Trees and Binary Relations

Debdeep Mukhopadhyay
IIT Madras

Outline

- Tree : definition and its relation with binary relations
- Properties
- Expressions:
  - Balancedness
  - Infix, Postfix and Prefix
- Expression Trees
- Traversal of Trees: Inorder, Postorder and Preorder

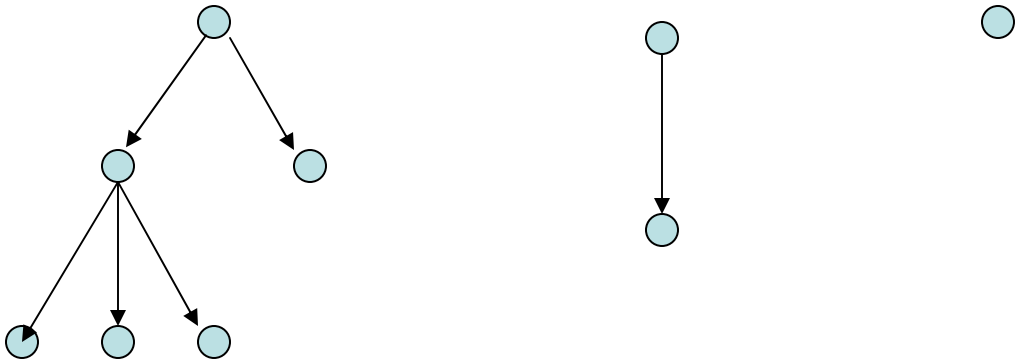
Trees and Relations

- They are an important class of binary relations
- They have many applications in CS: data structures, design of algorithms, analyzing networks etc
- Can be expressed and formalized using Directed Graphs (Digraphs)

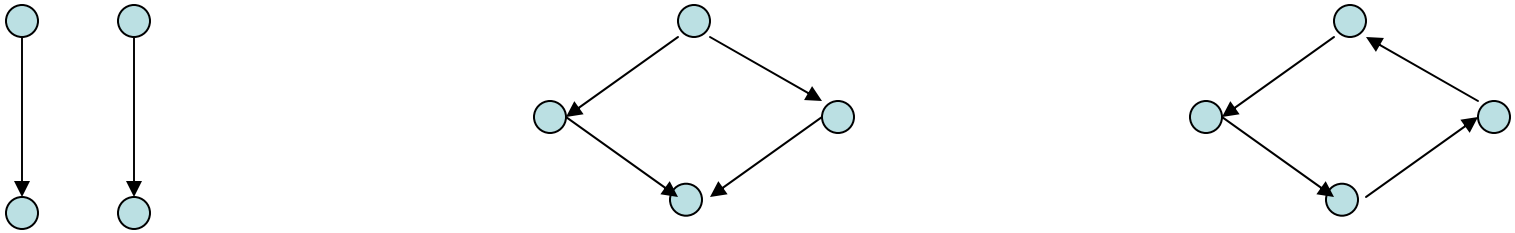
Trees and Relations

- A Tree is a digraph with a nonempty set of nodes such that:
  - (i) there is exactly one node, called the root, which has indegree 0
  - (ii) every node other than the root has indegree 1
  - (iii) for every node a of the tree, there is a directed path from the root to a.
- **Alternate Defn:** A tree is a connected, directed acyclic graph (see your graph theory text book)

Examples



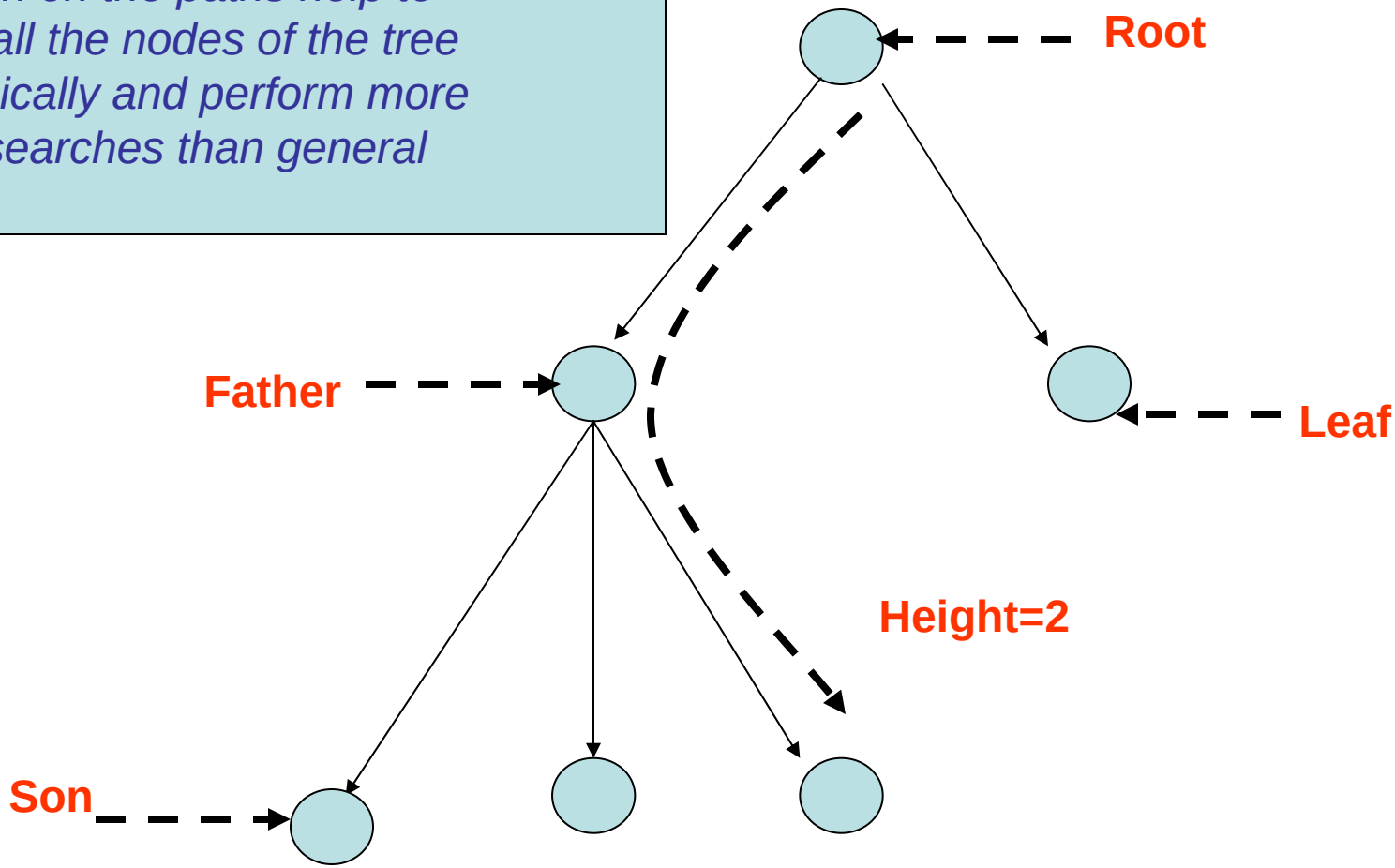
Trees



Not-Trees

Terms

Restriction on the paths help to traverse all the nodes of the tree algorithmically and perform more efficient searches than general digraphs.



Properties

- They are many
- Few important results:
  - There is a unique directed path from the root r to any node a.
  - Every directed path is a simple path
  - There are no loops on nodes of trees

Restricted Trees

- Applications of trees involve restricted classes
- If every node has n or fewer sons, it is called n-ary tree.
- If every node has either n or zero sons they are called complete n-ary trees.
- We shall be dealing with binary trees in more detail.

One Application of Trees to CS: Manipulation of Expressions

Normal form of Expressions

- Expressions of all kinds are defined recursively
- Basis: Variables, Integers and Real Numbers

- Induction: If E_1 and E_2 are arithmetic operations:
  1. E_1+E_2
  2. E_1-E_2
  3. $E_1 X E_2$
  4. E_1 / E_2

  Infix operators

  5. If E is an arithmetic expression, then so is (-E) (prefix operator)

Balanced Parenthesis

- ((a+b)χ((c+d)-e)): Balanced
- Definition of “balanced parenthesis” is governed by two rules (iterative rule):
  1. A balanced string has an equal number of left and right parenthesis.
  2. As we move from left to right along the string, the profile of the string never becomes negative. Here the profile is the running total of the number of left parenthesis minus the number of right parenthesis.

Recursive Rule

- Basis: The empty string is balanced
- Induction: If x and y are strings of balanced parenthesis, then (x)y is also a string of balanced parenthesis.

- Both the notions can be proved to be same.

Infix, Postfix and Prefix

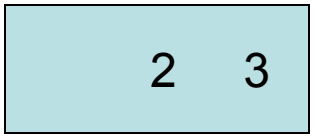
- Infix: A+B
- Postfix: AB+
- Prefix: +AB
- Convert A+(B*C) from Infix to Postfix form:
  - A+(BC*)
  - A(BC*)+
  - ABC*+
- *The only rule that has to be remembered is:*
  - *the operations with higher precedence are converted first, and*
  - *that after a portion of the expression has been converted into postfix it is to be treated as a single operand.*

Advantage of Postfix form

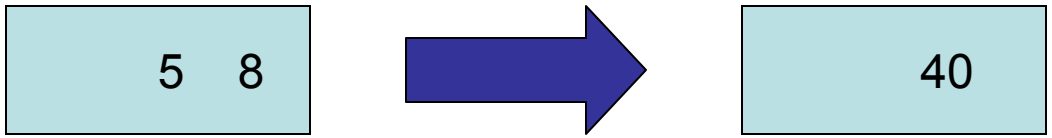
- Do not need to remember precedences
- No parenthesis required
- Very easy to evaluate
- For an expression of length N, the order is O(N).

Evaluate Postfix Expression

- Evaluate: 23+8*
- Push the operands, 2, 3 until you get an operator.



- When you get the operator +, pop the two top-most elements and apply the operator on them. Then push the result, 5 into the stack. Continue pushing 8. Then apply * to 8 and 5 (after popping them). Thus, the expression evaluates to 40.



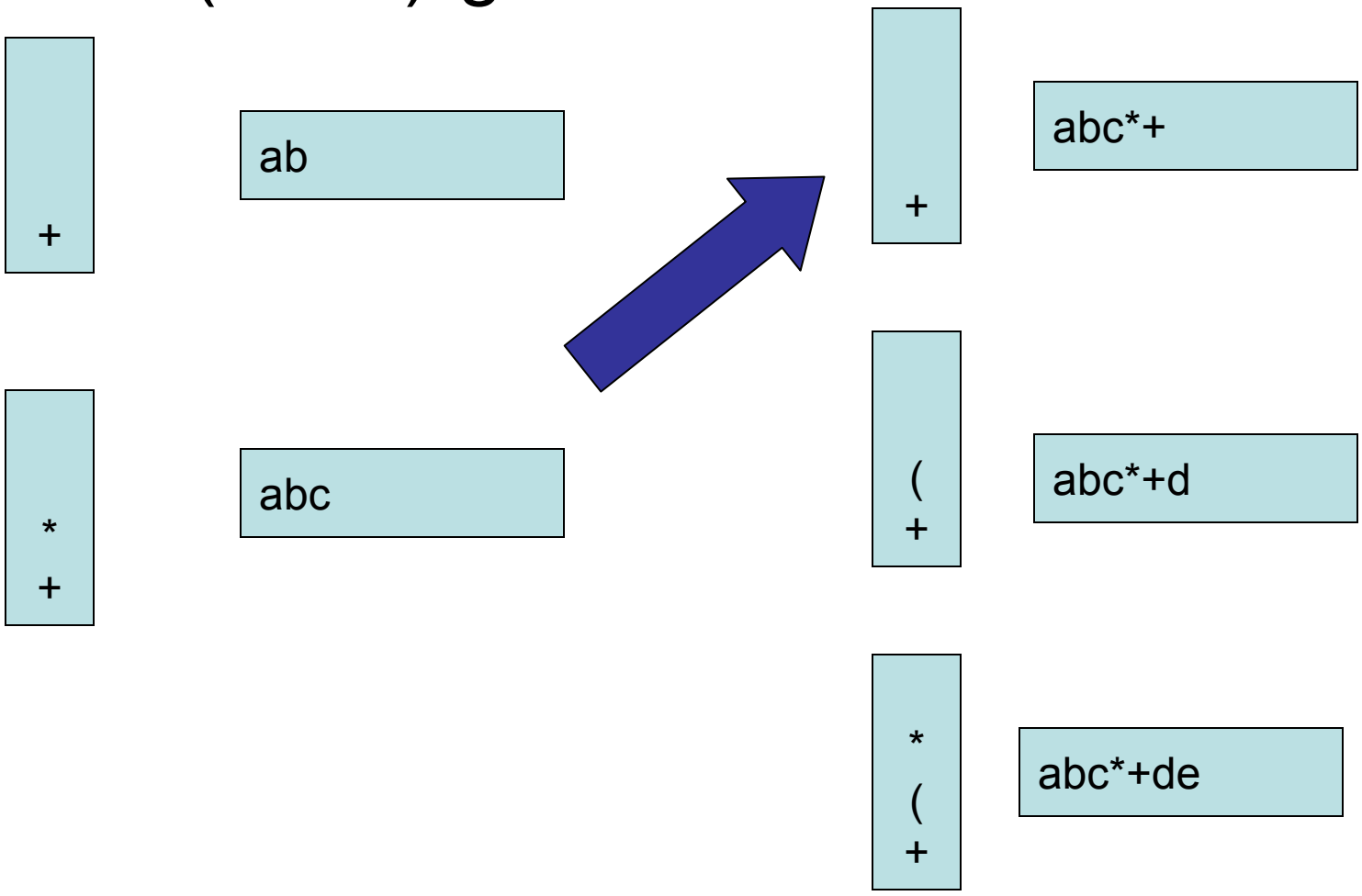
Converting an infix to postfix form

- Lets concentrate on the expression:
  - a+b*c+(d*e+f)*g
  - We shall convert the infix string to postfix.
  - Correct answer is: abc*+de*f+g*+
  - An easy way of implementing shall involve stacks.

- + has lowest priority, (has highest priority.
- Start with an empty stack.
- When an operand is read, it is placed onto the output.
- Operators are pushed into a stack, including the (.
- If we see a), we pop the stack, writing the symbols until we see the (. Note we do not output (.
- If we see operators like +, * we push them into the stack if their priority is higher than the element at the stack top.
- If we see operators like (, we push them into the stack.
- Finally, if the end of the stack is reached pop the stack and write the output.

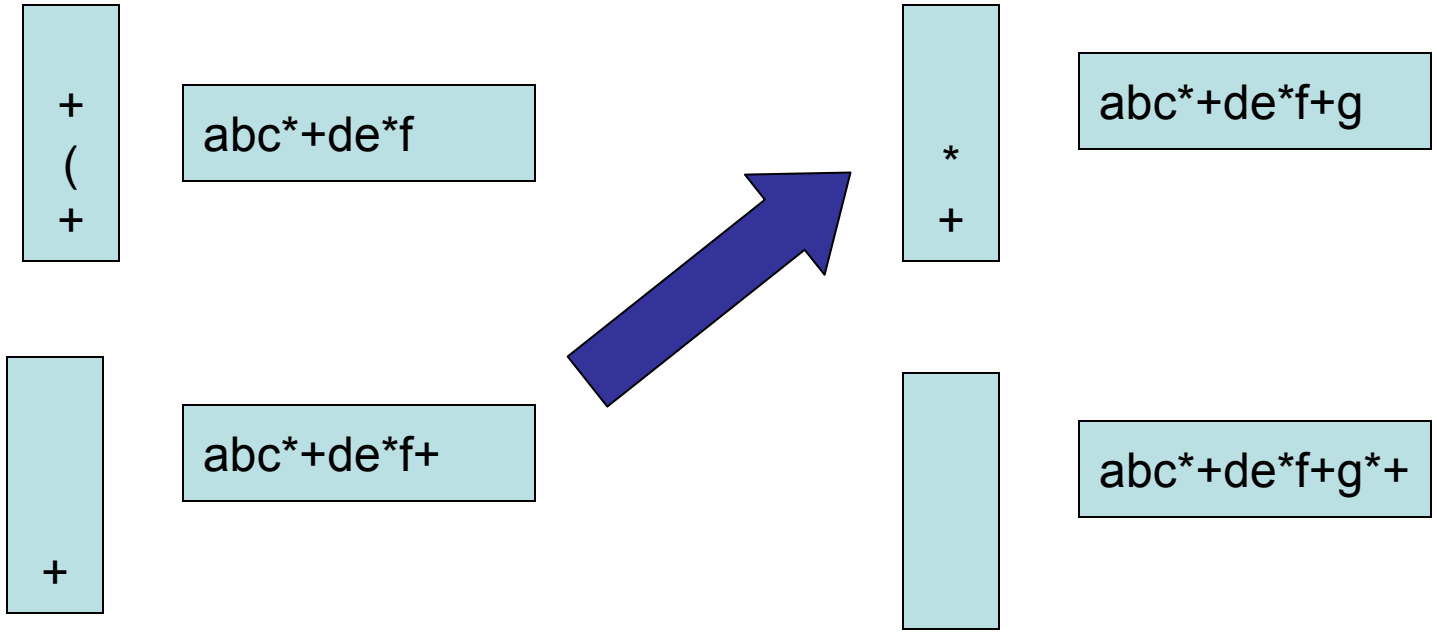
Example

- a+b*c+(d*e+f)*g



Example

- a+b*c+(d*e+f)*g

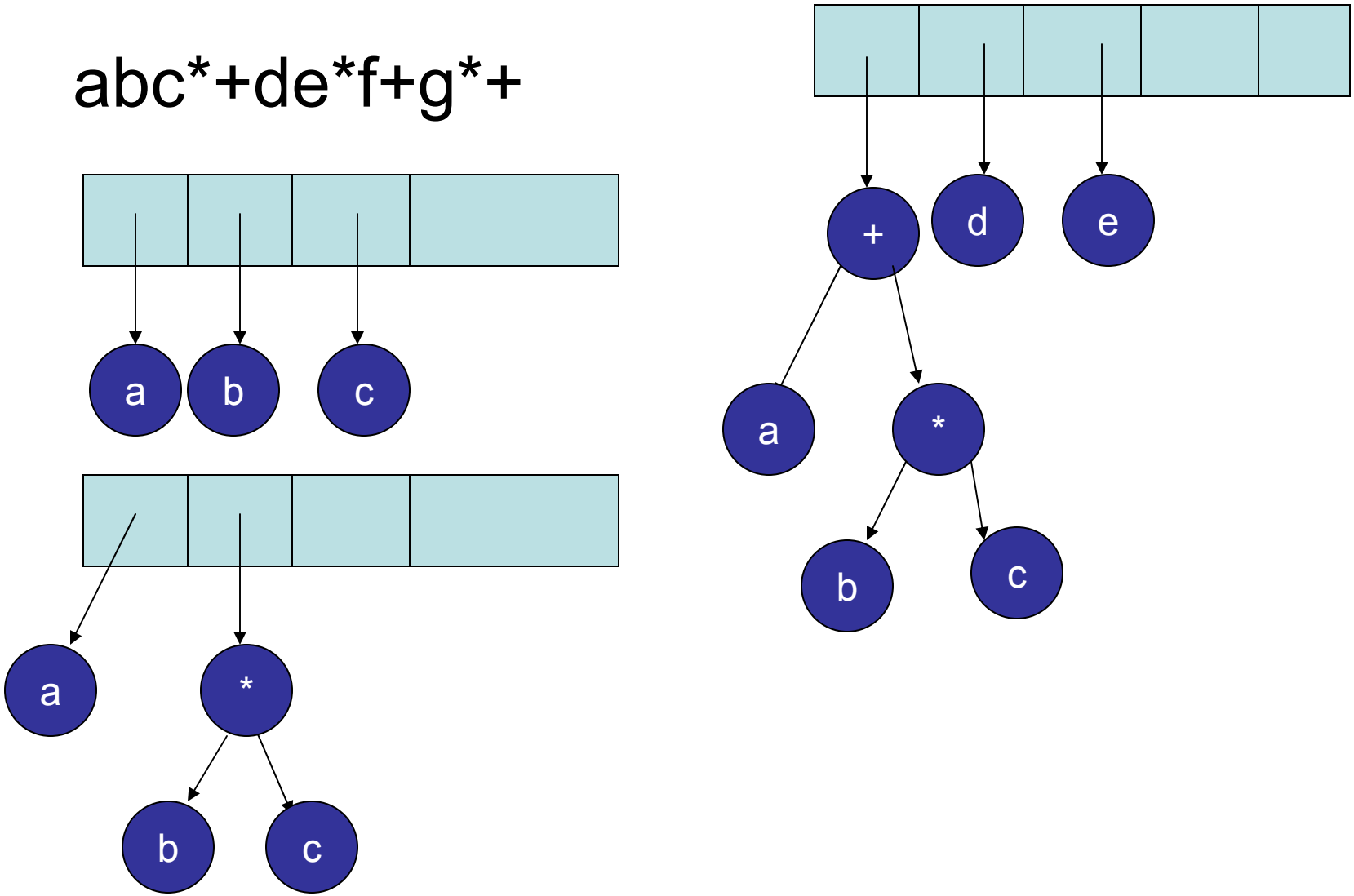


Expression Trees

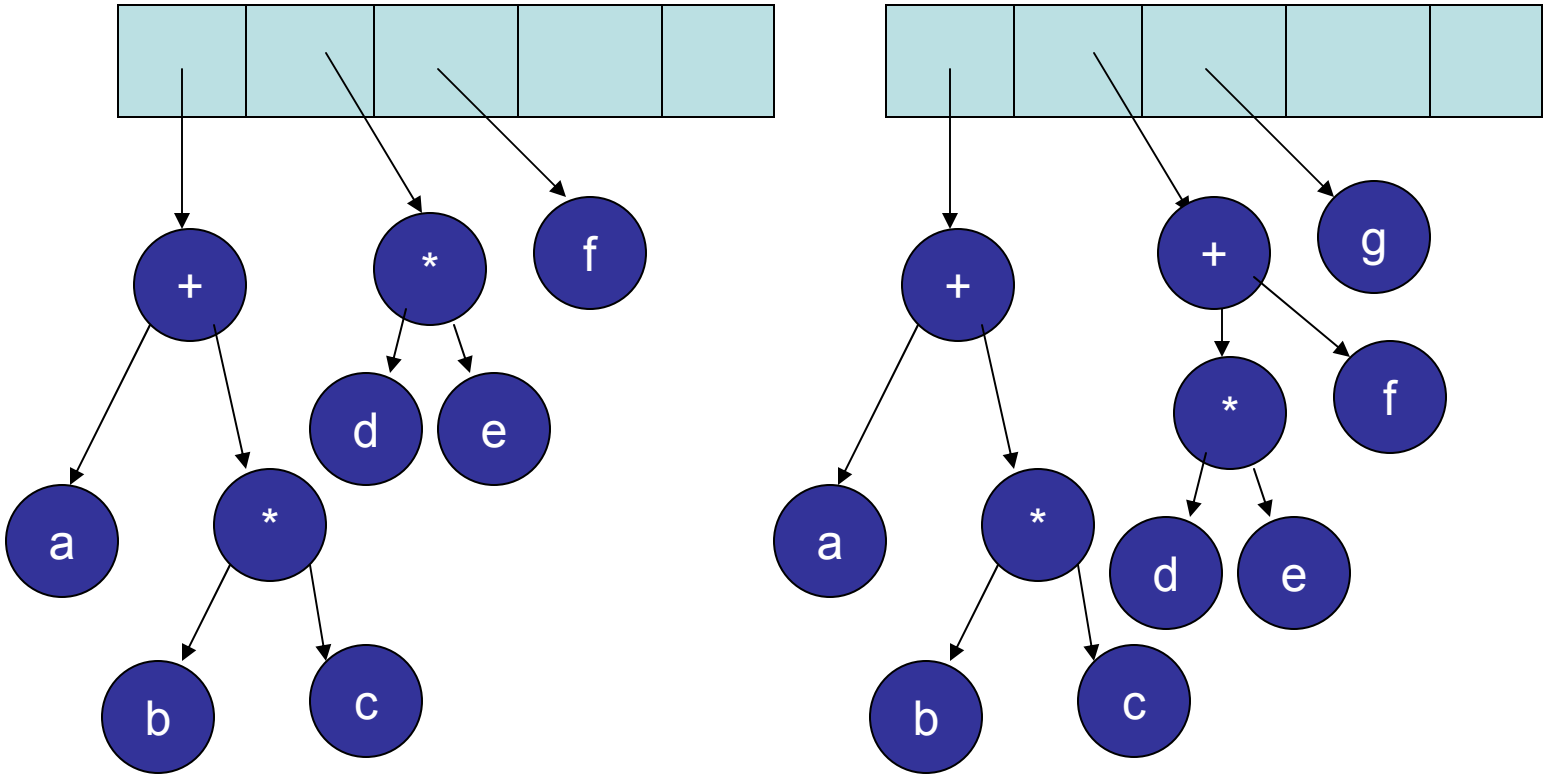
- Expression Trees are binary trees and compact representations of expressions
- We shall discuss a technique to convert postfix expressions to trees.
- **Method:**
  - If the symbol is an operand, create a one-node tree and push its pointer to a stack.
  - If the symbol is an operator, we pop twice to obtain pointers T_1 and T_2 to two trees.
  - If T_1 is first popped, then form a tree with the operator as root. The left child points to T_2 and right to T_1. Then push the new pointer to the stack.

Example

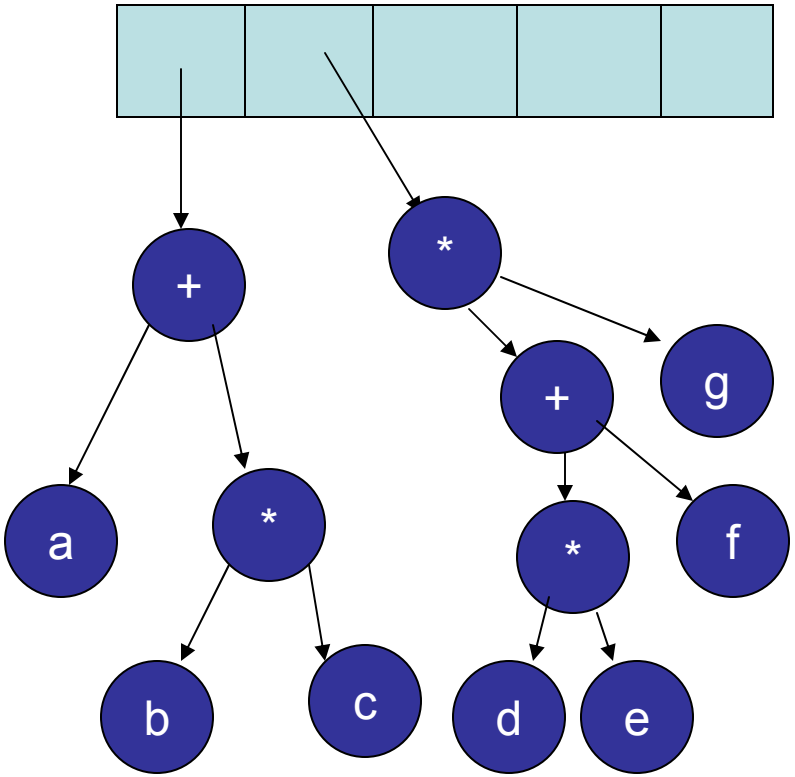
abc*+de*f+g*+

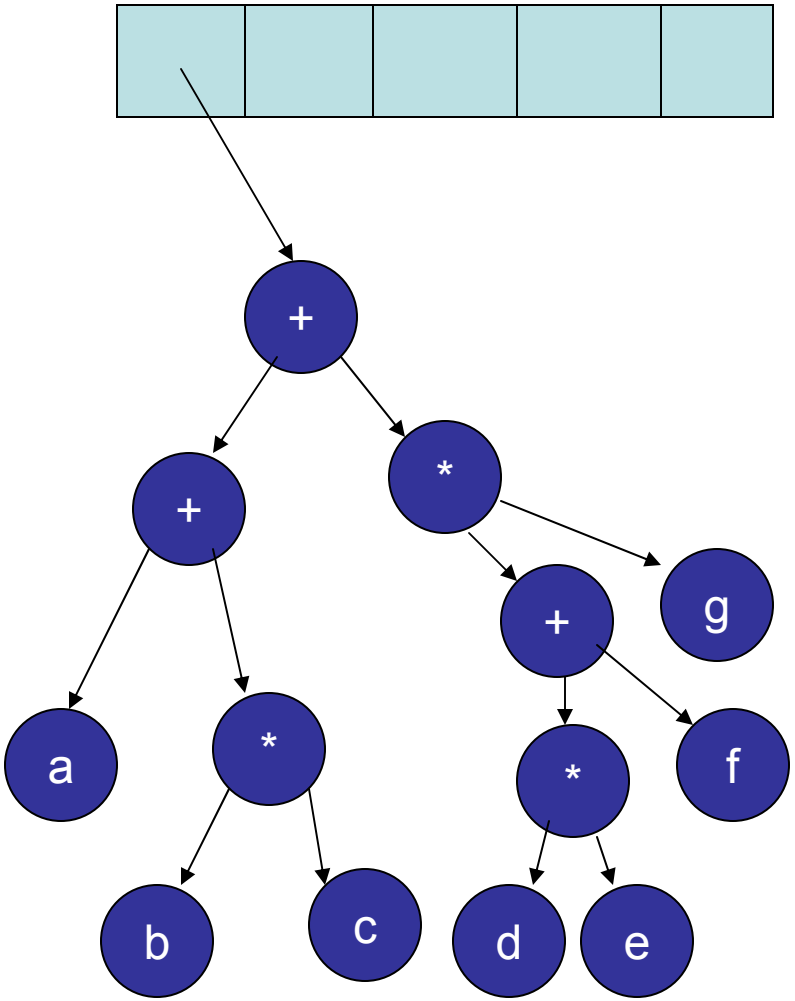


Example



Example



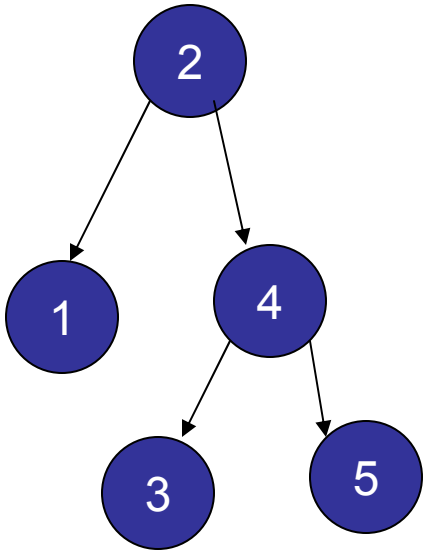


Height of the tree

- A binary tree with n nodes, n>0, is of height at least $\lfloor \log n \rfloor$
- Once, the tree is constructed we need to traverse the tree.
- There are three distinct methods for traversal:
  - Inorder
  - Preorder
  - Postorder
- The techniques may be recursively defined

Inorder

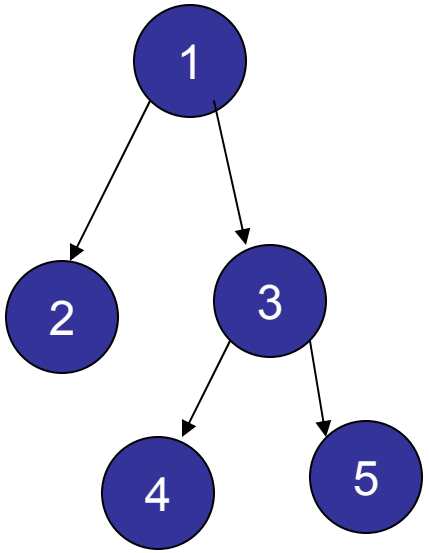
1. Traverse the left tree(wrt root), in order.
2. Traverse the root.
3. Traverse the right tree(wrt root), in order.



Example: Labels define the order of traversal

Preorder

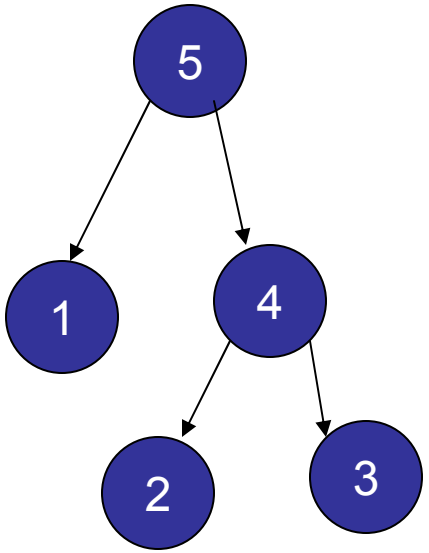
1. Traverse the root.
2. Traverse the left tree(wrt root), in order.
3. Traverse the right tree(wrt root), in order.



Example: Labels define the order of traversal

Postorder

1. Traverse the left tree(wrt root), in order.
2. Traverse the right tree(wrt root), in order.
3. Traverse the root.

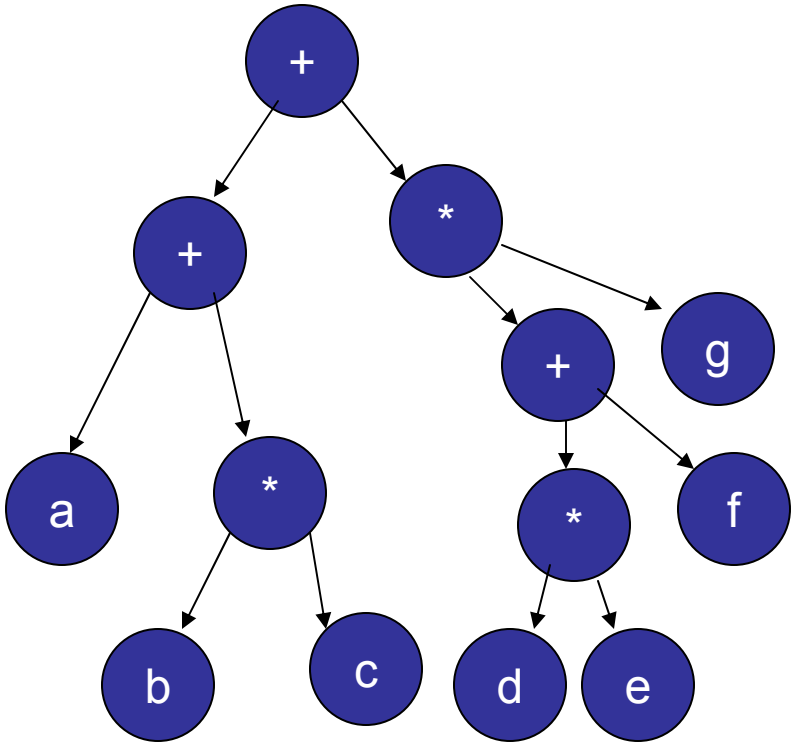


Example: Labels define the order of traversal

Traversing the expression tree

- Postorder Traversal
- abc*+de*f+g*+

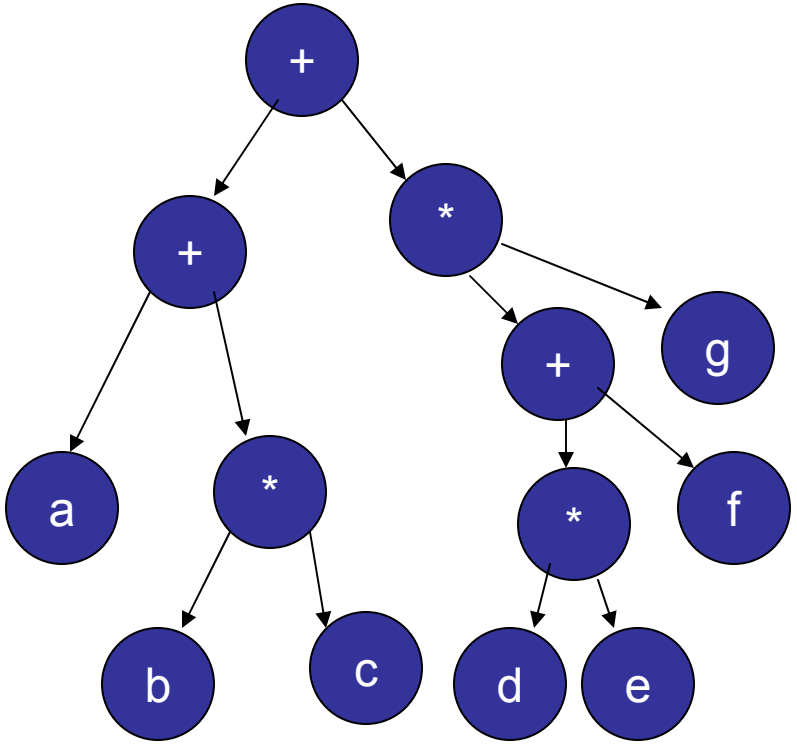
  (gives back the postfix expression)



Evaluate the expression tree

- Use Recursion

1. If the root is an operand, return value.
2. Else evaluate the left tree.
3. Evaluate the right tree.
4. Apply the operator on the two returned results.
5. Return the result.



Result: a+b*c+(d*e+f)*g

A possible representation

- #define OPERATOR 0
- #define OPERAND 1
- struct nodetype{
  shortint utype;
  union{
  char chinfo;
  float numinfo;
  }info;
  struct nodetype *left;
  struct nodetype *right;
  };
  typedef struct nodetype *NODEPTR;