

Trees



Rupesh Nasre.
rupesh@iitm.ac.in

Web: ~rupesh/teaching/pds/spw2019

Summer Projects and Workshop
June 2019

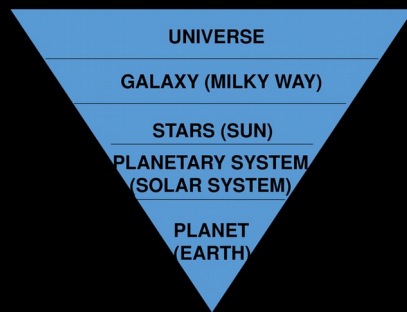
Manager-Employee Relation



Google Maps

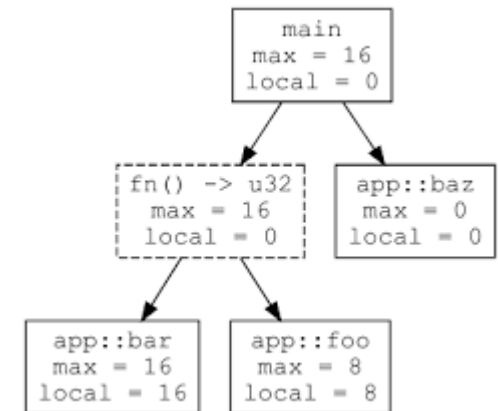


Hierarchy of the Universe

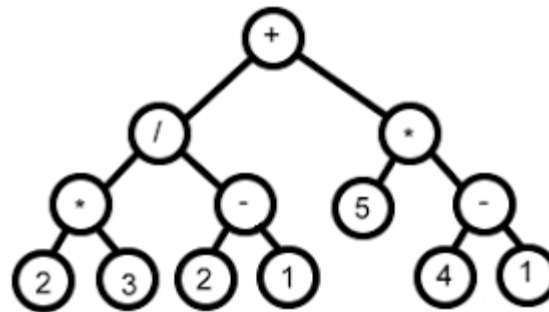


Planetary Hierarchy

Trees



Modeling Computation



Expression tree for $2*3/(2-1)+5*(4-1)$

Expression Evaluation

Nomenclature

- Root
 - Stem
 - Branches
 - Leaves
 - ~~Fruits~~
 - ~~Flowers~~
- Edges



Definition

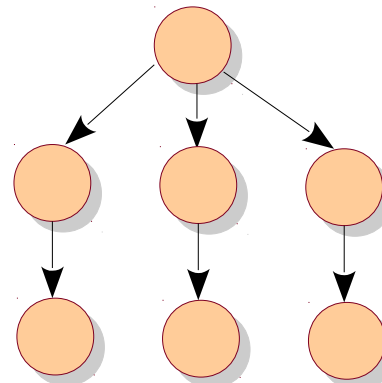
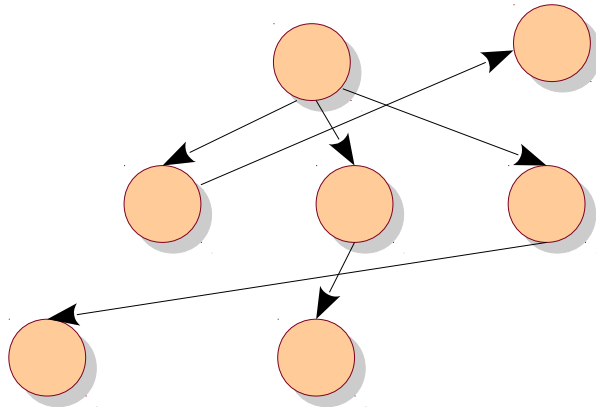
A **tree** is a collection of **nodes**.

It could be empty.

// base case

Otherwise, it contains a **root** node,
connected to zero or more (**child**) nodes,
each of which is a tree in itself!

// recursive



Alternatively, a tree is a collection of nodes and **directed** edges, such that each node except one has a single **parent**. The node without a parent node is the root.

Nomenclature

Root has no parent.

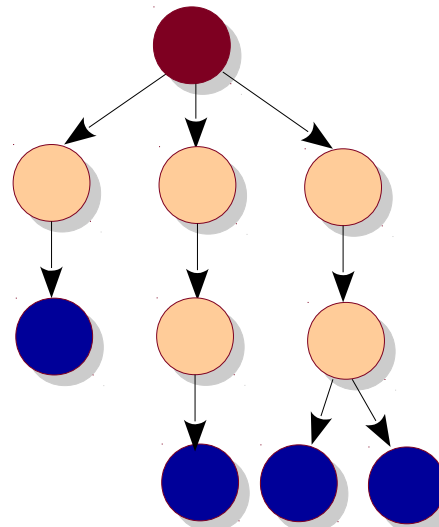
Leaves have no children.

Non-leaves are **internal nodes**.

Each node is reachable from the root.

The whole tree can be accessed via root.

Each node can be viewed as the root of its unique subtree.

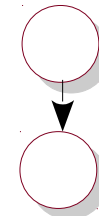


Empty Tree

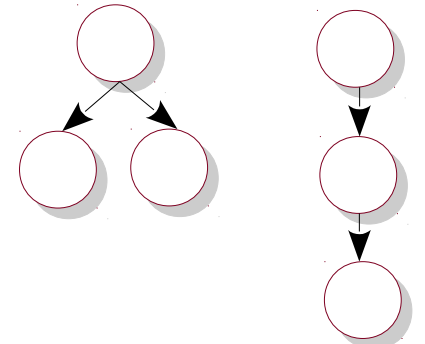
Tree with one node



Tree with two nodes

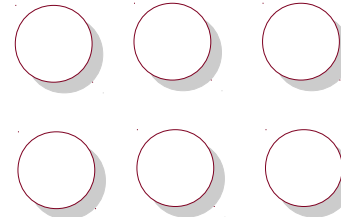


Trees with three nodes



Properties

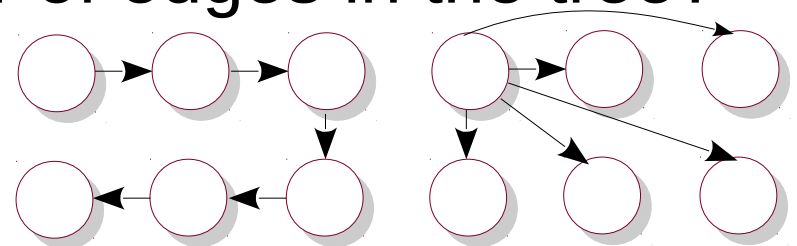
- A tree has six nodes.



- What is the minimum number of edges in the tree?

- What is the maximum?

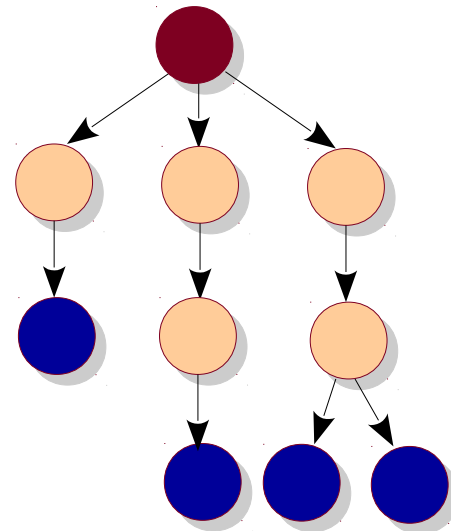
- Generalization for N nodes?



- How many (undirected) paths exist between two nodes?

More Nomenclature

- Sibling
 - What is the maximum number of siblings a node may have in an N node tree?
- Grandparent, grandchild
- Ancestor, descendant
- Path, length
- Height, depth



Exercises

- Given (a pointer to) a node in an employee tree, list all its direct and indirect subordinates.
- Same as above with the name of the employee given.
- Find distance between two nodes.
- Find tree diameter (max. distance).
- Convert infix to postfix.
- Mirror a tree.
- Find if there is a directed path from p to q .

Learning Outcomes

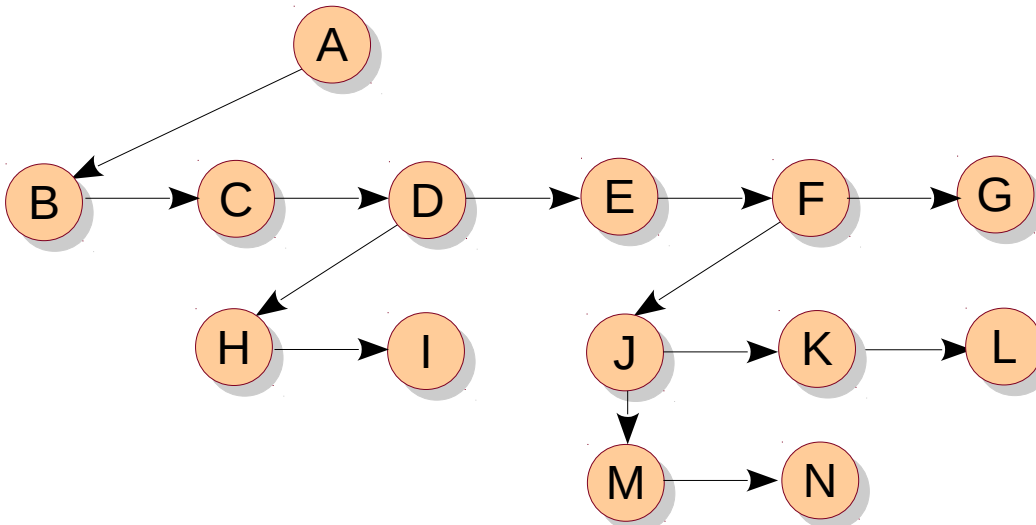
- Apply tree data structure in relevant applications.
- Construct trees in C++ and perform operations such as insert.
- Perform traversals on trees.
- Analyze complexity of various operations.

Implementation

- A challenge is that the maximum number of children is unknown, and may vary dynamically.

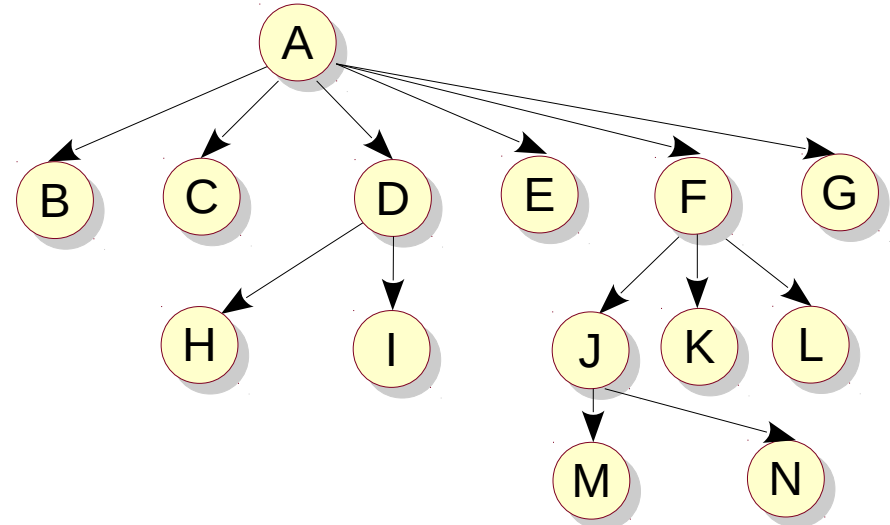
```
typedef struct TreeNode *PtrToNode;  
  
struct TreeNode {  
    int data;  
    PtrToNode firstChild;  
    PtrToNode nextSibling;  
};
```

C



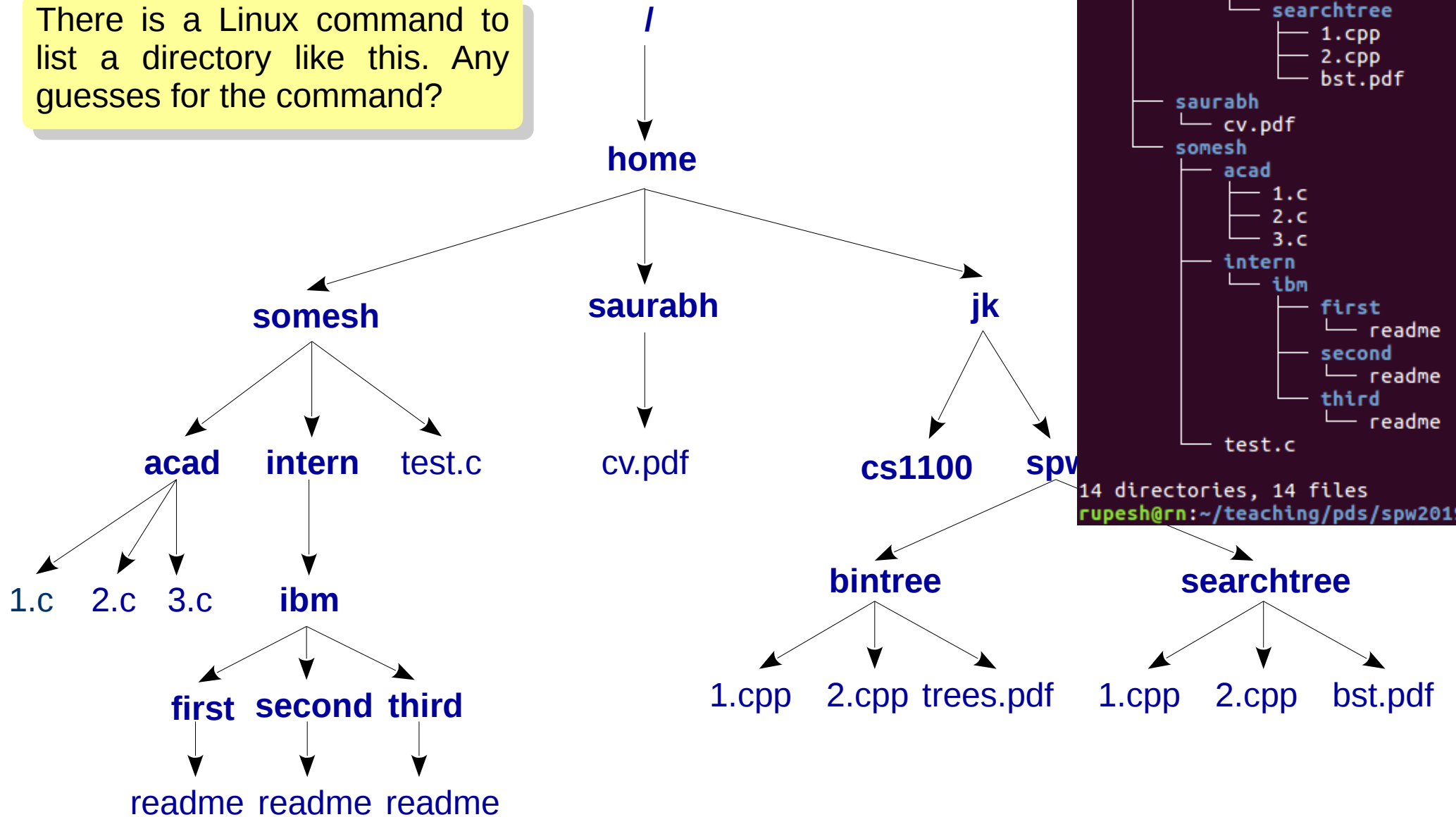
```
#include <vector>  
  
typedef struct TreeNode *PtrToNode;  
  
struct TreeNode {  
    int data;  
    std::vector<PtrToNode> children;  
};
```

C++



Directory Listing

There is a Linux command to list a directory like this. Any guesses for the command?



```
rupesh@rn:~/teaching/pds/spw2019
├── home
│   ├── jk
│   │   ├── cs1100
│   │   │   ├── bintree
│   │   │   │   ├── 1.cpp
│   │   │   │   ├── 2.cpp
│   │   │   │   └── trees.pdf
│   │   │   └── searchtree
│   │   │       ├── 1.cpp
│   │   │       ├── 2.cpp
│   │   │       └── bst.pdf
│   │   ├── saurabh
│   │   │   └── cv.pdf
│   │   └── somesh
│   │       ├── acad
│   │       │   ├── 1.c
│   │       │   ├── 2.c
│   │       │   └── 3.c
│   │       ├── intern
│   │       │   └── ibm
│   │       │       ├── first
│   │       │       │   └── readme
│   │       │       ├── second
│   │       │       │   └── readme
│   │       │       └── third
│   │       │           └── readme
│   │       └── test.c
│   └── test.c
└── test.c
14 directories, 14 files
rupesh@rn:~/teaching/pds/spw2019
```

Switch to code.

2.cpp and 3.cpp

Slides and code at <http://www.cse.iitm.ac.in/~rupesh/teaching/pds/spw2019/>
C++ basics at <http://www.cse.iitm.ac.in/~rupesh/teaching/ooaia/jan18/>

Traversals

- Preorder
 - Process each node before processing its children.
 - Children can be processed in any order.
- Postorder
 - Process each node after processing its children.
 - Children can be processed in any order.
- Preorder and postorder are examples of Depth-First Traversal.
 - Children of a node are processed before processing its siblings.
 - The other way is called Breadth-First or Level-Order Traversal.

Preorder

Iterative

```
void Tree::preorder() {  
    std::stack<PtrToNode> stack;  
    stack.push(root);  
  
    while (!stack.empty()) {  
        PtrToNode rr = stack.top();  
        stack.pop();  
        if (rr) {  
            rr->print();  
            for (auto child: rr->children)  
                stack.push(child);  
        }  
    }  
}
```

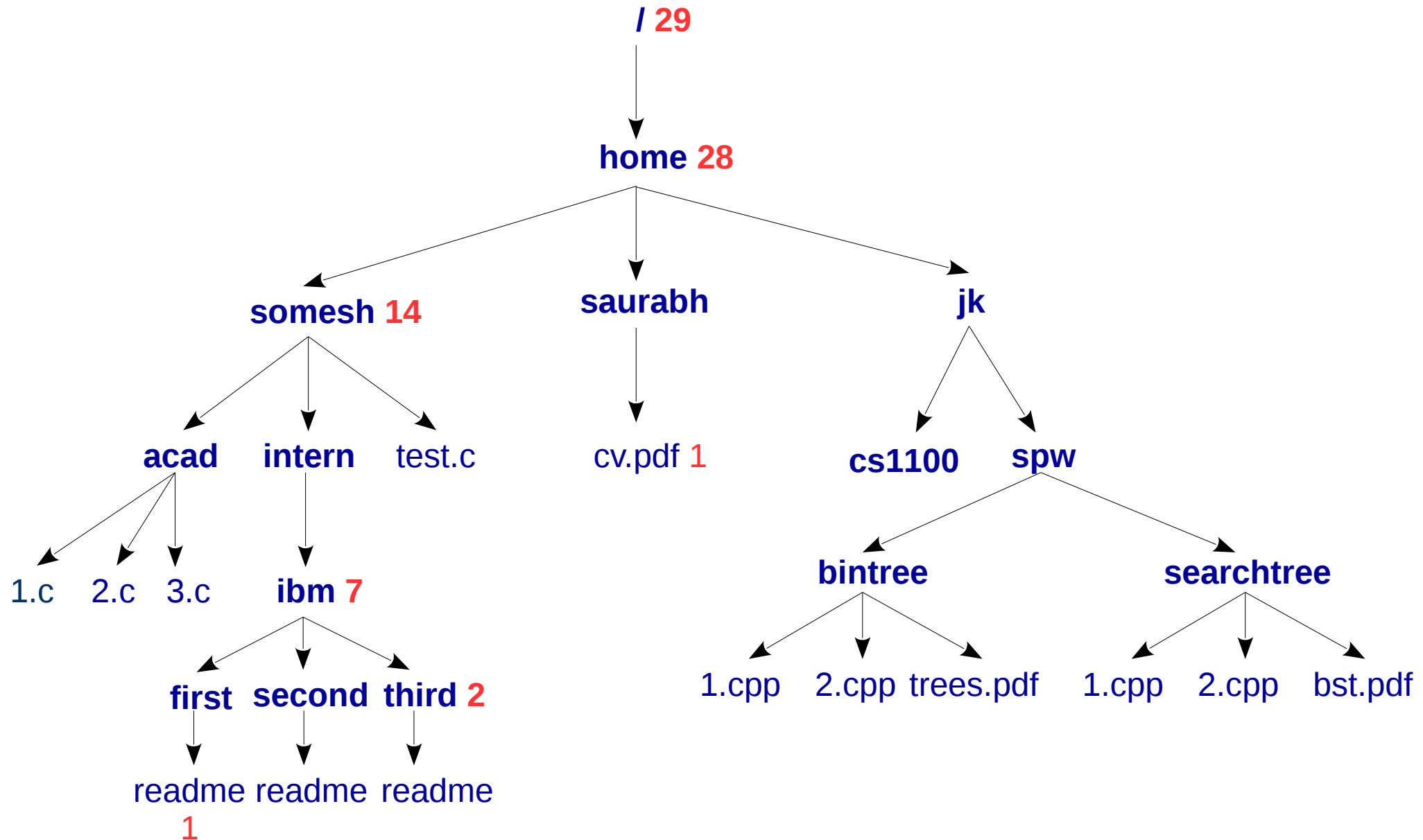
Recursive

```
void Tree::preorder(PtrToNode rr) {  
    if (rr) {  
        rr->print();  
        for (auto child: rr->children)  
            preorder(child);  
    }  
}  
  
void Tree::preorder() {  
    preorder(root);  
}
```

Switch to code: 4.cpp, 6.cpp

Labwork: Indent files as per their depth.

Find full size of each directory



Postorder

Iterative

Try it out in the lab.

Recursive

```
void Tree::postorder(PtrToNode rr) {  
    if (rr) {  
        for (auto child:rr->children)  
            postorder(child);  
        rr → print();  
    }  
}  
void Tree::postorder() {  
    postorder(root);  
}
```

Switch to code: 5.cpp

Story so far...

- **General trees**
 - arbitrary number of children
 - Resembles several situations such as employees, files, ...
- **Special trees**
 - Fixed / bounded number of children
 - Resembles situations such as expressions, boolean flows, ...
 - All the children may not be present.

K-ary Trees

```
typedef struct TreeNode *PtrToNode;

struct TreeNode {
    int data;
    PtrToNode firstChild;
    PtrToNode nextSibling;
};
```

```
#include <vector>
typedef struct TreeNode *PtrToNode;

struct TreeNode {
    int data;
    std::vector<PtrToNode> children;
};
```

For a fixed K

```
typedef struct TreeNode *PtrToNode;

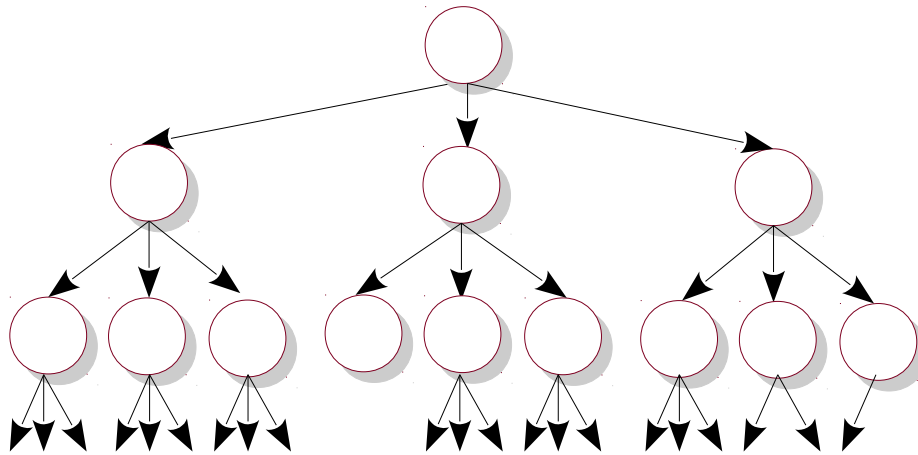
struct TreeNode {
    int data;
    PtrToNode children[K];
};
```

When K == 2

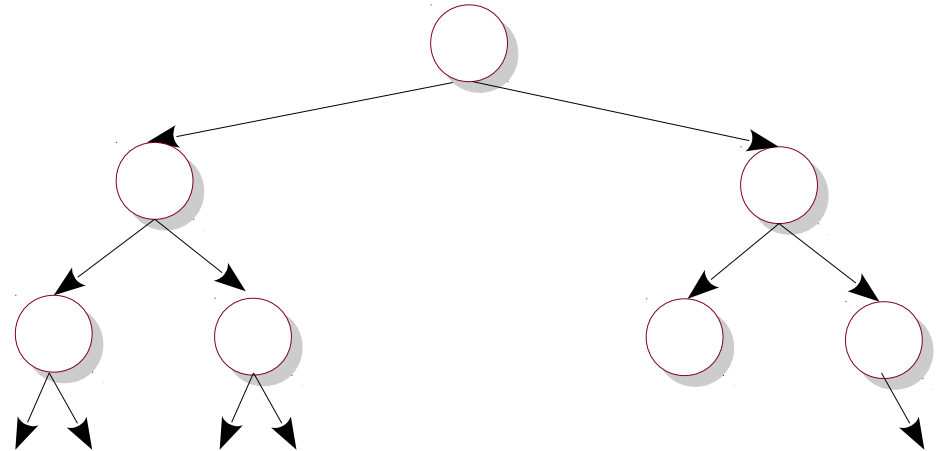
```
typedef struct TreeNode *PtrToNode;

struct TreeNode {
    int data;
    PtrToNode left;
    PtrToNode right;
};
```

K-ary Trees



Ternary



Binary

For a fixed K

```
typedef struct TreeNode *PtrToNode;  
  
struct TreeNode {  
    int data;  
    PtrToNode children[K];  
};
```

When K == 2

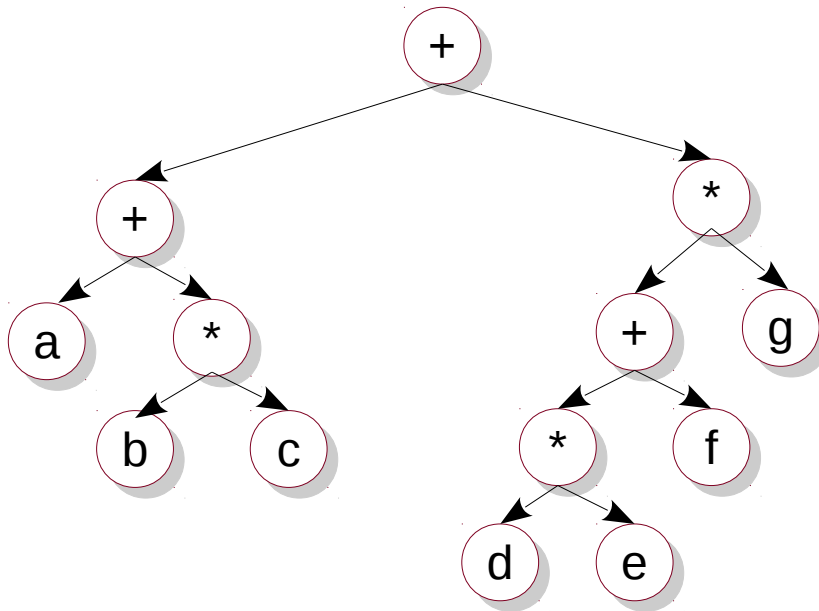
```
typedef struct TreeNode *PtrToNode;  
  
struct TreeNode {  
    int data;  
    PtrToNode left;  
    PtrToNode right;  
};
```

Properties of Binary Trees

- For an N node binary tree ($N > 0$):
 - What is the maximum height? N
 - What is the minimum height? $\log_2(N)$
 - How many NULL pointers? $N + 1$
 - How many min/max leaves? $0/1, N - 1$
- What is the maximum number of nodes a binary tree of height H may have? $2^H - 1$
- Full nodes (nodes with two children):
 - how many minimum, maximum? $0, N / 2 - 1$
- Show that $\# \text{full nodes} + 1 == \# \text{leaves}$ in a non-empty binary tree.

Expression Trees

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



Where did the parentheses go?

Can we write the expression itself in a way that no parentheses are required?

Traversals

- preorder (NLR)
- postorder (LRN)
- inorder (LNR)

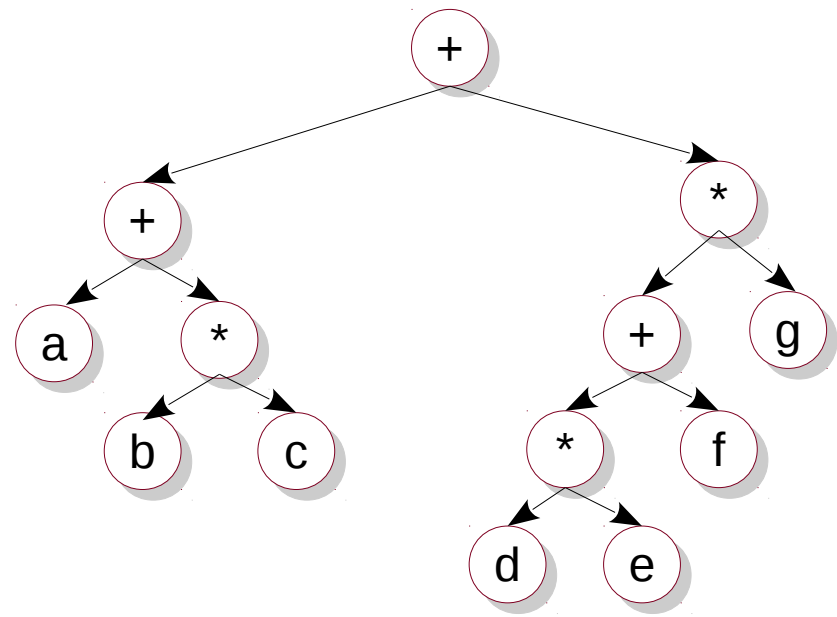
```
void Tree::inorder(PtrToNode rr) {  
    if (rr) {  
        inorder(rr->left);  
        rr->print();  
        inorder(rr->right);  
    }  
}  
void Tree::inorder() {  
    inorder(root);  
    std::cout << std::endl;  
}
```

Find output of this code on
this example tree.

a+b*c+d*e+f*g

Actual expression:

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



Traversals

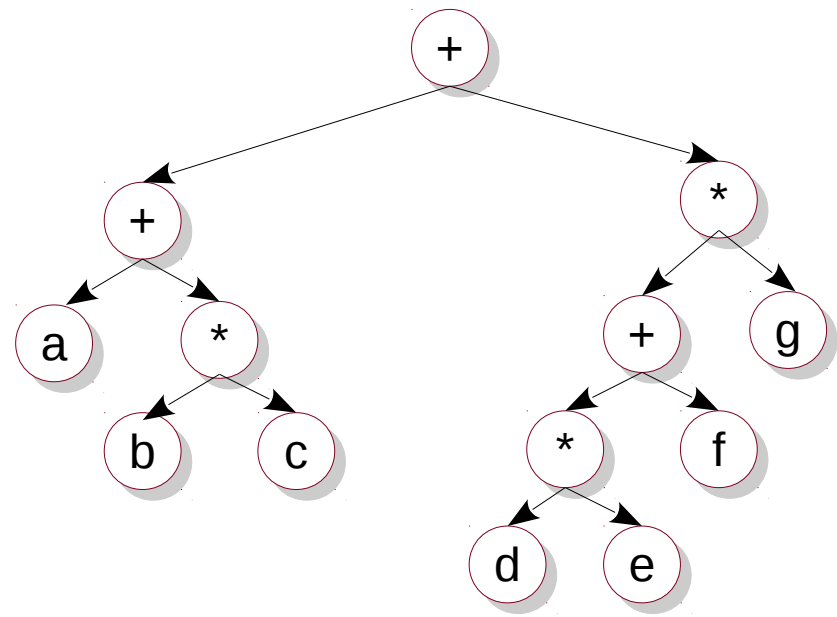
- preorder
- **Postorder** (7.cpp)
- inorder

```
void Tree::postorder(PtrToNode rr) {  
    if (rr) {  
        postorder(rr->left);  
        postorder(rr->right);  
        rr->print();  
    }  
}  
void Tree::postorder() {  
    postorder(root);  
    std::cout << std::endl;  
}
```

Find output of this code on this example tree.

abc*+de*f+g*+

Operator precedence encoded.



Infix, Prefix, Postfix

Infix	Prefix	Postfix
$A + B * C + D$		
$(A + B) * (C + D)$		
$A * B + C * D$		
$A + B + C + D$		
$A * B * C + D$		

Infix, Prefix, Postfix

Infix	Prefix	Postfix
$A + B * C + D$	$++A * B C D$	$A B C * + D +$
$(A + B) * (C + D)$	$* + A B + C D$	$A B + C D + *$
$A * B + C * D$	$+ * A B * C D$	$A B * C D * +$
$A + B + C + D$	$+++A B C D$	$A B + C + D +$
$A * B * C + D$	$+ ** A B C D$	$A B * C * D +$

- The order of operands (A, B, C, D) remain the same in all the expressions.
- Operators in prefix are in the opposite order compared to their postfix versions.

Evaluating postfix

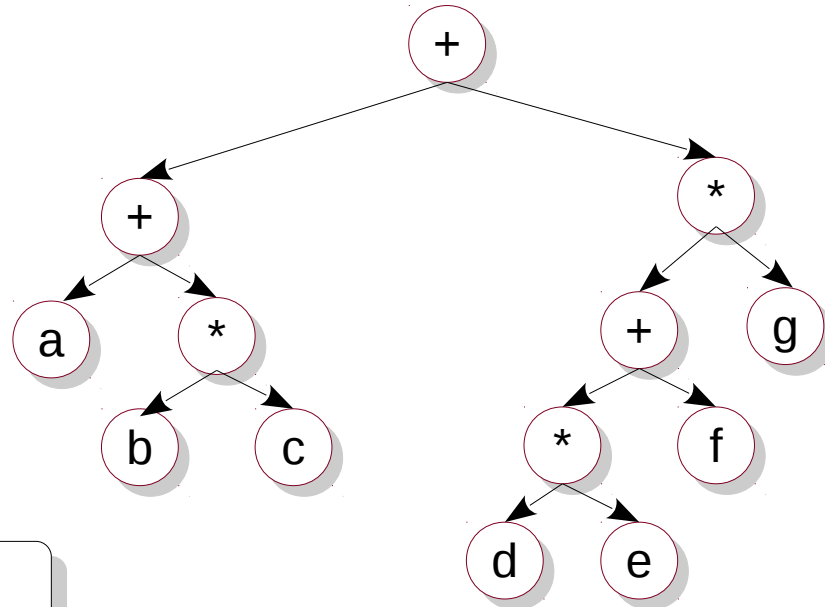
- Find the value of $5\ 1\ 2\ 3\ *\ -\ 4\ +\ 6\ *\ -$.
- Write a program to evaluate a postfix expression.
 - Assume digits, +, −, *, /.

For each symbol in the expression
If the symbol is an **operand**
 Push its value to a stack
Else if the symbol is an **operator**
 Pop two nodes from the stack
 Apply the operator on them
 Push result to the stack

Switch to postfixeval.cpp

Postfix to Expression Tree

a b c * + d e * f + g * +



For each symbol in the expression
If the symbol is an **operand**
Push its node to stack
Else if the symbol is an **operator**
Pop two nodes from the stack
Connect those to the operator
Push root of the tree to stack

Switch to postfix2tree.cpp

Operations on Trees

- **Insert:** our addChild would take care of this.
 - Given pointers, this is constant time operation.
- **Remove:** Update parent's pointer to NULL (and free memory).
 - What if the node getting removed has children?
 - Based on the above answer, the complexity could be $O(1)$ or $O(N)$
- **Search:** Our tree traversals can help.
 - Can a tree contain duplicate values?
 - This is $O(N)$, since the whole tree needs to be searched **in the worst case**.

Some Questions?

- What if a child node is common to two parents?
 - Ancestry
- Can the edge be undirected?
- Can the edges have weights?
- Can there be multiple roots?
- Can there be multiple edges between two nodes?
- Is it okay to draw a tree with root at the bottom?

Exercises

- Given (a pointer to) a node in an employee tree, list all its direct and indirect subordinates.
- Same as above with the name of the employee given.
- Find distance between two nodes.
- Find tree diameter (max. distance).
- Convert infix to postfix.
- Mirror a tree.
- Find if there is a directed path from p to q .

Learning Outcomes

- Apply tree data structure in relevant applications.
- Construct trees in C++ and perform operations such as insert.
- Perform traversals on trees.
- Analyze complexity of various operations.