
Árvores

Fabrício J. Barth

BandTec - Faculdade de Tecnologia Bandeirantes

Setembro de 2011

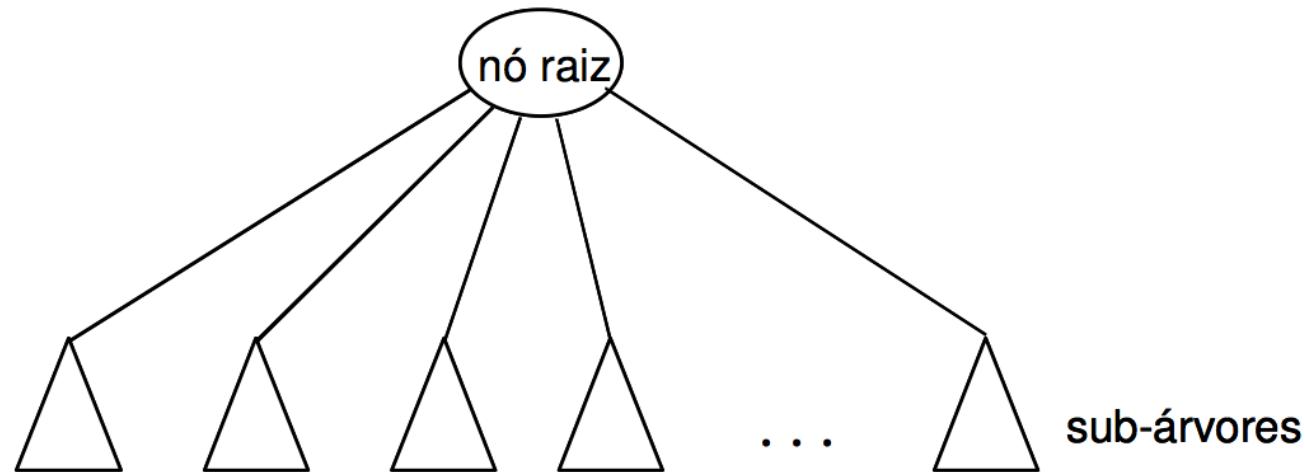
Tópicos

- Introdução
- Árvores binárias
 - ★ Implementação em Java
 - ★ Ordens de percurso em árvores binárias
 - ★ Altura de uma árvore
- Árvores com número variável de filhos
 - ★ Implementação em Java
 - ★ Altura de uma árvore
 - ★ Topologia

Introdução - Árvore

Um conjunto de nós tal que:

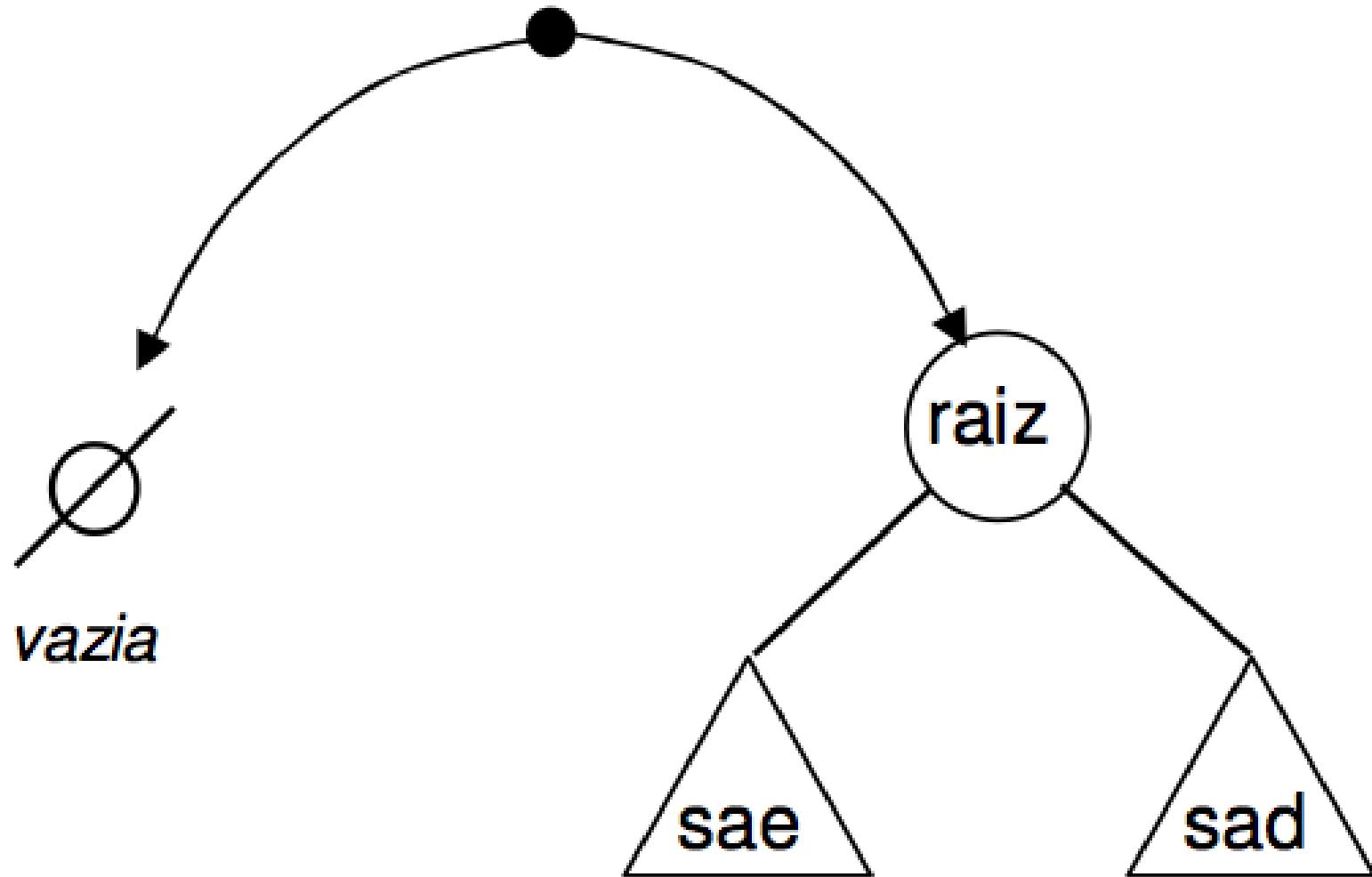
- existe um nó r , denominado **raiz**, com zero ou mais sub-árvore, cujas raízes estão ligadas a r .



- os nós raízes destas sub-árvore s̄o os **filhos** de r .
- os **nós internos** da árvore s̄o os nós com filhos.
- as **folhas** ou **nós externos** da árvore s̄o os nós sem filhos.

Árvores Binárias

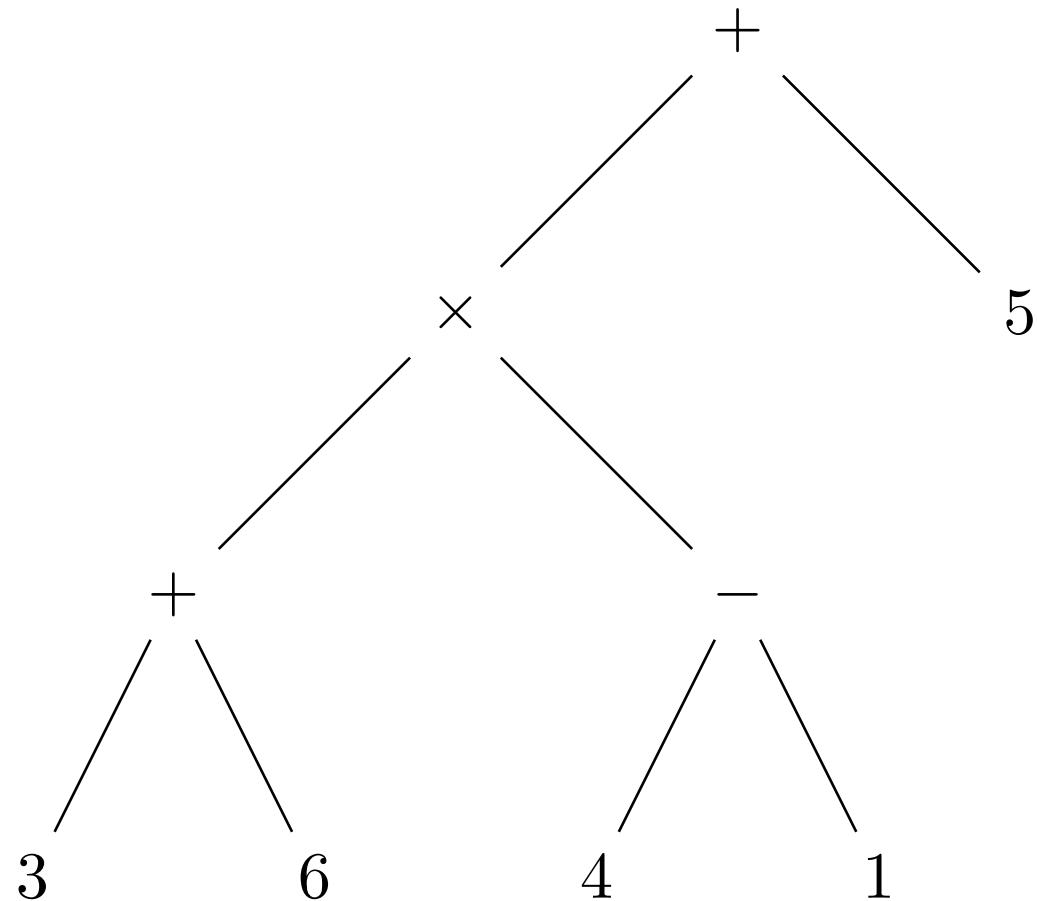
- uma árvore em que cada nodo tem zero, um ou dois filhos.
- uma árvore binária é:
 - ★ uma árvore vazia, ou;
 - ★ um nodo raiz com duas sub-árvores: a sub-árvore da direita (sad); a sub-árvore da esquerda (sae).



Árvores Binárias - Exemplo

Árvore binária representando expressões aritméticas:

- nós folhas representam operandos
- nós internos operadores
- exemplo: $(3 + 6) \times (4 - 1) + 5$



Árvores binárias - implementação

- A representação de uma árvore é feita através de uma referência para o nodo raiz da árvore.
- Representação de um nodo da árvore:

```
1 public class Nodo {  
2     private char c;  
3     private Nodo esq;  
4     private Nodo dir;  
5 }
```

Interface do tipo Árvore

- Nodo criaSemFilhos(char c)
- Nodo criaComFilhos(char c, Nodo esq, Nodo dir)
- boolean vazia(Nodo n)
- boolean pertence(char c, Nodo n)
- void imprime(Nodo n)

Árvores Binárias - implementação

```
1  public static Nodo criaSemFilhos(char c) {  
2      return new Nodo(c,null,null);  
3  }
```

Árvores Binárias - implementação

```
1  public static Nodo criaComFilhos(char c,
2                                  Nodo esq,
3                                  Nodo dir) {
4      return new Nodo(c,esq,dir);
5  }
```

Árvores Binárias - implementação

```
1  public static boolean vazia(Nodo n) {  
2      return n == null;  
3  }
```

Árvores Binárias - implementação

```
1  public static boolean pertence(char c, Nodo n) {  
2      if(vazia(n))  
3          return false;  
4      else  
5          return c == n.getc() ||  
6              pertence(c,n.getEsq()) ||  
7              pertence(c,n.getDir());  
8  }
```

Árvores Binárias - implementação

```
1  public static void imprime(Nodo n) {  
2      if(!vazia(n)){  
3          System.out.print(n.getC()+" - ");  
4          imprime(n.getEsq());  
5          imprime(n.getDir());  
6      }  
7  }
```

Árvores Binárias - exemplo de uso

```
1  public Main(){
2      Nodo raiz = ArvoreBinaria.criaComFilhos('a',
3                                         ArvoreBinaria.criaSemFilhos('d'),
4                                         ArvoreBinaria.criaSemFilhos('e'));
5
6      if(ArvoreBinaria.pertence('f', raiz))
7          System.out.println("encontrou");
8      else
9          System.out.println("nao encontrou");
10
11     ArvoreBinaria.imprime(raiz);
12 }
```

Ordens de percurso em árvores binárias

```
1  public static void imprimePreOrdem(Nodo n) {
2      if(!vazia(n)){
3          System.out.print(n.getC()+" - ");
4          imprimePreOrdem(n.getEsq());
5          imprimePreOrdem(n.getDir());
6      }
7  }
8
9  public static void imprimeInOrdem(Nodo n) {
10     if(!vazia(n)){
11         imprimeInOrdem(n.getEsq());
12         System.out.print(n.getC()+" - ");
13         imprimeInOrdem(n.getDir());
14     }
15 }
16
17 public static void imprimePosOrdem(Nodo n) {
18     if(!vazia(n)){
19         imprimePosOrdem(n.getEsq());
20         imprimePosOrdem(n.getDir());
21         System.out.print(n.getC()+" - ");
22     }
23 }
```

Exemplos concretos do uso de árvores

- Estrutura de diretórios e arquivos de um sistema operacional
- Análise semântica de equações matemáticas
- *É uma estrutura usada por vários outros algoritmos...*

Altura de árvore binárias

- Propriedade fundamental de árvores: só existe um caminho da raiz para qualquer nó.
- Altura de uma árvore: comprimento do caminho mais longo da raiz até uma das folhas.
 - ★ a altura de uma árvore com um único nó raiz é zero
 - ★ a altura de uma árvore vazia é -1

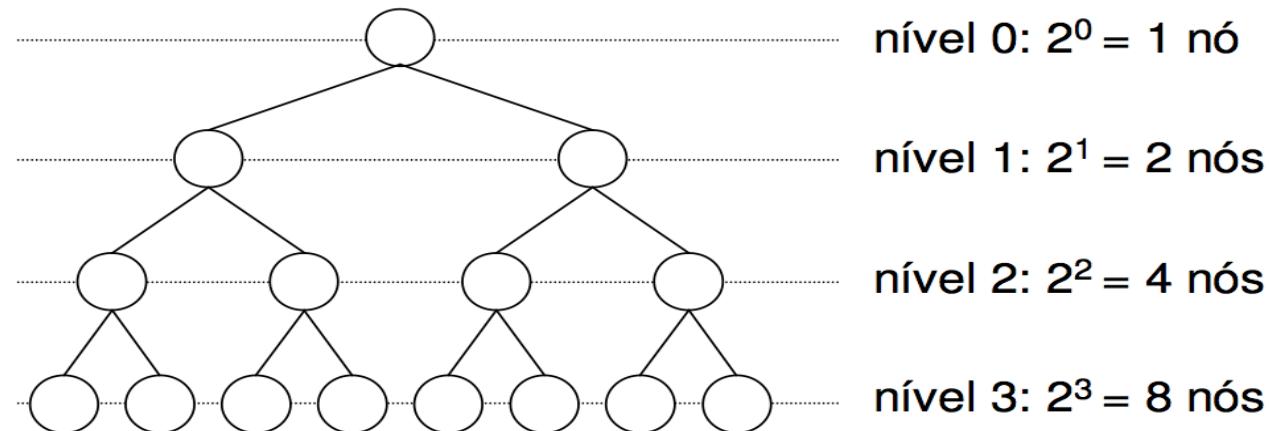
Altura de árvore binárias

- Nível de um nó:
 - ★ a raiz está no nível 0, seus filhos diretos estão no nível 1, ...
 - ★ o último nível da árvore é a altura da árvore.

Altura de árvore binárias

- Árvore cheia:
 - ★ todos os seus nós internos tem duas sub-árvores associadas.
 - ★ número n de nós de uma árvore cheia de altura h é igual a

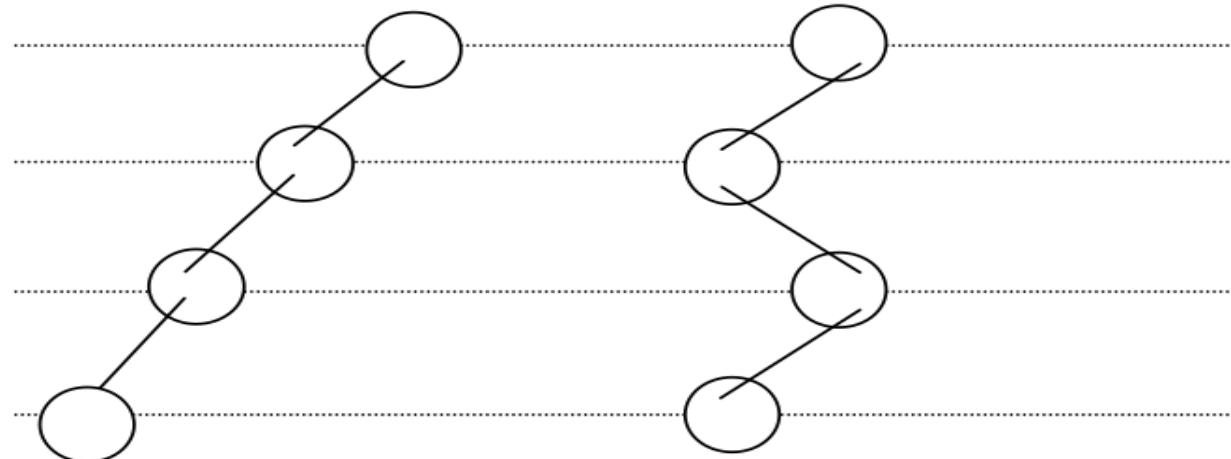
$$n = s^{k+1} - 1 \quad (1)$$



Altura de árvore binárias

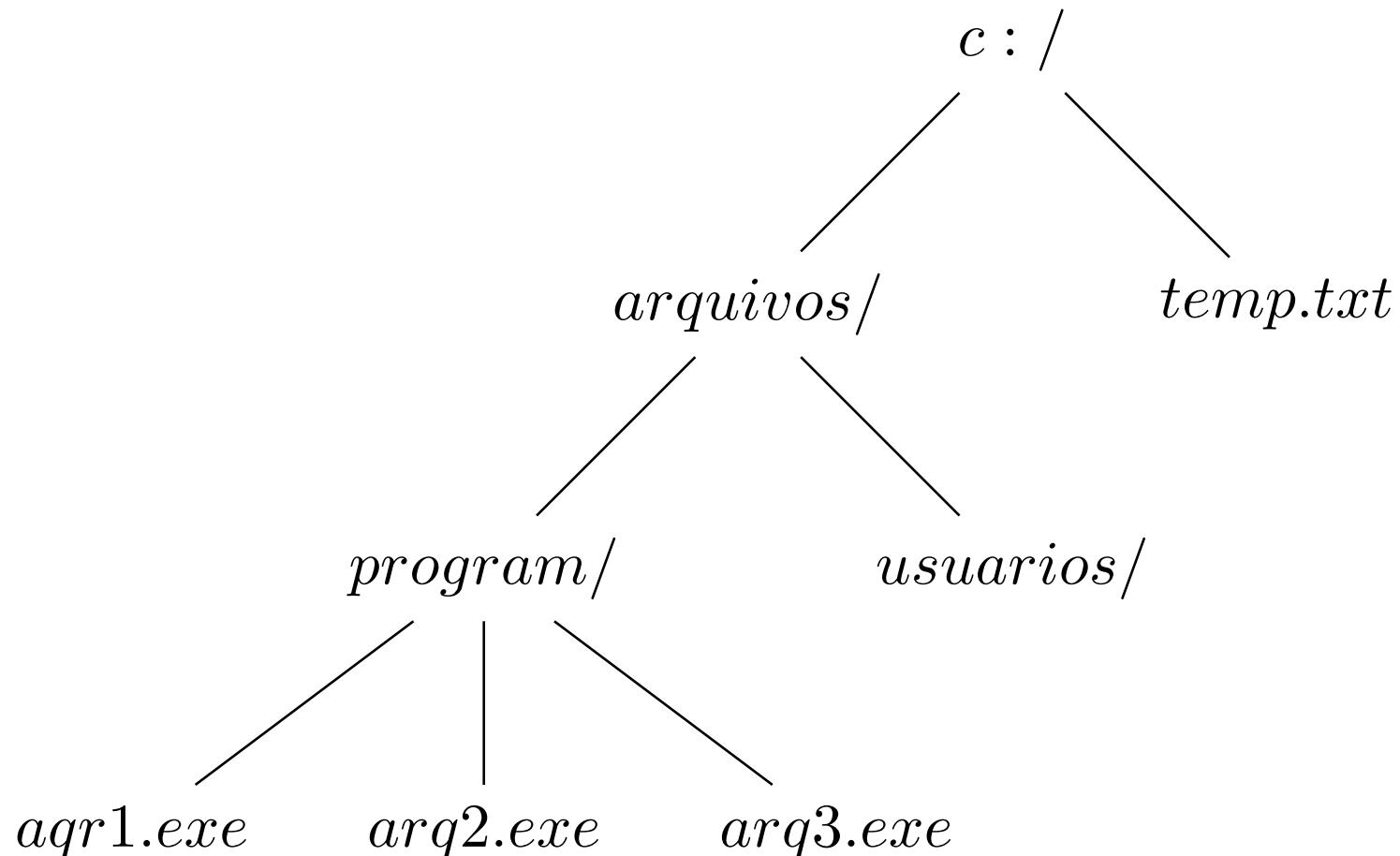
- Árvore degenerada:
 - ★ todos os seus nós internos tem uma única sub-árvore associada.
 - ★ número n de nós de uma árvore cheia de altura h é igual a

$$n = h + 1 \quad (2)$$



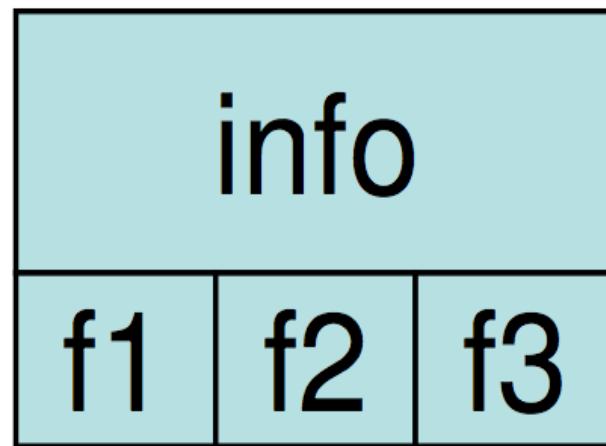
Árvores com número variável de filhos

Cada nó pode ter mais que duas sub-árvores associadas.



Árvores com até 3 nós

```
1 public class NodoTernario {  
2     private String conteudo;  
3     private NodoTernario[] filhos =  
4         new NodoTernario[3];  
5 }
```



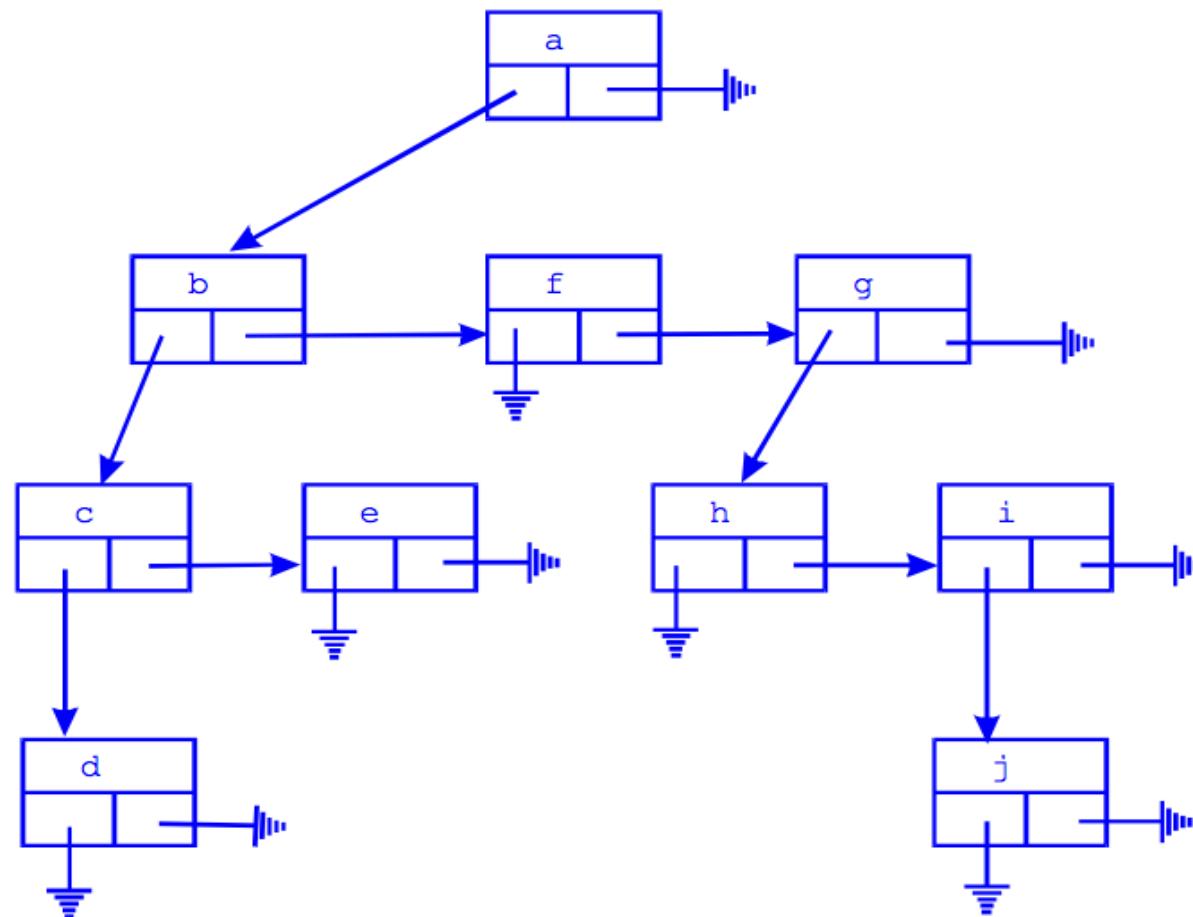
Procurando por um elemento

```
1  public static boolean pertence(String c, NodoTernario n)
2      if(vazia(n)) return false; else{
3          if (c.equals(n.getConteudo())){
4              return true;
5          }else{
6              NodoTernario [] f = n.getFilhos();
7              for(int i=0; i<f.length; i++){
8                  if(pertence(c,f[i])){return true;}
9              }
10             return false;
11     }}
```

Árvores com até N nós fixos

```
1 public class NodoTernario {  
2     private String conteudo;  
3     private NodoTernario[] filhos =  
4         new NodoTernario[N];  
5 }
```

Representação de árvore com número variável de filhos



Funções de uma árvore n-ária

- Nodo create(char c)
- void insert(Nodo a, Nodo sub_a)
- void print(Nodo a)
- boolean search(Nodo n, char c)
- int getAltura(Nodo n)

Nodo de uma árvore n-ária

```
1 public class Nodo {  
2  
3     private char c;  
4     private Nodo prim;  
5     private Nodo prox;  
6  
7 }
```

Criação de nodos em uma árvore n-ária

```
1  public Nodo create(char c){  
2      Nodo n = new Nodo();  
3      n.setC(c);  
4      n.setPrim(null);  
5      n.setProx(null);  
6      return n;  
7  }
```

Inserção em uma árvore n-ária

```
1  /*
2   * inseri uma nova sub-arvore como filha
3   * de um dado no
4   */
5  public void insert(Nodo a, Nodo sub_a){
6      //inserindo no inicio.
7      sub_a.setProx(a.getPrim());
8      a.setPrim(sub_a);
9  }
```

Impressão de uma árvore n-ária

```
1  public void print(Nodo a){  
2      System.out.println(a.getc());  
3      for(Nodo p = a.getPrim(); p!=null; p=p.getProx()){  
4          print(p); //imprime cada sub-arvore filha  
5      }  
6  }
```

Busca em uma árvore n-ária

```
1  public boolean search(Nodo n, char c){  
2      if(n.getc()==c){  
3          return true;  
4      }else{  
5          for(Nodo p=n.getPrim(); p!=null; p=p.getProx()){  
6              if(search(p,c))  
7                  return true;  
8          }  
9      }  
10     return false;  
11 }
```

Atributos de uma árvore n-ária

```
1  public int getAltura(Nodo n){  
2      int hmax = -1; /*caso de arvore sem filhos*/  
3      for(Nodo p=n.getPrim(); p!=null; p=p.getProx()){  
4          int h = getAltura(p);  
5          if(h > hmax)  
6              hmax = h;  
7      }  
8      return hmax + 1;  
9  }
```

Exemplo de construção e utilização de uma árvore n-ária

```
1  public Main(){  
2      //cria nodos como folha  
3      ArvoreNaria arv = new ArvoreNaria();  
4      Nodo a = arv.create('a');  
5      Nodo b = arv.create('b');  
6      Nodo c = arv.create('c');  
7      Nodo d = arv.create('d');  
8      Nodo e = arv.create('e');  
9      Nodo f = arv.create('f');
```

```
1      //monta a hierarquia
2      arv.insert(a, b);
3      arv.insert(a, c);
4      arv.insert(c, d);
5      arv.insert(a, e);
6      arv.insert(d, f);
7
8      arv.print(a);
9      System.out.println(arv.search(a, 'd'));
10     System.out.println(arv.search(e, 'd'));
11     System.out.println(arv.getAltura(a));
12 }
```

Topologia Binária

- Representação de um nodo de uma árvore n-ária ≡ representação de um nodo de uma árvore binária.
- Um nodo possui a informação que deve ser armazenada e duas referências para sub-árvores:
 - ★ árvore binária: referências para as sub-árvores à esquerda e à direita.
 - ★ árvore n-ária: referências para a primeira sub-árvore filha e para a sub-árvore irmã.

Resumo

- Árvore **binária**.
- Árvore com **número variável de filhos** (n -ária).

Material de **consulta** e referência

- Capítulo 13 do livro: “Introdução a Estruturas de Dados” do Waldemar Celes, Renato Cerqueira e José Lucas Rangel.
- Imagens retiradas do site da disciplina de Programação II da PUC do Rio de Janeiro
<http://www.inf.puc-rio.br/inf1007/>.