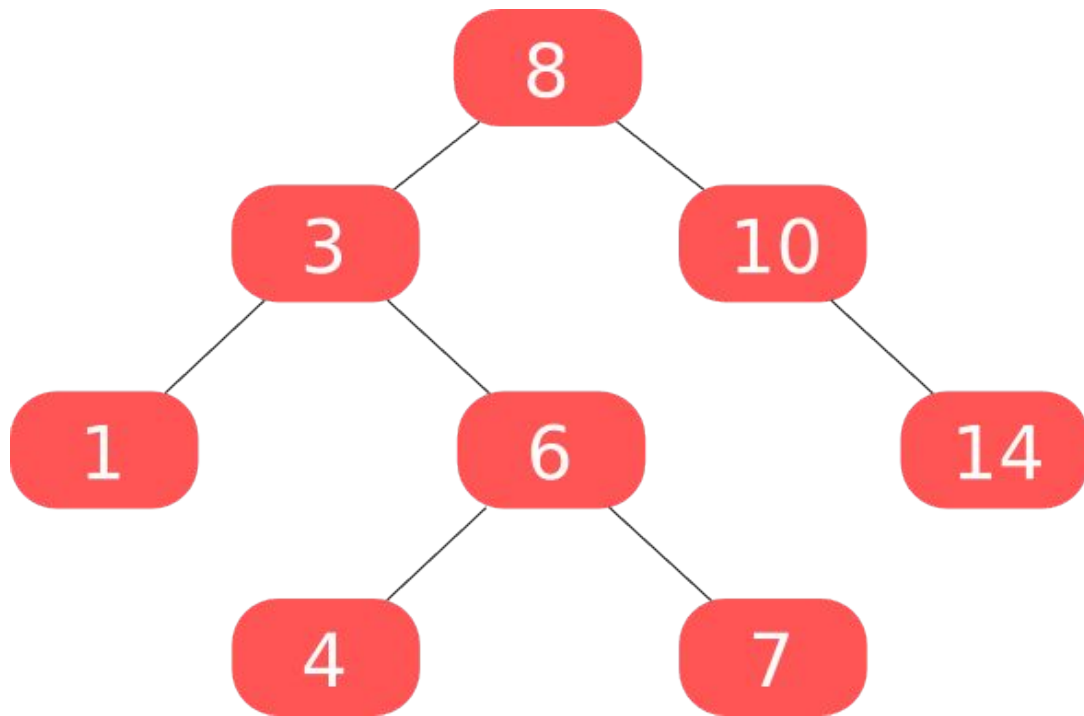
Repita o exemplo anterior e adicione os números: 2,4,6,8,5,7 e 0.

# Operações com árvores

- Construindo uma árvore;
- Inserindo ordenado;
- **Buscando elementos na árvore;**
- Calculando a altura de uma árvore;
- Percorrendo e Mostrando;

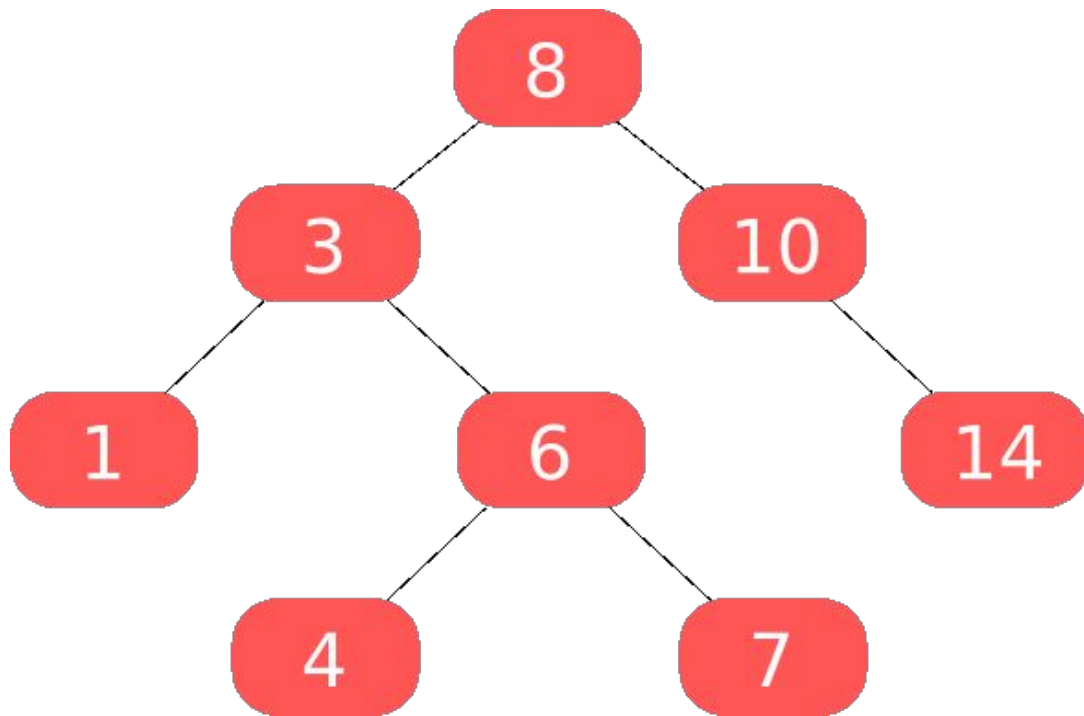
# Buscando elementos na árvore

- Por exemplo, buscando o valor 4:
- Duas alternativas de implementação:
  - Busca linear
  - Busca recursiva



# Buscando elementos na árvore

- Por exemplo, buscando o valor 4:
- Duas alternativas de implementação:
  - Busca linear
  - Busca recursiva



# Busca linear

tree.py

```
class Arvore:
    ...

def buscaLinear(no, chave):
    while no is not None:
        if no.chave == chave:
            return no
        elif chave > no.chave:
            no = no.direita
        else:
            no = no.esquerda
    return None

...

if buscaLinear(arvore, 6) is not None: # Retorna o NO ou None se nao encontrou
    print 'valor encontrado\n'
else:
    print 'valor nao encontrado\n'
```



# Busca recursiva

tree.py

```
class Arvore:
    ...
def buscaRecursiva(no, chave):
    if no is None:
        print str(chave) + ' nao foi encontrado na arvore\n'
        return None
    if no.chave == chave:
        print str(chave) + ' foi encontrado na arvore\n'
        return no
    if chave > no.chave:
        buscaRecursiva(no.direita, chave)
    else:
        buscaRecursiva(no.esquerda, chave)
```

# Busca recursiva

tree.py

```
class Arvore:
    ...

def buscaRecursiva(no, chave):
    ...

...

buscaRecursiva(arvore, 6) #Busca que imprime na propria funcao
```

# Exercício

Façam uma busca pelos elementos 21, 2, 12 e 0.

Utilizando as duas estratégias.

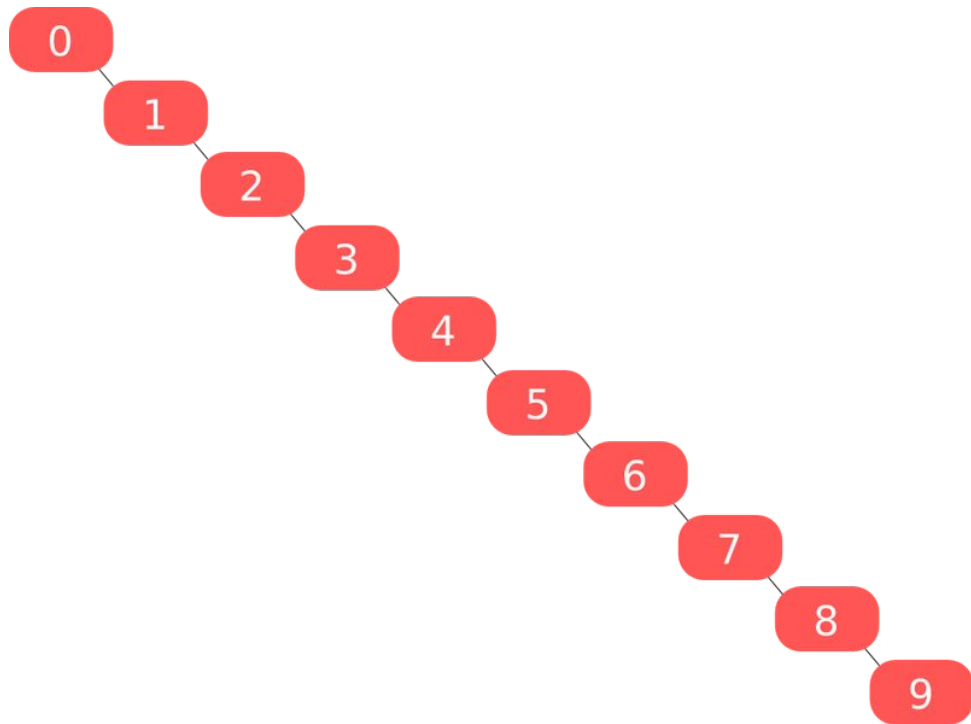
# Operações com árvores

- Construindo uma árvore;
- Inserindo ordenado;
- Buscando elementos na árvore;
- **Calculando a altura de uma árvore;**
- Percorrendo e Mostrando;

# Altura de Árvores

Vocês acham que essa é uma árvore possível?

Ela está equilibrada?



# Altura de Árvores

- O conceito de altura de uma árvore é definido pela quantidade de arestas no **caminho mais longo entre a raiz e as folhas**.
- Os nós de uma árvore binária possuem graus zero, um ou dois.
- Um nó de grau zero é denominado folha.
- A profundidade de um nó é a distância deste nó até a raiz. Um conjunto de nós com a mesma profundidade é denominado nível da árvore.
- **A maior profundidade de um nó, é a altura da árvore.**

# Altura de uma Árvore

tree.py

```
class Arvore:
    ...
def maximo(a, b):
    if a > b:
        return a
    return b

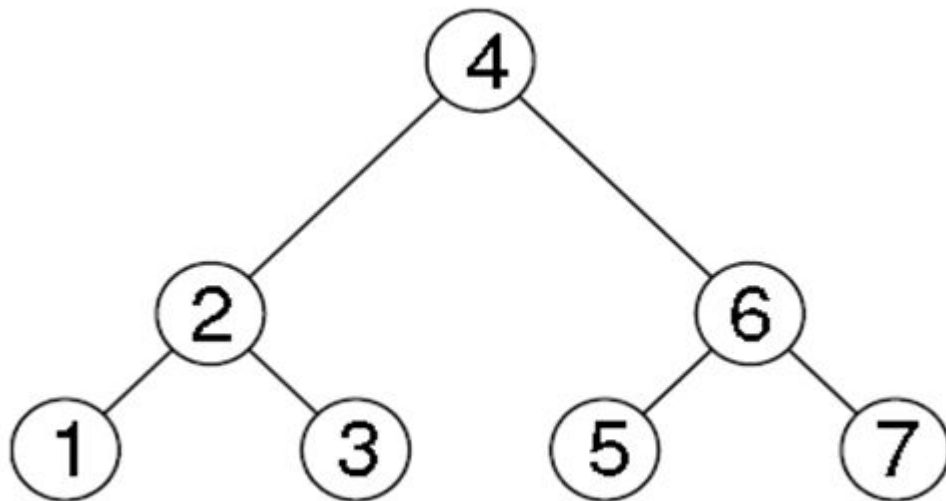
def altura(no):
    if no is None:
        return 0
    return 1 + maximo( altura(no.esquerda), altura(no.direita) )

...

print 'Altura : %d' % altura(arvore) # Retorna a altura da arvore (int)
```

# Exercício

Crie a seguinte árvore binária e calcule sua altura





# Operações com árvores

- Construindo uma árvore;
- Inserindo ordenado;
- Buscando elementos na árvore;
- Calculando a altura de uma árvore;
- **Percorrendo e Mostrando; - Próxima Aula**

# Dúvidas?



[alanamm.prof@gmail.com](mailto:alanamm.prof@gmail.com)