Lição 4

Árvores Binárias

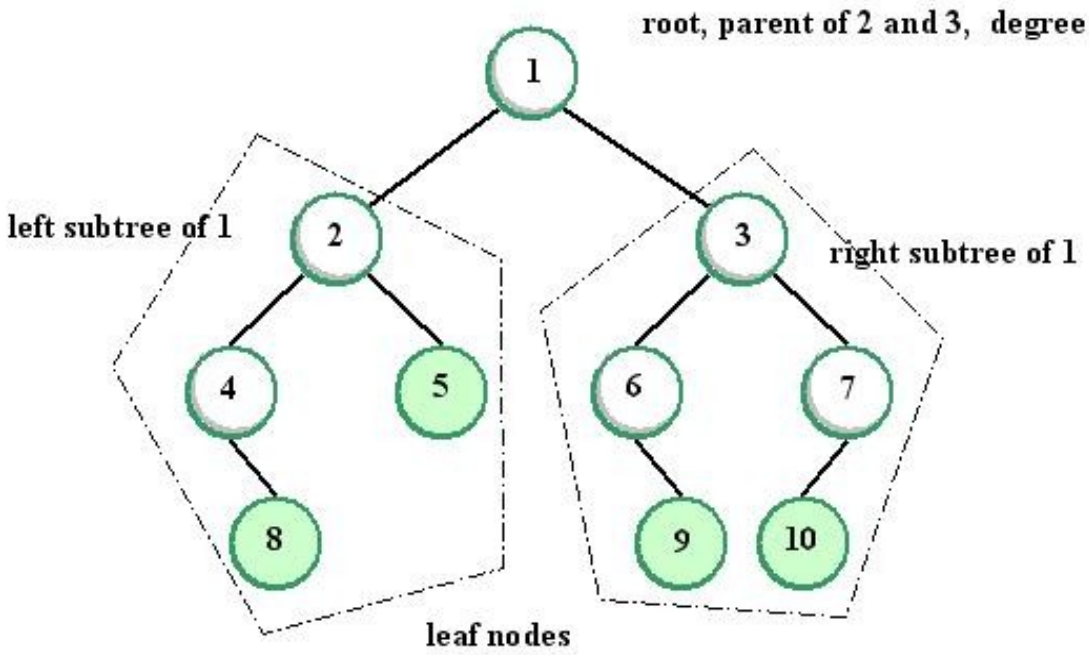
Objetivos

Ao final desta lição, o estudante será capaz de:

- Explicar os conceitos básicos e definições relacionadas a árvores binárias
- Identificar as propriedades de uma árvore binária
- Enumerar os diferentes tipos de árvores binárias
- Discutir como as árvores binárias são representadas na memória dos computadores
- Percorrer árvores binárias usando três algoritmos de varredura: pré-ordem, em ordem e pós-ordem
- Discutir aplicações da varredura em árvores binárias
- Usar *heaps* e o algoritmo *heapsort* para classificar um conjunto de elementos

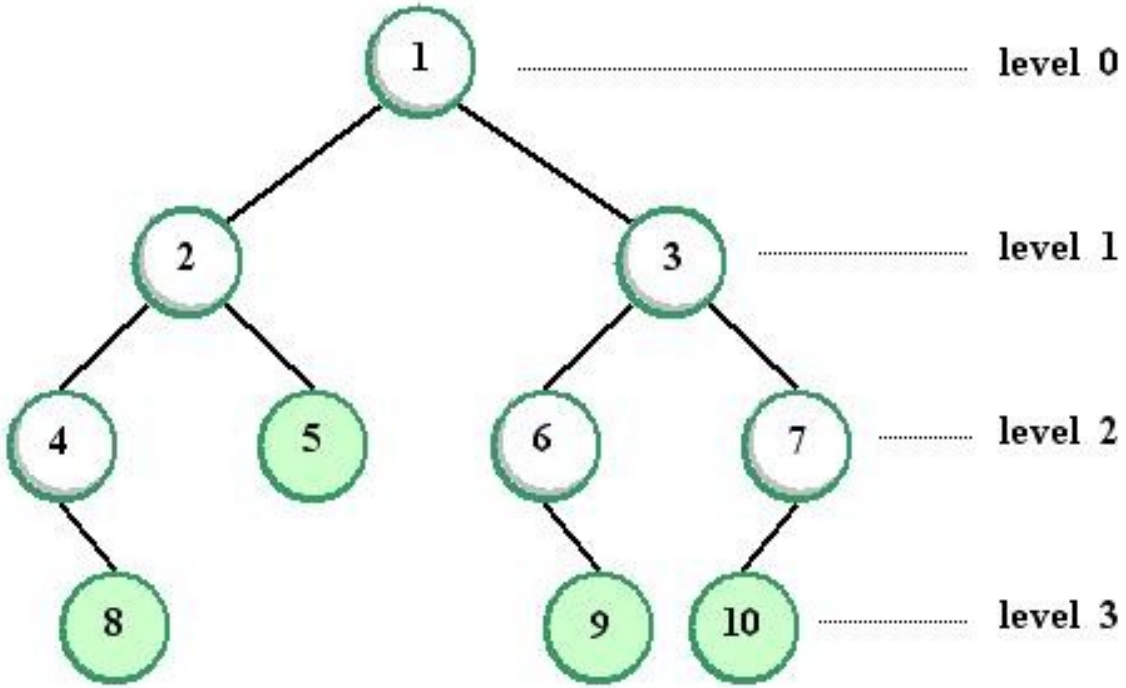
Árvore Binária

- Um ADT (*abstract data type* = tipo de dados abstrato) que é naturalmente hierárquico
- Coleção de nodes que pode estar vazia ou pode consistir de uma raiz e duas árvores binárias distintas chamadas de sub-árvores à esquerda e à direita

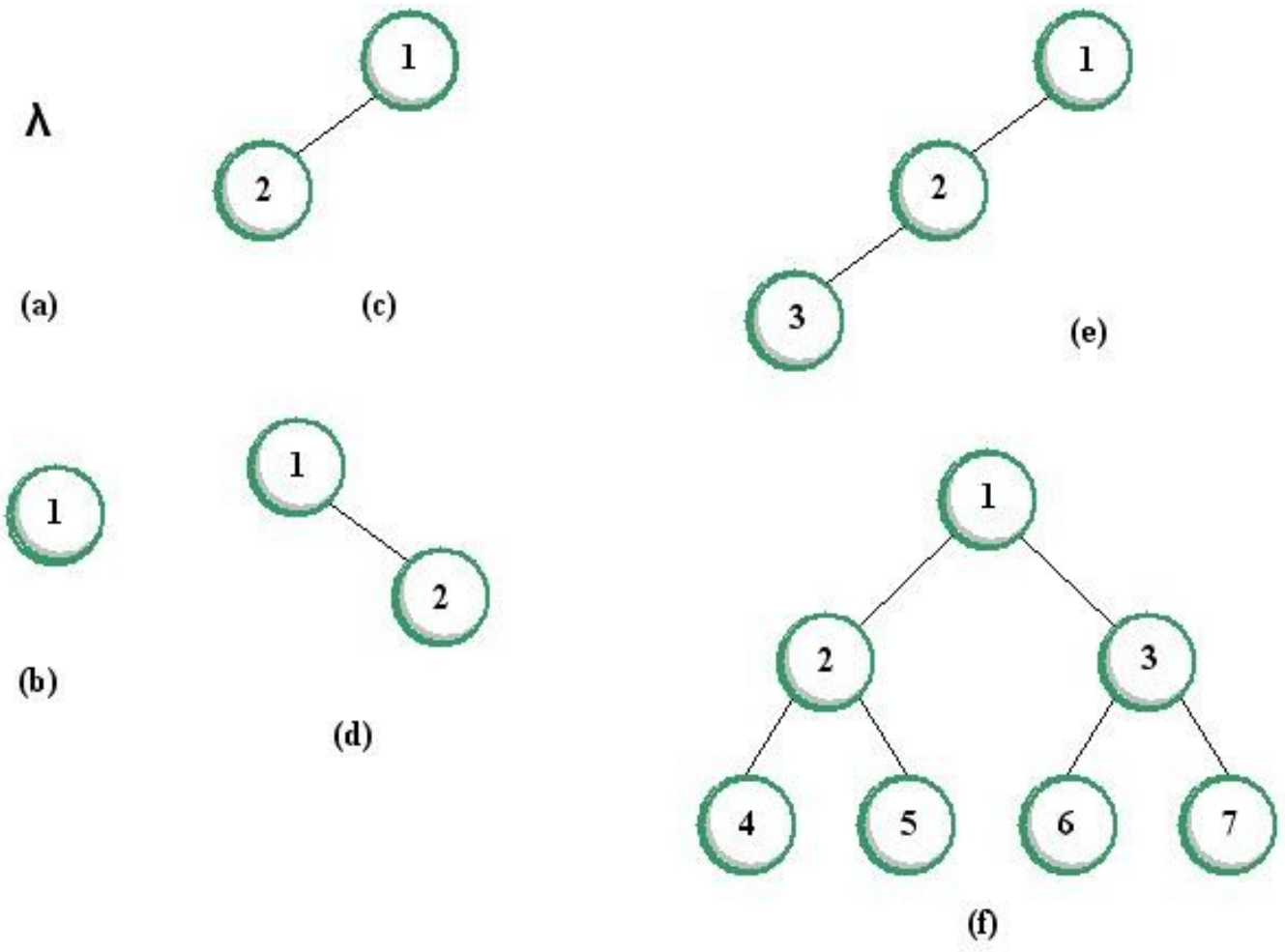


Árvore Binária

- Nível de um *node* - distância do *node* à raiz da árvore
- Altura ou Profundidade de uma árvore
- *Node* externo – node sem *child*; caso contrário é interno
- Árvore binária balanceada – tem 0 ou 2 *children*



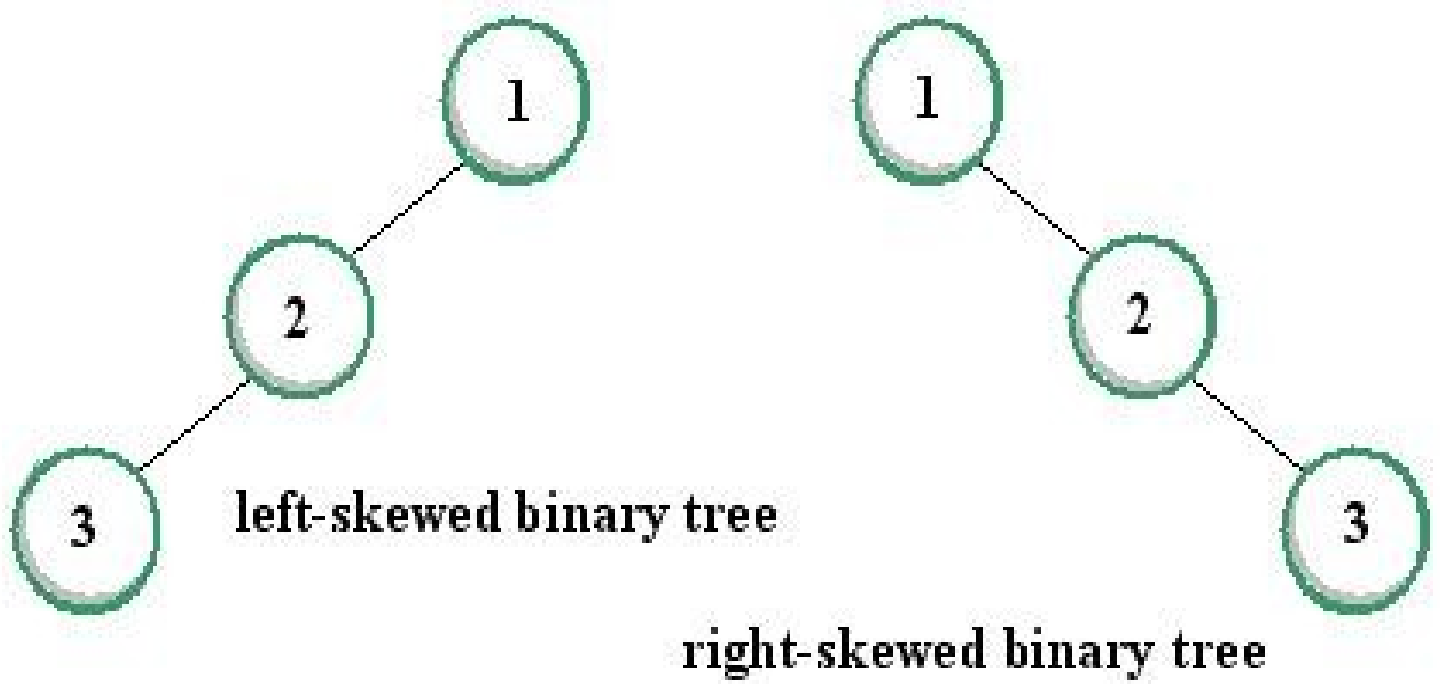
Árvore Binária



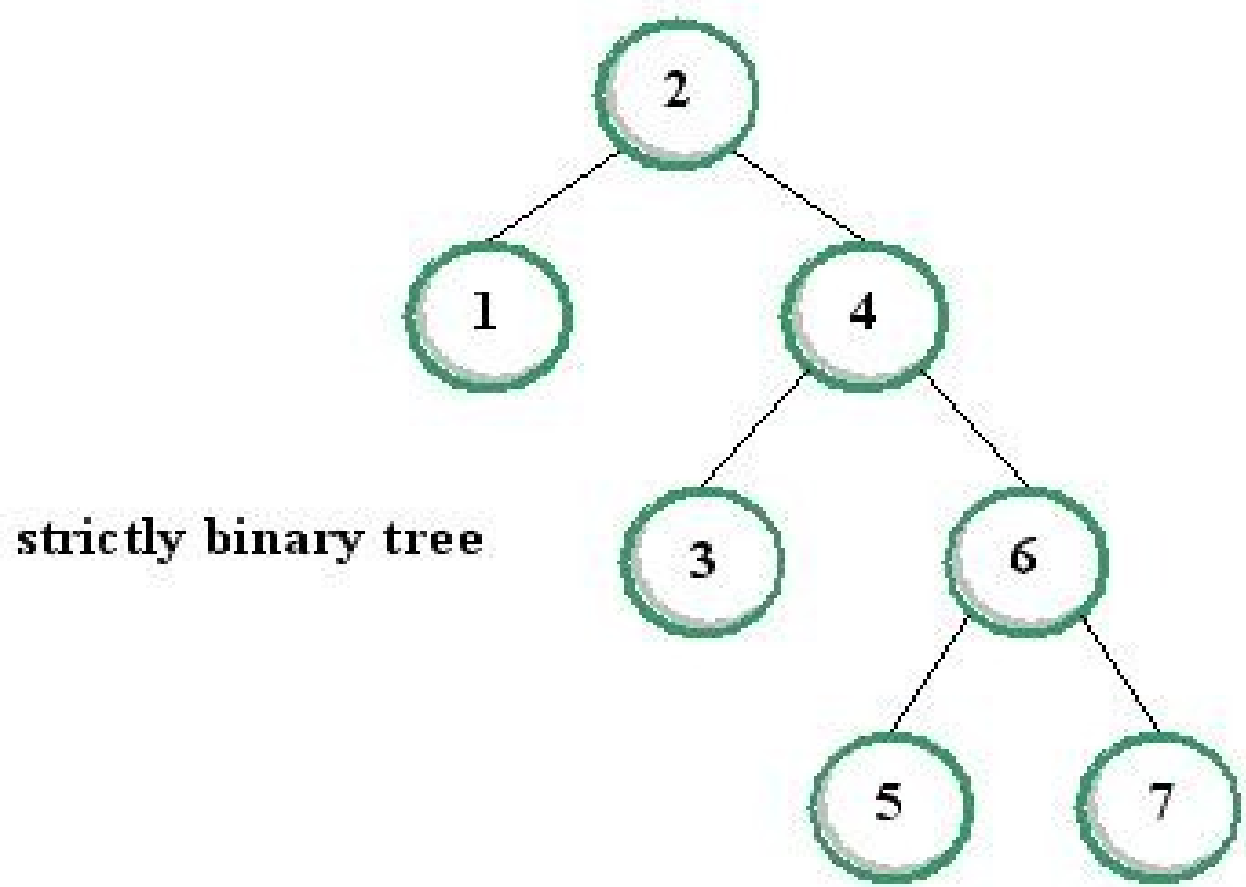
Propriedades

- Árvore binária (equilibrada/balanceada) de profundidade **k**:
  - Número máximo de *nodes* no nível **i** é $\mathbf{2^i}$, $i \geq 0$
  - Número de *nodes* é no mínimo **2k + 1** e no máximo $\mathbf{2^{k+1} - 1}$
  - Número de *nodes* externos é no mínimo **h+1** e no máximo $\mathbf{2^k}$
  - Número de *nodes* internos é no mínimo **h** e no máximo $\mathbf{2^k - 1}$
  - Se $\mathbf{n_o}$ é o número de *nodes*-folha e $\mathbf{n_2}$ é o número de *nodes* de grau 2 numa árvore binária, então $\mathbf{n_o = n_2 + 1}$

Tipos: Árvore Binária Degenerada à Direita (ou Esquerda)

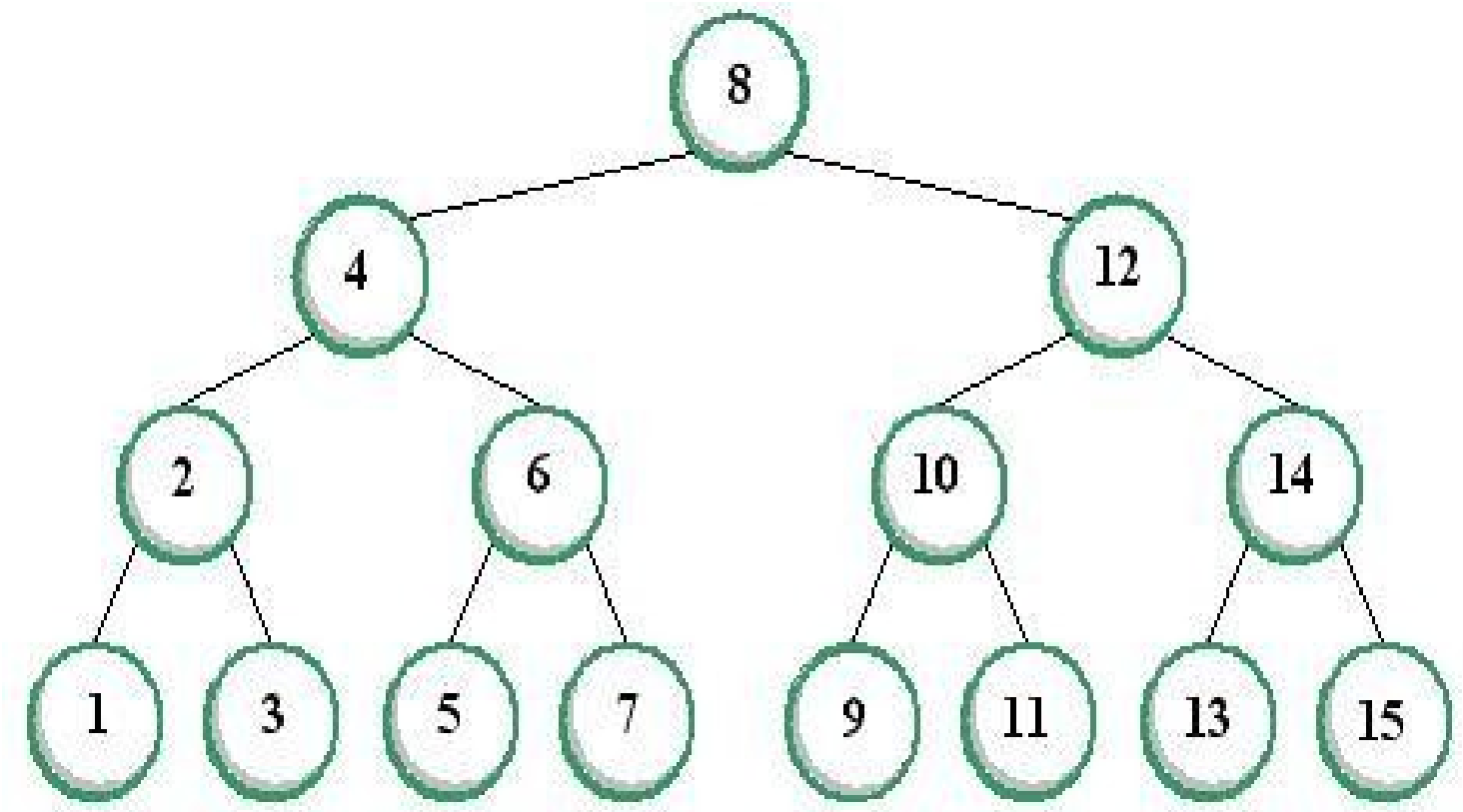


Tipos: Árvore Estritamente Binária

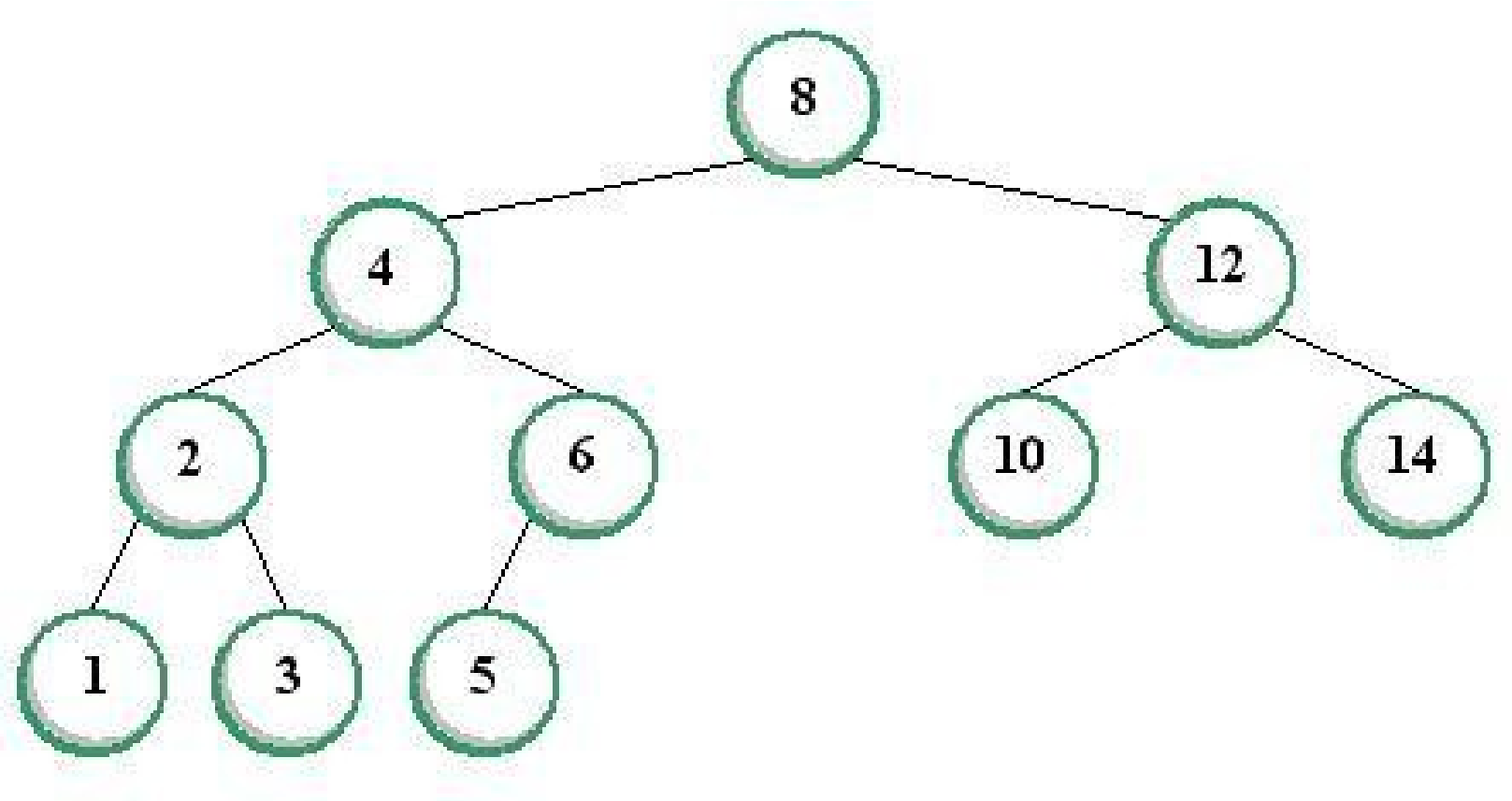


strictly binary tree

Tipos: Árvore Binária Cheia

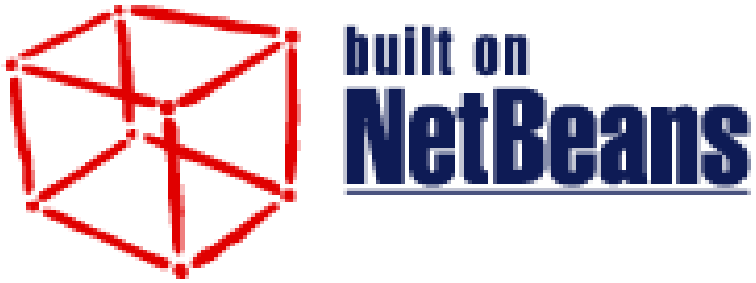


Tipos: Árvore Binária Completa

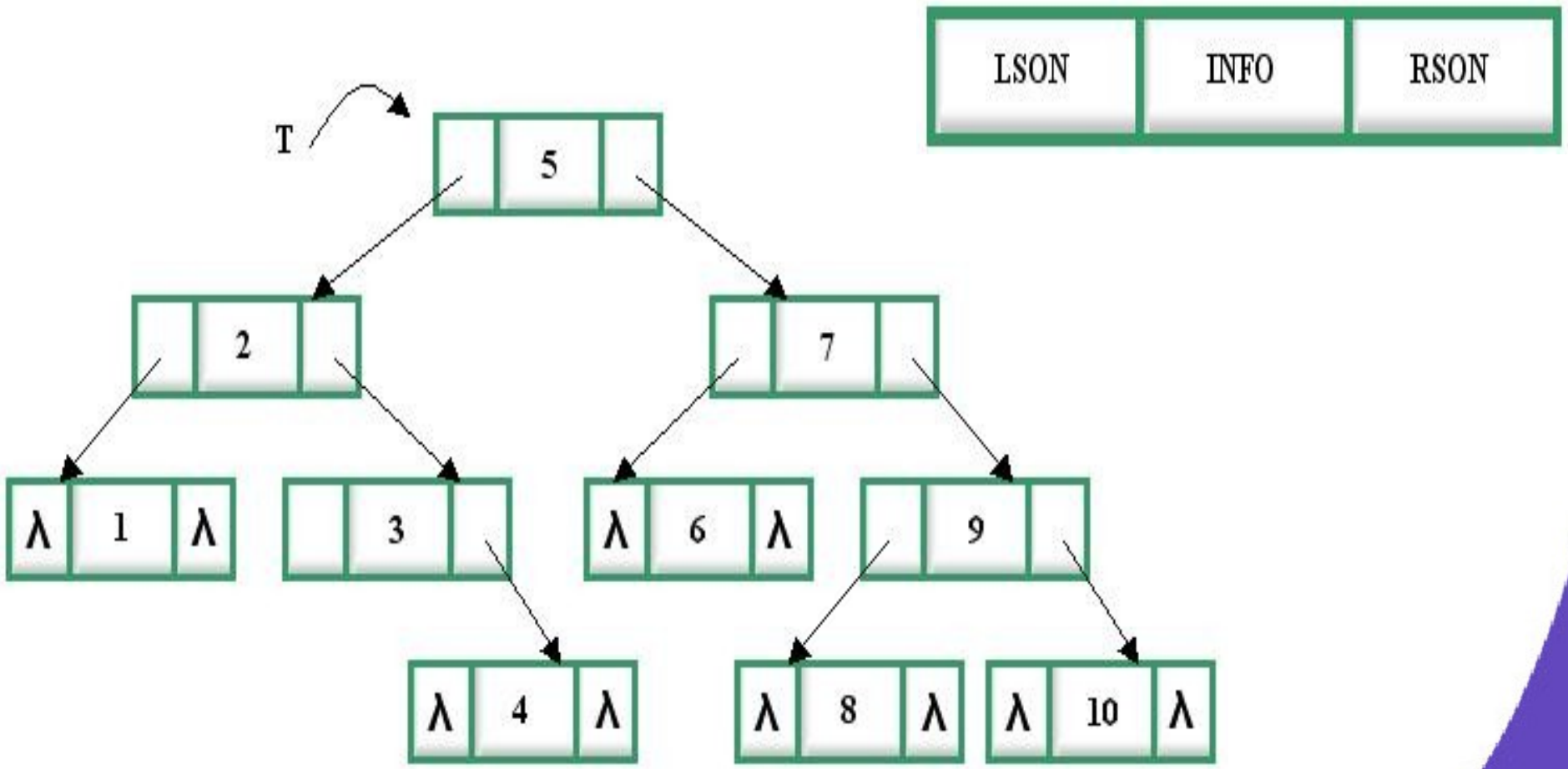


Representação

- Passaremos agora para o NetBeans



Representação

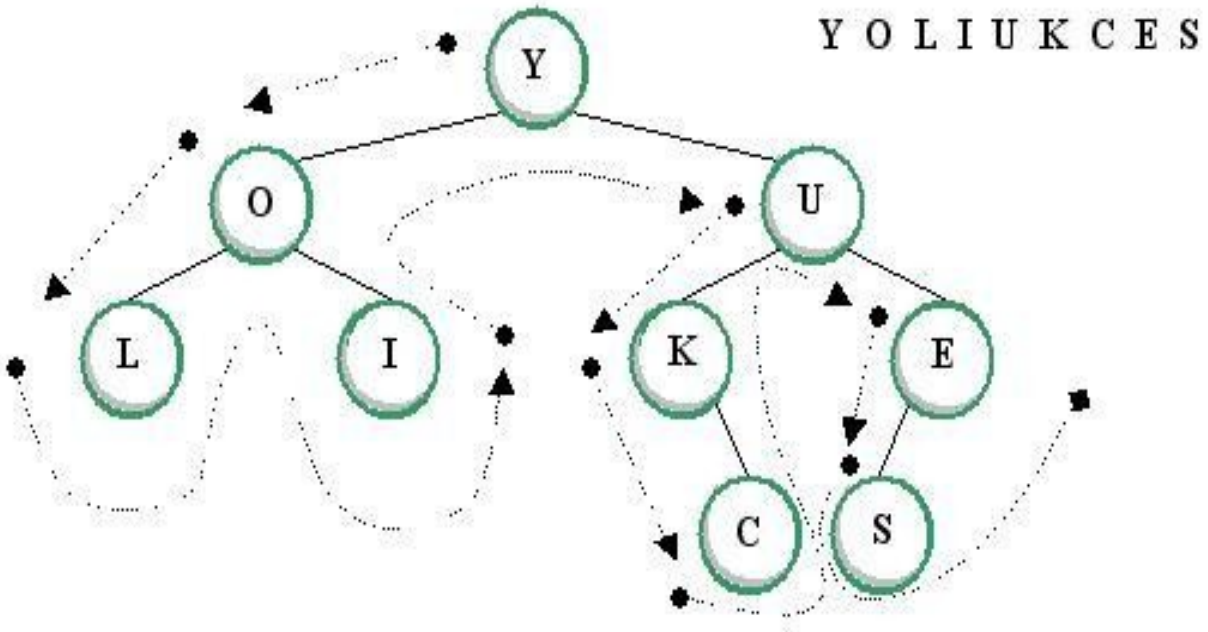


Busca

- Procedimento que pesquisa os *nodes* de uma árvore binária
- Maneira linear
- Pré-ordem, Em ordem e Pós-ordem

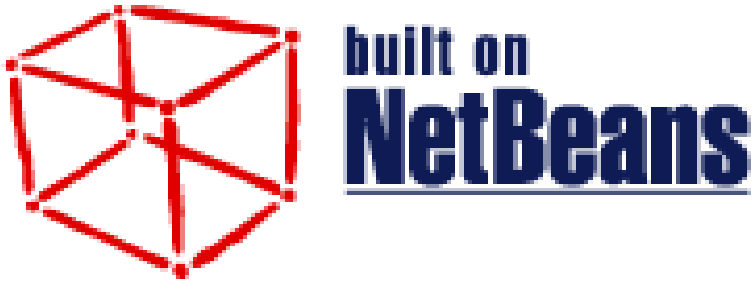
Busca: Pré-ordem

- Visite a raiz
- Percorra a sub-árvore da esquerda em pré-ordem
- Percorra a sub-árvore da direita em pré-ordem



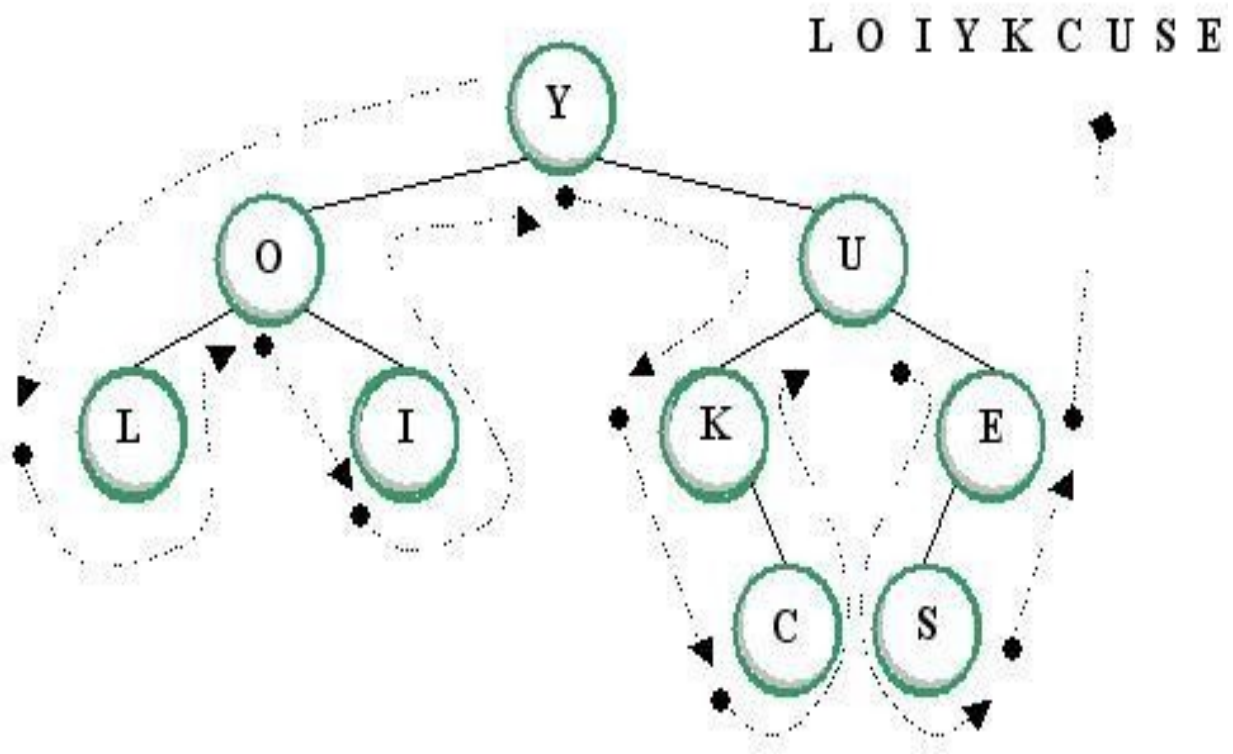
Busca: Pré-ordem

- Passaremos agora para o NetBeans



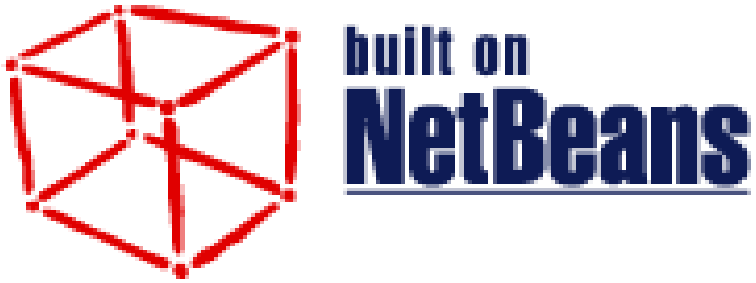
Busca: Em ordem

- Percorra a sub-árvore da esquerda em ordem
- Visite a raiz
- Percorra a sub-árvore da direita em ordem



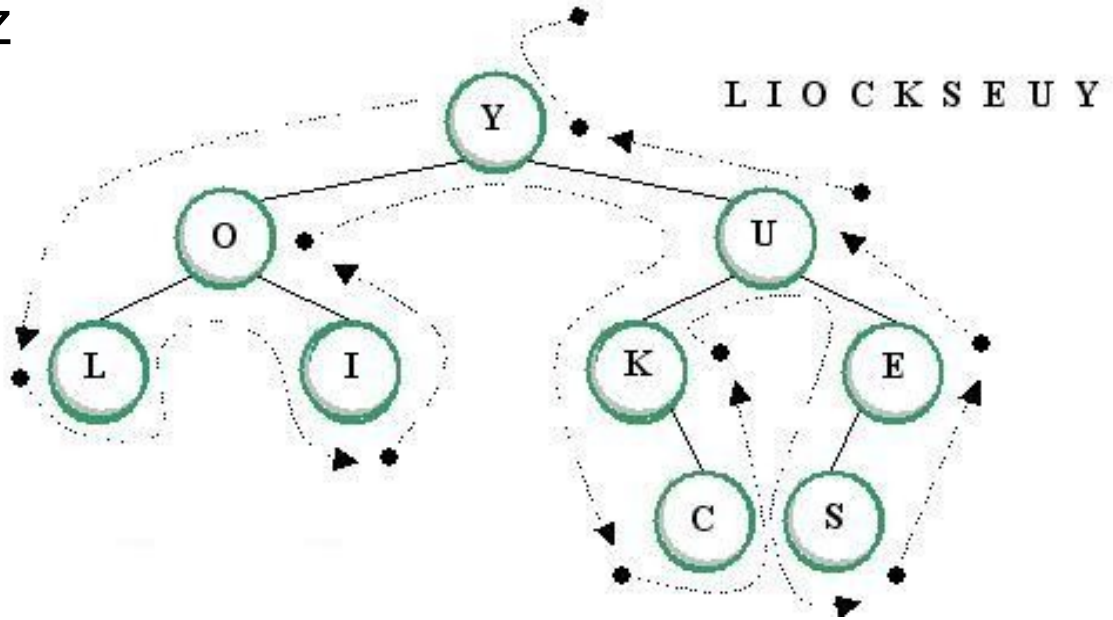
Busca: Em ordem

- Passaremos agora para o NetBeans



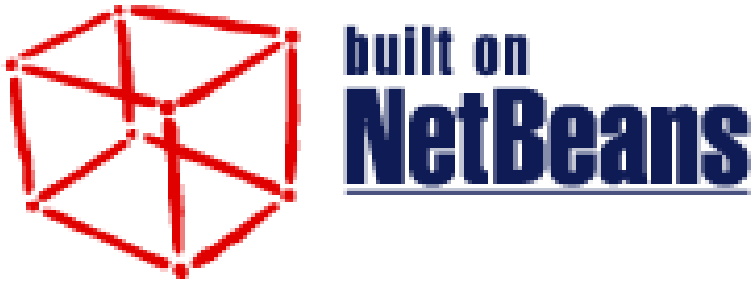
Busca: Pós-ordem

- Se árvore binária vazia, não faça nada (fim da varredura)
- Caso contrário:
  - Percorra a sub-árvore da esquerda em pós-ordem
  - Percorra a sub-árvore da direita em pós-ordem
  - Visite a raiz



Busca: Pós-ordem

- Passaremos agora para o NetBeans



Busca: Ocorrência de Pesquisas

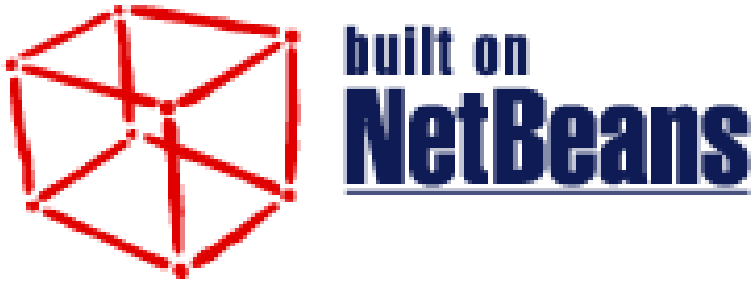
- Pré-ordem e em ordem:
  - Na descida à esquerda quando a sub-árvore da esquerda é percorrida
  - Na subida à esquerda depois que a sub-árvore da esquerda é percorrida
- Pós-ordem:
  - Na descida à esquerda quando a sub-árvore da esquerda é percorrida
  - Na subida à esquerda quando a sub-árvore da esquerda já foi percorrida
  - Na subida à direita quando a sub-árvore da direita já foi percorrida

Aplicação de Busca: Duplicação de uma Árvore Binária

- Percorra a sub-árvore da esquerda do *node* α em pós-ordem e faça uma cópia dela
- Percorra a sub-árvore da direita do *node* α em pós-ordem e faça uma cópia dela
- Faça uma cópia do *node* e anexe as cópias de suas sub-árvores da esquerda e da direita

Aplicação de Busca: Duplicação de uma Árvore Binária

- Passaremos agora para o NetBeans

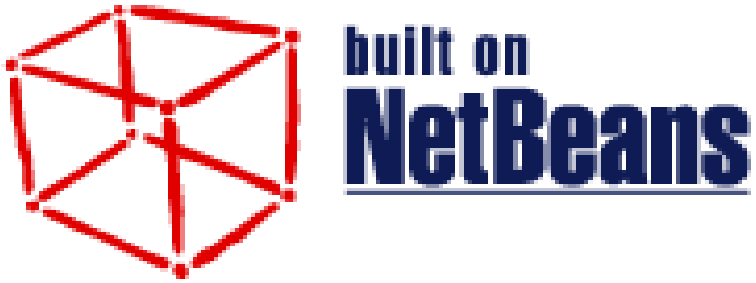


Aplicação de Busca: Equivalência entre Duas Árvores Binárias

- Verifique se o *node* α e o *node* β contêm os mesmos dados
- Percorra as sub-árvores da esquerda dos *nodes* α e β em pré-ordem e verifique se são equivalentes
- Percorra as sub-árvores da direita dos *nodes* α e β em pré-ordem e verifique se são equivalentes

Aplicação da Busca: Equivalência entre Duas Árvores Binárias

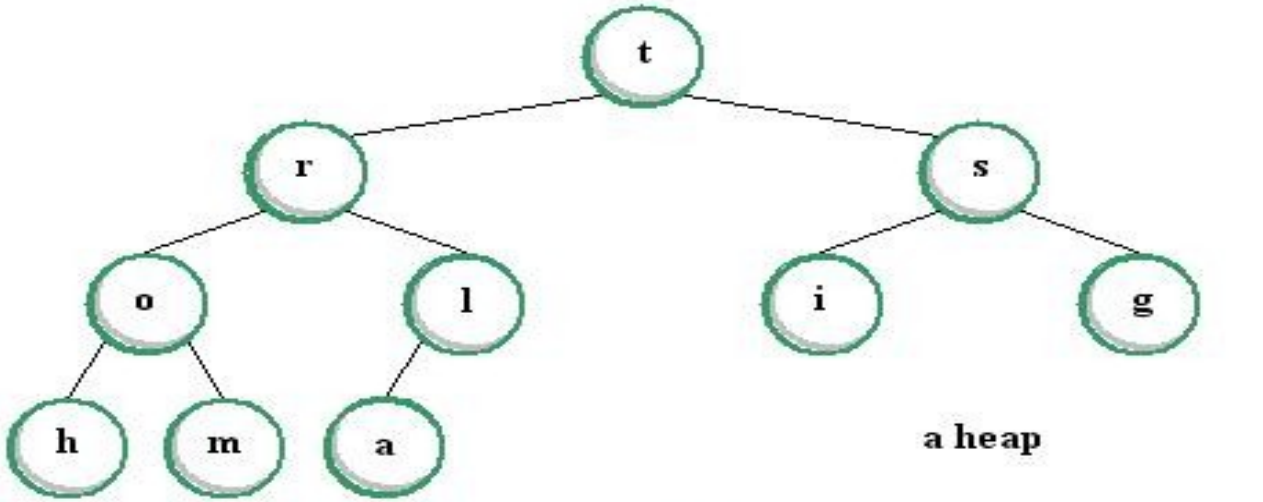
- Passaremos agora para o NetBeans



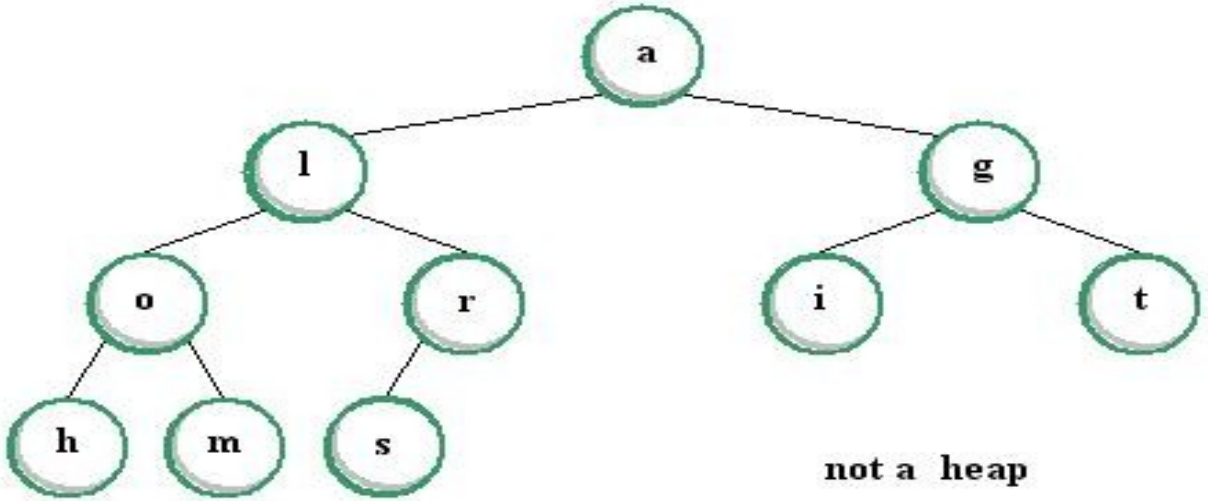
Aplicação de Árvore Binária: *Heaps* e *Heapsort*

- Completa quando tem elementos armazenados em seus *nodes*
- Satisfaz a propriedade da **ordem-*heap***
- Elementos armazenados em um *heap* satisfazem a ordem total:
    - Transitividade
    - Tricotomia

Heaps

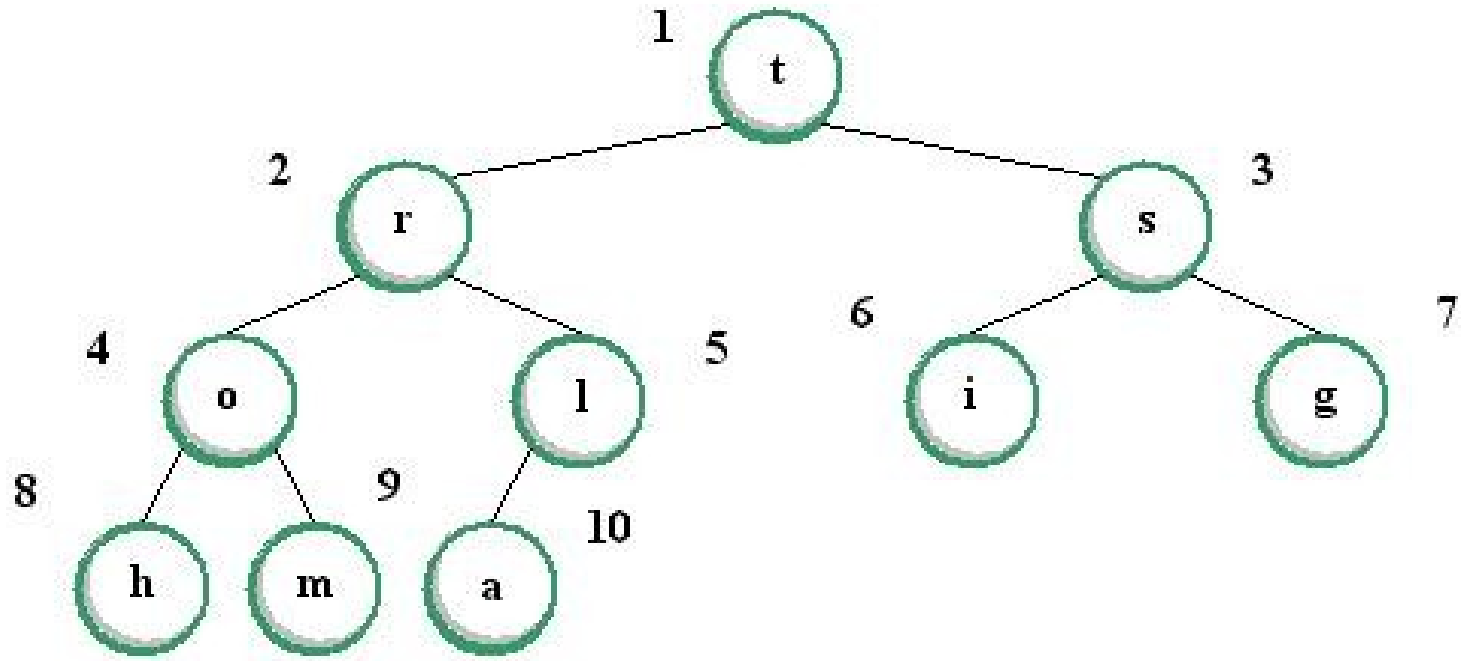


a heap



not a heap

Representação Seqüencial

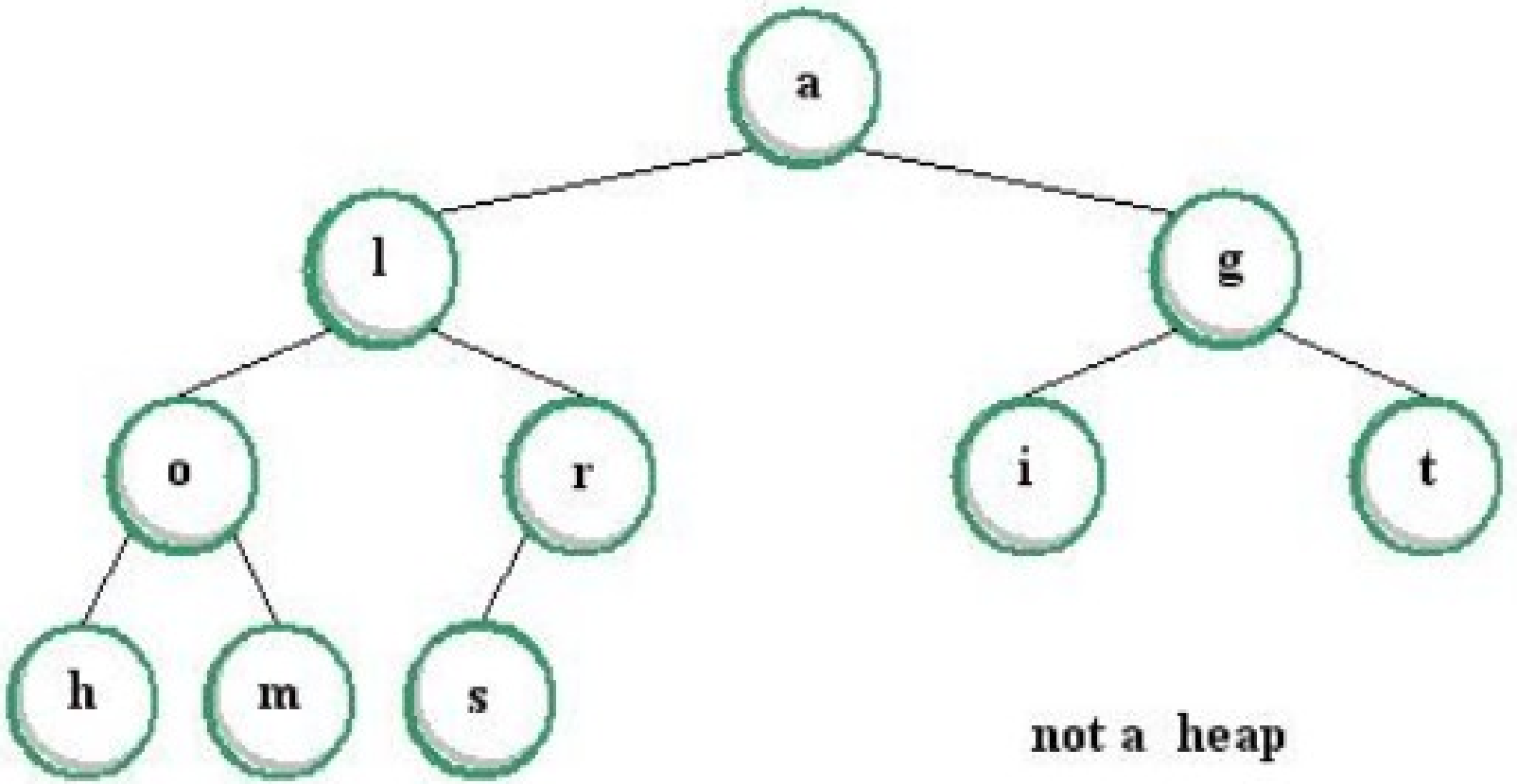


i →	1	2	3	4	5	6	7	8	9	10
KEY :	t	r	s	o	l	i	g	h	m	a

Representação Seqüencial

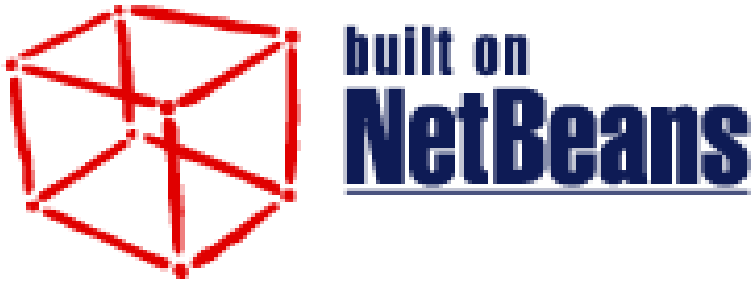
- Propriedades de um *heap* representado como uma árvore binária completa:
- Se **2i ≤ n**, o *child* esquerdo do *node* **i** é **2i**; senão, **i** não tem *child* esquerdo
- Se **2i < n** (**2i + 1 ≤ n** em Java), o *child* direito de **i** é **2i + 1**; senão, **i** não tem *child* direito
- Se **1 < i ≤ n**, o pai do *node* **i** é $\lfloor (i)/2 \rfloor$

Heaps e Shift-Up



Heaps e Shift-Up

- Passaremos agora para o NetBeans



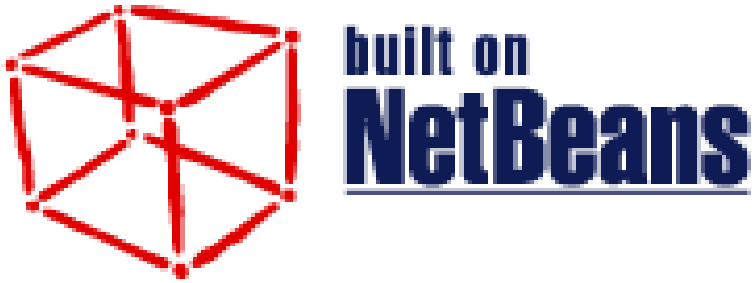
Heapsort

- Algoritmo *Heapsort* apresentado em 1964 por R. W. Floyd e J. W. J. Williams

1. Atribua as chaves a serem ordenadas aos *nodes* de uma árvore binária completa
2. Converta esta árvore binária em um *heap* aplicando o método *shift-up* aos seus *nodes* em ordem reversa de nível
3. Repita o seguinte até que o *heap* esteja vazio:
   a) Remova a chave na raiz do *heap* (o menor valor no *heap*) e coloque-o na saída
   b) Extraia do *heap* o *node* folha mais à direita do nível mais baixo, obtenha sua chave e armazene-a na raiz do *heap*
   c) Aplique *shift-up* à raiz para converter a árvore binária em um *heap* novamente

Heapsort

- Passaremos agora para o NetBeans



Sumário

- Definições e Conceitos Relacionados
  - Propriedades e Tipos de Árvores Binárias
- Representação das Árvores Binárias
- Percorrendo Árvores Binárias
  - Busca pré-ordem, em ordem e pós-ordem
- Aplicações de Busca em Árvores Binárias
  - Duplicação de uma Árvore Binária
  - Equivalência entre Duas Árvores Binárias
- Aplicação de Árvore Binária: *Heaps* e o Algoritmo *Heapsort*
  - *Shift-Up*, Representação Seqüencial de uma Árvore Binária Completa e Algoritmo *Heapsort*

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**