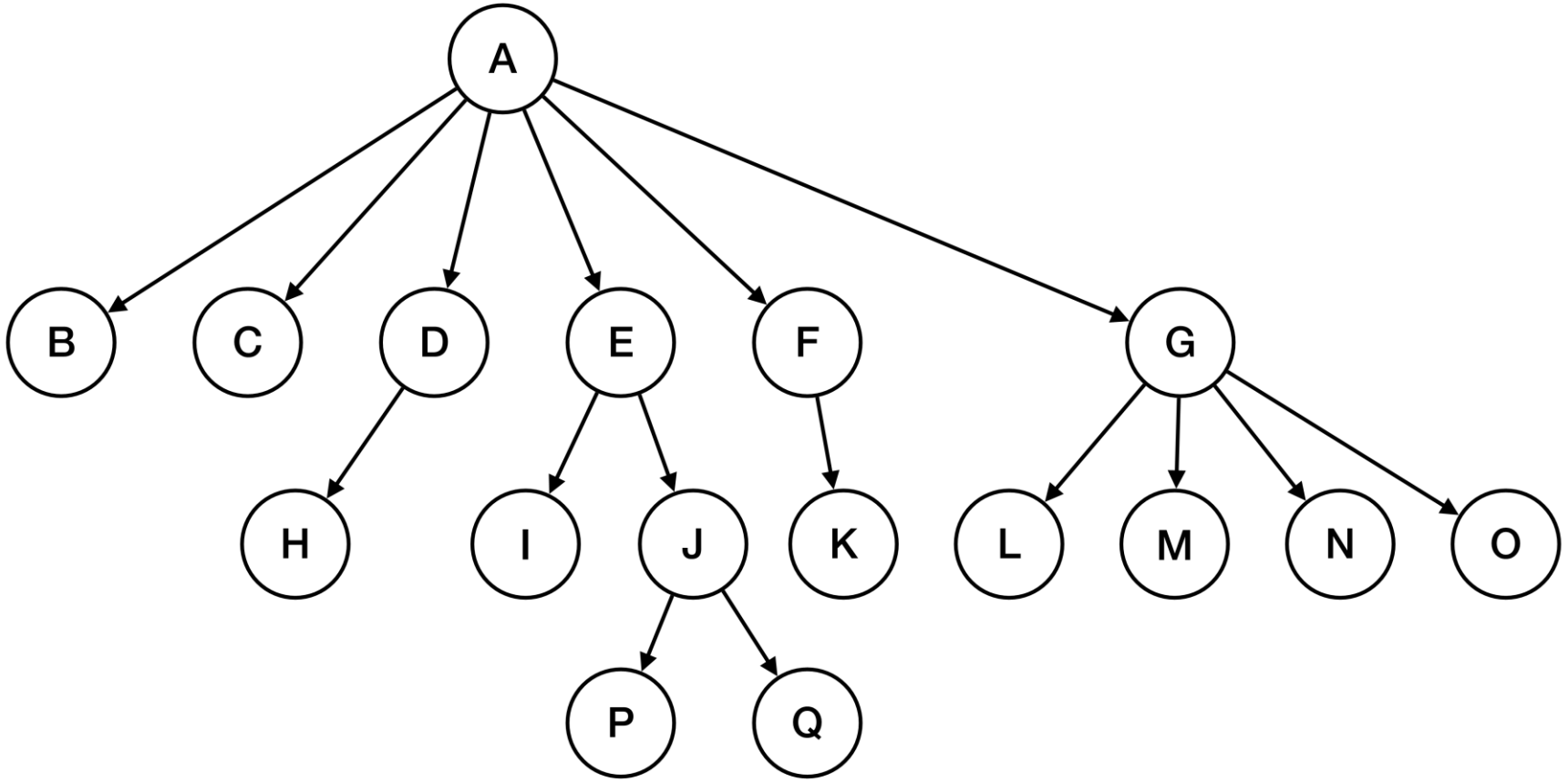
بسم الله الرحمن الرحیم

ساختمان‌های داده

جلسه ۱۵

مجتبی خلیلی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

مثال

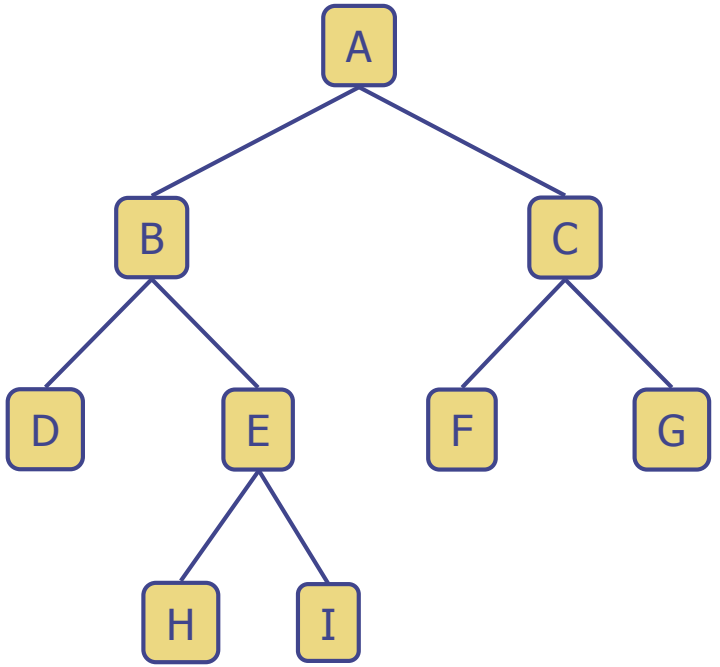


Binary Tree

Binary Trees

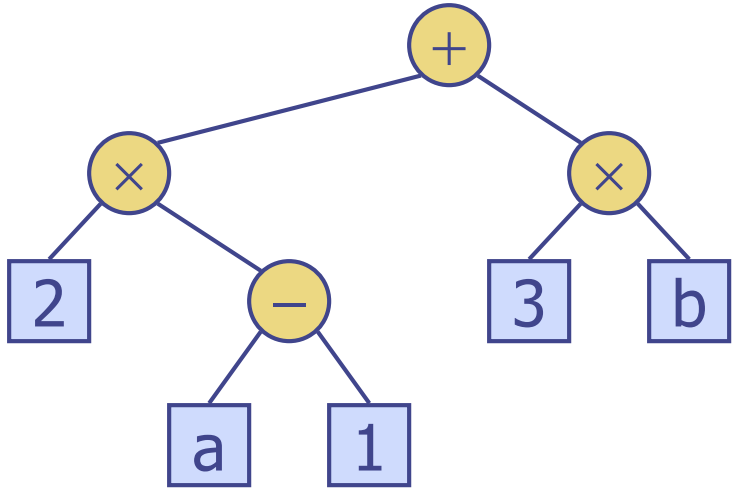
- A binary tree is a tree with the following properties:
  - Each internal node has at most two children (exactly two for proper or full binary trees)
  - The children of a node are an ordered pair
- We call the children of an internal node left child and right child
- Left subtree & Right subtree
- 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



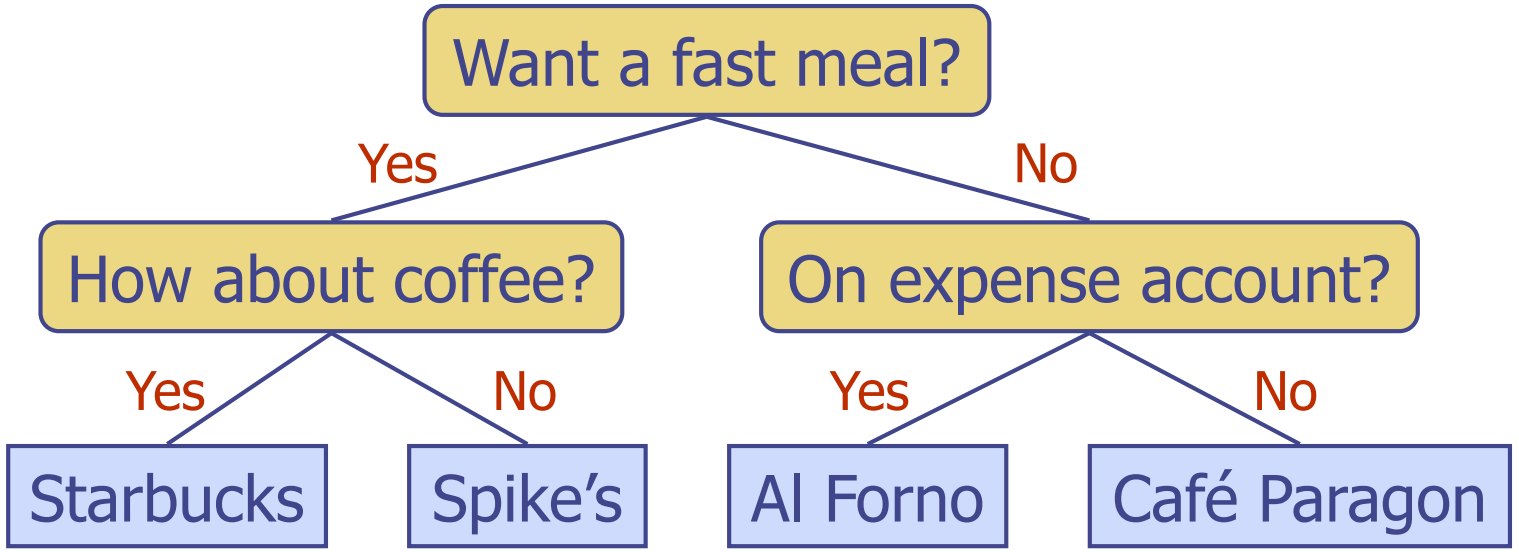
Arithmetic Expression Tree

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



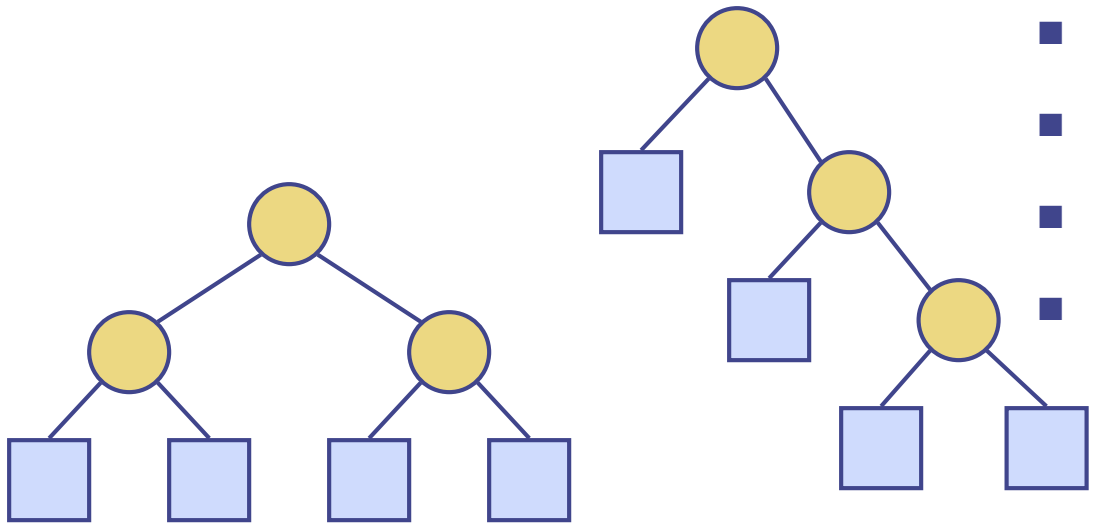
Decision Tree

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



Properties of Proper Binary Trees

- Notation
  - n number of nodes
  - e number of external nodes
  - i number of internal nodes
  - h height



- Properties:
  - $e = i + 1$
  - $n = 2e - 1$
  - $h \leq i$
  - $h \leq (n - 1)/2$
  - $e \leq 2^h$
  - $h \geq \log_2 e$
  - $h \geq \log_2 (n + 1) - 1$

Properties of Proper Binary Trees

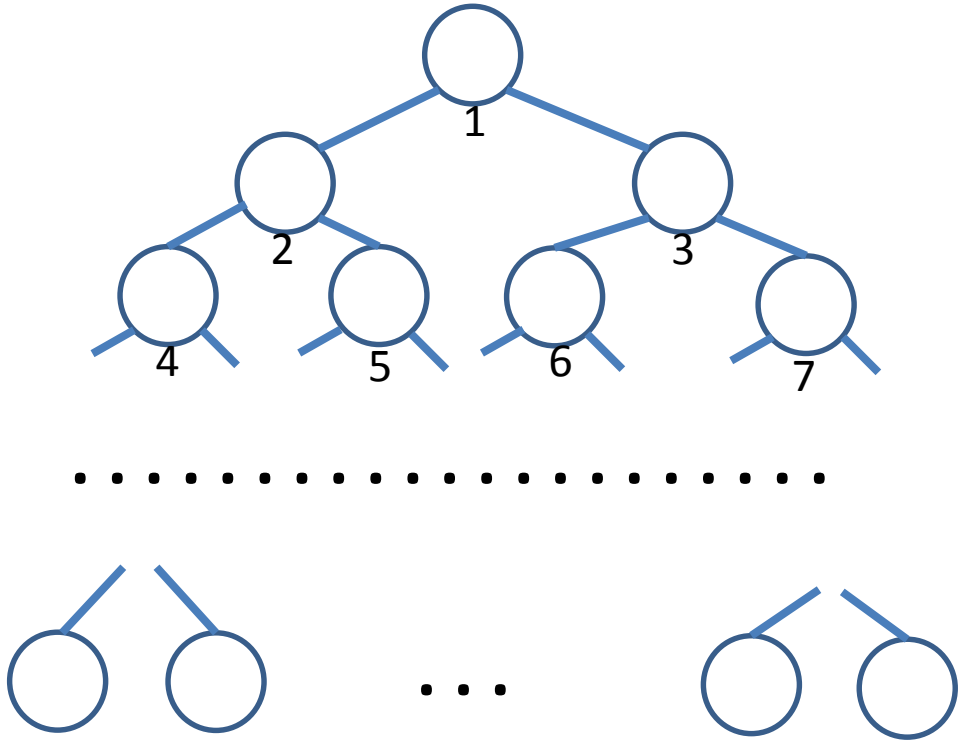
- Notation
  - n number of nodes
  - e number of external nodes
  - i number of internal nodes
  - h height

- Properties:
  - $e = i + 1$
  - $n = 2e - 1$

$$n = i + e$$
$$e = i + 1$$
$$n = 2e - 1$$

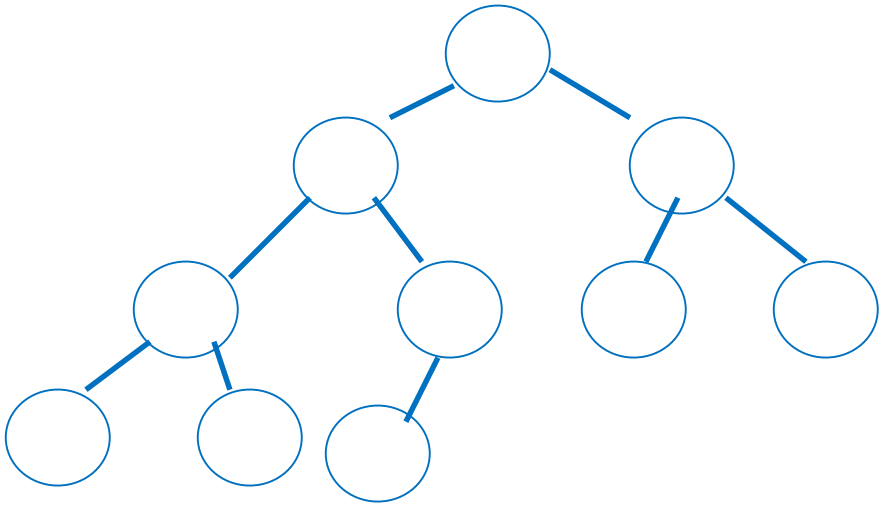
Other types of Binary Trees

- **Perfect**– each internal node has exactly 2 children and all the leaves are on the same level.



Other types of Binary Trees

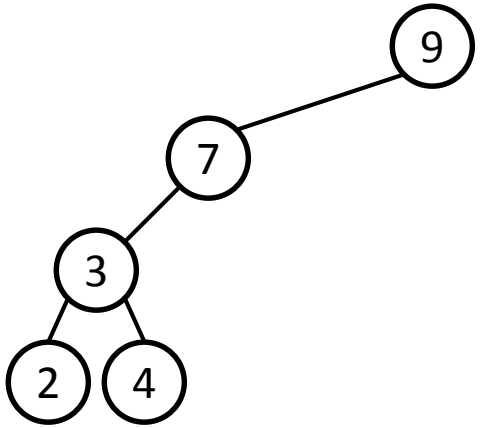
- **Complete tree***:* a binary tree in which every level, except possibly the deepest is completely filled. At depth n, the height of the tree, all nodes are as far left as possible.



- گره بعدی کجا؟

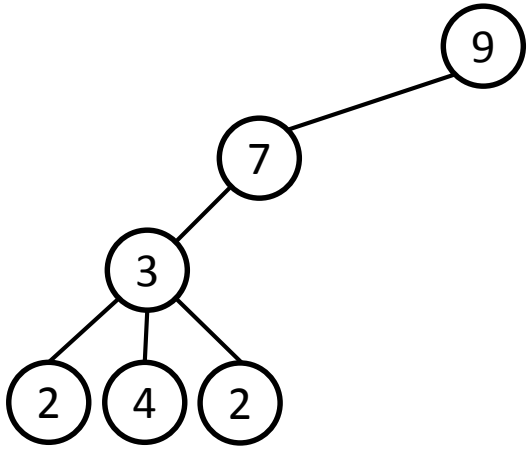
مثال

- چه نوعی؟



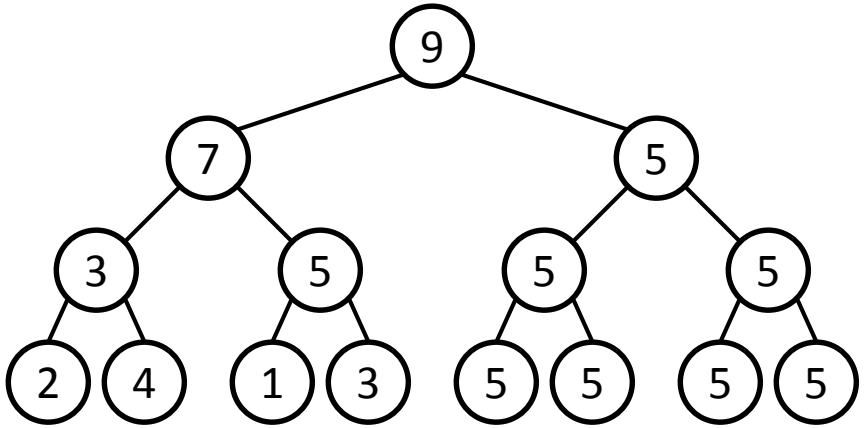
مثال

- چه نوعی؟



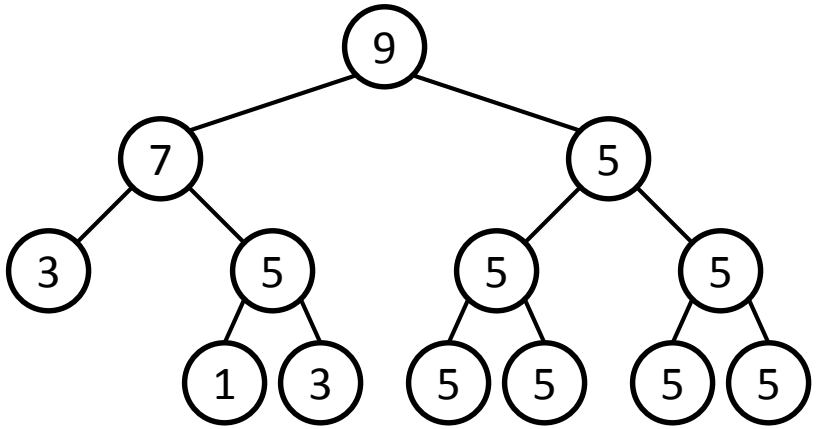
مثال

- چه نوعی؟



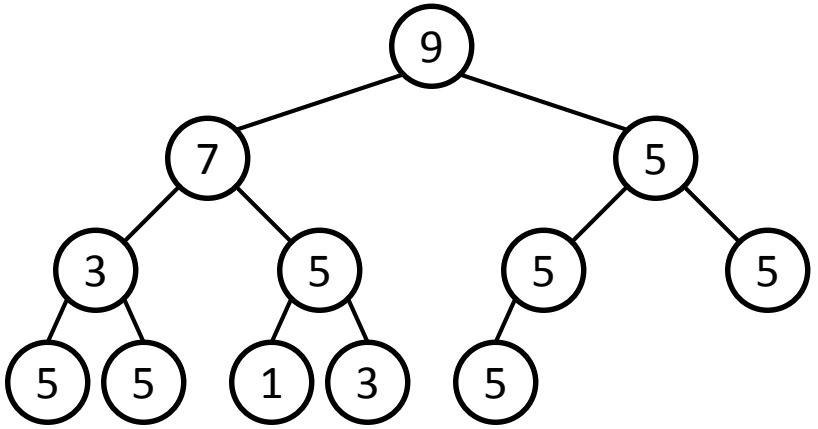
مثال

- چه نوعی؟



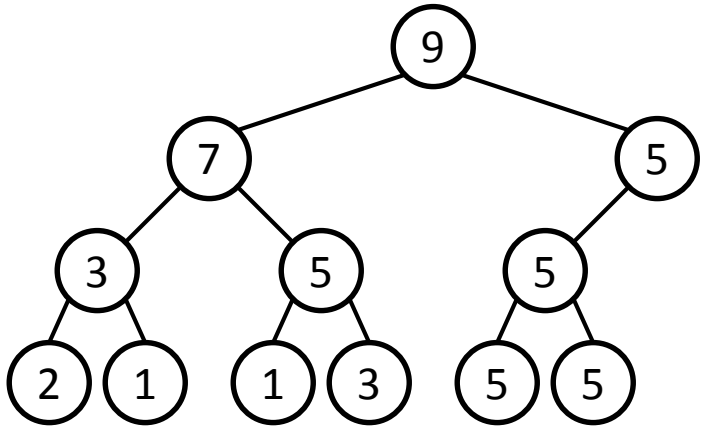
مثال

- چه نوعی؟



مثال

- چه نوعی؟



مثال

- بیشینه ارتفاع یک درخت باینری proper با ۱۳ گره؟

مثال

- بیشینه ارتفاع یک درخت باینری proper با ۸ گره؟

مثال

- تعداد گره ها در یک درخت perfect با ارتفاع h؟

$$n = 1 + 2 + 4 + 8 + \cdots 2^h$$

$$= 1 + x + x^2 + x^3 + \cdots x^h = \frac{x^{h+1} - 1}{x - 1}, \qquad x = 2$$

BinaryTree ADT

IUT-ECE

- 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

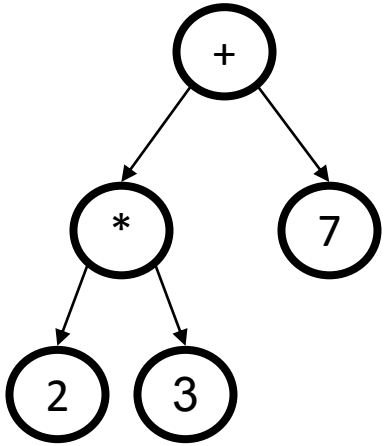
سوال

- درخت یک ADT است یا یک ساختمان داده؟

Evaluate Arithmetic Expressions

- مطابق تعاریف قبلی مان:

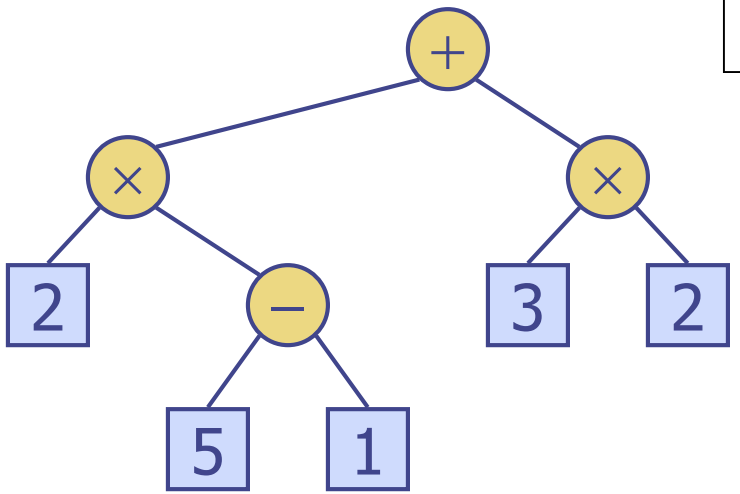
- Pre-order: + * 2 3 7
- In-order: 2 * 3 + 7
- Post-order: 2 3 * 7 +



Evaluate Arithmetic Expressions

- Specialization of a postorder traversal
  - recursive method returning the value of a subtree
  - when visiting an internal node, combine the values of the subtrees

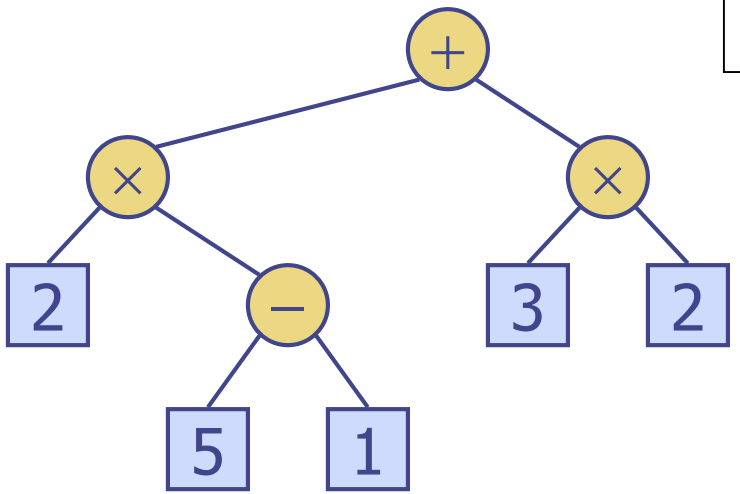
```
Algorithm evalExpr(v)
    if v.isExternal()
        return v.element()
    else
        x ← evalExpr(v.left())
        y ← evalExpr(v.right())
        ◊ ← operator stored at v
        return x ◊ y
```



Print Arithmetic Expressions

- Specialization of an inorder traversal
  - print operand or operator when visiting node
  - print “(“ before traversing left subtree
  - print “)“ after traversing right subtree

```
Algorithm printExpr(v)
    if ¬ v.isExternal()
        print (“(“)
        printExpr(v.left())
    print(v)
    if ¬ v.isExternal()
        printExpr(v.right())
        print (“)“)
```

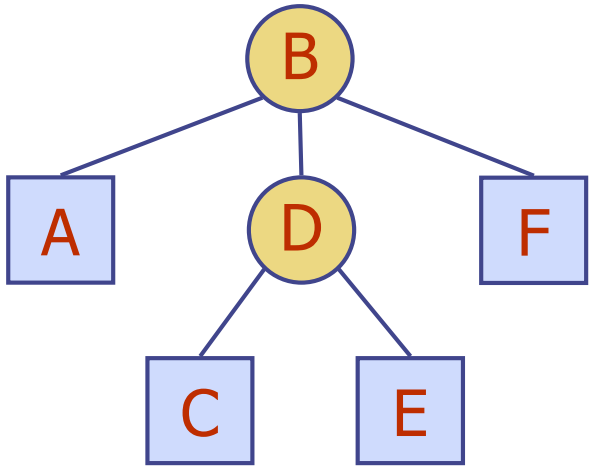


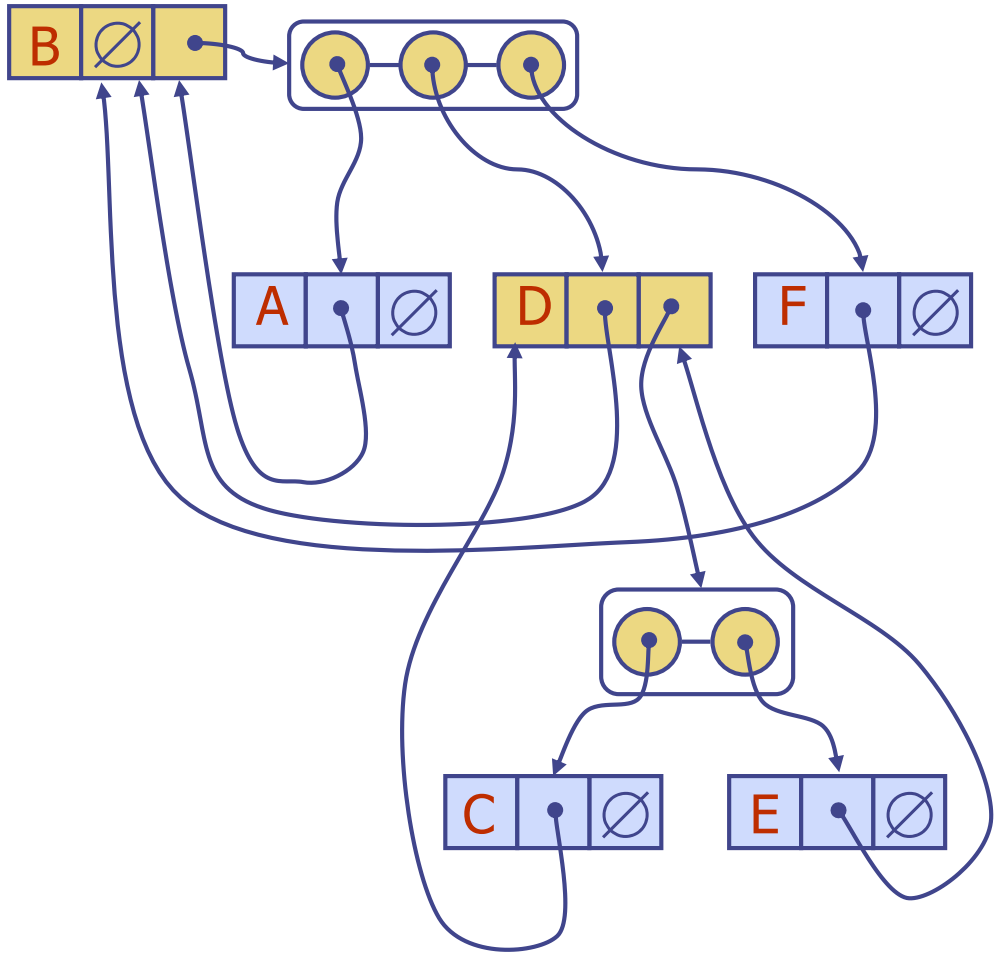
((2 × (5 − 1)) + (3 × 2))

How to represent trees in programming language?

Recall: Linked Structure for Trees

- A node is represented by an object storing
  - Element
  - Parent node
  - Sequence of children nodes
- Node objects implement the Position ADT

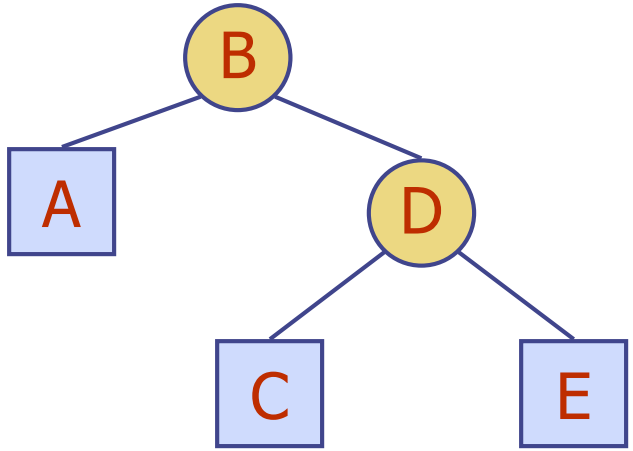


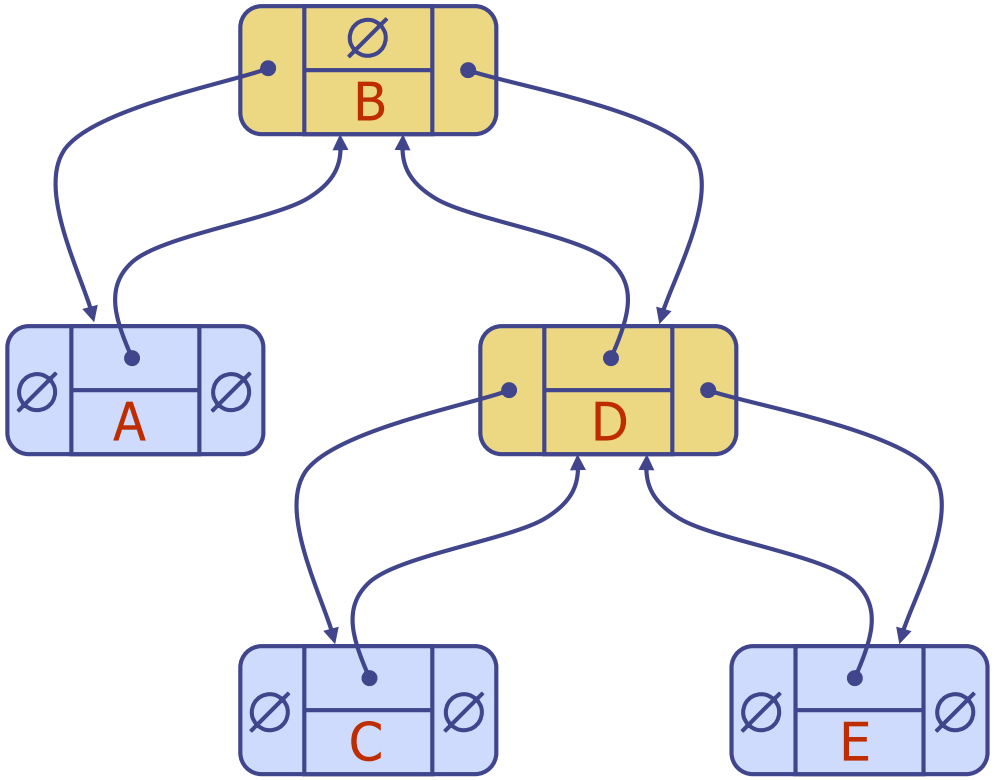


Linked Structure for Binary Trees

IUT-ECE

- A node is represented by an object storing
  - Element
  - Parent node
  - Left child node
  - Right child node
- Node objects implement the Position ADT





Linked Structure for Binary Trees

IUT-ECE

```
struct Node {                                          // a node of the tree
  Elem    elt;                                         // element value
  Node*   par;                                         // parent
  Node*   left;                                        // left child
  Node*   right;                                       // right child
  Node() : elt(), par(NULL), left(NULL), right(NULL) { } // constructor
};
```

Code Fragment 7.17: Structure Node implementing a node of a binary tree.

Linked Structure for Binary Trees

IUT-ECE

```
class Position {                                    // position in the tree
private:
  Node* v;                                          // pointer to the node
public:
  Position(Node* _v = NULL) : v(_v) { }             // constructor
  Elem& operator*()                                 // get element
    { return v->elt; }
  Position left() const                             // get left child
    { return Position(v->left); }
  Position right() const                            // get right child
    { return Position(v->right); }
  Position parent() const                           // get parent
    { return Position(v->par); }
  bool isRoot() const                               // root of the tree?
    { return v->par == NULL; }
  bool isExternal() const                           // an external node?
    { return v->left == NULL && v->right == NULL; }
  friend class LinkedBinaryTree;                    // give tree access
};
typedef std::list<Position> PositionList;           // list of positions
```

Code Fragment 7.18: Class Position implementing a position in a binary tree.

```
typedef int Elem;                                    // base element type
class LinkedBinaryTree {
protected:
  // insert Node declaration here...
public:
  // insert Position declaration here...
public:
  LinkedBinaryTree();                                // constructor
  int size() const;                                  // number of nodes
  bool empty() const;                                // is tree empty?
  Position root() const;                             // get the root
  PositionList positions() const;                    // list of nodes
  void addRoot();                                    // add root to empty tree
  void expandExternal(const Position& p);            // expand external node
  Position removeAboveExternal(const Position& p); // remove p and parent
  // housekeeping functions omitted...
protected:                                           // local utilities
  void preorder(Node* v, PositionList& pl) const;    // preorder utility
private:
  Node* _root;                                       // pointer to the root
  int n;                                             // number of nodes
};
```

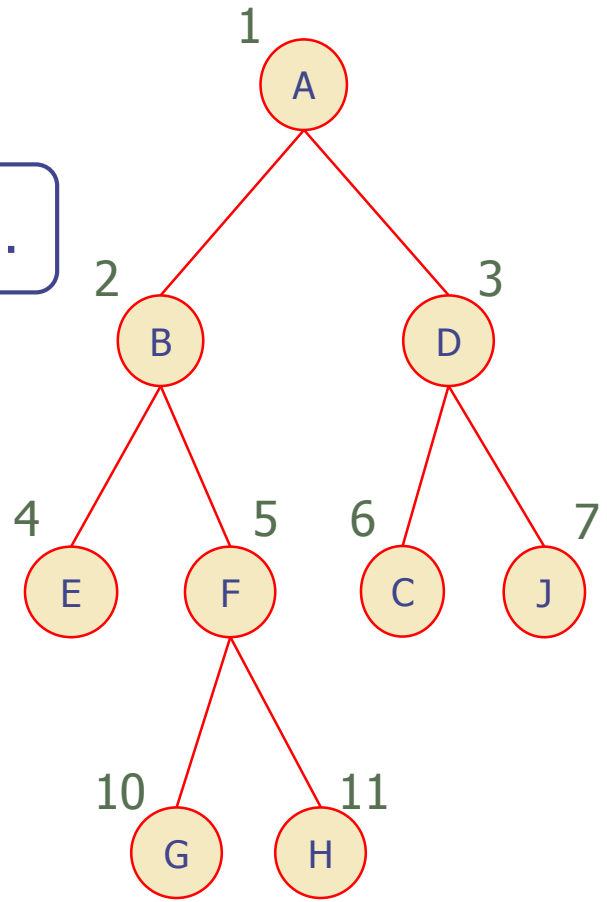
Code Fragment 7.19: Implementation of a LinkedBinaryTree class.

```
LinkedBinaryTree::LinkedBinaryTree()                    // constructor
  : _root(NULL), n(0) { }
int LinkedBinaryTree::size() const                      // number of nodes
  { return n; }
bool LinkedBinaryTree::empty() const                    // is tree empty?
  { return size() == 0; }
LinkedBinaryTree::Position LinkedBinaryTree::root() const // get the root
  { return Position(_root); }
void LinkedBinaryTree::addRoot()                        // add root to empty tree
  { _root = new Node; n = 1; }
```

Code Fragment 7.20: Simple member functions for class LinkedBinaryTree.

Array-Based Representation of Binary Trees

- Nodes are stored in an array A



- Node v is stored at A[rank(v)]
  - rank(root) = 1
  - if node is the left child of parent(node),
    rank(node) = 2 · rank(parent(node))
  - if node is the right child of parent(node),
    rank(node) = 2· rank(parent(node)) + 1

Array-Based Representation of Binary Trees

- root, parent, left, right, isExternal?