Árvores

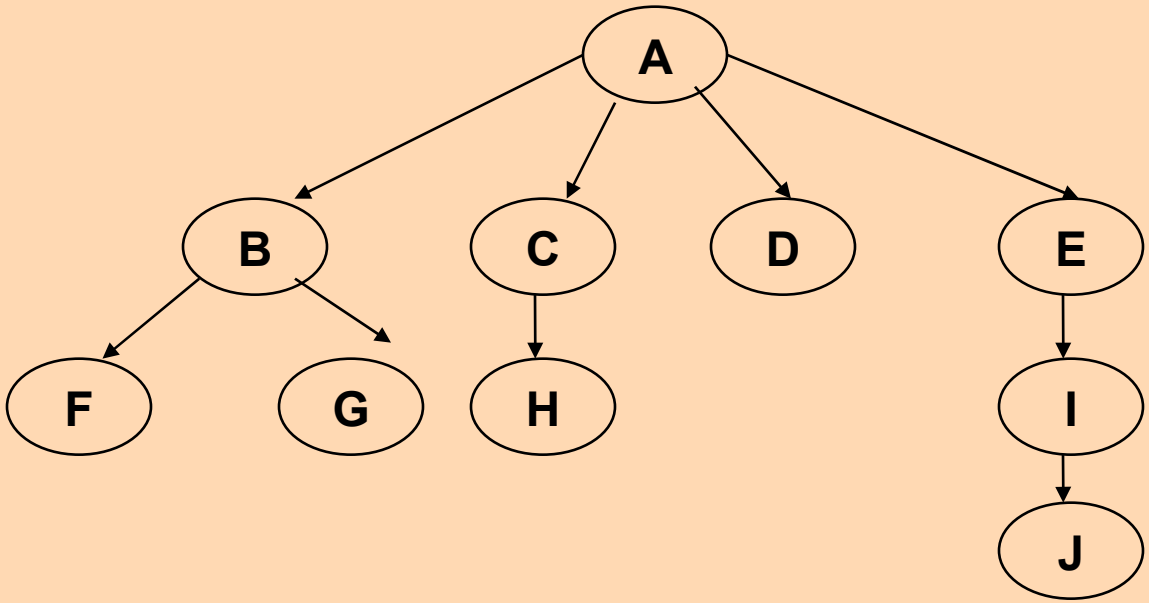
Algoritmos e Estruturas de Dados

2005/2006

FEUP

Árvores

- Conjunto de nós e conjunto de arestas que ligam pares de nós
  - Um nó é a *raiz*
  - Com excepção da raiz, todo o nó está ligado por uma aresta a 1 e 1 só nó (o pai)
  - Há um caminho único da raiz a cada nó; o *tamanho do caminho* para um nó é o número de arestas a percorrer

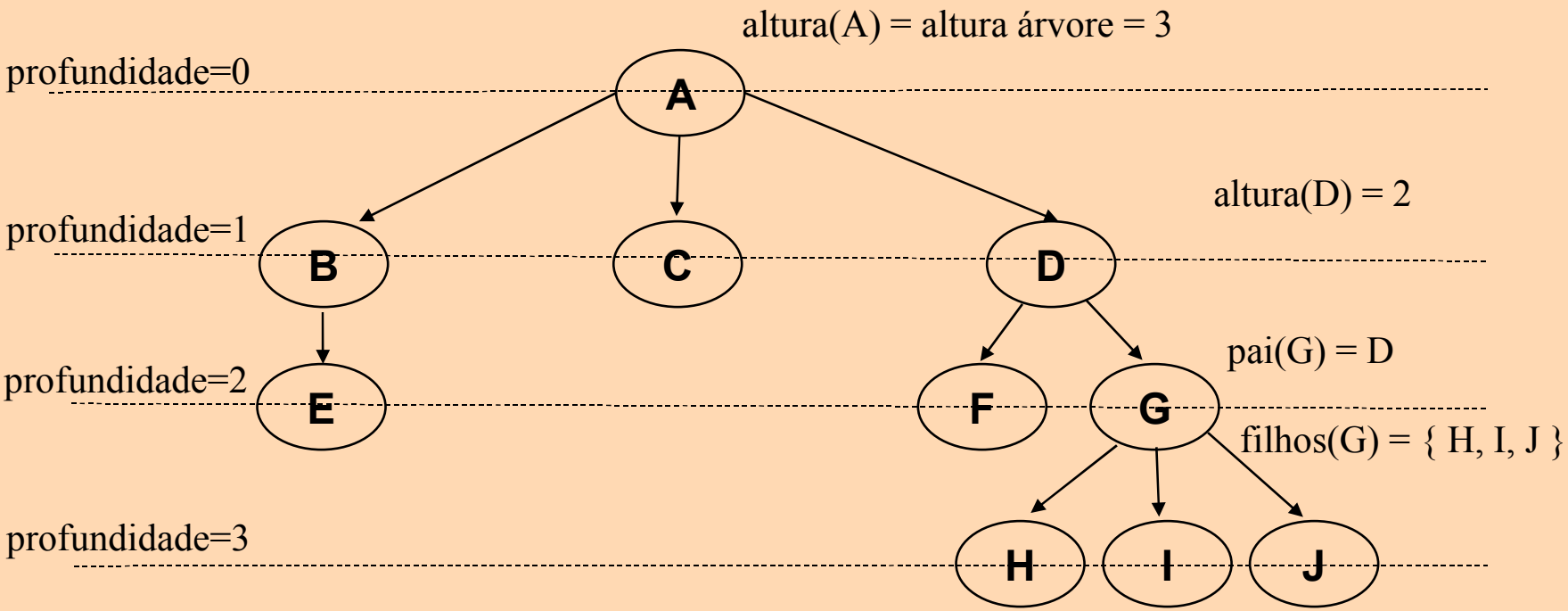


Nós sem descendentes: folhas

Árvores

- Ramos da árvore
  - Árvore de *N* nós tem *N-1* ramos
- Profundidade de um nó
  - Comprimento do caminho da raiz até ao nó
    - Profundidade da raiz é 0
    - Produndidade de um nó é 1 + a profundidade do seu pai
- Altura de um nó
  - Comprimento do caminho do nó até à folha a maior profundidade
    - Altura de uma folha é 0
    - Altura de um nó é 1 + a altura do seu filho de maior altura
  - Altura da árvore: altura da raiz
- Se existe caminho do nó u para o nó v
  - u é antepassado de v
  - v é descendente de u
- Tamanho de um nó: número de descendentes

Árvores


altura(A) = altura árvore = 3
profundidade=0
A
profundidade=1
B
C
D
altura(D) = 2
profundidade=2
E
F
G
pai(G) = D
filhos(G) = { H, I, J }
profundidade=3
H
I
J

Árvores binárias

- Uma árvore binária é uma árvore em que cada nó *não tem mais que dois filhos*

- Propriedades:
  - Uma árvore binária não vazia com profundidade *h* tem no mínimo *h+1*, e no máximo $2^{h+1}-1$ nós
  - A profundidade de uma árvore com *n* elementos (*n*>0) é no mínimo $log_2 n$, e no máximo *n-1*
  - A profundidade média de uma árvore de n nós é O(√n)

Árvores

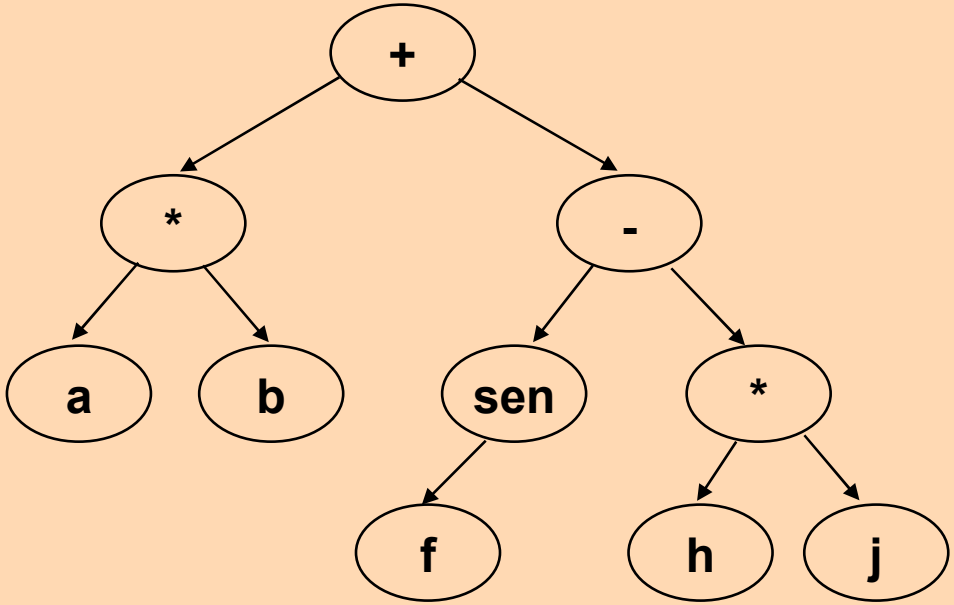
- Percorrer árvores

  Os elementos de uma árvore (binária) podem ser enumerados por quatro ordens diferentes. As três primeiras definem-se recursivamente:

  - ***Pré-ordem***: Primeiro a raiz, depois a sub-árvore esquerda, e finalmente a sub-árvore direita
  - ***Em-ordem***: Primeiro a sub-árvore esquerda, depois a raiz, e finalmente a sub-árvore direita
  - ***Pós-ordem***: Primeiro a sub-árvore esquerda, depois a sub-árvore direita, e finalmente a raiz
  - ***Por nível***: Os nós são processados por nível (profundidade) crescente, e dentro de cada nível, da esquerda para a direita

Árvores

- Percorrer árvores - exemplo



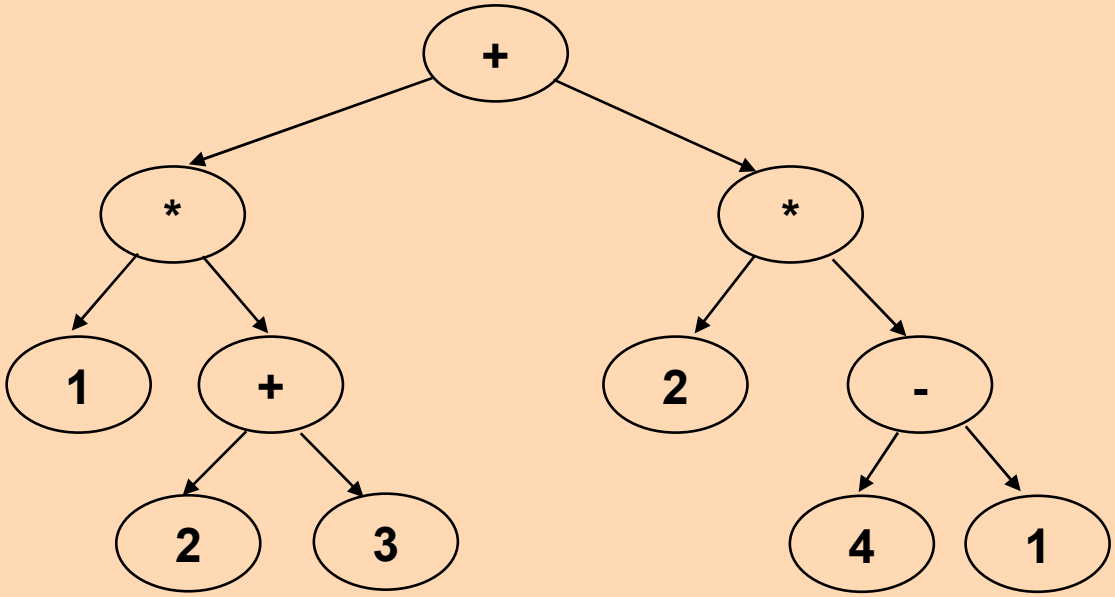
Pré-ordem	+ * a b – sen f * h j
Em-ordem	a * b + f sen – h * j
Pós-ordem	a b * f sen h j * - +
Por nível	+ * - a b sen * f h j

Árvores binárias: implementação

- Operações:
  - Criar uma árvore vazia
  - Determinar se uma árvore está vazia
  - Criar uma árvore a partir de duas sub-árvores
  - Eliminar os elementos da árvore (esvaziar a árvore)
  - Definir iteradores para percorrer a árvore
  - Imprimir uma árvore
  - ...

Árvores binárias: aplicações

Expressões aritméticas



Expressão = 1 * (2 + 3) + (2 * (4 – 1))

Árvores binárias: aplicações

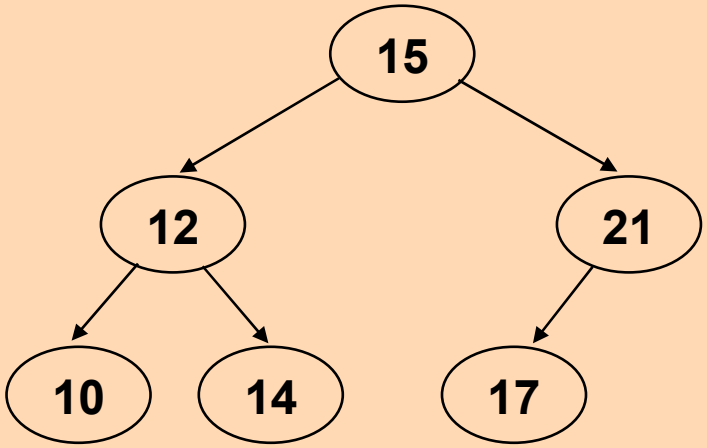
- *Construção da árvore de expressões*
  - O algoritmo é similar ao algoritmo de conversão infixa->RPN mas usa duas pilhas, uma para guardar os operadores e outra para guardar sub-árvores correspondentes a sub-expressões.
  - O algoritmo procede da seguinte forma:
    - Números são transformados em árvores de 1 elemento e colocados na pilha de operandos.
    - Operadores são tratados como no programa de conversão de infixa -> RPN (usando uma pilha apenas para operadores e '(').
    - Quando um operador é retirado da pilha, duas sub-árvores (operandos) são retirados da pilha de operandos e combinados numa nova sub-árvore, que por sua vez é colocada na pilha.
    - Quando a leitura da expressão chega ao fim, todos os operadores existentes na pilha são processados.

Árvores binárias de pesquisa

- *Árvore binária de pesquisa*

  Árvore binária, sem elementos repetidos, que verifica a seguinte propriedade:

  – Para **cada nó**, todos os valores da sub-árvore esquerda são menores, e todos os valores da sub-árvore direita são maiores, que o valor desse nó



Árvores binárias de pesquisa

- Estrutura linear com elementos ordenados
  - A pesquisa de elementos pode ser realizada em *O(log n)*
  - ... mas não inserção ou remoção de elementos

- Estrutura em árvore binária
  - pode manter o tempo de acesso logarítmico nas operações de inserção e remoção de elementos
  - ***Árvore binária de pesquisa***
    - mais operações do que árvore binária básica: pesquisar, inserir, remover
    - objectos nos nós devem ser comparáveis (*Comparable*)

Árvores binárias de pesquisa

- Pesquisa
  - usa a propriedade de ordem na árvore para escolher caminho, eliminando uma sub-árvore a cada comparação
- Inserção
  - como pesquisa; novo nó é inserido onde a pesquisa falha
- Máximo e mínimo
  - procura, escolhendo sempre a subárvore direita (máximo), ou sempre a sub-árvore esquerda (mínimo)
- Remoção
  - Nó folha : apagar nó
  - Nó com 1 filho : filho substitui o pai
  - Nó com 2 filhos: elemento é substituído pelo menor da sub-árvore direita (ou maior da esquerda); o nó deste tem no máximo 1 filho e é apagado.