



Conteúdo

1. Introdução
2. Listas
3. Pilhas e Filas
4. Árvores
5. Árvores de Pesquisa
 - Árvore Binária e Árvore AVL
 - Árvore N-ária e Árvore B
6. Tabelas de Dispersão (Hashing)
7. Métodos de Acesso a Arquivos
8. Métodos de Ordenação de Dados



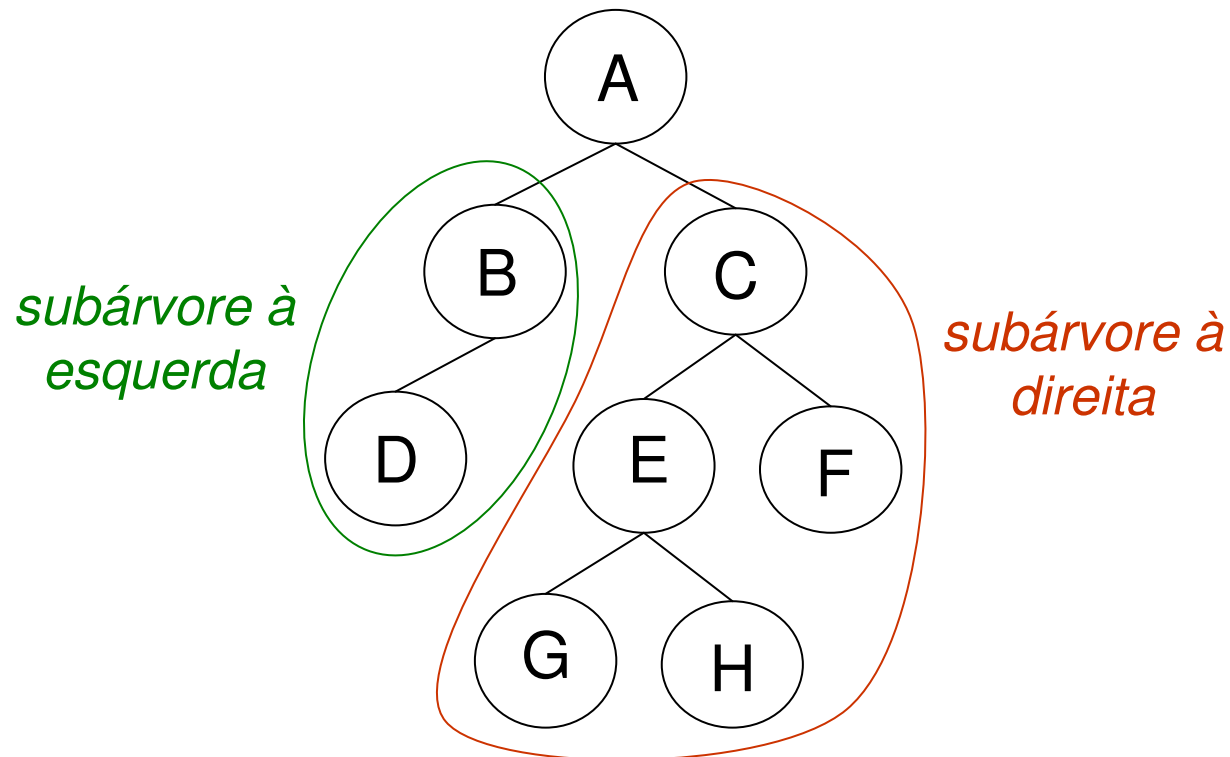


Árvores Binárias



Árvore Binária

- Uma **árvore binária** é uma árvore **N-ária** com $N = 2$.
- Distingue-se entre uma subárvore esquerda e uma direita.





Árvore Binária

Uma **árvore binária** é uma árvore ordenada com as seguintes propriedades:

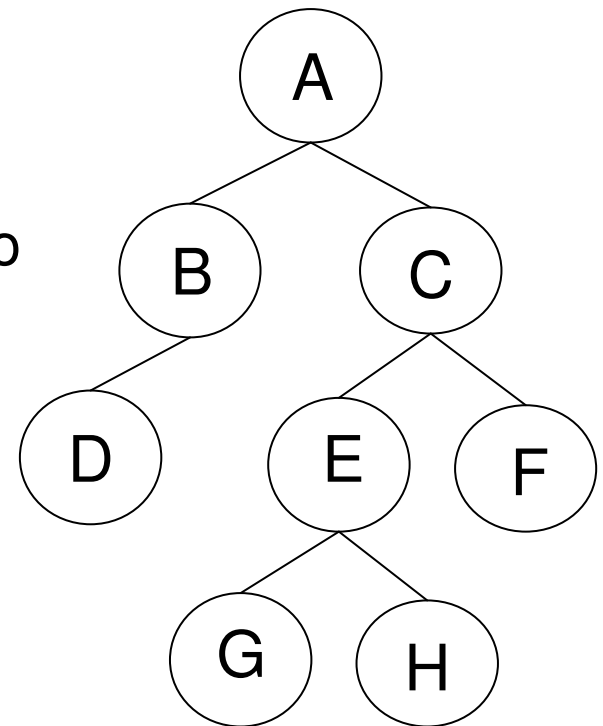
- Todos os nodos tem no máximo dois filhos.
- Cada nodo filho é rotulado como sendo um filho da direita ou um filho da esquerda.
- O filho da esquerda precede o filho da direita na ordenação dos filhos de um nodo.



Propriedades

As propriedades definidas para árvores N-árias também são aplicadas às árvores binárias:

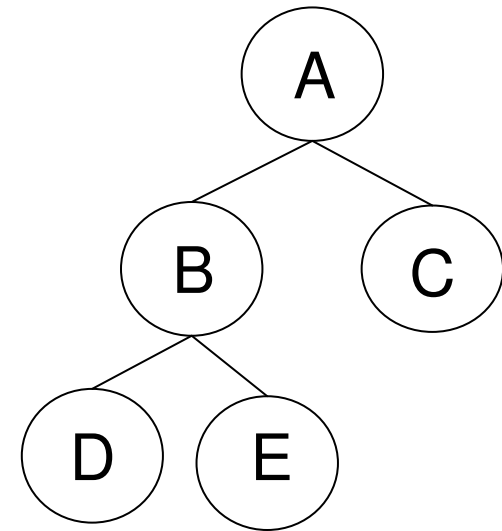
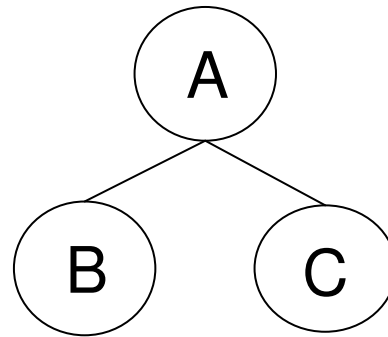
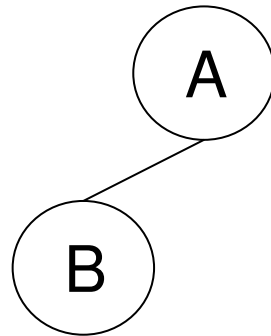
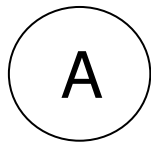
- Grau de um Nodo
- Nodo Folha e Nodo Interno
- Caminho e Comprimento de um Caminho
- Profundidade de um Nodo
- Nodo Pai e Nodo Filho
- Nodo Irmão
- Nodo Ancestral e Descendente
- Altura de uma Árvore





Propriedade

- “O número máximo de nodos em uma árvore binária A com altura $h(A)$ é $2^{h(A)+1}-1$ ”.



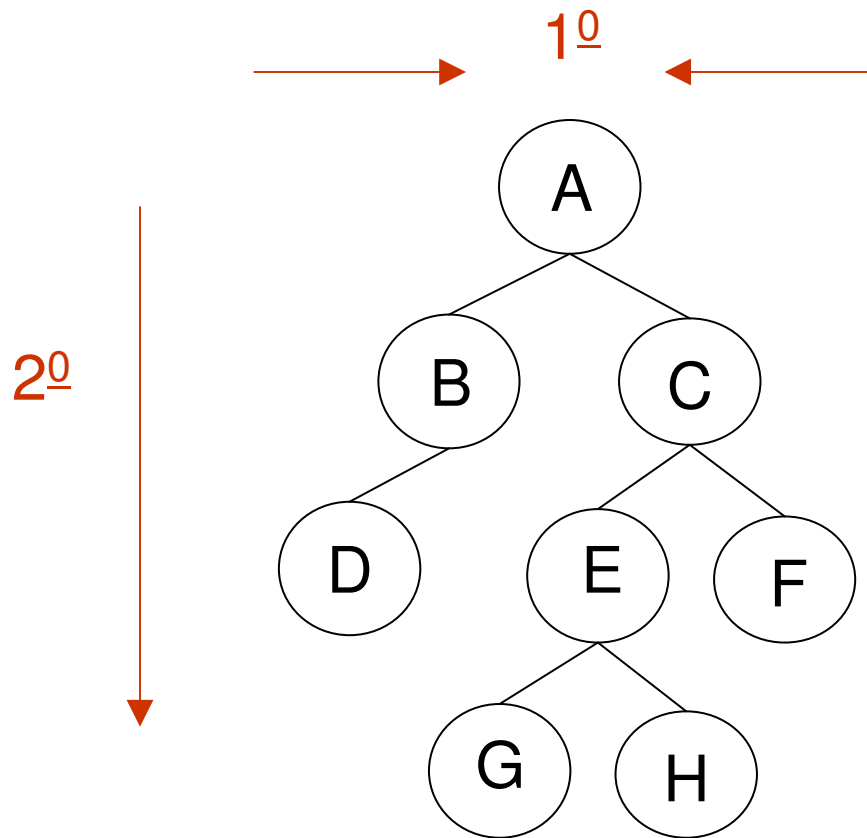


Caminhamento em Árvores Binárias



Busca em Largura

- Percorre a árvore por **ordem de profundidade**

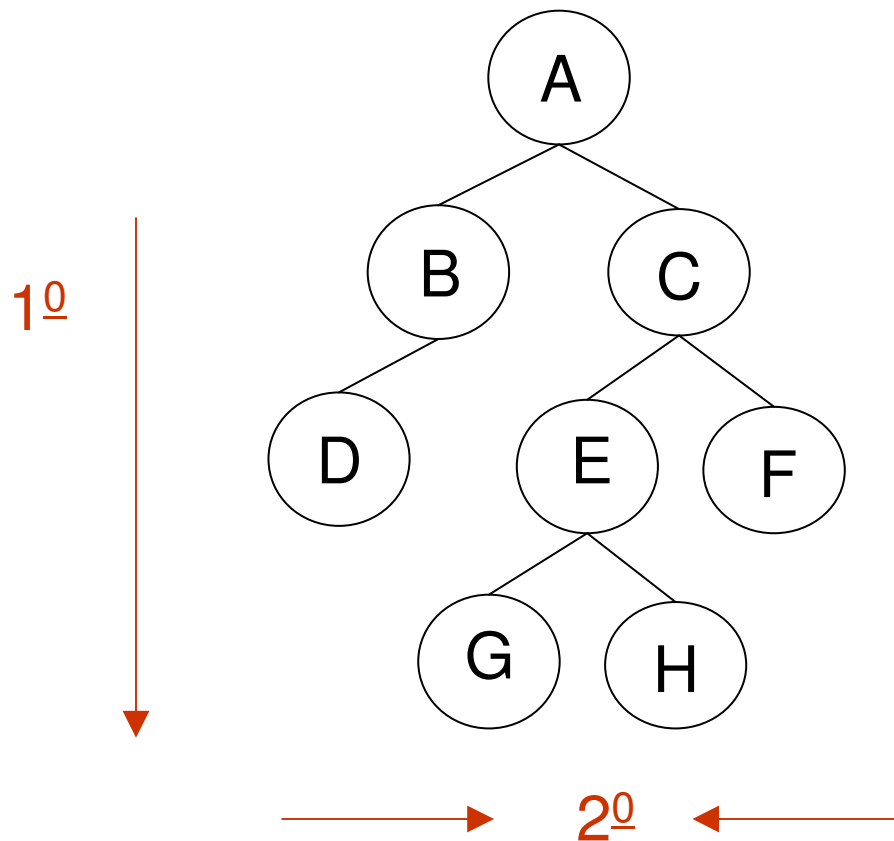


busca pela ESQ:
A-B-C-D-E-F-G-H

busca pela DIR:
A-C-B-F-E-D-H-G

Busca em Profundidade

- Percorre a árvore por **ordem de sub-árvore** (recursivamente)





Busca em Profundidade

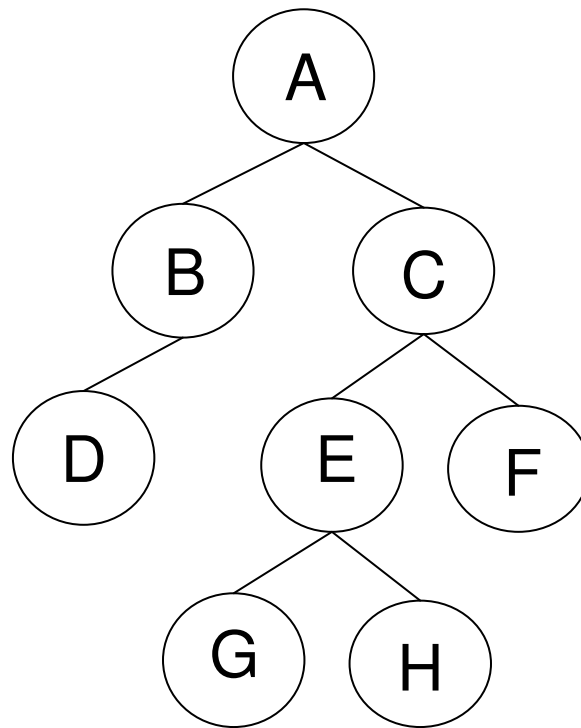
- Tipos de pesquisa em uma busca em profundidade:
 - pré-ordem (ou pré-fixada)
 - pós-ordem (ou pós-fixada)
 - in-ordem (ou in-fixada ou central)



Pré-Ordem em Árvore Binária

Passos:

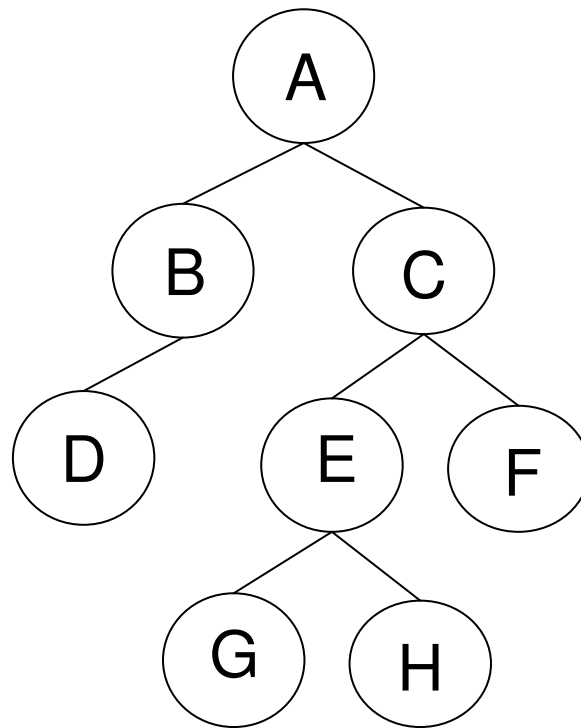
- Visita o nodo raiz
- Pesquisa em pré-ordem a subárvore à ESQ
- Pesquisa em pré-ordem a subárvore à DIR



Pré-Ordem em Árvore Binária

Passos:

- Visita o nodo raiz
- Pesquisa em pré-ordem a subárvore à ESQ
- Pesquisa em pré-ordem a subárvore à DIR

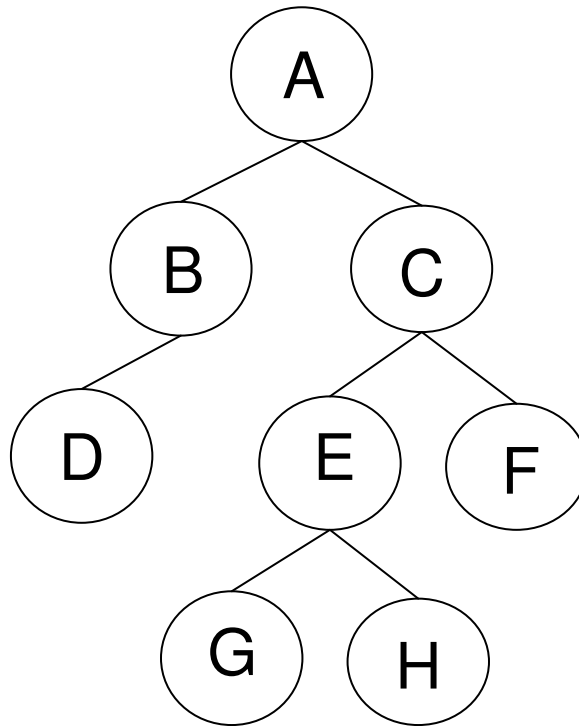


pré-ordem:
A-B-D-C-E-G-H-F

Pós-Ordem em Árvore Binária

Passos:

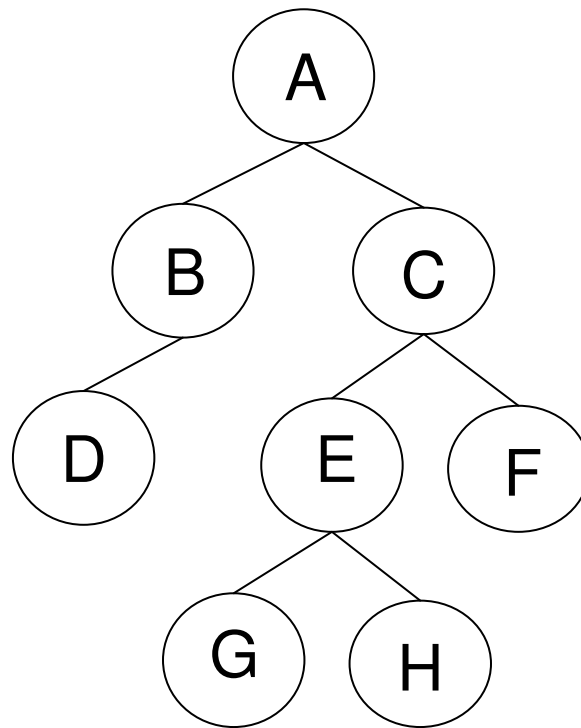
- Pesquisa em pós-ordem a subárvore à ESQ
- Pesquisa em pós-ordem a subárvore à DIR
- Visita o nodo raiz



Pós-Ordem em Árvore Binária

Passos:

- Pesquisa em pós-ordem a subárvore à ESQ
- Pesquisa em pós-ordem a subárvore à DIR
- Visita o nodo raiz

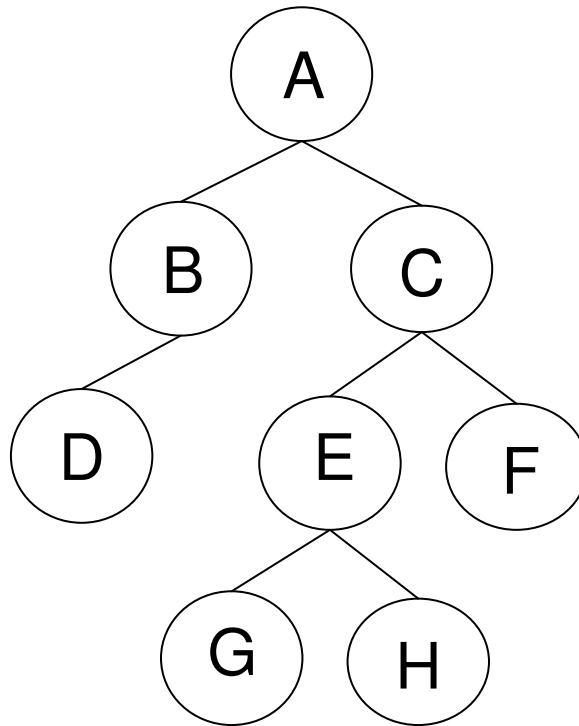


pós-ordem:
D-B-G-H-E-F-C-A

In-Ordem em Árvore Binária

Passos:

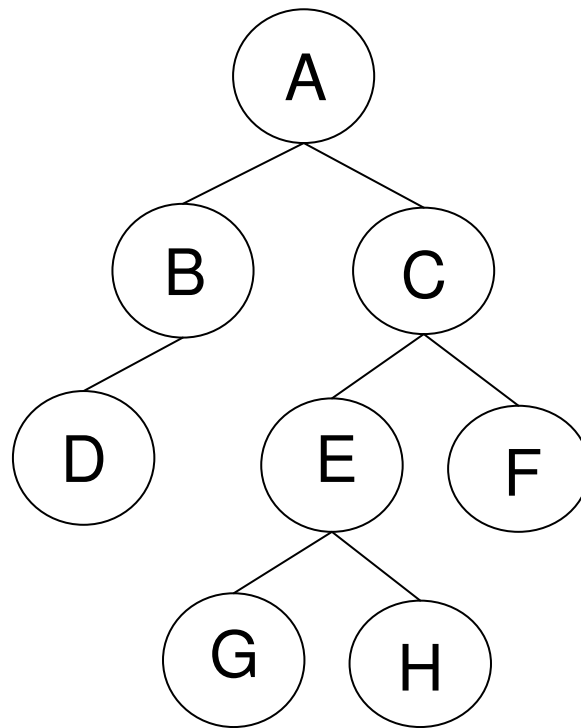
- Pesquisa em in-ordem a subárvore à ESQ
- Visita o nodo raiz
- Pesquisa em in-ordem a subárvore à DIR



In-Ordem em Árvore Binária

Passos:

- Pesquisa em in-ordem a subárvore à ESQ
- Visita o nodo raiz
- Pesquisa em in-ordem a subárvore à DIR

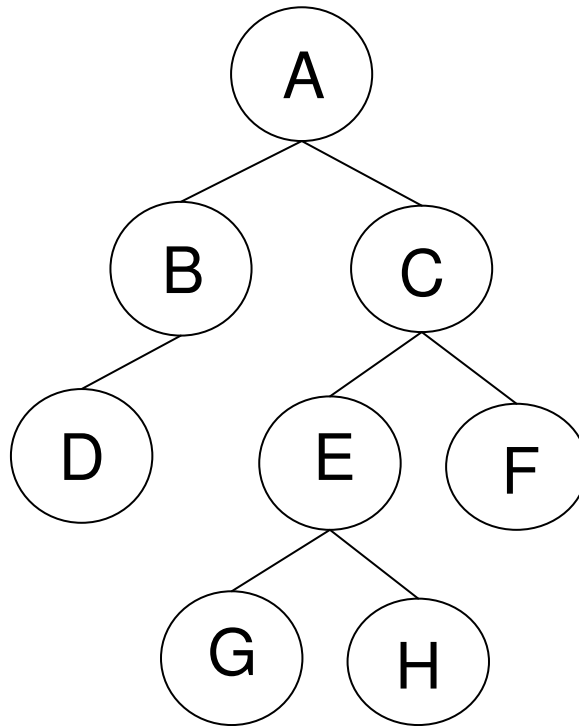


in-ordem:
D-B-A-G-E-H-C-F

Modelagem Física de Árvores Binárias

➡ Alternativas de implementação:

- Encadeamento
- Array



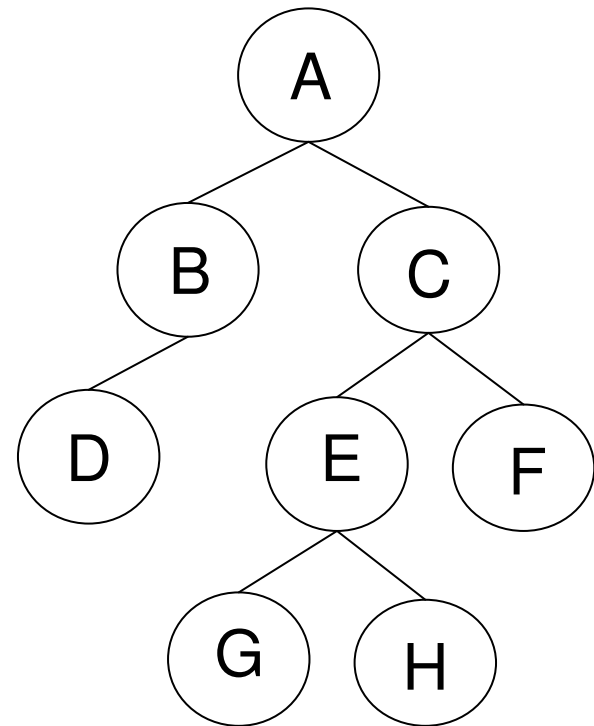
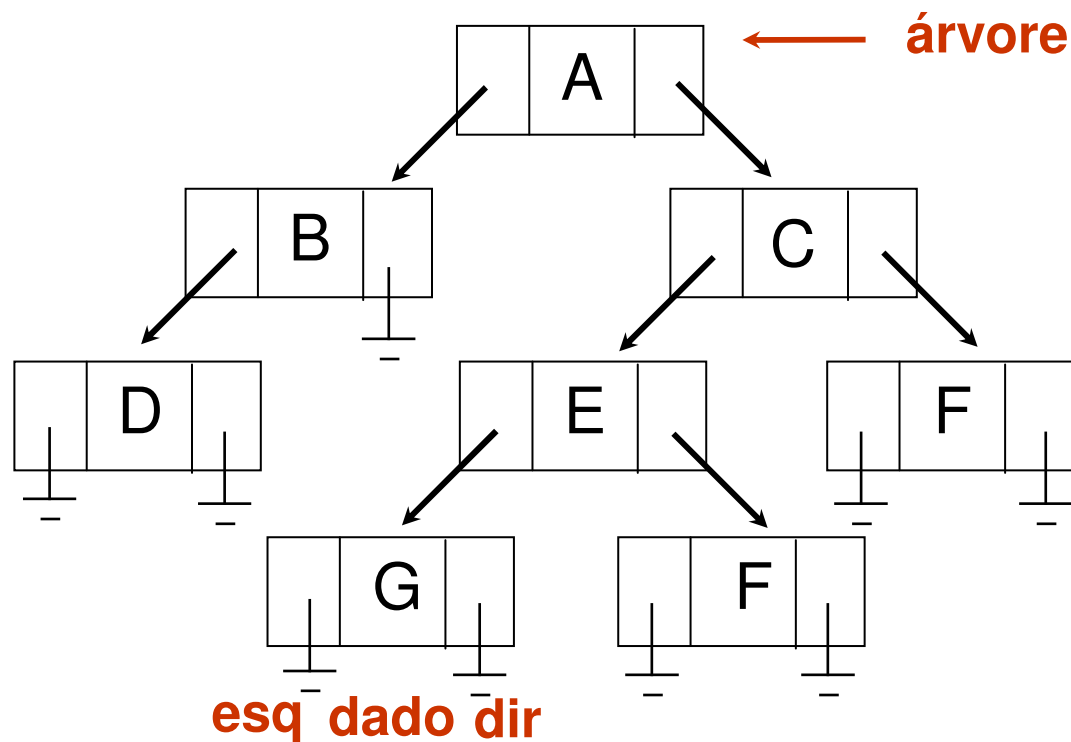


Árvore Binária com Encadeamento



Árvore Binária com Encadeamento

- Alternativa 1:





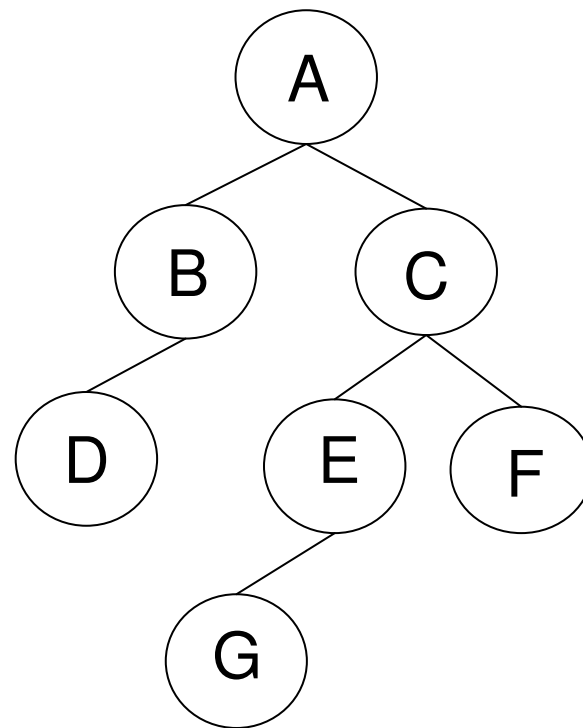
Árvore Binária com Array



Árvore com Array

Alternativa 1: (nodo, número de filhos)

A	2	B	1	D	0	C	2	E	1	G	0	F	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---



Árvore com Array

Alternativa 1: (nodo, número de filhos)

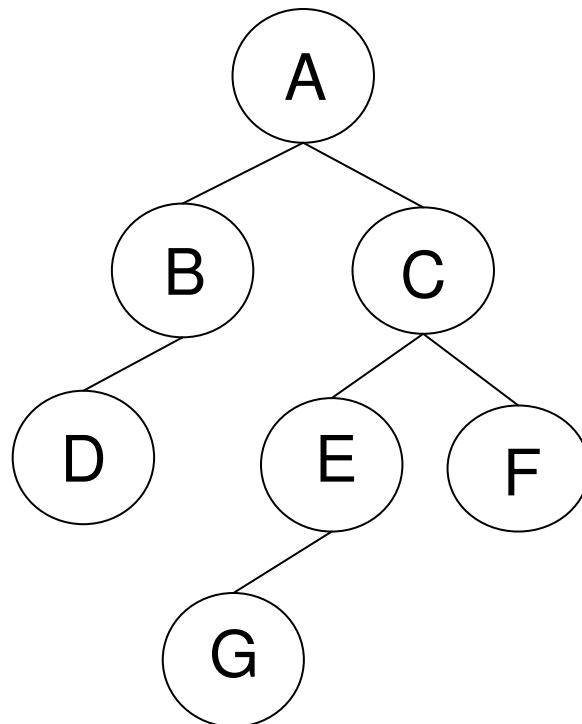
A	2	B	1	D	0	C	2	E	1	G	0	F	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---

- Desvantagens:
 - dado adicional (número filhos)
 - inserção / remoção:
 - achar a posição correta para inserir/remover um nodo
 - deslocamento
 - consultar o pais de um nodo
 - consultar os filhos de um nodo

Árvore com Array

Alternativa 2: (nodo, posição do nodo pai)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
A	-1	B	0	D	2	C	0	E	6	G	8	F	6



Árvore com Array

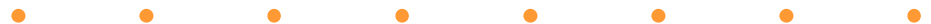
Alternativa 2: (nodo, posição do nodo pai)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
A	-1	B	0	D	2	C	0	E	6	G	8	F	6

- Desvantagens:
 - dado adicional (posição pai)
 - inserção:
 - buscar pai para descobrir o valor da sua posição
 - inserção / exclusão:
 - deslocamento e atualização de referências aos nodos pais no vetor
 - consultar os filhos de um nodo



Interface da Árvore Binária





Interface da Árvore Binária



```
public interface ArvoreBinaria<E> extends ArvoreGeral<E>{  
  
    public E removeDaRaiz () throws ExcecaoOperacaoInvalida;  
    public ArvoreBinaria<E> retornaEsquerda ();  
    public ArvoreBinaria<E> retornaDireita ();  
    public boolean temEsquerda ();  
    public boolean temDireita ();  
    public int retornaAltura ();  
    public Iterator<ArvoreGeral<E>> retornaSubArvoresEmInOrdem();  
    public Iterator<E> retornaElementosEmInOrdem();  
}
```





Interface da Árvore Binária

```
public interface ArvoreGeral<E> extends EstruturaDeDados {  
  
    public void atribuiRaiz (E elemento);  
    public E retornaRaiz();  
    public boolean contem (E elemento);  
    public ArvoreGeral<E> retornaPai (E elemento) throws  
        ExcecaoElementoInexistente;  
    public boolean temFilhos ();  
    public int numeroFilhos ();  
    public Lista<ArvoreGeral<E>> retornaFilhosDaRaiz ();  
  
    ...  
}
```





Interface da Árvore Binária

continuação da Árvore Geral

...

```
public Iterator<ArvoreGeral<E>> retornaSubArvoresEmLargura();  
public Iterator<ArvoreGeral<E>> retornaSubArvoresEmPreOrdem();  
public Iterator<ArvoreGeral<E>> retornaSubArvoresEmPosOrdem();  
public Iterator<E> retornaElementosEmLargura();  
public Iterator<E> retornaElementosEmPreOrdem();  
public Iterator<E> retornaElementosEmPosOrdem();  
}
```





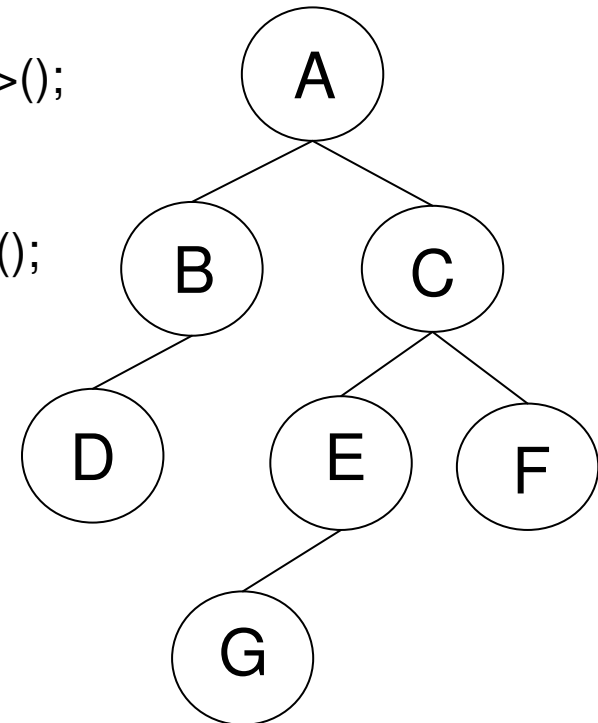
Classe ArvoreBinariaEncadeada

```
public class ArvoreBinariaEncadeada<E> implements ArvoreBinaria<E> {  
  
    private E elem;  
    private ArvoreBinariaEncadeada<E> esq;  
    private ArvoreBinariaEncadeada<E> dir;  
    private ArvoreBinariaEncadeada<E> pai;  
  
    public ArvoreBinariaEncadeada () {  
        this.elem = null;  
        this.esq = null;  
        this.dir = null;  
        this.pai = null; }  
}
```



Classe ArvoreBinariaEncadeada

```
public void atribuiRaiz (E elemento){  
    if (elemento != null){  
        if (this.elem == null){  
            this.esq = new ArvoreBinariaEncadeada<E>();  
            this.esq.pai = this;  
            this.dir = new ArvoreBinariaEncadeada<E>();  
            this.dir.pai = this;  
        }  
        this.elem = elemento;  
    }  
}
```



Classe ArvoreBinariaEncadeada

```
public int numeroElementos (){
```

```
    if (this.elem == null)
```

```
        return 0;
```

```
    else
```

```
        return this.esq.numeroElementos() + this.dir.numeroElementos() + 1;
```

```
}
```

