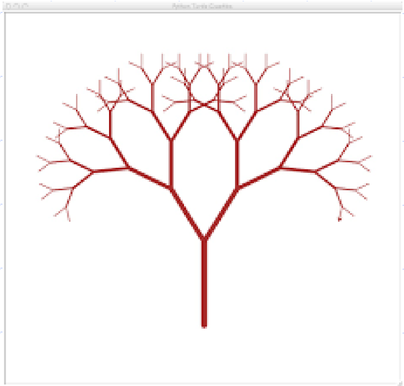
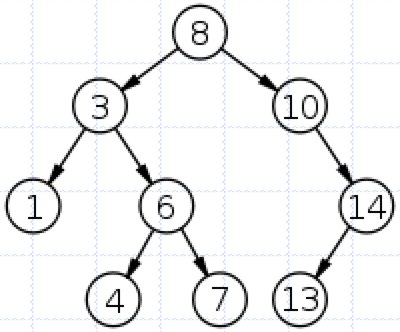
Binary Trees

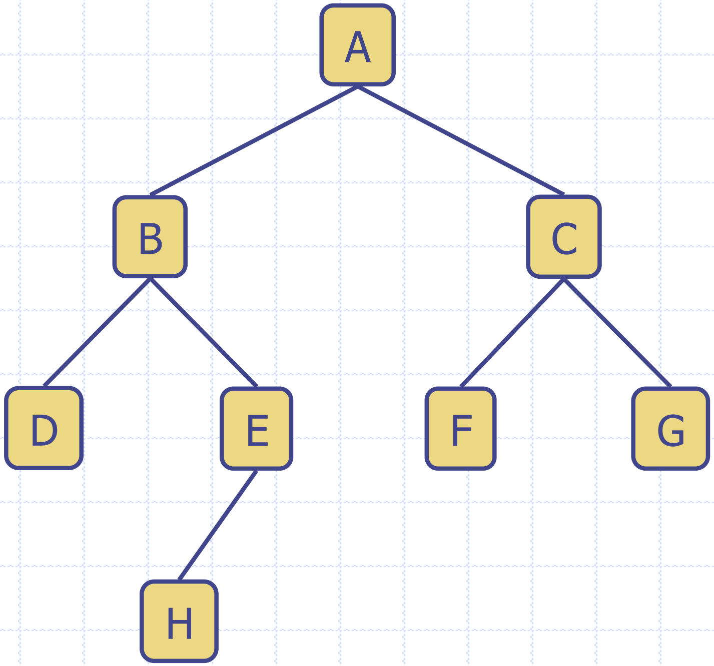




Binary Tree (BT)

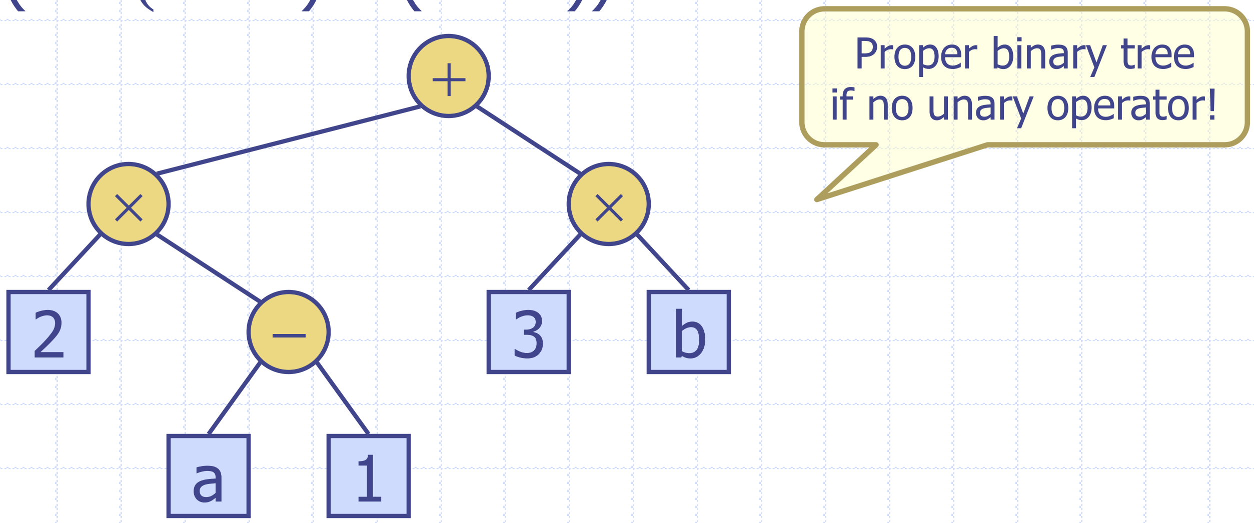
- A binary tree is a tree with the following properties:
  - Each internal node has one or two children (exactly two for proper binary trees)
  - The children of a node are an ordered pair
- We call the children of an internal node left child and right child
- Alternative recursive definition: a binary tree is either
  - a tree consisting of a single node, or
  - a tree whose root has an ordered pair of children, each of which is a binary tree

- Applications:
  - arithmetic expressions
  - decision processes
  - searching



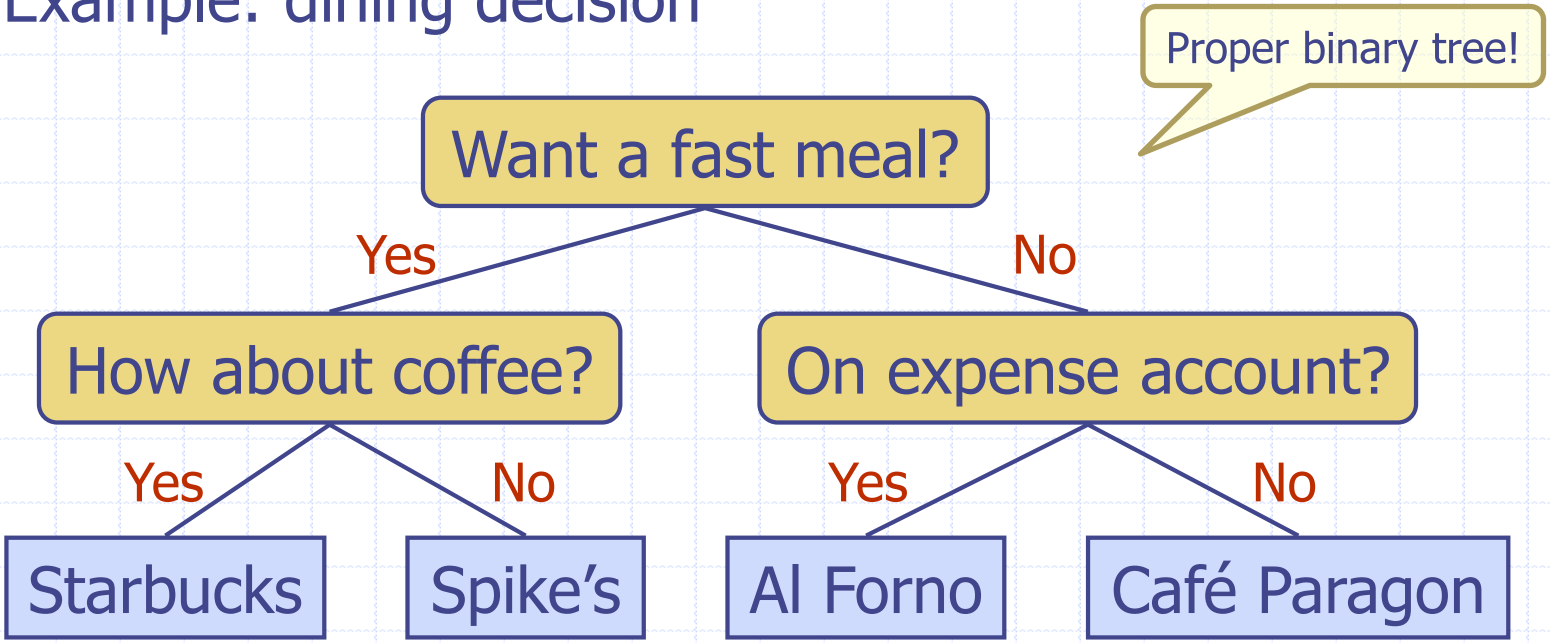
Example: Arithmetic Expression Trees

- Binary tree associated with an arithmetic expression
  - internal nodes: operators
  - external nodes: operands
- Example: arithmetic expression tree for the expression $(2 \times (a - 1) + (3 \times b))$



Example: Decision Trees

- Binary tree associated with a decision process
  - internal nodes: questions with yes/no answer
  - external nodes: decisions
- Example: dining decision

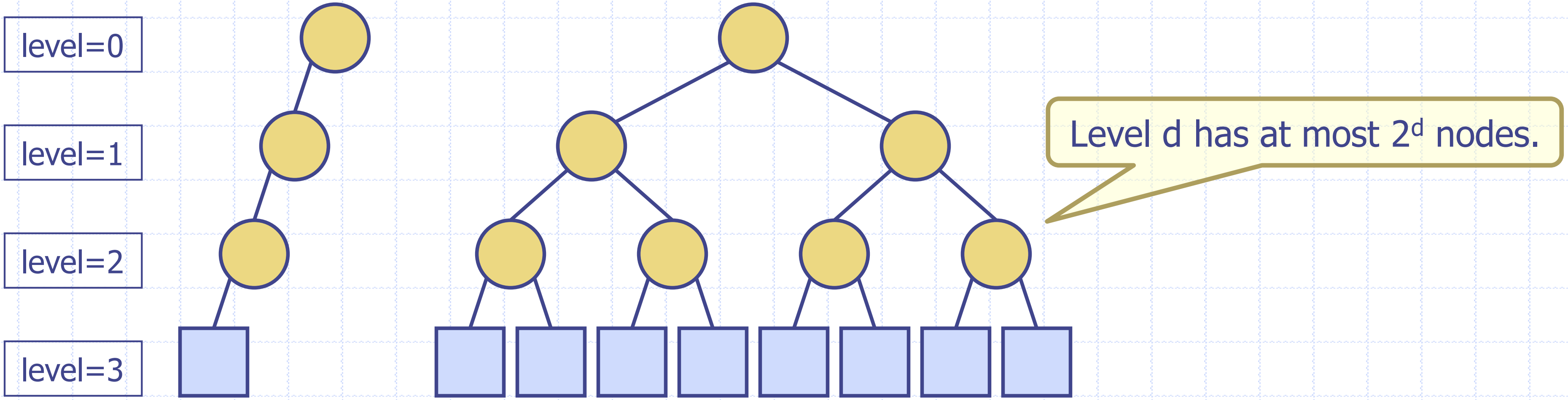


Properties of Binary Trees

Quiz!

- Notation
  - n: # of nodes
  - n_e: # of external nodes
  - n_i: # of internal nodes
  - h: height

- Properties:
  - $1 \leq n_e \leq 2^h$
  - $h \leq n_i \leq 2^h - 1$
  - $h+1 \leq n \leq 2^{h+1} - 1$
  - $\log_2(n+1) - 1 \leq h \leq n - 1$

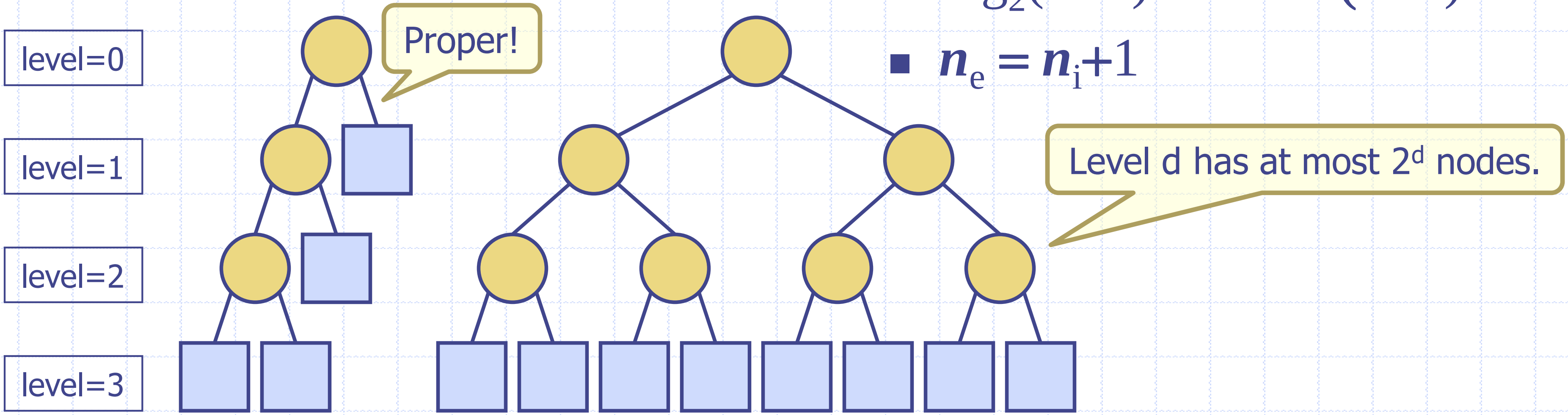


Properties of Proper Binary Trees

- Notation
  - n: # of nodes
  - n_e: # of external nodes
  - n_i: # of internal nodes
  - h: height

- Properties:
  - $h+1 \le n_e \le 2^h$
  - $h \le n_i \le 2^h - 1$
  - $2h+1 \le n \le 2^{h+1} - 1$
  - $\log_2(n+1)-1 \le h \le (n-1)/2$
  - $n_e = n_i + 1$

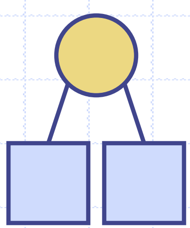
Quiz!



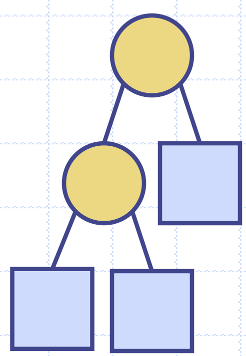
Proof by Induction: $n_e = n_i + 1$ for Proper Binary Trees

- Prove that $n_e = n_i + 1$
  - When $n_i = 1$ ...
  - When $n_i = 2$ ...
  - Assume the identity holds when $n_i \le k$, then when $n_i = k+1$ ...

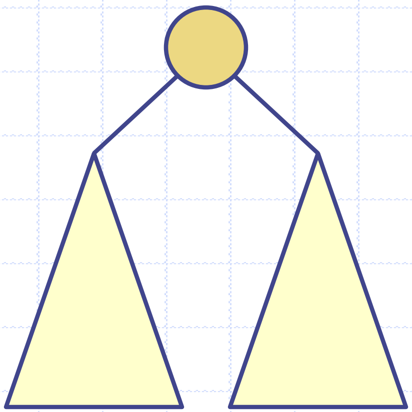
$n_i = 1$



$n_i = 2$



$n_i = k+1$



Quiz

- Given a proper binary tree with n node
  - Explain why n is always odd.
  - If n is 51, what is the numbers of internal and external nodes, respectively?

Proof by Induction: $n_0 = n_2+1$ for General Binary Trees

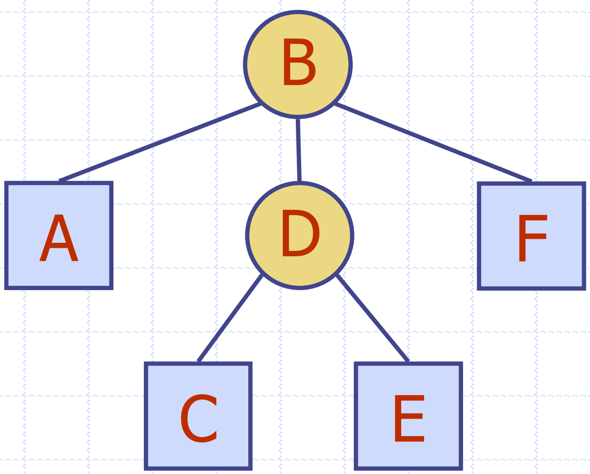
- n_k : # of nodes with k children, k=0, 1, 2
- Proof by induction…

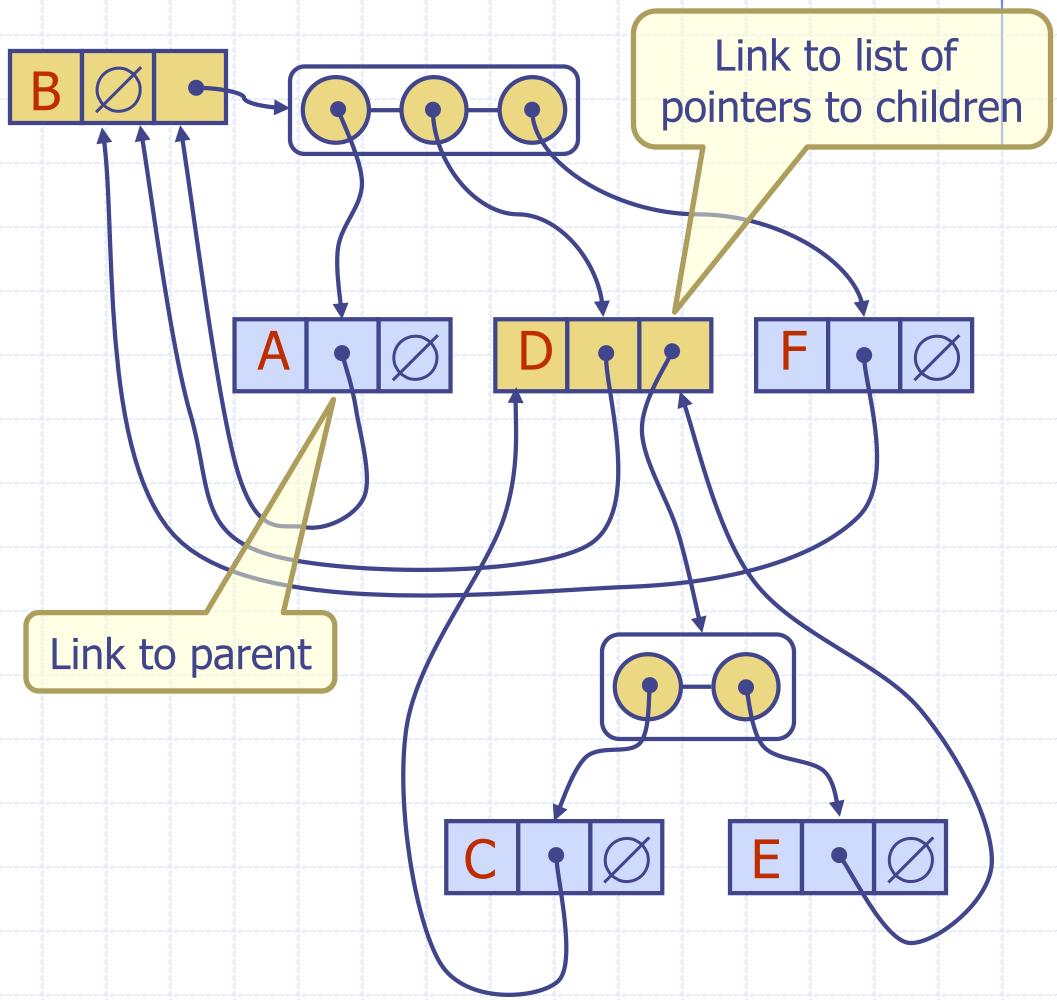
BinaryTree ADT

- The BinaryTree ADT extends the Tree ADT, i.e., it inherits all the methods of the Tree ADT
- Additional methods:
  - position p.left()
  - position p.right()
- Update methods may be defined by data structures implementing the BinaryTree ADT
- Proper binary tree: Each node has either 0 or 2 children

Linked Structure for Trees

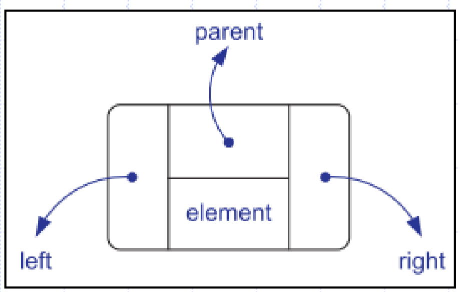
- A node is represented by an object storing
  - Element
  - Parent node
  - Sequence of children nodes

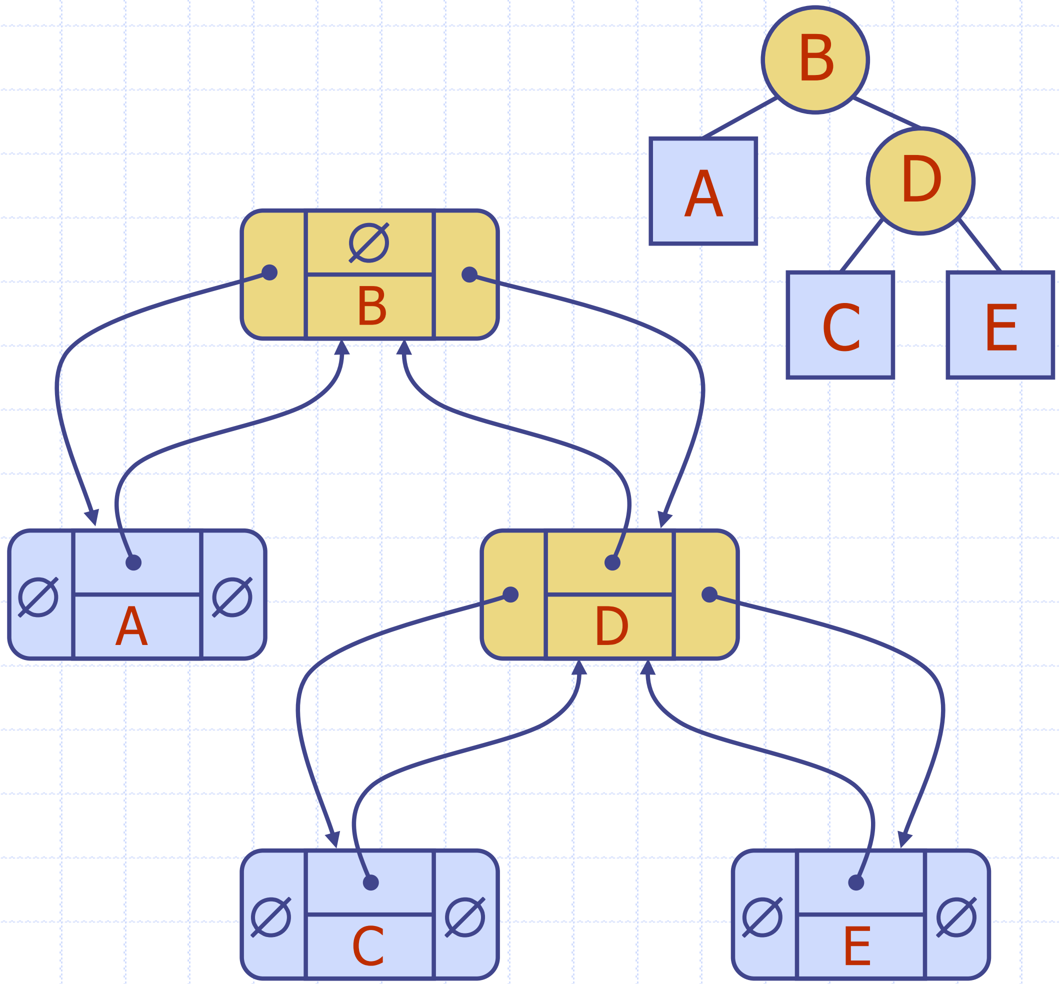




Linked Structure for Binary Trees

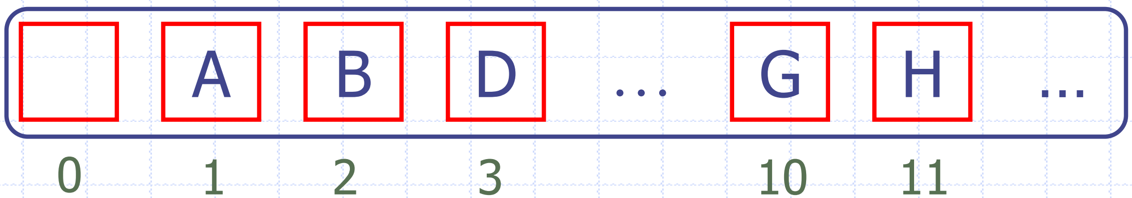
- A node is represented by an object storing
  - Element
  - Parent node
  - Left child node
  - Right child node

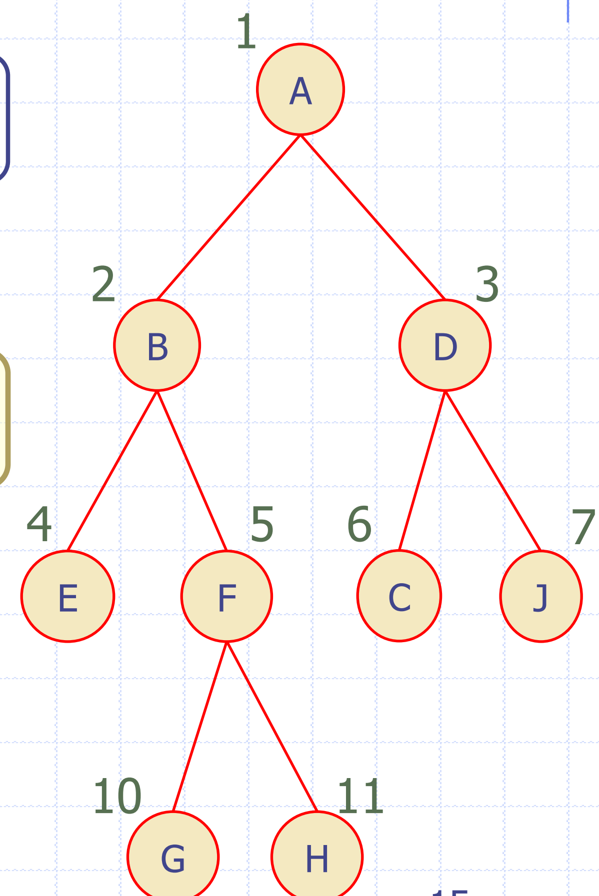




Vector Representation of Binary Trees

- Nodes of tree T are stored in vector S





- Node v is stored at S[f(v)]
  - f(root) = 1
  - if v is the left child of parent(v),
    f(v) = 2*f(parent(v))
  - if v is the right child of parent(v),
    f(v) = 2*f(parent(v)) + 1

f() is known as level numbering

Quiz!

Vector Representation of Binary Trees: More Examples

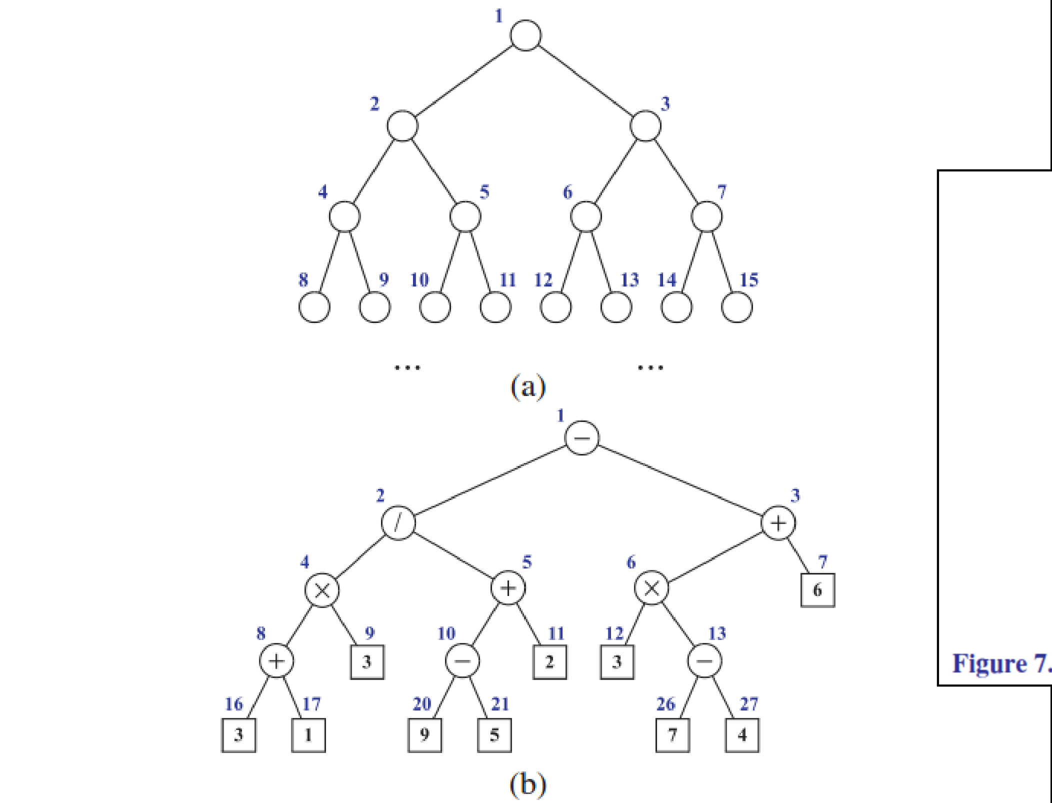


Figure 7.16: Binary tree level numbering: (a) general scheme; (b) an example.

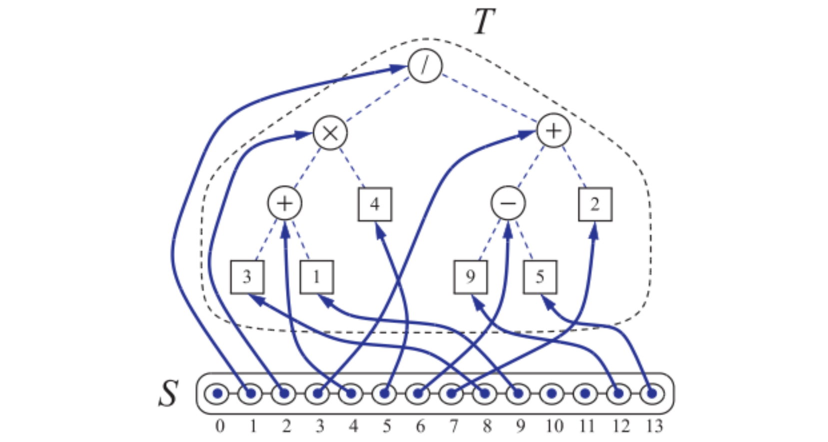
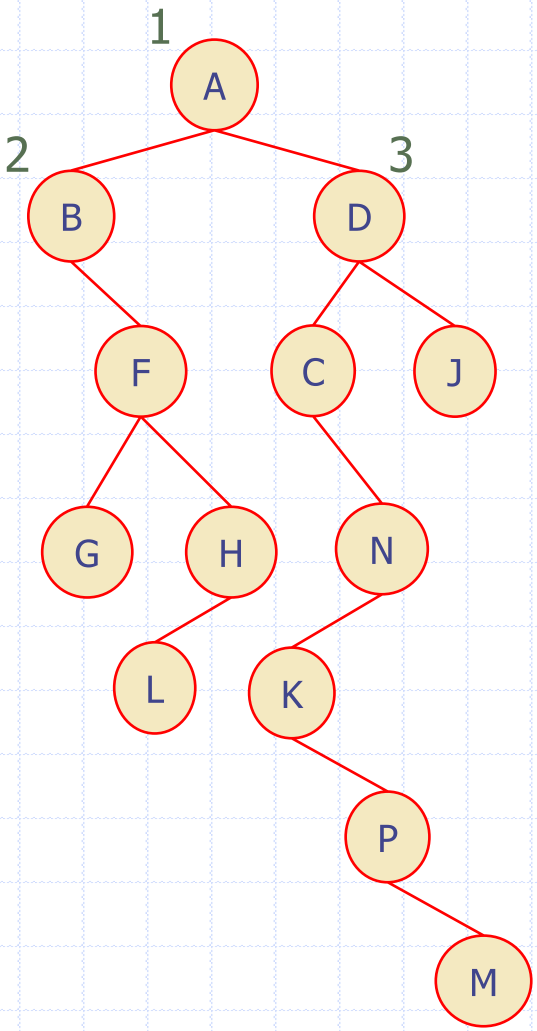


Figure 7.17: Representation of a binary tree T by means of a vector S.

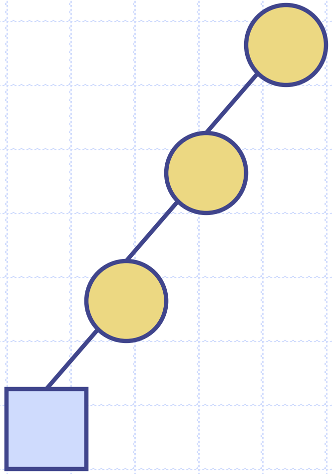
Quiz

- The binary tree is stored level-by-level in a one-dimensional array. What are the indices of nodes N and M?



Vector Representation of Binary Trees: Analysis

- Notation
  - n: # of nodes in tree T
  - N: size of vector S



- Time complexity

Operation	Time
left, right, parent, isExternal, isRoot	$O(1)$
size, empty	$O(1)$
root	$O(1)$
expandExternal, removeAboveExternal	$O(1)$
positions	$O(n)$

- Space complexity
  - O(N), which is $O(2^n)$ in the worse case

Major drawback!
(So we always want to keep trees as shallow as possible!)

Traversal of Binary Trees

- Basic types of traversal
  - Preorder
  - Postorder
  - Inorder
  - Level order

```
Algorithm binaryPreorder(T, p):
  perform the "visit" action for node p
  if p is an internal node then
    binaryPreorder(T, p.left())
    binaryPreorder(T, p.right())
```

```
Algorithm binaryPostorder(T, p):
  if p is an internal node then
    binaryPostorder(T, p.left())          {re
    binaryPostorder(T, p.right())         {re
  perform the "visit" action for the node p
```

Postorder Traversal for BT

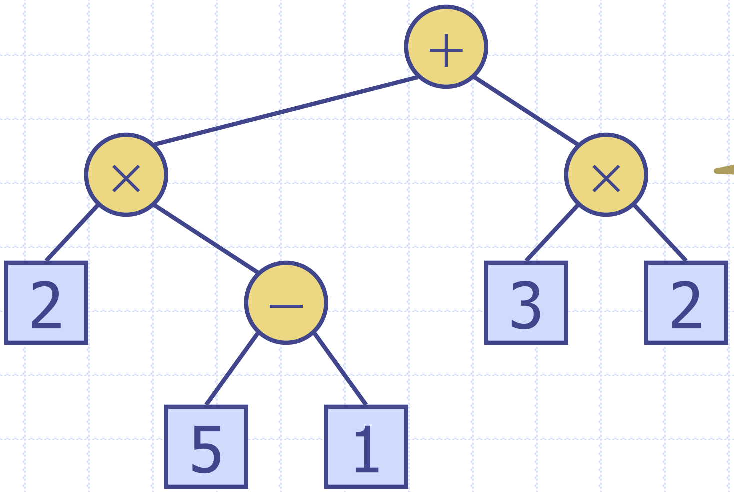
- Can be applied to any "bottom-up" evaluation problems
  - Evaluate an arithmetic expression
  - Directory size computation (for general trees)

Evaluate Arithmetic Expressions

- Based on postorder traversal
  - Recursive method returning the value of a subtree
  - When visiting an internal node, combine the values of the subtrees

Algorithm *evalExpr*(*v*)
  if *v.isExternal*()
    return *v.element*()

  x ← *evalExpr*(*v.left*())
  y ← *evalExpr*(*v.right*())
  ◊ ← operator stored at *v*
  return *x* ◊ *y*

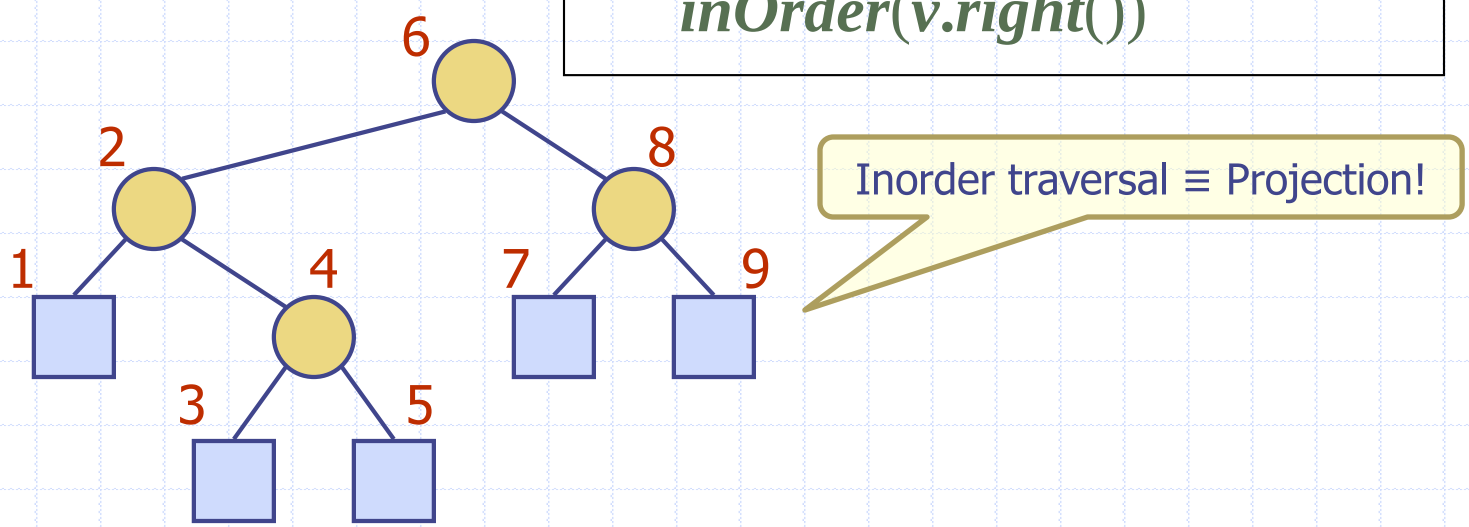


Postorder traversal ≡ Postfix notation

Inorder Traversal

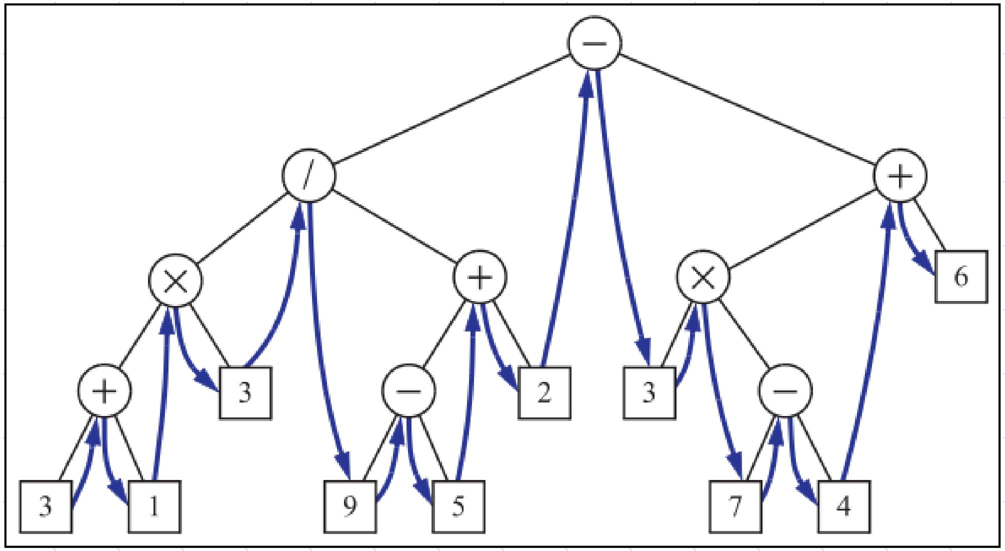
- Inorder traversal: a node is visited after its left subtree and before its right subtree
- Application
  - Draw a binary tree
  - Print arithmetic expressions with parentheses

Algorithm *inOrder*(*v*)
    if *v is NULL*
        return
    inOrder(*v.left*())
    visit(*v*)
    inOrder(*v.right*())



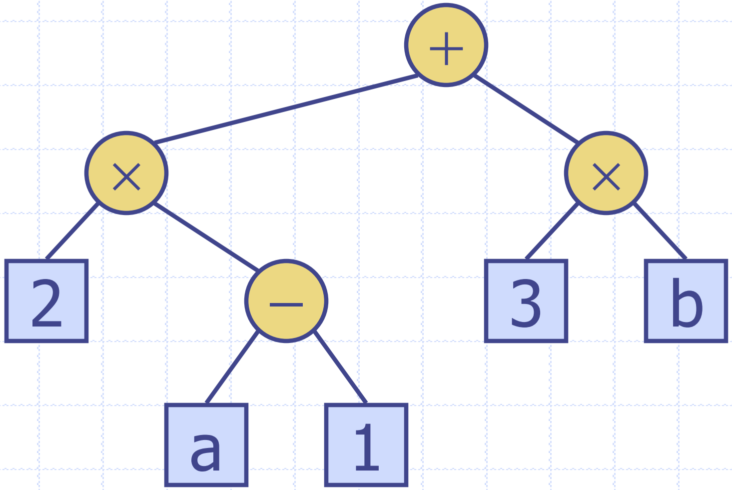
Inorder Traversal: Examples

- Properties of inorder traversal
  - Very close to infix notation
  - Can be obtained by tree projection



Print Arithmetic Expressions

- Based on inorder traversal
  - Print operand or operator when visiting node
  - Print "(" before traversing left subtree
  - Print ")" after traversing right subtree



Algorithm *printExpression*(*v*)
    if *v is NULL*
        return
    print("(")
    inOrder(*v.left*())
    print(*v.element*())
    inOrder(*v.right*())
    print (")")

$$((2 \times (a - 1)) + (3 \times b))$$

Draw a Binary Tree

- Since inorder traversal is equivalent to tree projection, it is easy to draw a binary tree using inorder traversal.

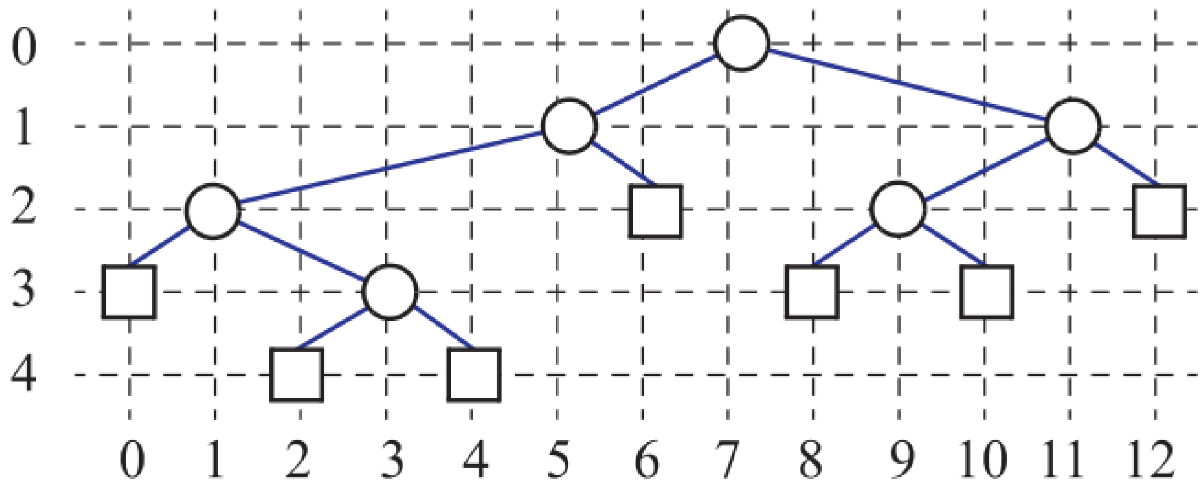
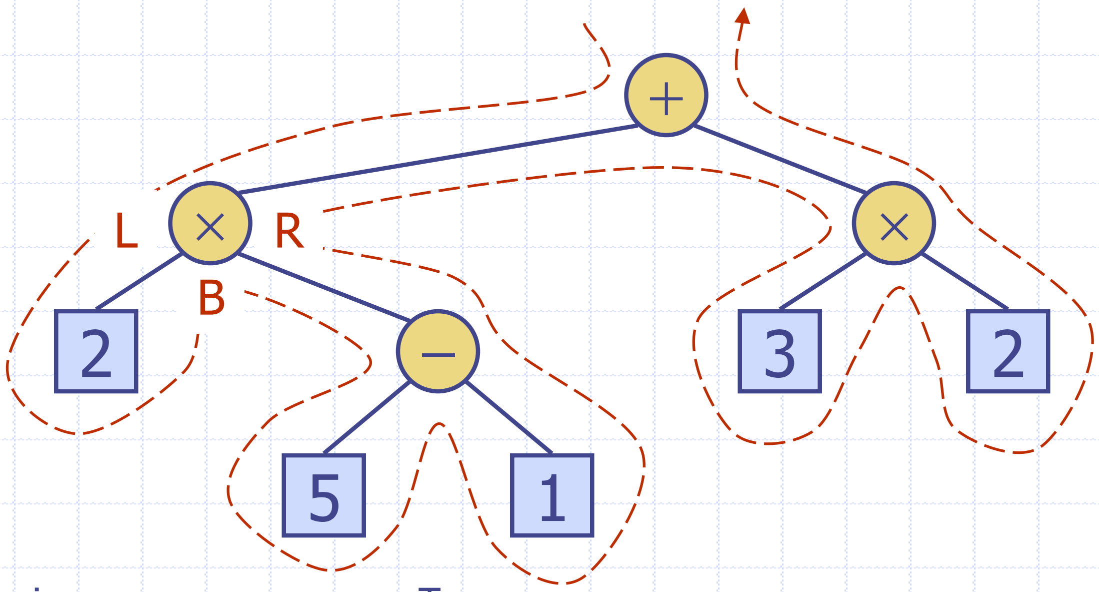


Figure 7.20: The inorder drawing algorithm for a binary tree.

Euler Tour Traversal

- Generic traversal of a binary tree
- Includes a special cases the preorder, postorder and inorder traversals
- Walk around the tree and visit each node three times:
  - On the left (preorder) ➔ + x 2 – 5 1 x 3 2
  - From below (inorder) ➔ 2 x 5 – 1 + 3 x 2
  - On the right (postorder) ➔ 2 5 1 – x 3 2 x +

Quiz!



Euler Tour Traversal

- Applications
  - Determine the number of descendants of each node in a tree
  - Fully parenthesize an arithmetic expression from an expression tree

Quiz

- Determine a binary tree
  - Inorder=[c q f p x z k y]
  - Preorder=[f c q z x p y k]
- Determine a binary tree
  - Inorder=[z b j d e f m c]
  - Postorder=[b z d e m c f j]
- Determine a binary tree
  - Preorder=[a b c]
  - Postorder=[c b a]

From General to Binary Trees

- How to transform a general tree to a binary tree ➔ Left-child right-sibling representation

Quiz!