Árvores

Fernando Lobo

Algoritmos e Estrutura de Dados

Árvores

- Uma árvore pode ser vista como uma generalização do conceito de lista.
- Usado para representar uma estrutura hierárquica.
  - Sistema de ficheiros num sistema operativo.
  - Organigrama de uma empresa.
  - etc.
- Também é usado como estrutura de dados para pesquisar informação de modo eficiente.

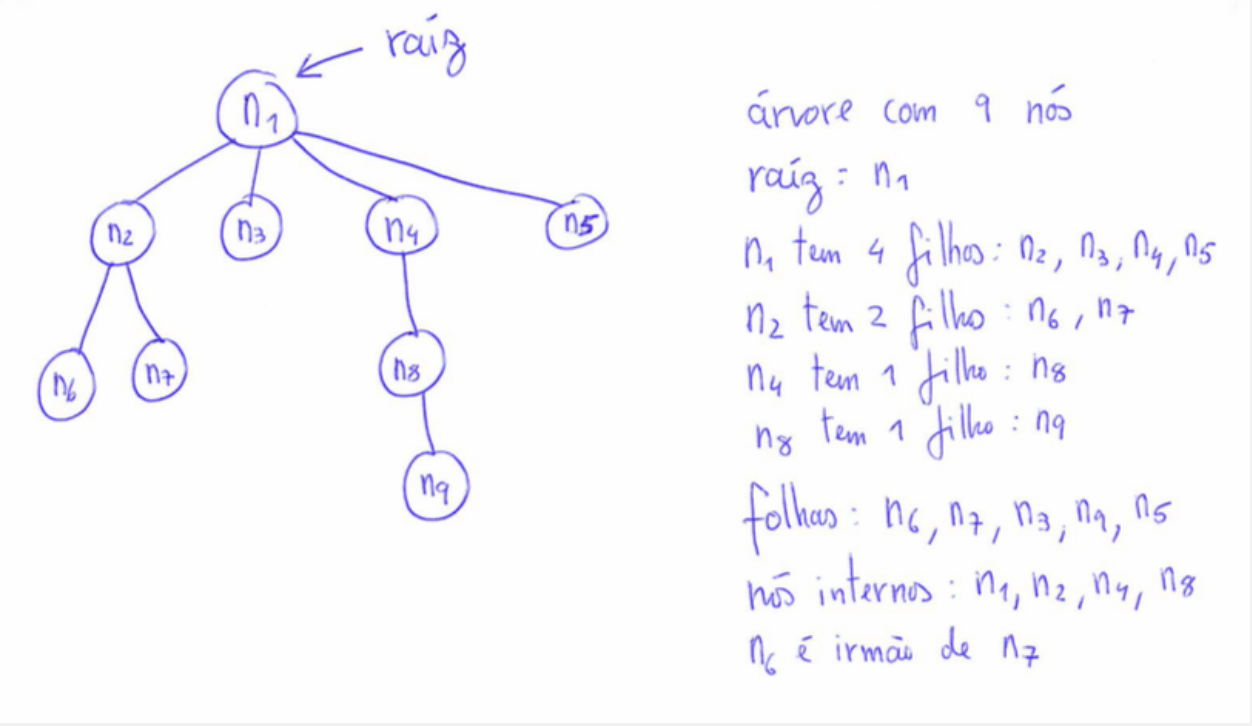
Terminologia

- nó
- raíz
- folha
- nó interno
- irmão
- caminho
- altura
- ascendente
- descendente

Terminologia

- A árvore é uma coleção de nós ligados entre si.
- Não há circuitos fechados.
- A terminologia é análoga à das árvores genealógicas.
  - Há um nó especial que se designa por *raíz*.
  - Um nó tem 0 ou mais filhos.
  - Todos os nós excepto a raíz têm 1 *pai*. A raíz não tem pai.
  - Um nó é *irmão* de outro nó se tiverem o mesmo pai.
  - Os nós sem filhos são *folhas* da árvore. Todos os outros são *nós internos*.

Exemplo



Terminologia (cont.)

- Seja $x_1, x_2, \ldots, x_k$, uma sequência de nós de tal forma que x_1 é pai de x_2, x_2 é pai de x_3, ..., e x_{k-1} é pai de x_k.
- A sequência $x_1, x_2, \ldots, x_k$ é um *caminho* na árvore de x_1 a x_k. Esse caminho tem comprimento $k - 1$. Um nó sozinho corresponde a um caminho de comprimento 0.
- A *altura* da árvore é o comprimento do caminho mais longo desde a raíz até uma folha.
- Se $x_1, x_2, \ldots, x_k$ for um caminho, então x_i é um *ascendente* de x_j, e x_j é um *descendente* de x_i, para todo $1 \leq i \leq j \leq k$.

No exemplo anterior...

- n_1, n_2, n_7, é um caminho na árvore.
- n_2 é ascendente de n_2, n_6, n_7.
- A raíz n_1 é ascendente de todos os nós.
- n_7 é descendente de n_7, n_2, n_1.
- A altura da árvore é 3.

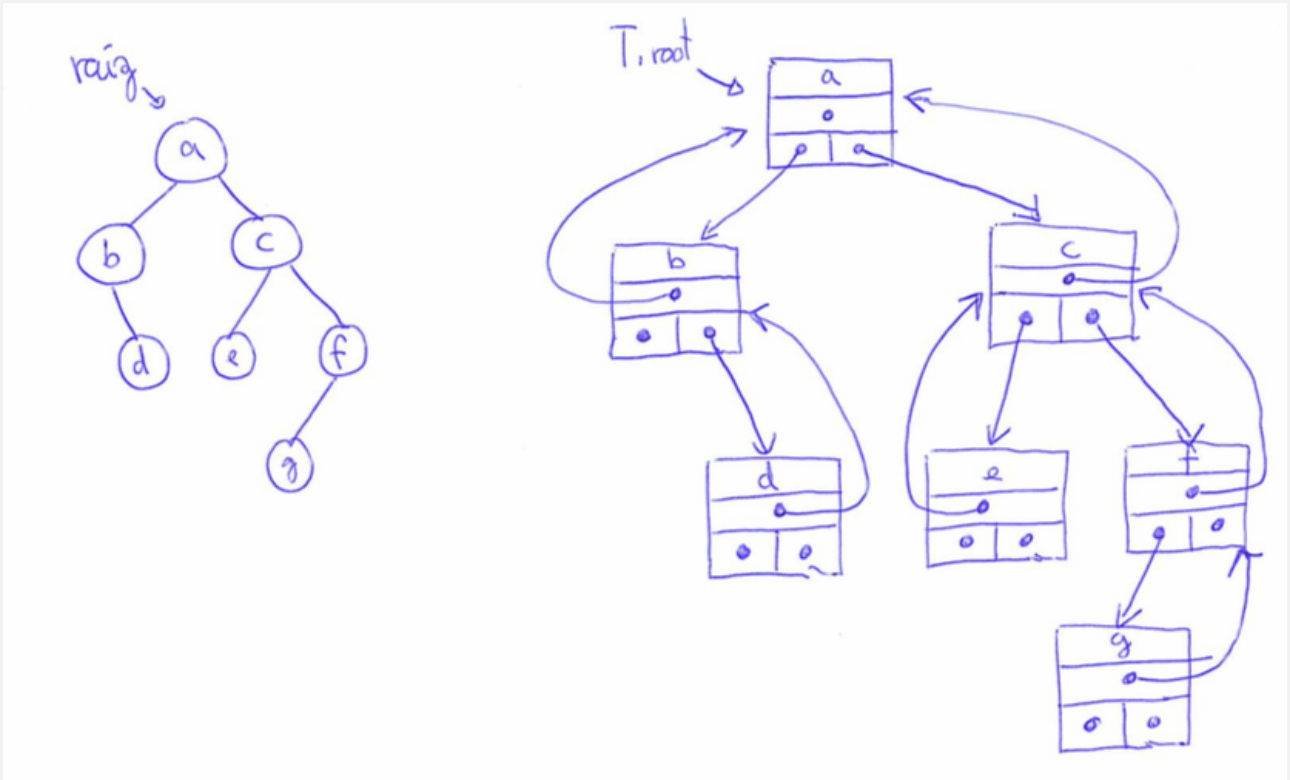
Casos particulares

- Árvore binária: cada nó tem no máximo 2 filhos.
- Árvore de aridade k: cada nó tem no máximo k filhos.

Árvore binária: implementação

- Cada nó é um objecto que contém:
  - **info**: informação que queremos guardar.
  - **left**: apontador para o filho esquerdo.
  - **right**: apontador para o filho direito.
  - **p**: apontador para o pai.
- Se T é uma árvore, $T.root$ aponta para a raíz.
- A raíz é o único nó da árvore que não tem pai: $T.root.p = \mathrm{NIL}$

Árvore binária: implementação



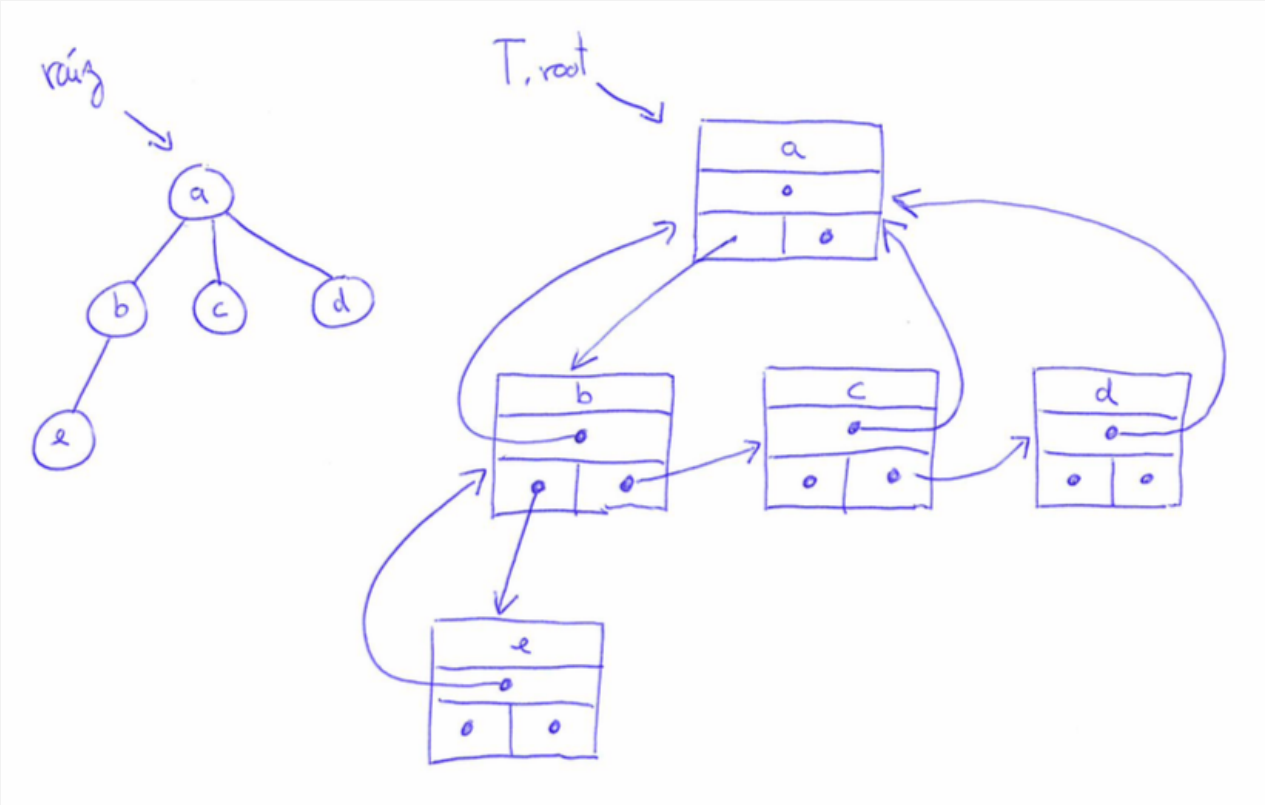
Árvore de aridade k: implementação

- Substitui-se os atributos *left* e *right* por k atributos.
- Ou melhor, por um array $child[1 \ldots k]$
- Cada elemento do array aponta para um filho (ou para NIL.)
- Tudo o resto é idêntico às árvores binárias.

Árvore genérica (sem limite de número de filhos)

- Usa-se a representação *left-child*, *right-sibling*.
  - x. *left-child* aponta para o filho mais à esquerda de x.
  - x. *right-sibling* aponta para o irmão de x imediatamente à sua direita.
  - Se x não tiver filhos, x. *left-child* aponta para NIL.
  - Se x for o filho mais à direita, x. *right-sibling* aponta para NIL.

Árvore genérica (sem limite de número de filhos)



Árvores ordenadas

- As árvores podem manter uma relação de ordem entre os nós.
- A ordem é da esquerda para a direita.
- Se x e y são irmãos, e x está à esquerda de y, então qualquer descendente de x é $\leq$ que qualquer descendente de y.

Tries

- Uma *trie* (em inglês pronuncia-se *try*) é uma árvore especial em que associamos a cada nó uma letra. É uma estrutura de dados muito usada para representar as palavras de um dicionário.
- Permite verificar de forma eficiente quais as palavras do dicionário que têm um determinado prefixo.
- A raíz da árvore não tem letra associada.
- O caminho desde a raíz até um nó x dá-nos uma sequência de letras que é um prefixo de um conjunto de palavras.
- Associa-se um bit a cada nó, para indicar se a sequência de letras desde a raíz até esse nó, é uma palavra do dicionário (1: sim; 0: não).

Trie