# Estrutura de Dados

## Árvore binária – Parte 01

Prof. Dr. Daniel Vecchiato

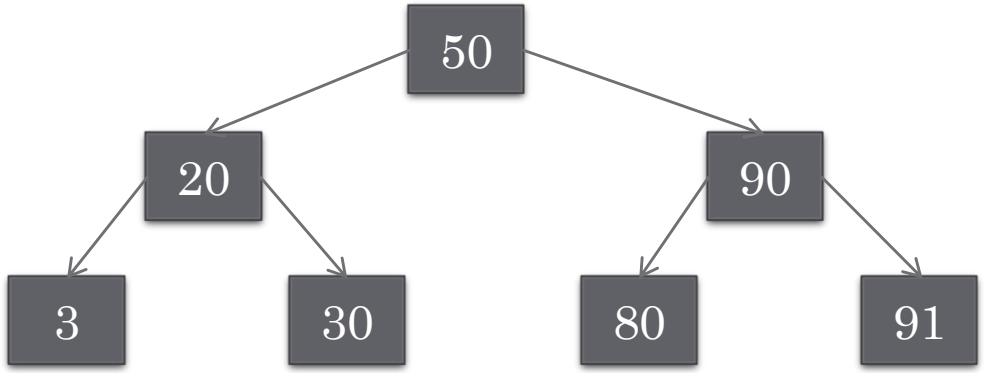
# Agenda

- Introdução
- Árvore binária
  - Definição
  - Classificação
  - Implementação básica
- Percorrendo a árvore binária
- Discussão
- Exercícios

# Introdução

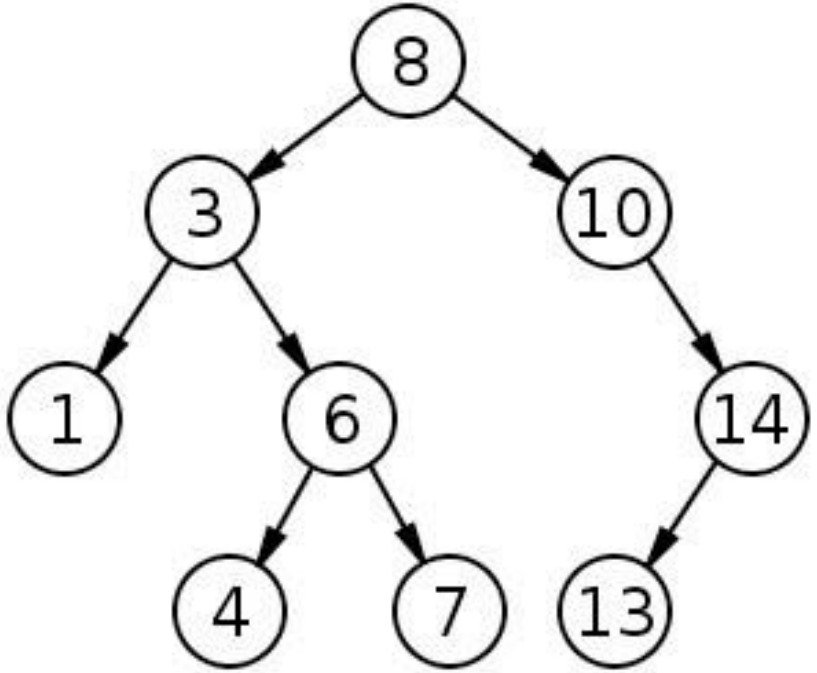
- Os mesmos dados podem ser representados/armazenados de formas diferentes


- Até então, todo o conteúdo foi estudado usando algo semelhante à estrutura de cima
- Hoje vamos começar a estudar a estrutura de baixo



# Introdução

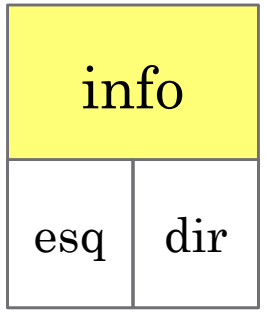
- Estrutura de dados: árvore




- Como você acha que deve ser a implementação da estrutura de uma árvore?

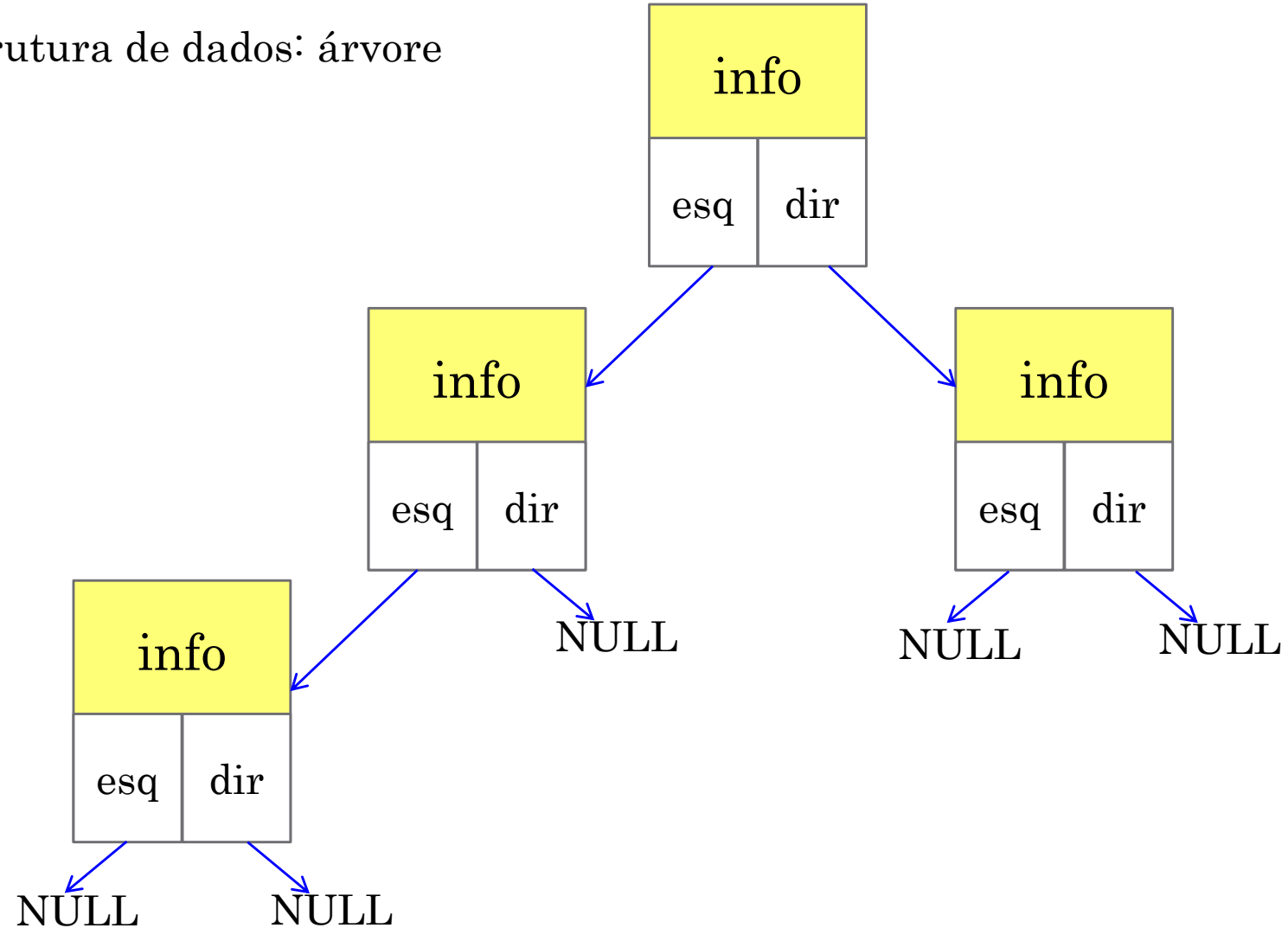
# Introdução

- Estrutura de dados: árvore



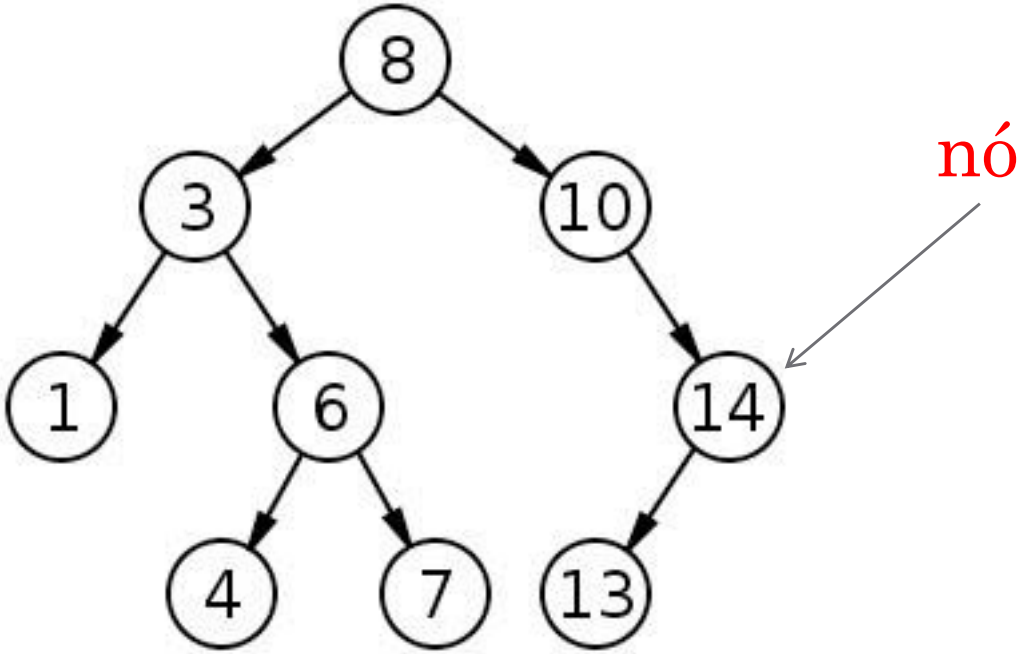
# Introdução

- Estrutura de dados: árvore



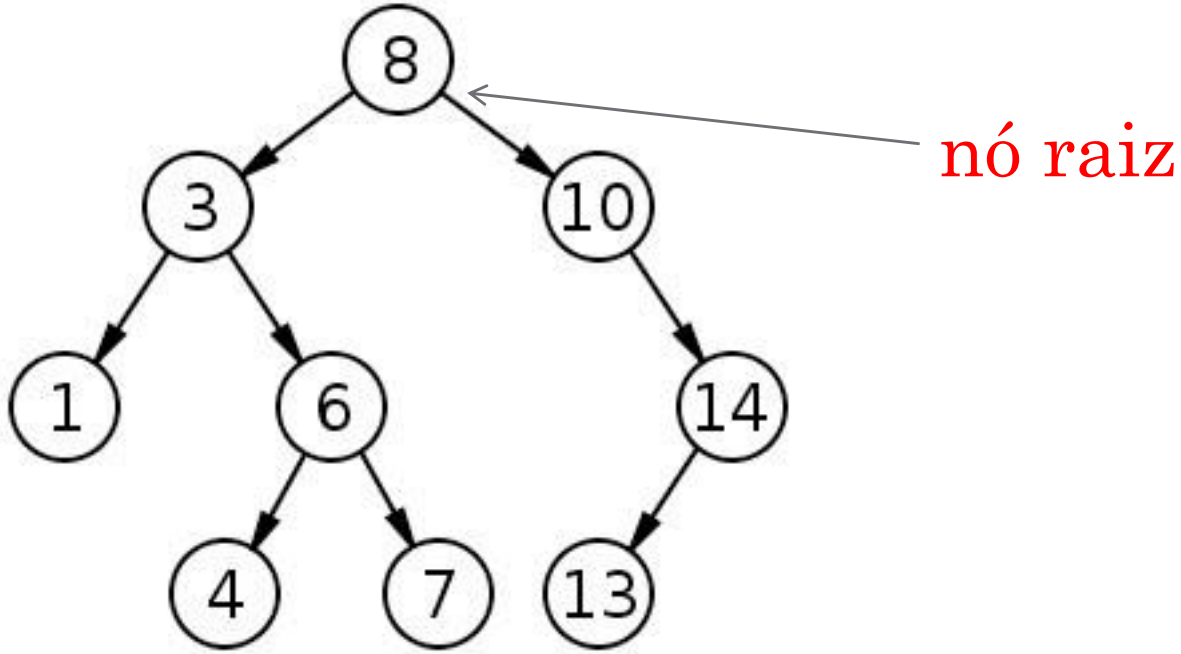
# Introdução

- Estrutura de dados: árvore
  - Termos
    - Nó
      - Item armazenado em uma árvore
      - É o equivalente às células nas listas encadeadas



# Introdução

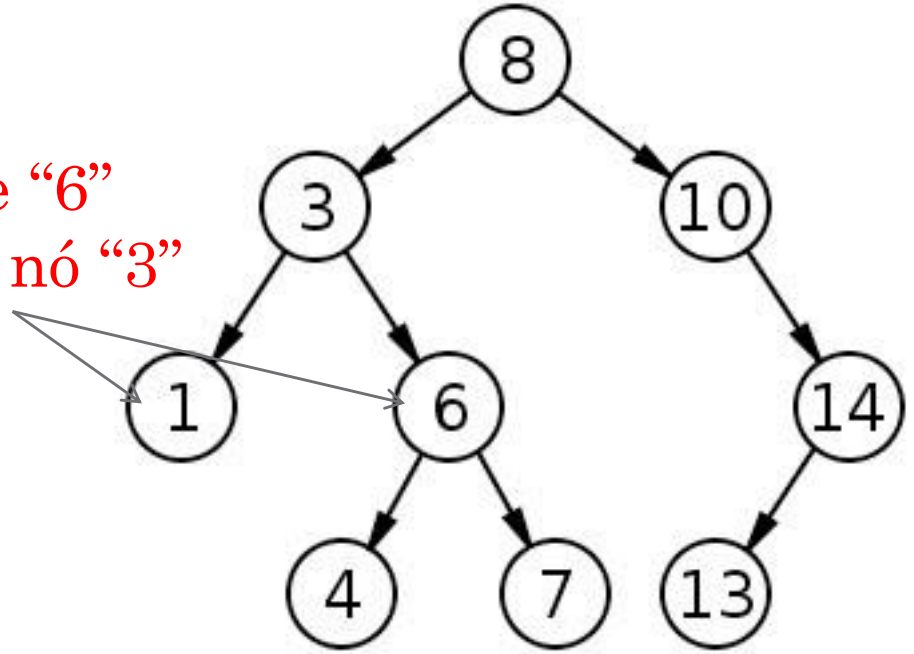
- Estrutura de dados: árvore
  - Termos
    - Raiz
      - Nó do início da árvore
      - Ponto de partida para acessar os outros nós da árvore



# Introdução

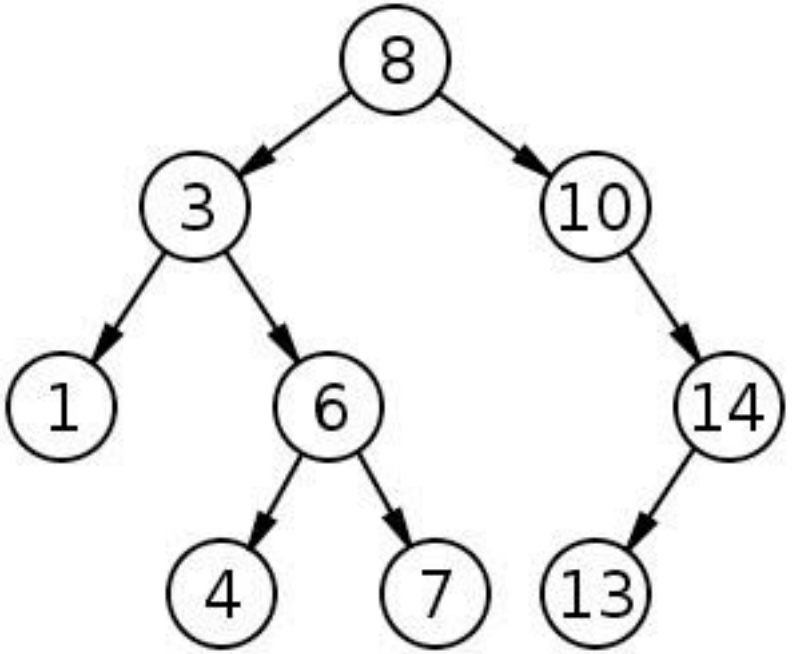
- Estrutura de dados: árvore
  - Termos
    - Filho
      - Nó diretamente abaixo de um determinado nó
      - Ideia semelhante pode ser aplicada ao termo “nó pai”

os nós “1” e “6”
são filhos do nó “3”



# Introdução

- Estrutura de dados: árvore
  - Termos
    - Folha
      - Um nó que não tem filho
      - São os últimos itens da árvore

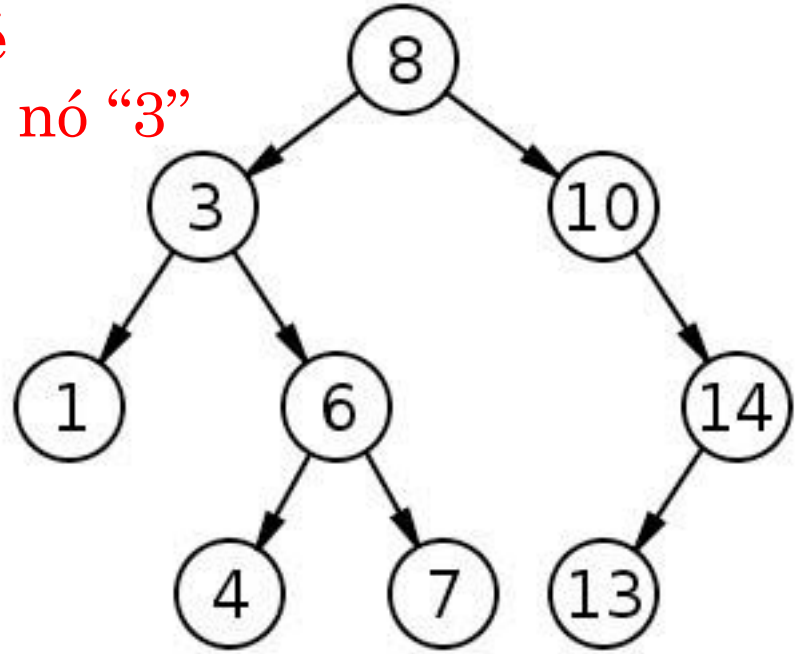



"1", "4", "7" e "13" são nós folhas

# Introdução

- Estrutura de dados: árvore
  - Termos
    - Descendente
      - Nó que pode ser alcançado a partir de determinado nó
      - Um nó pode ser alcançado efetuando vários `x = x->esq`, por exemplo

O nó “7” é descendente do nó “3”



# Introdução

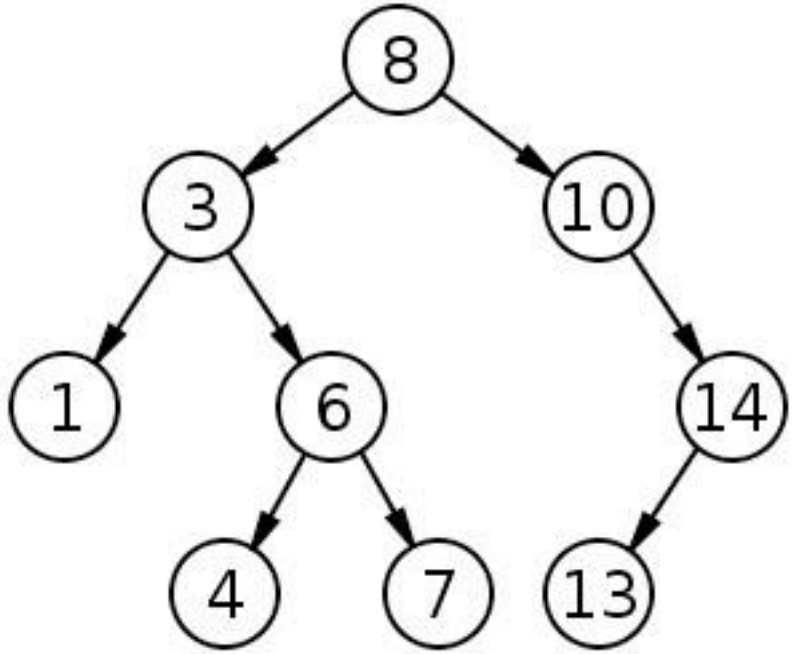
- Estrutura de dados: árvore
  - Termos
    - Nível
      - Conjunto de nós que estão na mesma distância da raiz
      - Ideia semelhante pode ser aplicado ao termo "altura"

Nível 0

Nível 1

Nível 2

Nível 3

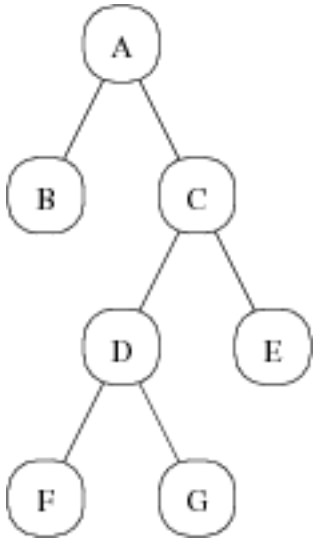


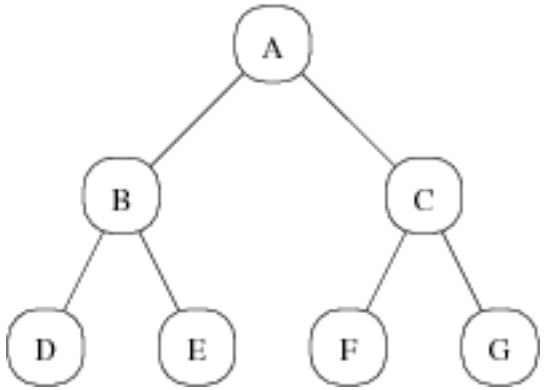
# Árvore binária

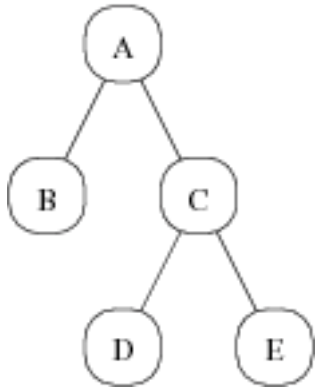
- Definição
  - Tipo de árvore que possui chave e 2 ponteiros para subárvores
    - além de outras informações associadas a respectiva chave
  - Em um nó com chave X:
    - As chaves na subárvore da **esquerda são menores** que X
    - As chaves na subárvore do **direita são maiores** que X

# Árvore binária

- Classificação
  - Estritamente binária
    - Todo nó tem zero ou dois nós como filhos

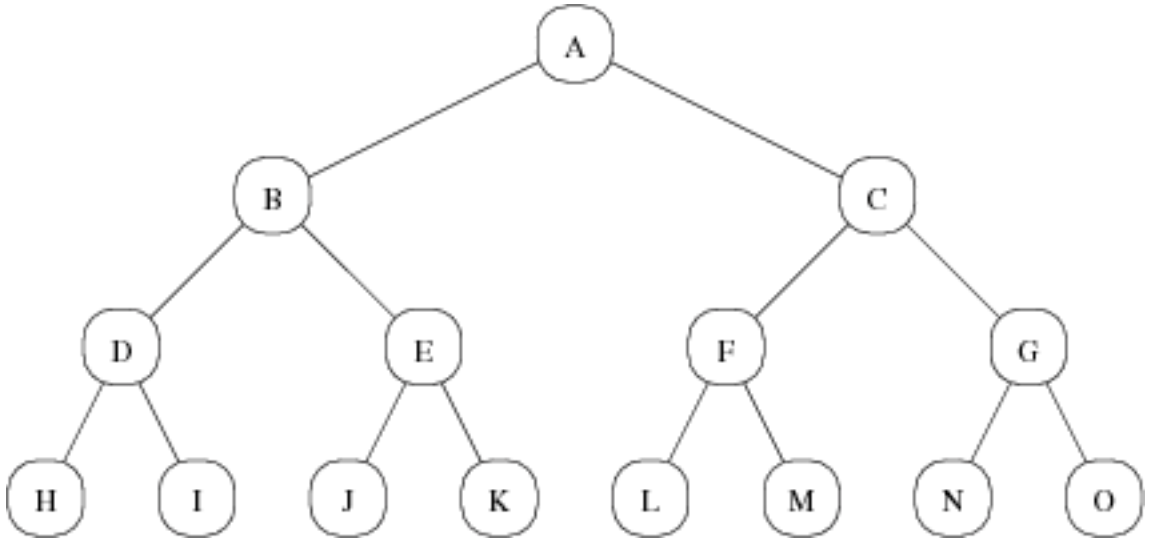


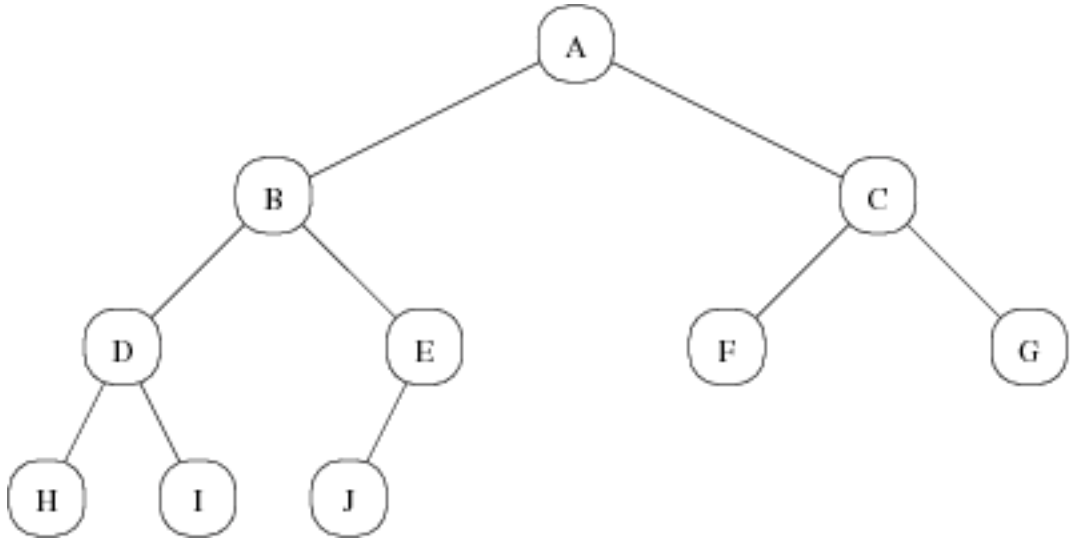
# Árvore binária

- Classificação
  - Completa
    - Todos os níveis tem o número máximo de elementos
    - Todos os nós folhas estão no nível n
    - Nesse caso, pode ser dito que “árvore binária completa de nível n”



# Árvore binária

- Classificação
  - Quase completa
    - Cada nó folha está no nível **n** ou no nível **n-1**



# Árvore binária

- Implementação básica

```
typedef struct {
	int chave;
	int informacao;
} TItem;


typedef struct No {
	TItem item;
	struct No *pEsq, *pDir;
} TNo;
```

# Árvore binária

- Implementação básica

```
TNo* criarNo (TItem x) {
        TNo *pAux = (TNo*) malloc (sizeof(TNo));
        pAux->item = x;
        pAux->pEsq = NULL;
        pAux->pDir = NULL;
        return pAux;
}

TNo* inserirNo (TNo *pR, TItem x) {
        if (pR == NULL)
                pR = criarNo (x);
        else
                if (x.chave < pR->item.chave)
                        pR->pEsq = inserirNo (pR->pEsq, x);
                else
                        pR->pDir = inserirNo (pR->pDir, x);
        return pR;
}
```

# Árvore binária

- Implementação básica

```
TItem* pesquisa (TNo* pR, int chave) {
	if (pR == NULL)
		return NULL;

	if (chave < pR->item.chave)
		return pesquisa (pR->pEsq, chave);

	if (chave > pR->item.chave)
		return pesquisa (pR->pDir, chave);

	return &(pR->item);
}
```

# Percorrendo a árvore binária

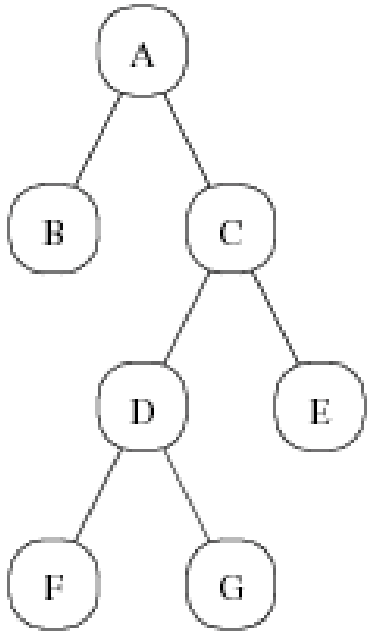
- Percorrer: passar por todos os nós da árvore, pelo menos 1 vez
- Como a árvore não é uma lista sequencial, não há uma ordem pré-definida de como deve percorrer a árvore
- Há três maneiras principais
  - Pré-ordem
  - Em ordem
  - Pós-ordem
- Elas se diferencia na ordem em que 3 operação são executadas
  - Obter informações do nó atual
  - Percorrer a subárvore esquerda do nó atual
  - Percorrer a subárvore direita do nó atual

# Percorrendo a árvore binária

- Pré-ordem
  - Também chamada “profundidade” ou “pré-fixa”
  - Ordem das operações
    - Obter informações do nó atual
    - Percorrer a subárvore esquerda em pré-ordem
    - Percorrer a subárvore direita em pré-ordem

```
void preOrdem (TNo *p) {
     if (p == NULL)
             return;
     printf ("%d\n", p->item.chave);
     preOrdem (p->pEsq);
     preOrdem (p->pDir);
}
```

  - Em que ordem estes nós seriam visitados?

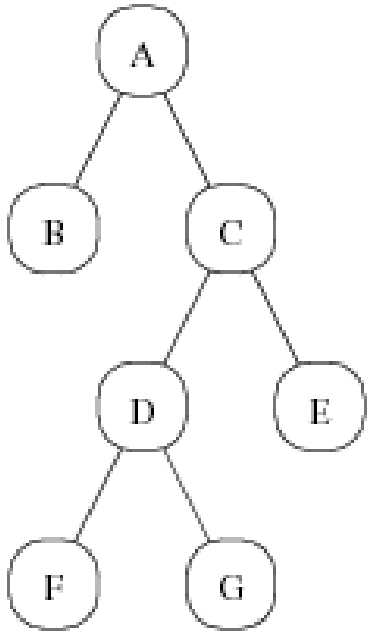


# Percorrendo a árvore binária

- Pré-ordem
  - Também chamada “profundidade” ou “pré-fixa”
  - Ordem das operações
    - Obter informações do nó atual
    - Percorrer a subárvore esquerda em pré-ordem
    - Percorrer a subárvore direita em pré-ordem

```
void preOrdem (TNo *p) {
     if (p == NULL)
             return;
     printf ("%d\n", p->item.chave);
     preOrdem (p->pEsq);
     preOrdem (p->pDir);
}
```

  - Em que ordem estes nós seriam visitados?
    - A B C D F G E

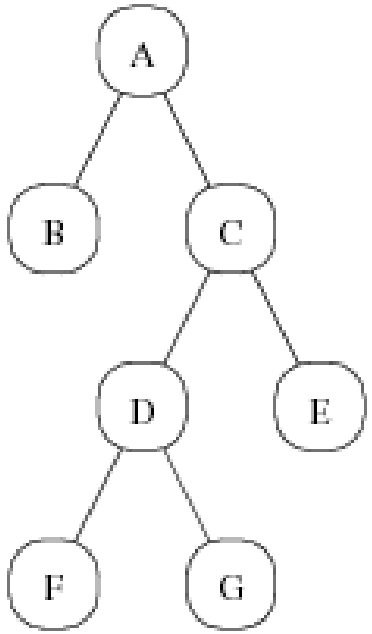


# Percorrendo a árvore binária

- Em ordem
  - Também chamada “central” ou “ordem simétrica”
  - Ordem das operações
    - Percorrer a subárvore esquerda em pré-ordem
    - Obter informações do nó atual
    - Percorrer a subárvore direita em pré-ordem

```
void emOrdem (TNo *p) {
     if (p == NULL)
             return;
     emOrdem (p->pEsq);
     printf ("%d\n", p->item.chave);
     emOrdem (p->pDir);
}
```

  - Em que ordem estes nós seriam visitados?

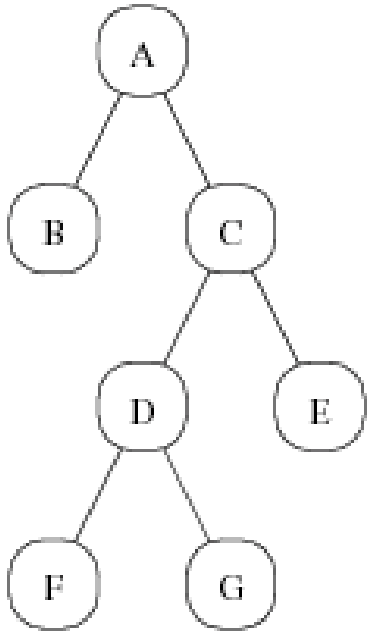


# Percorrendo a árvore binária

- Em ordem
  - Também chamada “central” ou “ordem simétrica”
  - Ordem das operações
    - Percorrer a subárvore esquerda em pré-ordem
    - Obter informações do nó atual
    - Percorrer a subárvore direita em pré-ordem

```
void emOrdem (TNo *p) {
     if (p == NULL)
             return;
     emOrdem (p->pEsq);
     printf ("%d\n", p->item.chave);
     emOrdem (p->pDir);
}
```

  - Em que ordem estes nós seriam visitados?
    - B A F D G C E

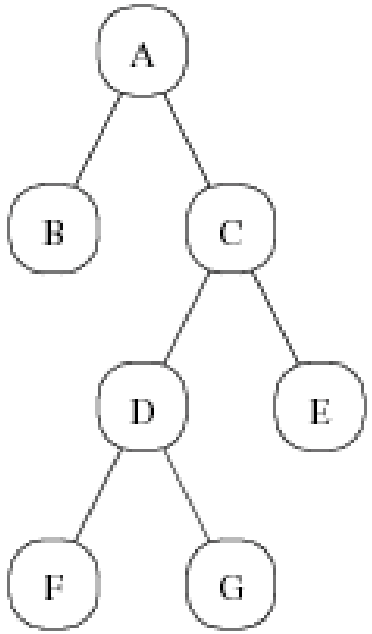


# Percorrendo a árvore binária

- Pós-ordem
  - Também chamada “pós-fixa”
  - Ordem das operações
    - Percorrer a subárvore esquerda em pós-ordem
    - Percorrer a subárvore direita em pós-ordem
    - Obter informações do nó atual

```
void posOrdem (TNo *p) {
     if (p == NULL)
             return;
     posOrdem (p->pEsq);
     posOrdem (p->pDir);
     printf ("%d\n", p->item.chave);
}
```

  - Em que ordem estes nós seriam visitados?

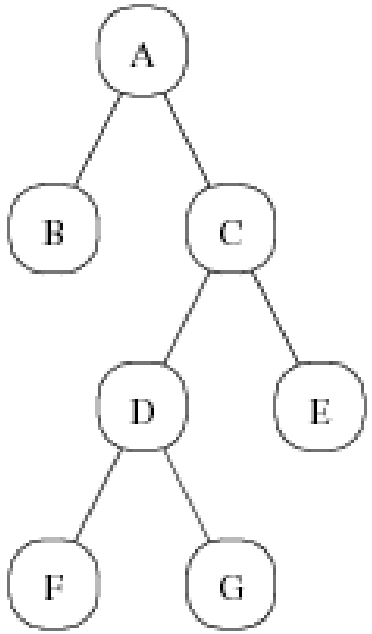


# Percorrendo a árvore binária

- Pós-ordem
  - Também chamada “pós-fixa”
  - Ordem das operações
    - Percorrer a subárvore esquerda em pós-ordem
    - Percorrer a subárvore direita em pós-ordem
    - Obter informações do nó atual

```
void posOrdem (TNo *p) {
     if (p == NULL)
             return;
     posOrdem (p->pEsq);
     posOrdem (p->pDir);
     printf ("%d\n", p->item.chave);
}
```

  - Em que ordem estes nós seriam visitados?
    - B F G D E C A



# Discussão

- Como a árvore deve ficar para ter o melhor desempenho nas pesquisas?

# Discussão

- Como a árvore deve ficar para ter o melhor desempenho nas pesquisas?
  - Balanceada

# Discussão

- E como ela deve ficar para ter o pior desempenho?
- Como deve estar o conjunto de dados para que aconteça isso?

# Discussão

- E como ela deve ficar para ter o pior desempenho?
- Como deve estar o conjunto de dados para que aconteça isso?

  - Semelhante a uma lista encadeada
  - Ordenado

# Discussão

- Quais os custos para inserção, pesquisa e obter os valores de forma ordenada?

# Discussão

- Quais os custos para inserção, pesquisa e obter os valores de forma ordenada?

  - Inserção e pesquisa
    - O(log n) para caso médio
    - O(n) para pior caso
  - Obter os registros em ordem: O(n)

# Exercícios

- Desenhe uma árvore binária após a inserção destes elementos:
  - 10, 20, 5, 8, 12, 22, 23, 24, 11, 13, 18
- Como seria mostrado os elementos percorrendo a árvore “em ordem”
- Podemos classificar esta árvore como estritamente binária, completa ou quase completa?
- Qual a raiz da árvore? Quantos nós existem? Quantos níveis? Qual a altura do nó 12?
- Implemente o algoritmo da árvore binária
- Faça duas funções
  - uma para encontrar o menor valor da árvore
  - e outra função para retornar o maior valor

# Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)
- Demaine, E., Devadas, S. Introduction to Algorithms (MIT OpenCourseWare), http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011
- http://www.ft.unicamp.br/liag/siteEd/