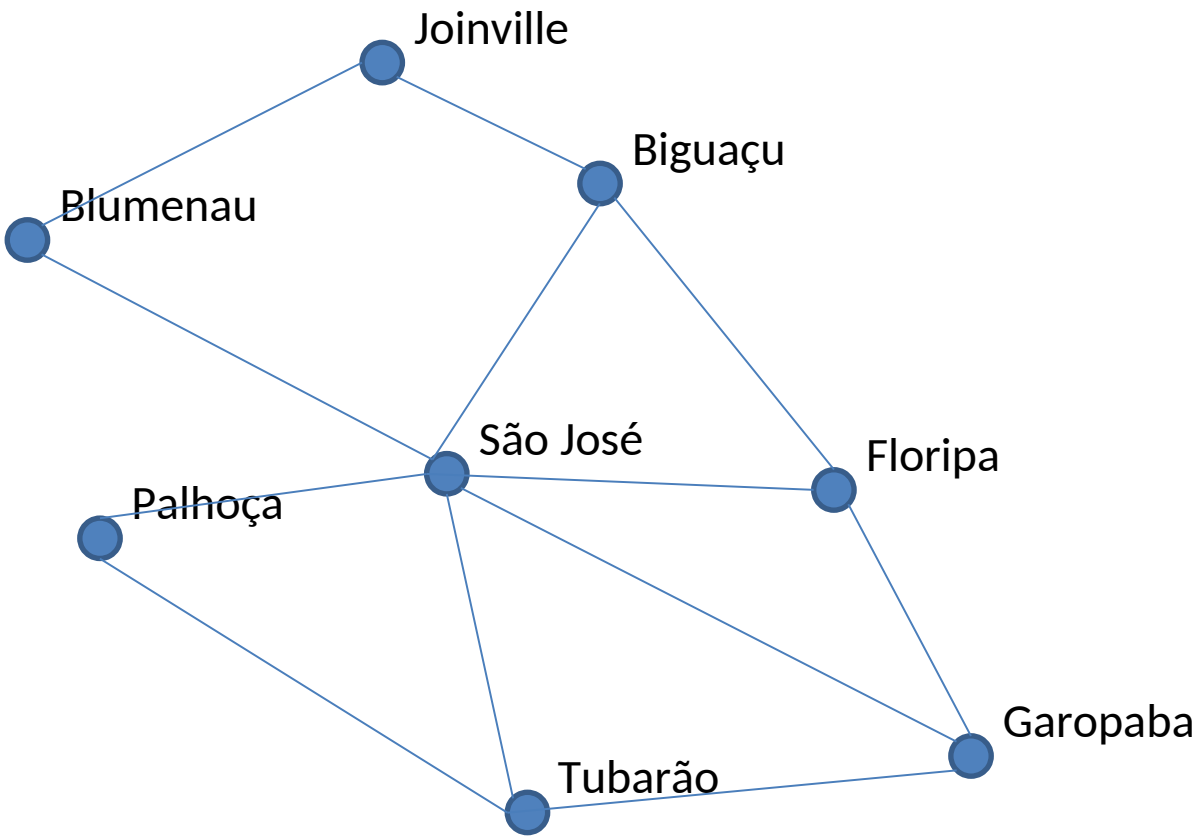
Motivação

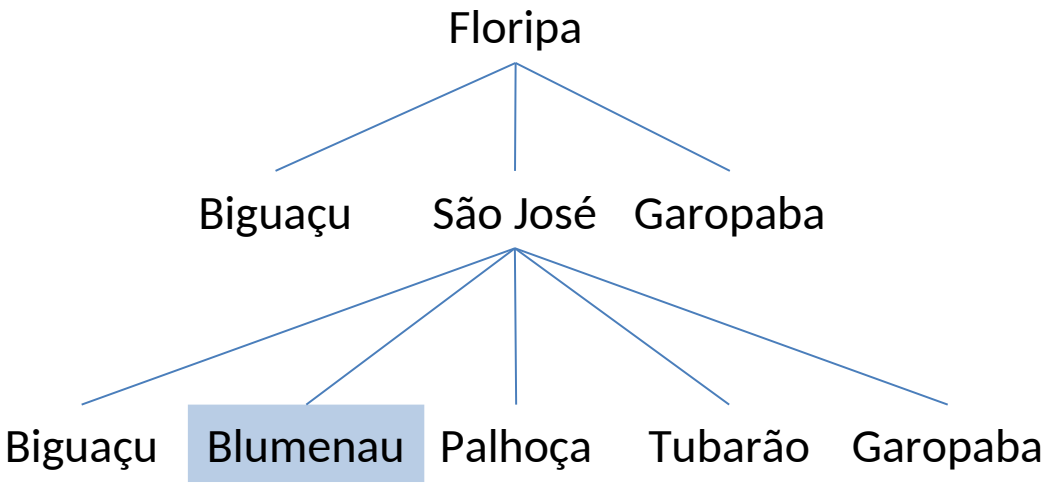
- Estrutura de dados muito utilizada
  - Permite a representação de dados de maneira hierárquica;
  - Fornece maneiras eficientes de busca;
- Quais são seus usos comuns?
  - Manipular dados hierárquicos
  - Manipular listar ordenadas de dados
  - Em algoritmos de roteamento de pacotes

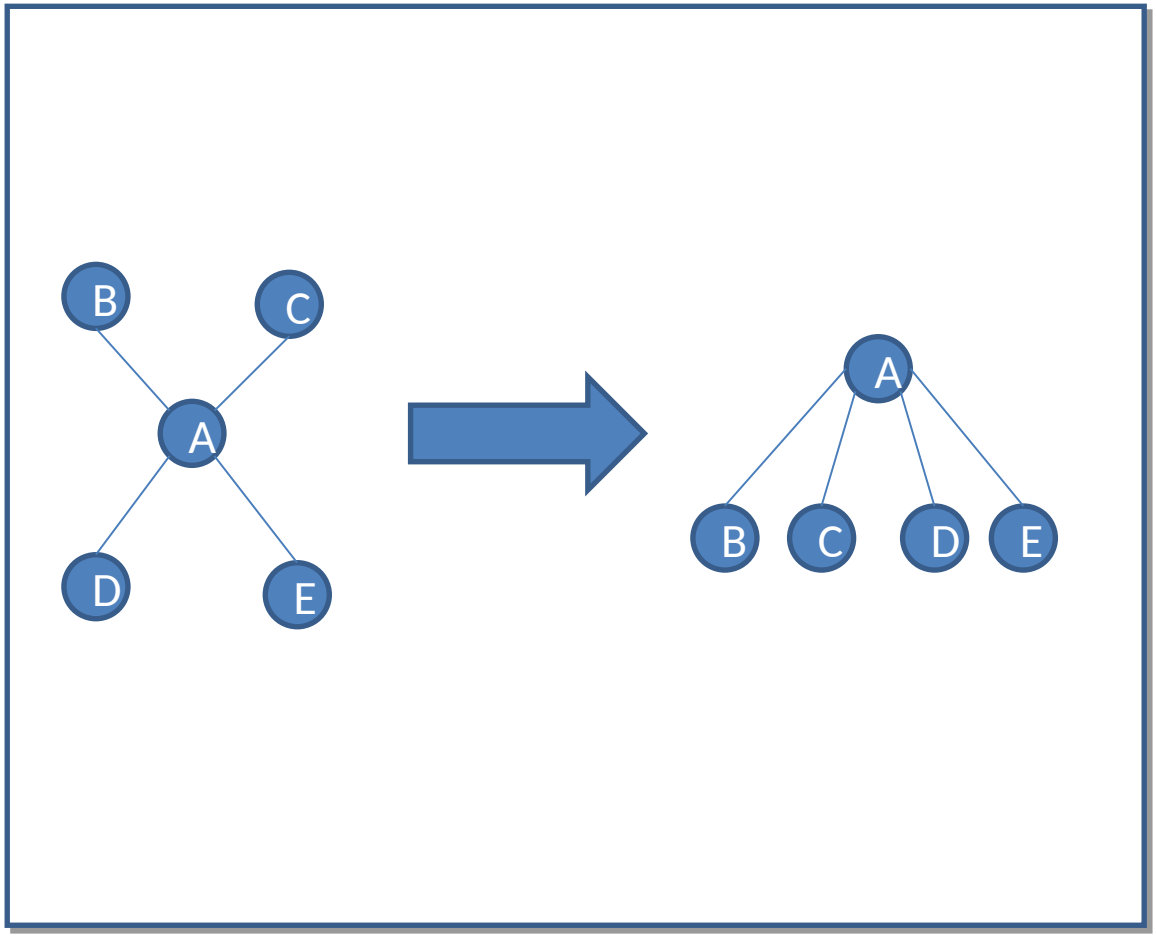
Motivação - exemplo



- Objetivo: Ir de Floripa para Blumenau!

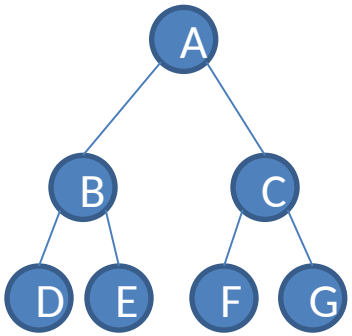
Árvore de Busca



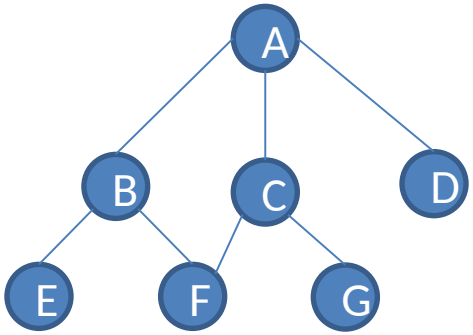

B
C
A
D
E
A
B
C
D
E

Árvores - Grafos

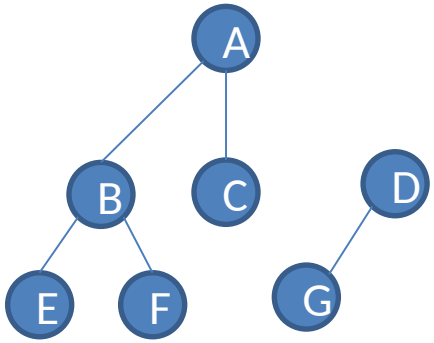
- É uma árvore nada mais é que um tipo particular de grafo. Para que um grafo seja uma árvore o mesmo deve ser:
  - Conectado,
  - acíclico,
  - Não orientado



- É uma árvore
  - Conectado
  - acíclico
  - Não orientado

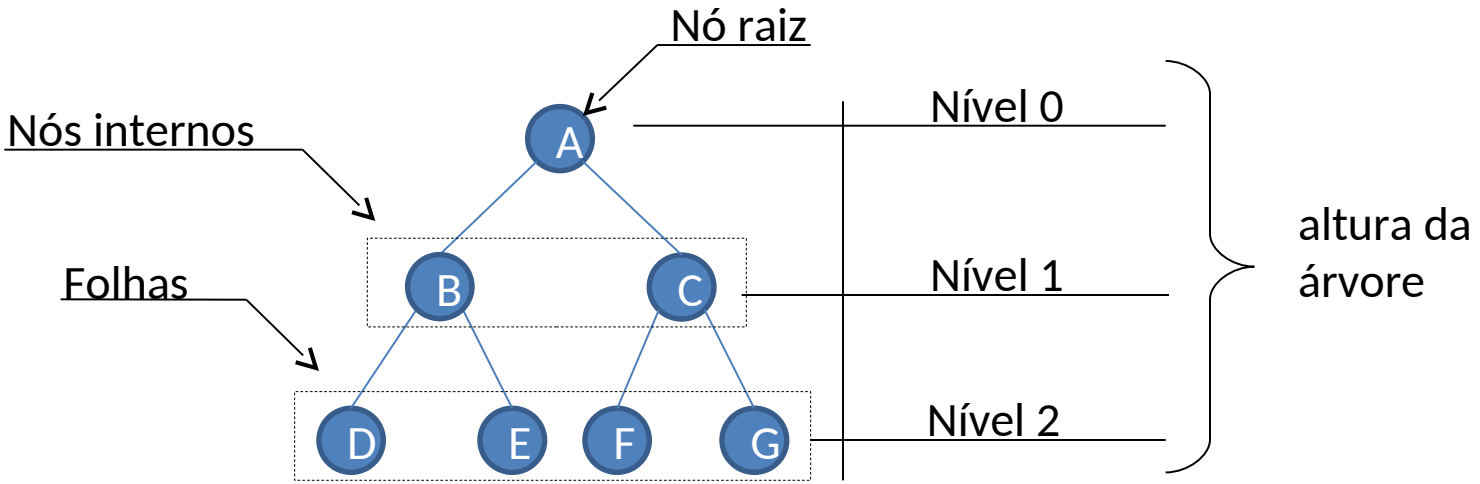


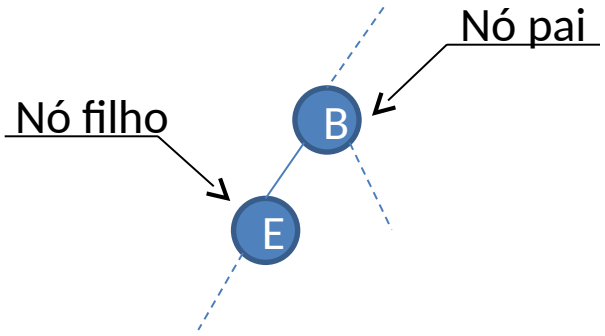
- Não é uma árvore
  - Conectado
  - Cíclico
  - Não orientado



- É uma arborecência
  - desconectado
  - acíclico
  - Não orientado

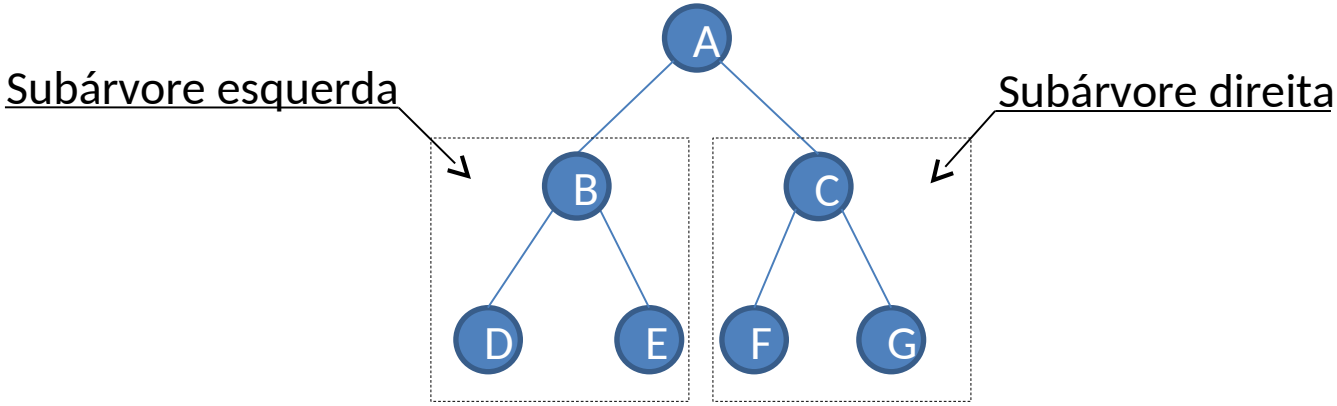
Nomenclatura





Árvores Binárias

- Todo nó possui exatamente dois nós filhos
  - Exceto os nos folha, que devem possuir exatamente 0 filhos
- É muito útil para modelar situações em que precisam ser tomadas decisões bidirecionais em cada ponto de um processo



Árvores Binárias - Definições

- Árvore Binária Completa de nível *d*
  - Todas as folhas estejam no nível *d*
- Se contiver *m* nós no nível *l*, ela conterá no máximo *2m* nós no nível *l+1*
- Uma árvore binária completa de nível *d* contém exatamente 2^l nós em cada lível *l* entre *0* e *d* (profundidade d com exatamente 2^d nós no nível *d*

Nível (*d*)	n. max. nós	Potência
0	1	2^0
1	2	2^1
2	4	2^2
3	8	2^3
4	16	2^4
5	32	2^5

Árvores Binárias - Definições

- Número total de nós

$$t_n = 2^0 + 2^1 + 2^2 + \ldots + 2^n = \sum_{j=0}^{d} 2^j \quad \xrightarrow{\text{Por indução}} \quad t_n = 2^{d+1} - 1$$

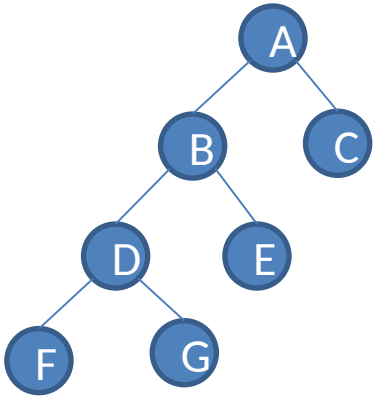
- Também é possível calcular o nível *d* de uma árvore binária completa se o número t_n for conhecido.

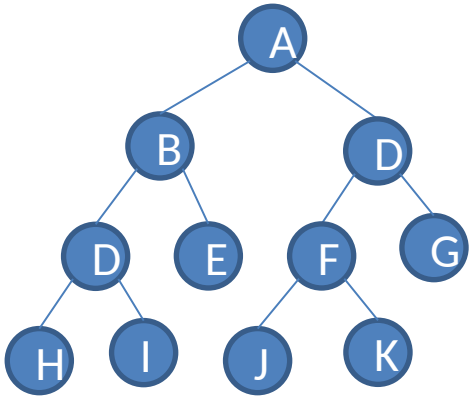
$$d = \log_2(t_n + 1) - 1$$

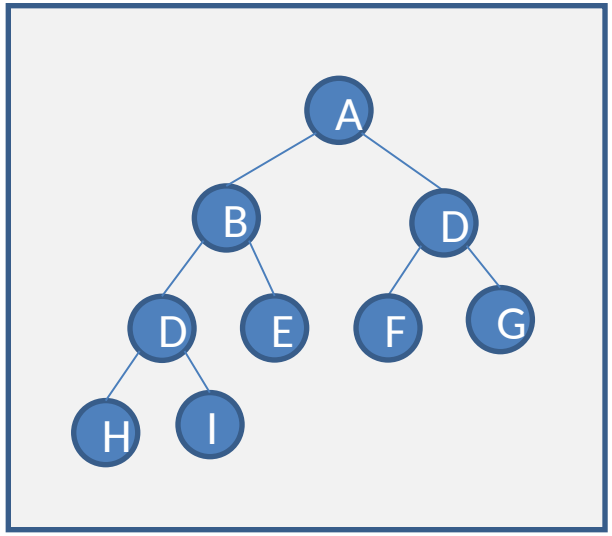
note que em geral, $\log_2 x < x$

$\log_2 x$	x
3	15
10	1.024
≈20	1.000.000

Árvore Binárias Quase Completas

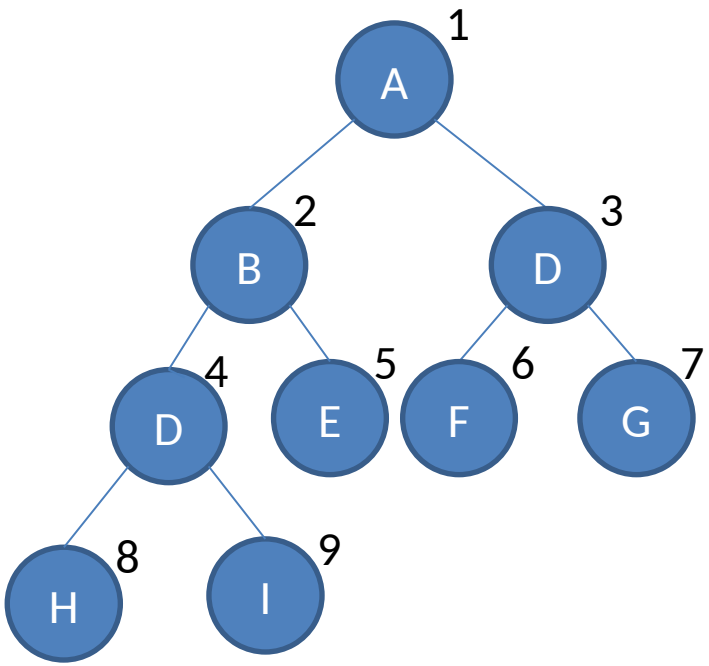






1. Todas as folhas da árvore devem estar localizadas no nível d ou no nível d-1;
2. Para cada nó nd na árvore com um descendente direto no nível d, todos os descendentes esquerdos de nd que forem folhas estiverem também no nível d.

Numeração de nós de árvores binárias



- Os nós de uma árvore binária quase completa podem ser numerados
- 1 para a raiz
- Filho esquerdo = dobro do n. do pai
- Filho direito = dobro + 1 do n. do pai
- A numeração ajuda na implementação como será visto na aula prática

(implementação por vetores)

Facilita na localização de itens

A	B	D	D	E	F	G	H	I

Árvores Binárias

Assim como vimos em nosso estudo sobre listas, árvores também podem ser armazenadas de forma estática ou dinâmica.

Começaremos nosso estudo com o armazenamento estático.

Quais campos seriam relevantes para cada nó?

Os campos *info*, *left*, *right* e *father*. Onde estes campos representam respectivamente a informação armazenada no nó, seu filho à esquerda, seu filho à direita e seu pai.

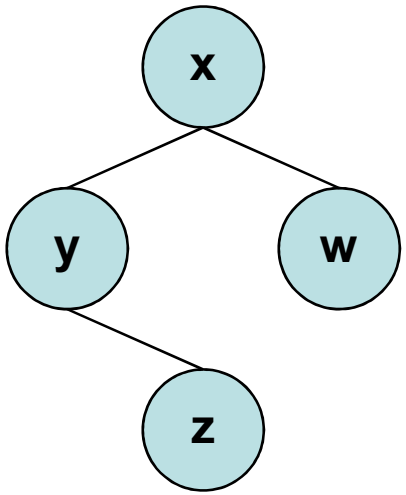
UNIVASF

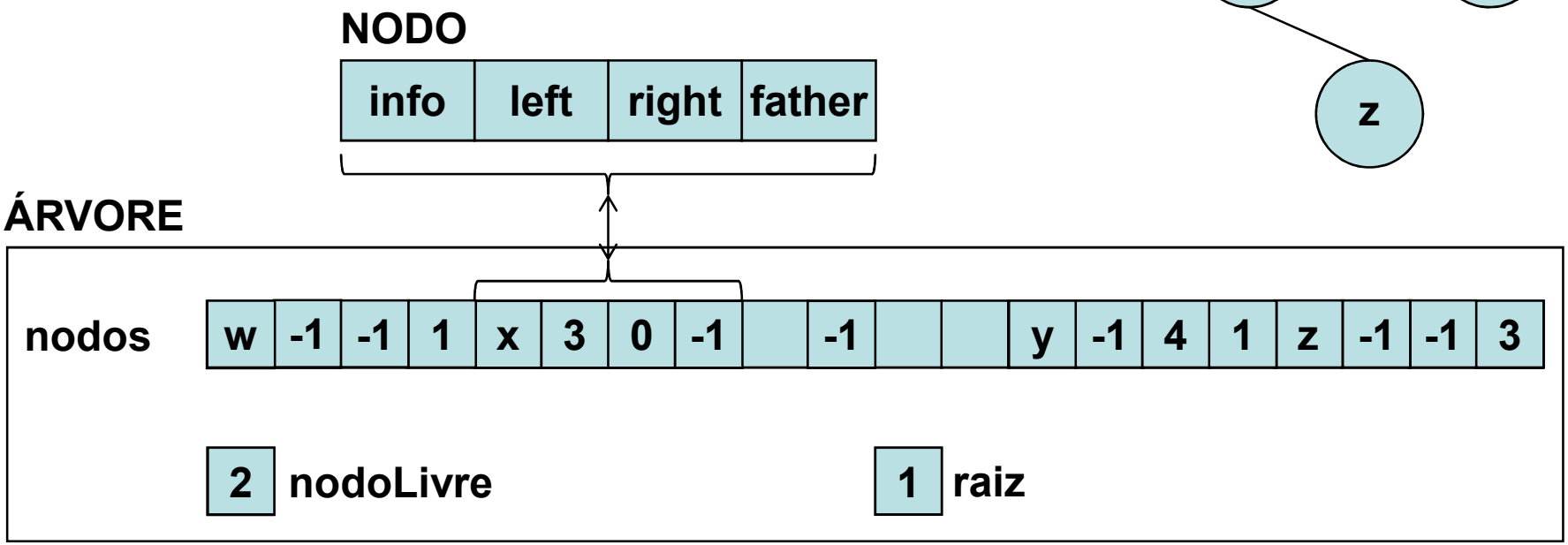
Árvores Binárias

Como vocês sugeririam o armazenamento estático?

Em um vetor?

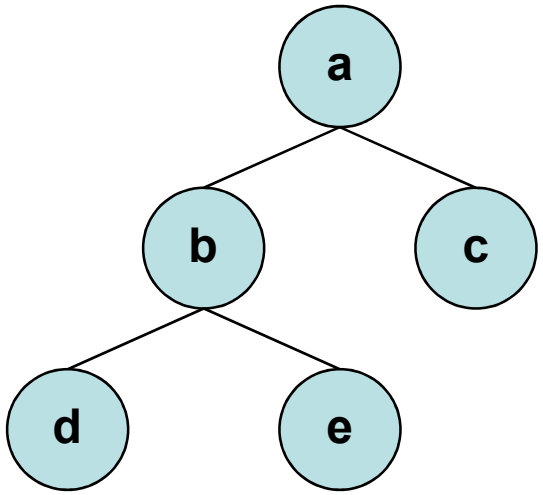
Como?





Árvores Binárias

Existem formas de armazenamento estático alternativas à apresentada. Por exemplo, vamos considerar a árvore binária abaixo.

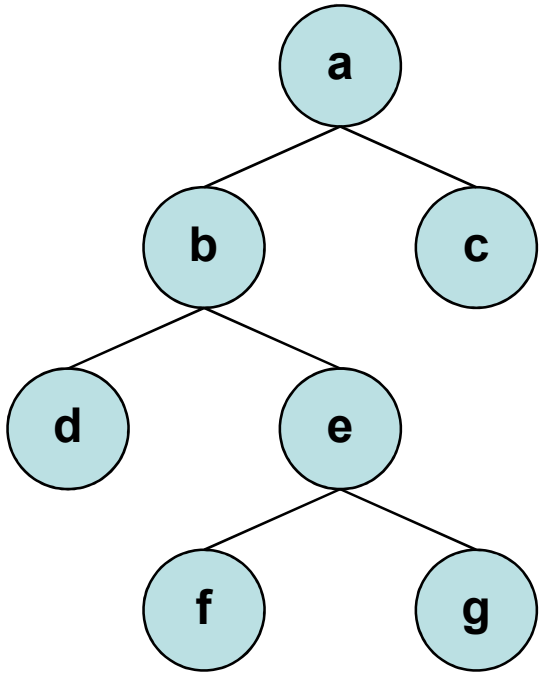


Esta Árvore poderia ser armazenada em um vetor da seguinte forma:

a	b	c	d	e

Árvores Binárias

Se a árvore anterior passasse a ter a seguinte configuração:



Como ficaria o armazenamento no vetor?

Assim

a	b	c	d	e	f	g

?

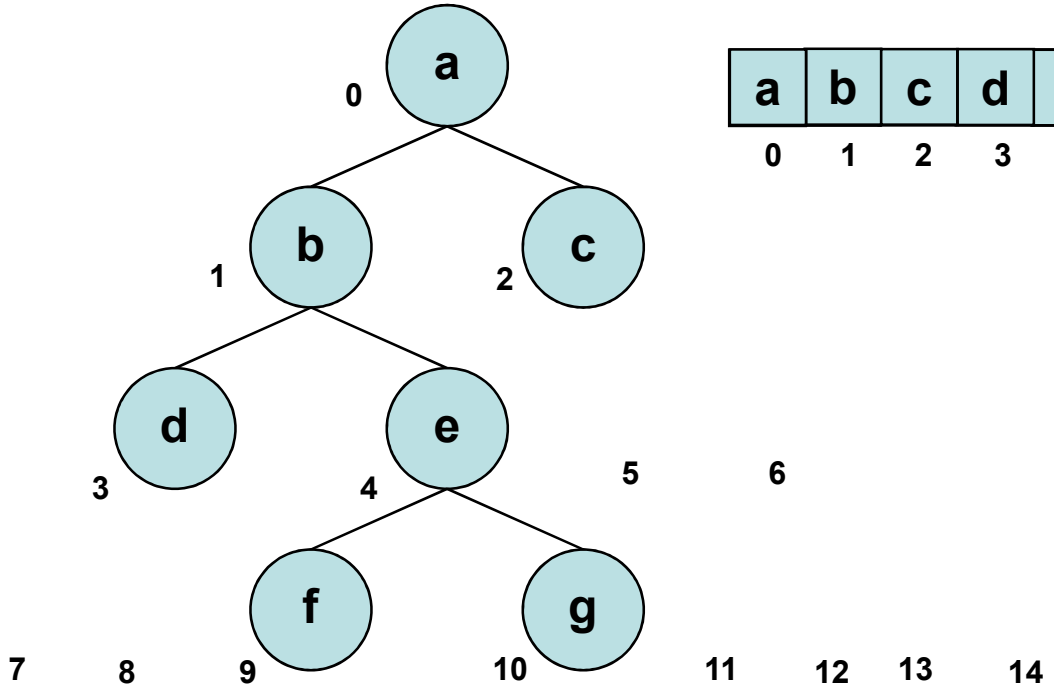
E se fosse assim:

a	b	c	d	e					f	g				

Vocês identificam alguma vantagem?

Árvores Binárias

Árvore:



Representação:

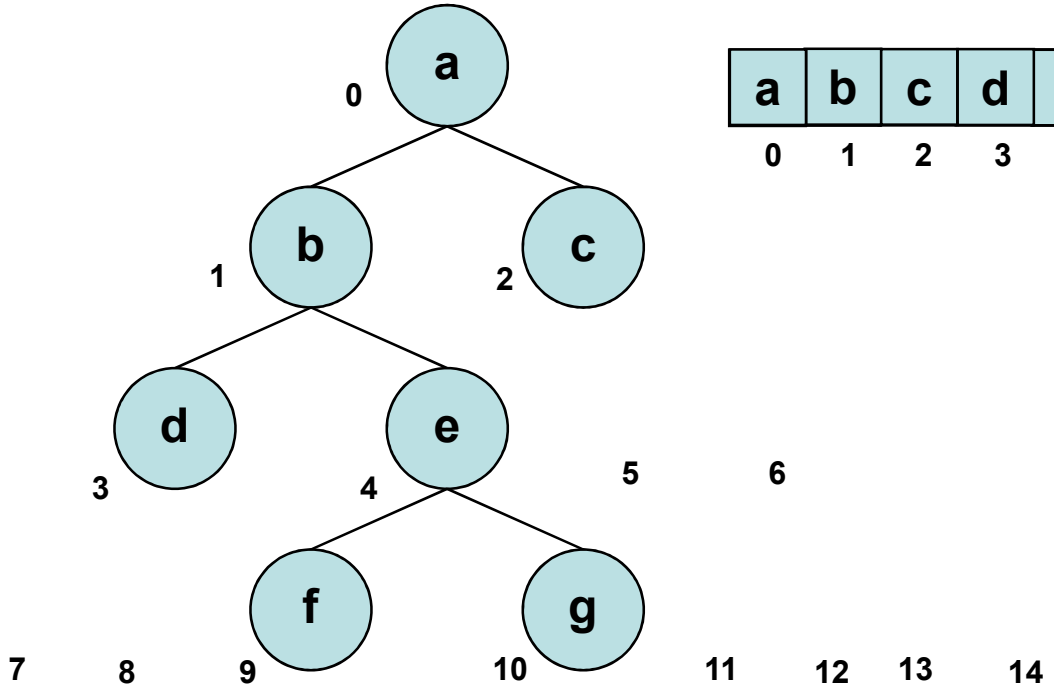
a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

A raiz está aonde?

No índice 0 (zero).

Árvores Binárias

Árvore:



Representação:

a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

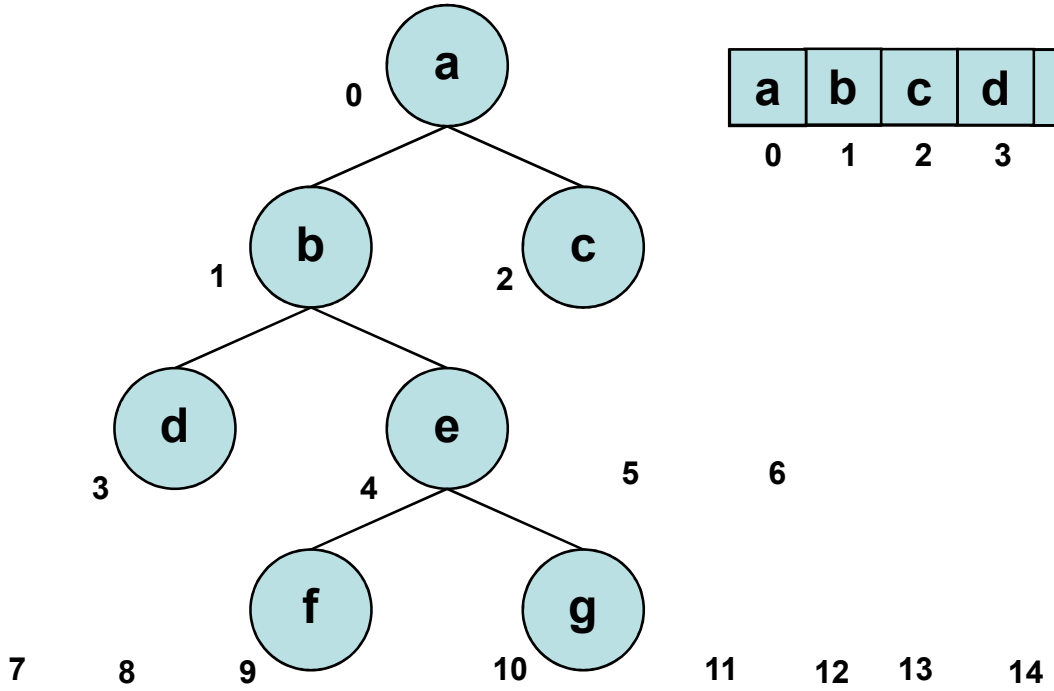
Como saber quem é o pai de um nó?

(p-1)/2.

p é índice do vetor que contém o nó.

Árvores Binárias

Árvore:



Representação:

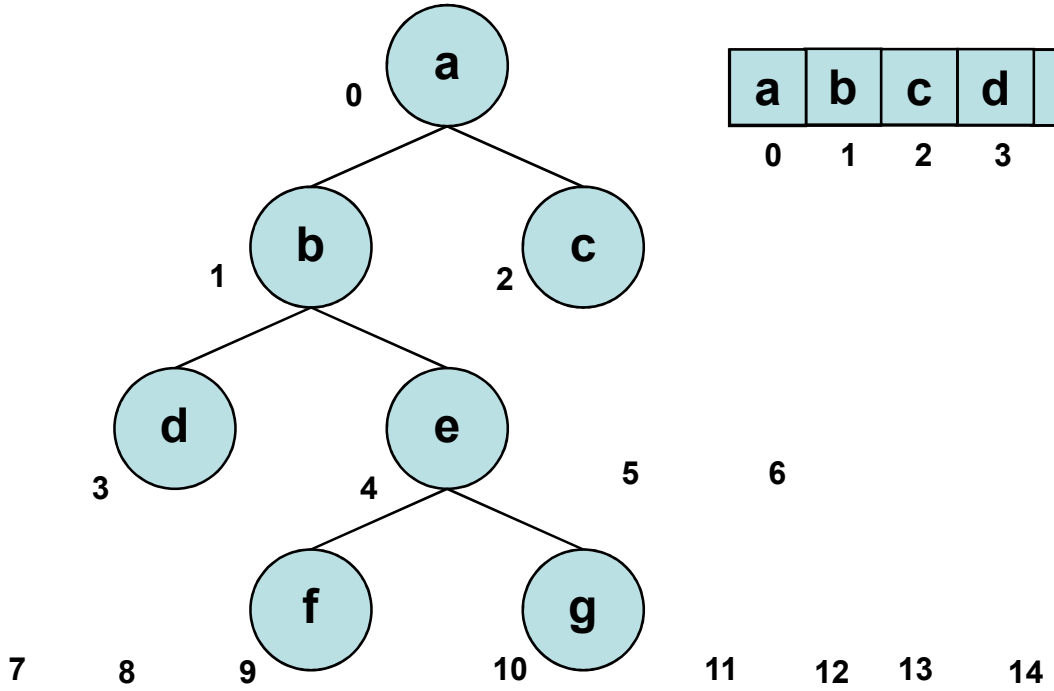
a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Como saber quem é o filho esquerdo de um nó?

2*p+1

Árvores Binárias

Árvore:



Representação:

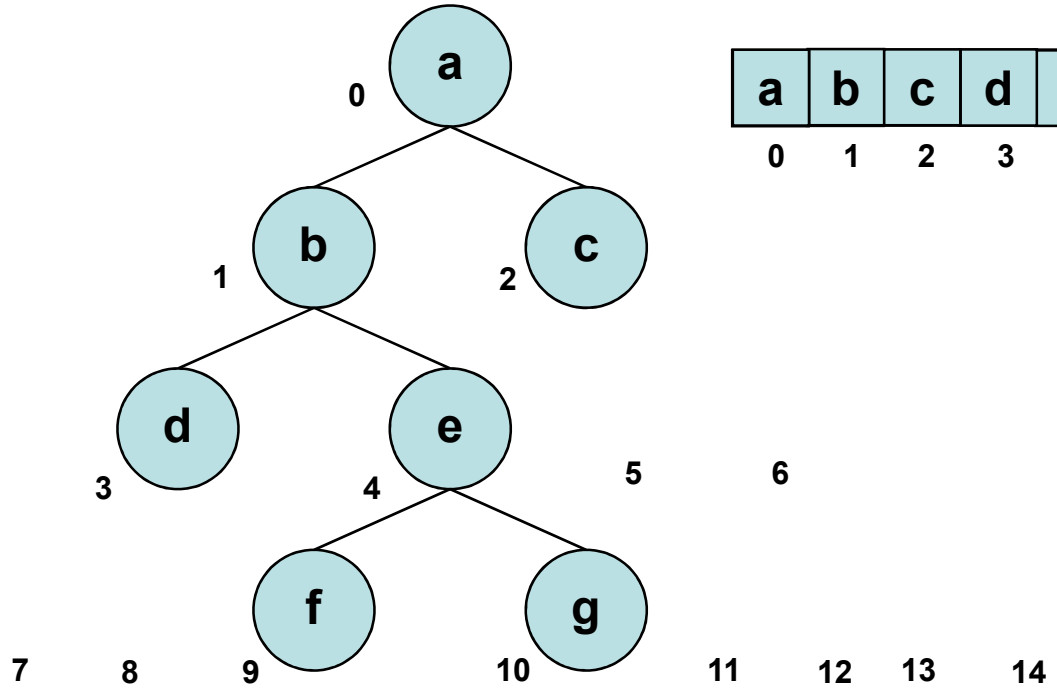
a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Como saber quem é o filho direito de um nó?

2*p+2

Árvores Binárias

Árvore:



Representação:

a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Como saber onde está o irmão direito de um nó?

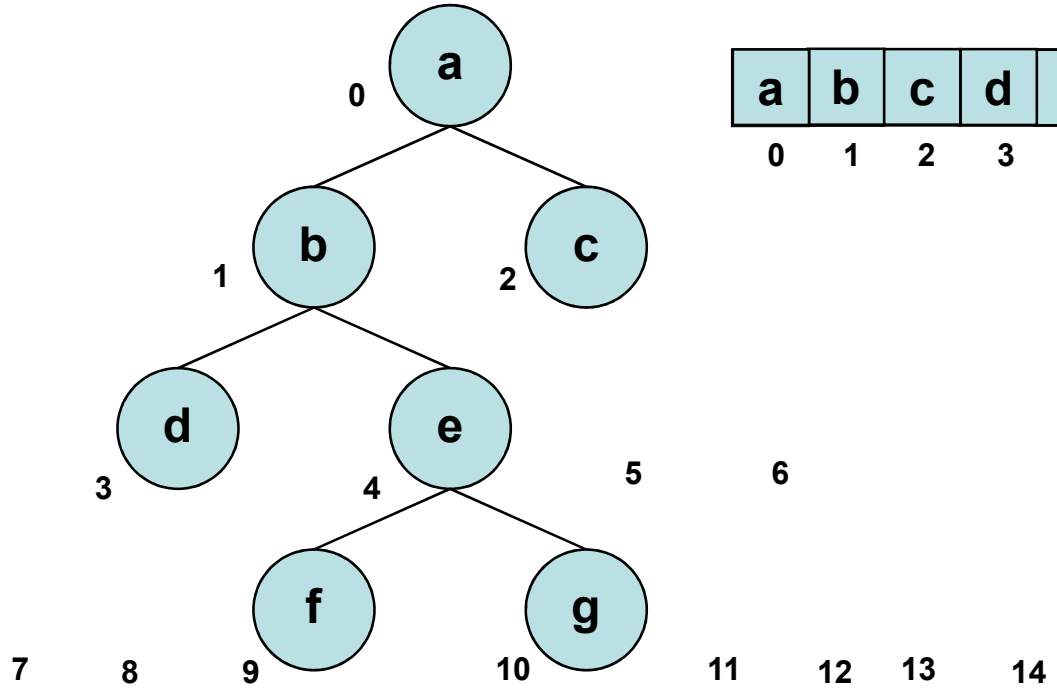
Primeiro, devemos saber se ele é um filho esquerdo.

Como saber?

É só verificar se p é impar.

Árvores Binárias

Árvore:



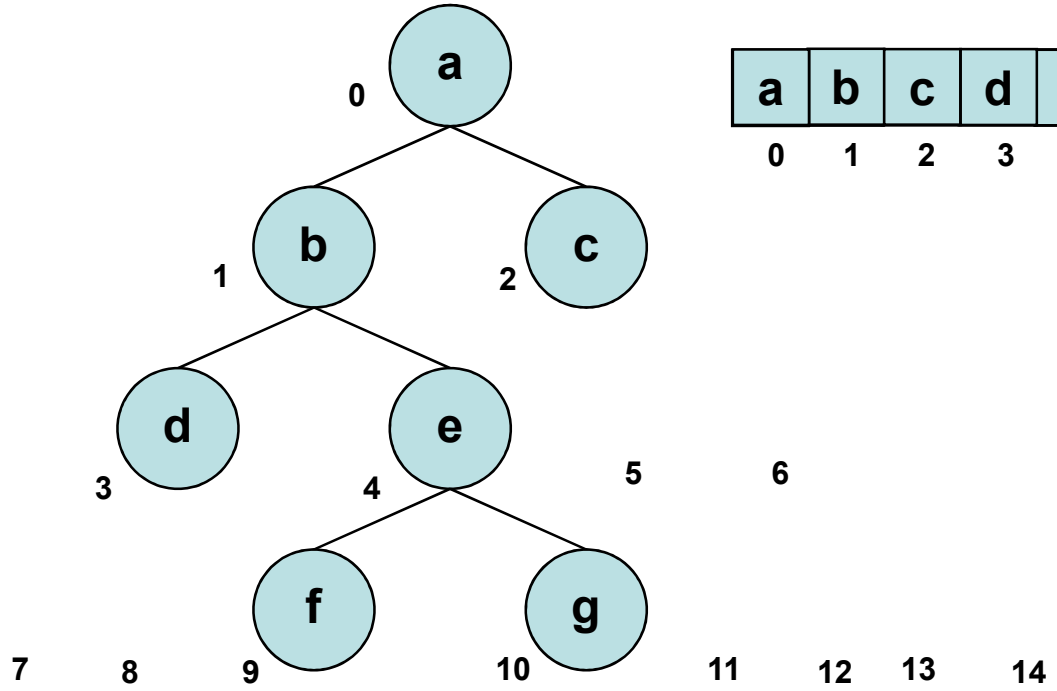
Representação:

a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Se um nó for um filho a esquerda, para saber onde está o seu irmão direito basta somar uma unidade a p.

Árvores Binárias

Árvore:



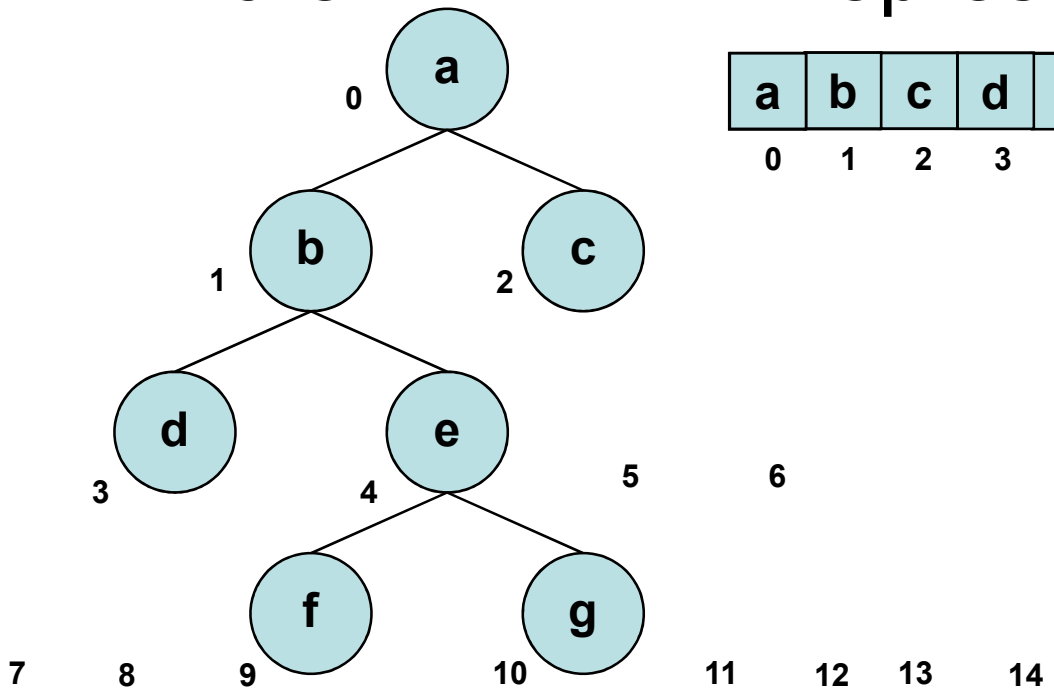
Representação:

a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

De forma análoga determinamos se um nó é um filho à direita e onde está o seu irmão esquerdo.

Árvores Binárias

Árvore:



Representação:

a	b	c	d	e					f	g				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Como saber se d não tem filho esquerdo e/ou direito?

Com a inserção de um campo adicional em cada nó, que identificará se este está ou não ocupado. Assim, ao computar onde está o filho direito de d (3*2+2=8) verifica-se o valor deste campo adicional.

Árvores Binárias

Esta forma de representação é denominada representação implícita em vetores de árvores binárias.

Para armazenar uma árvore binária completa com profundidade igual a d será necessário um vetor com $2^{d+1} - 1$.

A implementação da representação implícita em vetores de árvores binárias é uma boa proposta de exercício.

Contudo, visando otimizar a utilização da memória, temos a possibilidade de armazenar uma árvore binária dinamicamente.

UNIVASF

Percorrendo Árvores Binárias

- Todos estes três métodos podem ser definidos recursivamente e se baseiam em três operações básicas:
  - visitar a raiz,
  - percorrer a subárvore da esquerda e
  - percorrer a subárvore da direita.
- A única diferença entre estes métodos é a ordem em que estas operações são executadas.

Percorrendo Árvores Binárias

- As 3 formas de percorrer uma árvore binária são:

  - Caminhamento Pré-Ordem ou Pré-Fixado:
    1. Visite a raiz;
    2. Visite a subárvore esquerda;
    3. Visite a subárvore direita.

Percorrendo Árvores Binárias

- As 3 formas de percorrer uma árvore binária são:

  - Caminhamento Central:
    1. Visite a subárvore esquerda;
    2. Visite a raiz;
    3. Visite a subárvore direita.

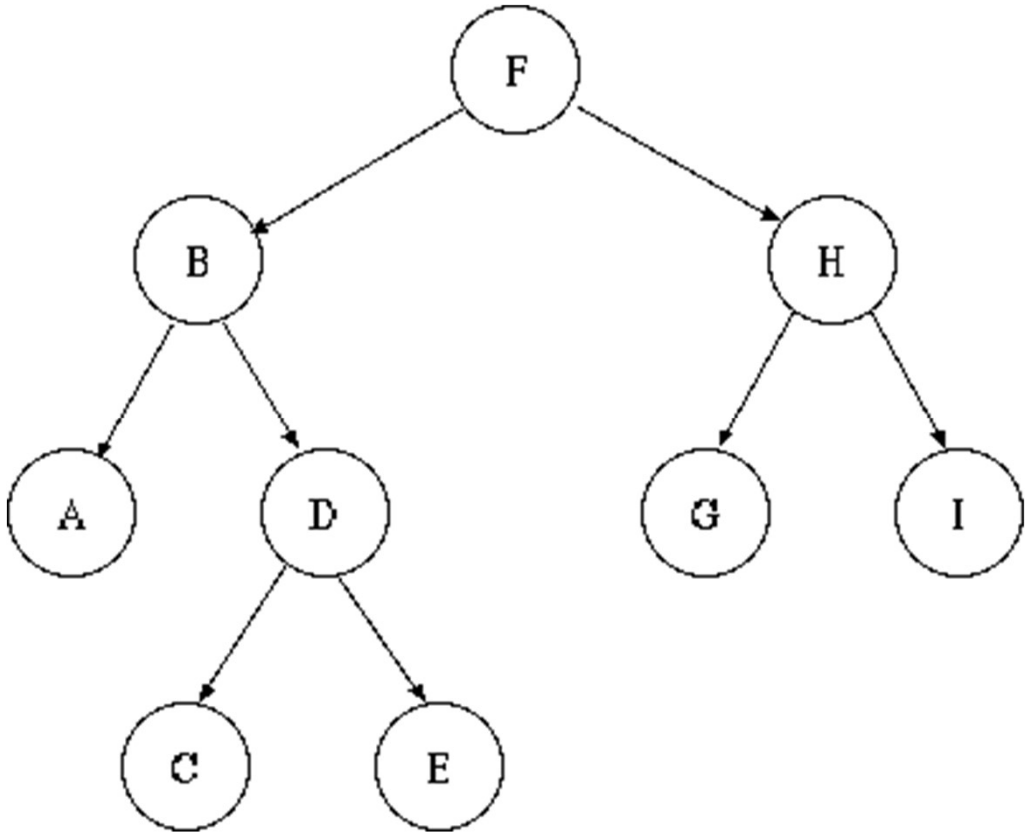
Percorrendo Árvores Binárias

- As 3 formas de percorrer uma árvore binária são:

  - Caminhamento Pós-Ordem ou Pós-Fixado:
    1. Visite a subárvore esquerda;
    2. Visite a subárvore direita;
    3. Visite a raiz.

Percorrendo Árvores Binárias

- **Pré-Ordem**:
  - F B A D C E H G I
- **Central**:
  - A B C D E F G H I
- **Pós-Ordem**:
  - A C E D B G I H F

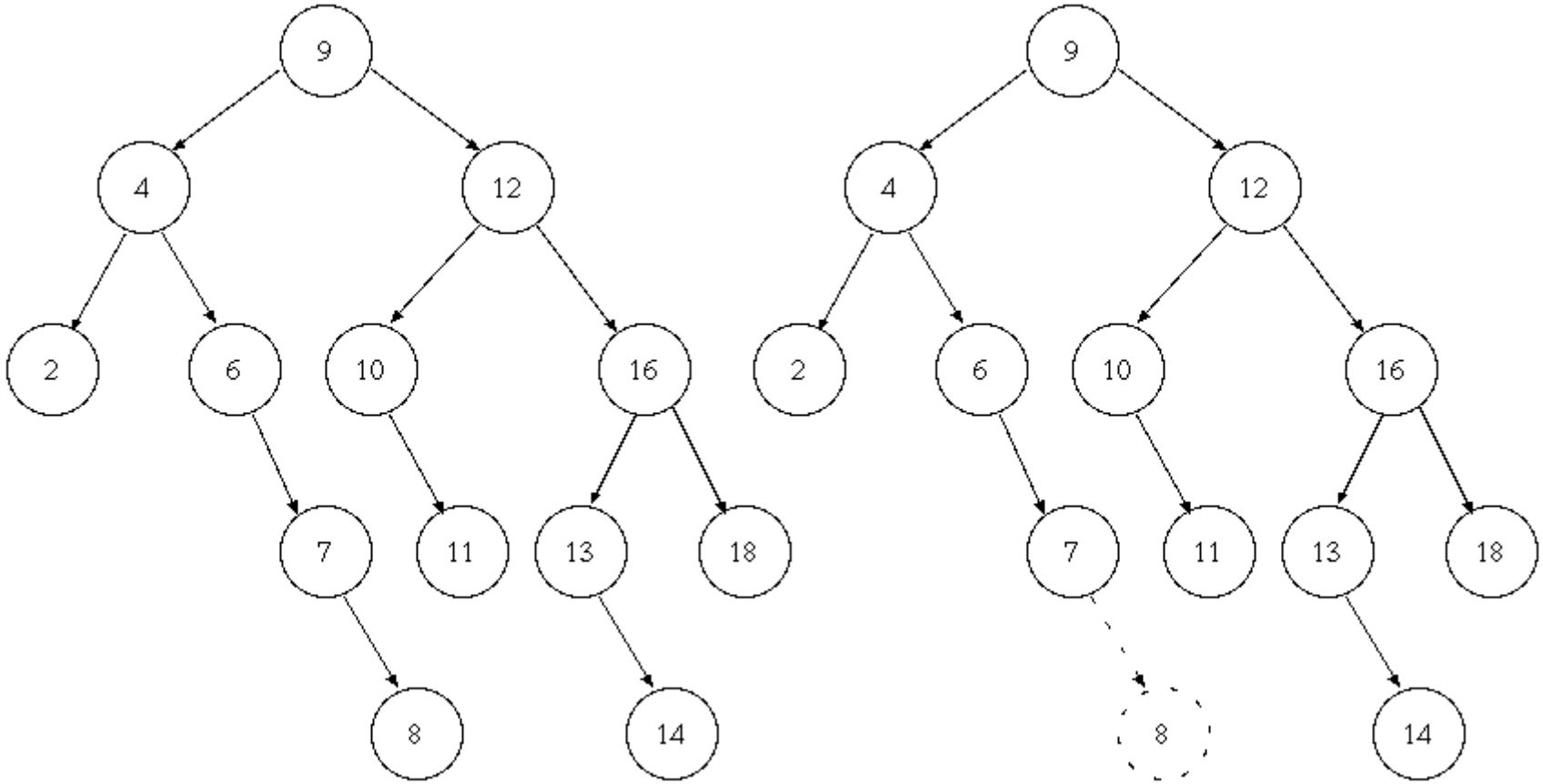


Removendo Nós de Árvores Binárias

- Para remover um nó de uma árvore binária devemos considerar três casos:
  1. nó sem filhos;
  2. nó com um único filho;
  3. nó com dois filhos.
- O caso de um nó sem filhos é o mais simples e significa apenas ajustar o ponteiro de seu pai.

Removendo Nós de Árvores Binárias

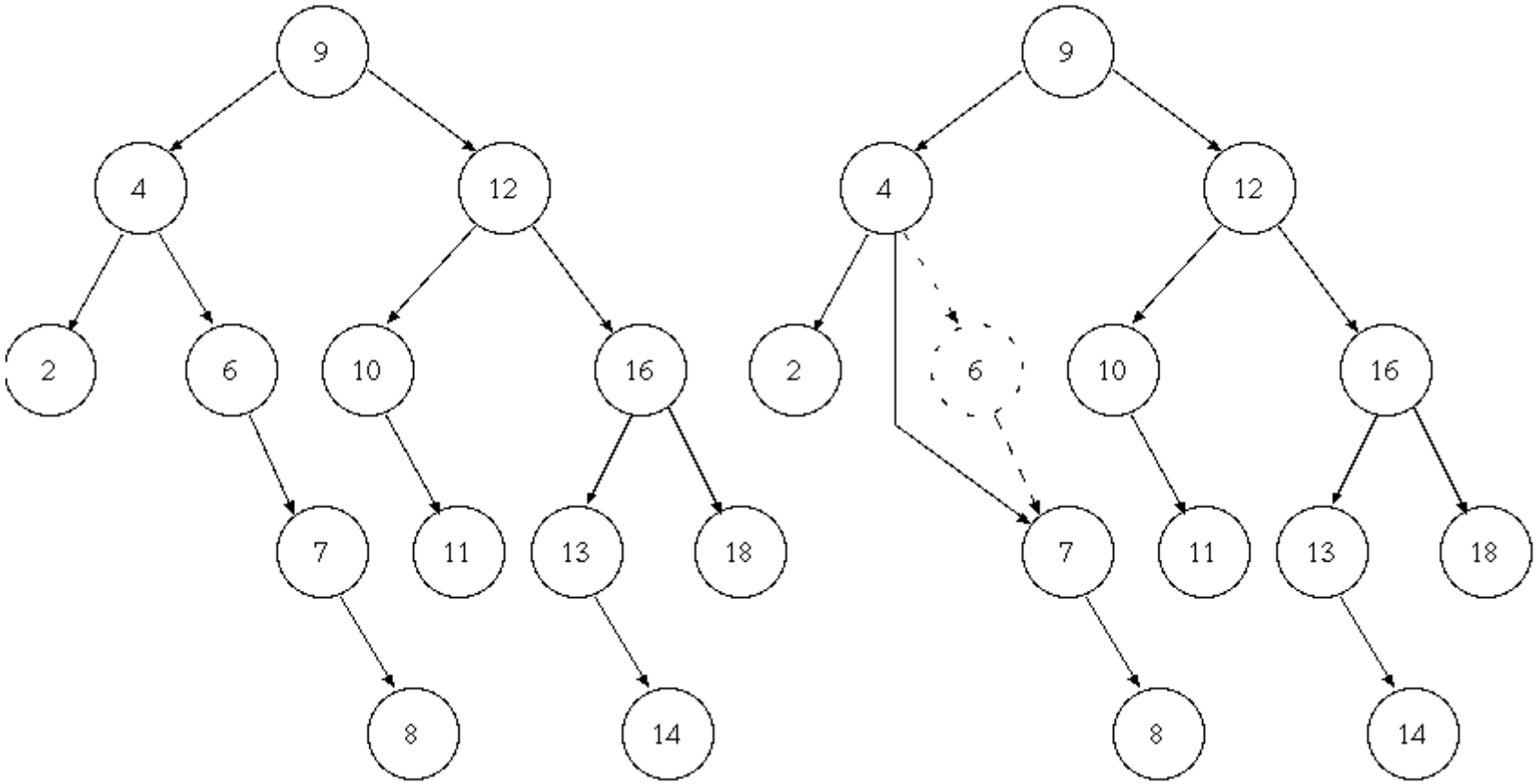
- A Figura a seguir ilustra este caso, onde o nó com o valor 8 é removido.



Removendo Nós de Árvores Binárias

- No caso do nó ter um único filho a mudança na árvore também é simples significa mover o nó filho daquele será removido uma posição para cima como está ilustrado na figura a seguir, onde o nó com o valor 6 é removido.

Removendo Nós de Árvores Binárias



Removendo Nós de Árvores Binárias

- O caso mais complexo é o do nó com dois filhos. Neste caso devemos procurar o sucessor s (ou antecessor) do nó deverá ocupar este lugar.
- Este nó (sucessor) é um descendente que está na subárvore da direita do nó e corresponde ao nó mais à esquerda desta árvore.
- Ele não tem filhos à esquerda e a sua árvore à direita pode ser movida para o lugar de s.

Removendo Nós de Árvores Binárias

- A Figura a seguir ilustra o caso de remoção do nó com o valor 12. Observe que o nó 13 (sucessor) assumiu o lugar do nó 12.