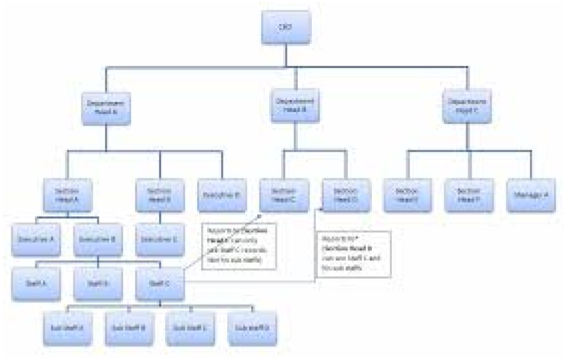
Trees

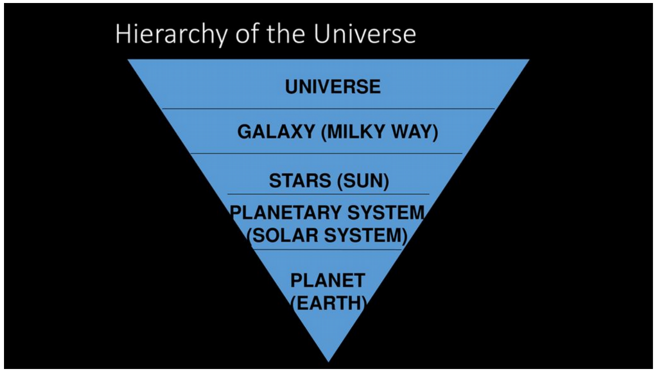
Rupesh Nasre.
rupesh@iitm.ac.in

July 2019

Manager-Employee Relation

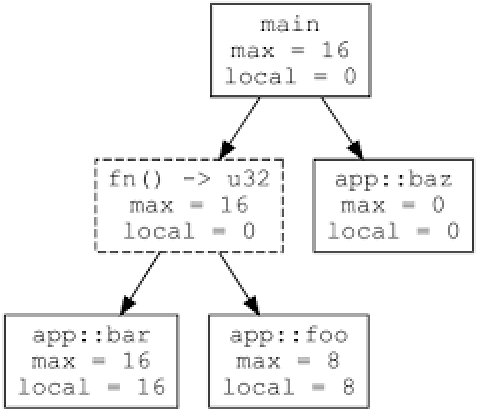


Google Maps

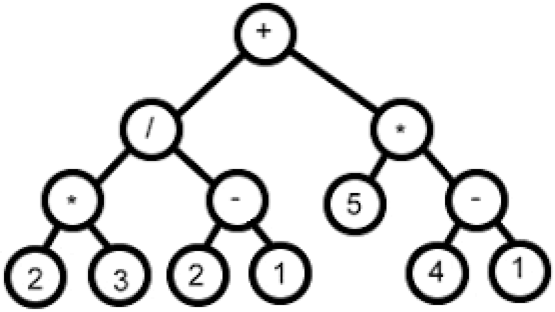


Planetary Hierarchy

Trees



Modeling Computation



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

Expression Evaluation

Nomenclature

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

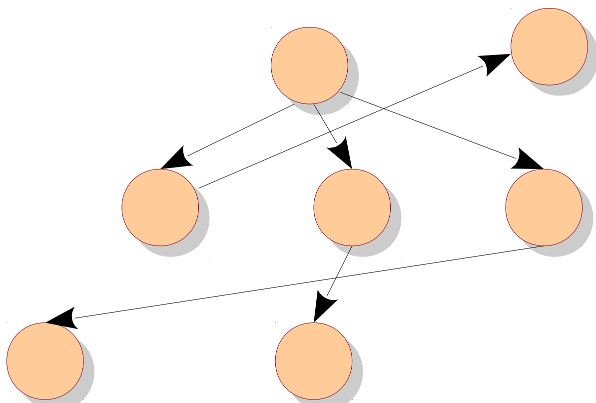
Edges

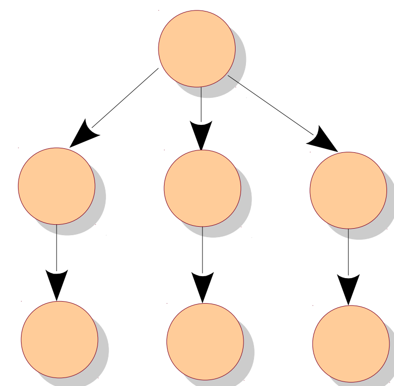
Definition

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*

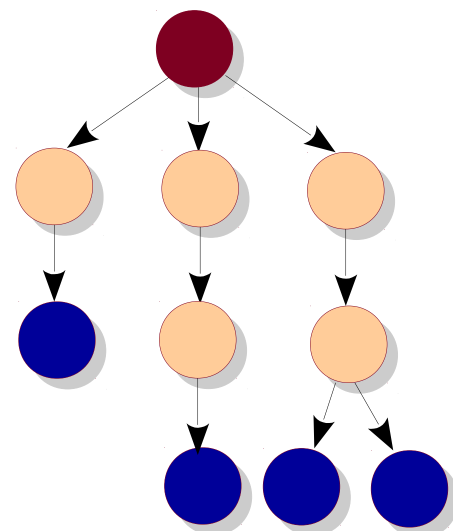




Alternatively, a tree is a collection of nodes and directed edges, such that each node except one has a single parent. The node without a parent node is the root.

Nomenclature

Root has no parent.
Leaves have no children.
Non-leaves are **internal nodes**.



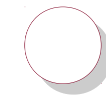
Each node is *reachable* from the root.

The whole tree can be accessed via root.

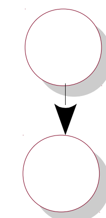
Each node can be viewed as the root of its unique *subtree*.

Empty Tree

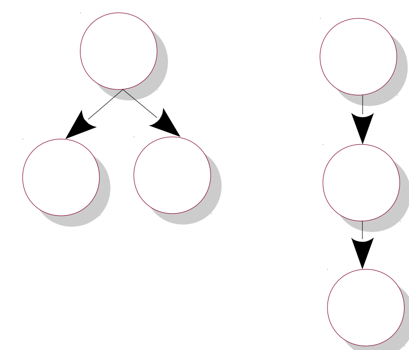
Tree with one node



Tree with two nodes

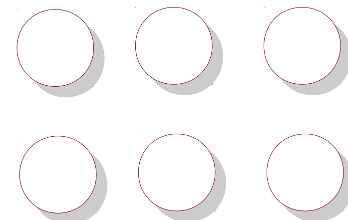


Trees with three nodes

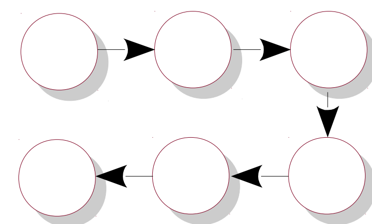


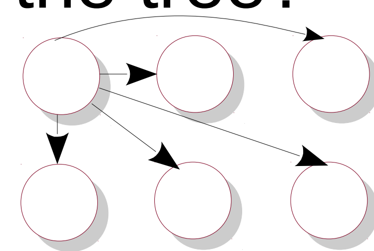
Properties

- A tree has six nodes.



  - What is the minimum number of edges in the tree?
  - What is the maximum?
  - Generalization for N nodes?

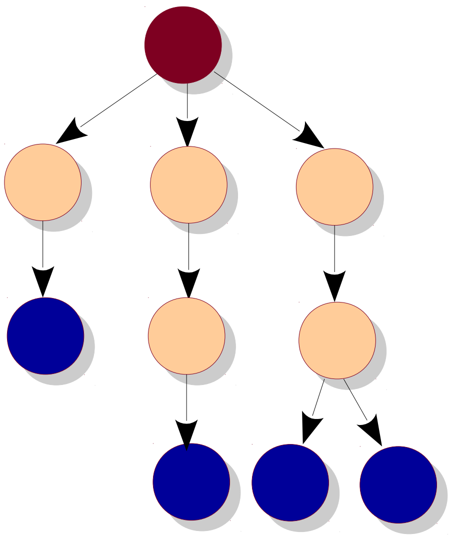




- How many (undirected) paths exist between two nodes?

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 (using a tree).
- 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.

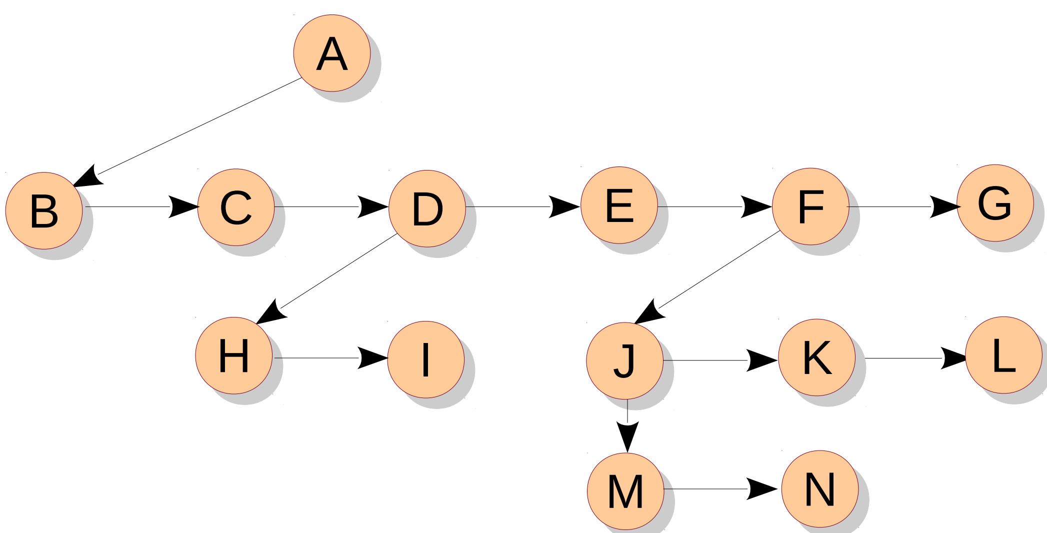
Implementation

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

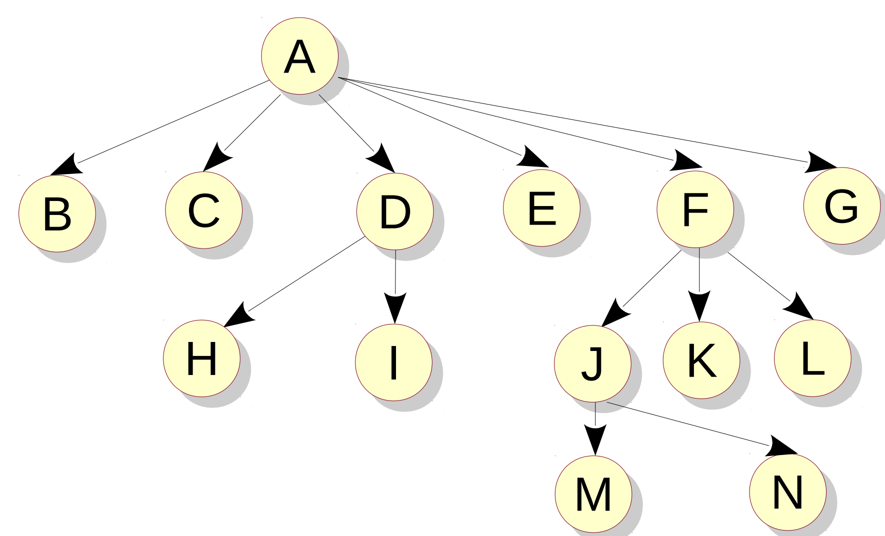
```c
typedef struct TreeNode *PtrToNode;

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

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

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

C

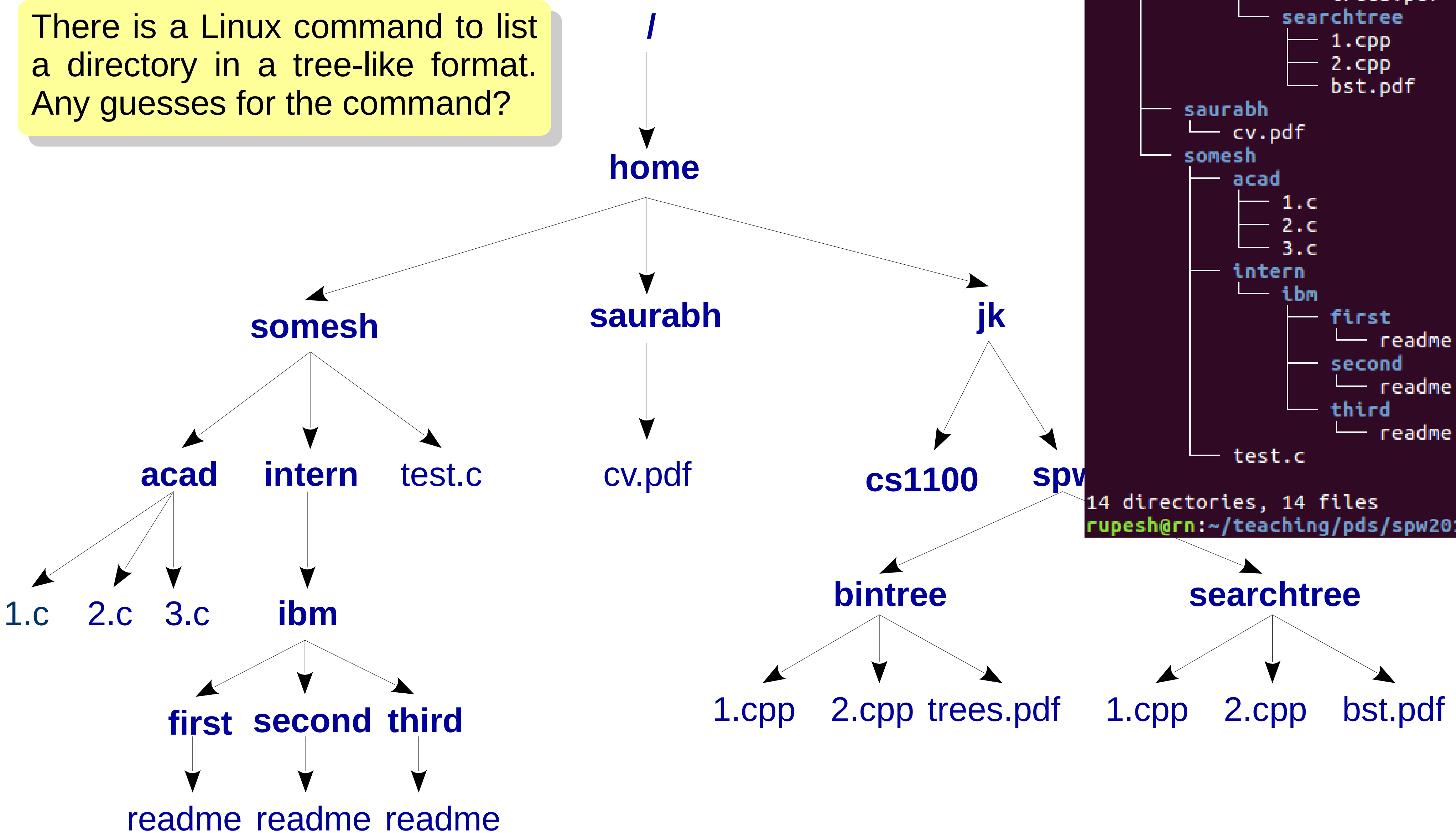


C++



Directory Listing

There is a Linux command to list a directory in a tree-like format. Any guesses for the command?




```
rupesh@rn:~/teaching/pds/spw201
.
└── home
    ├── jk
    │   ├── cs1100
    │   └── spw
    │       ├── 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

14 directories, 14 files
rupesh@rn:~/teaching/pds/spw201
```

Switch to code.

2.cpp and 3.cpp

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
Classwork: Indent files as per their depth. What is the code complexity? Note that indentation time also needs to be considered.

Surprize Quiz

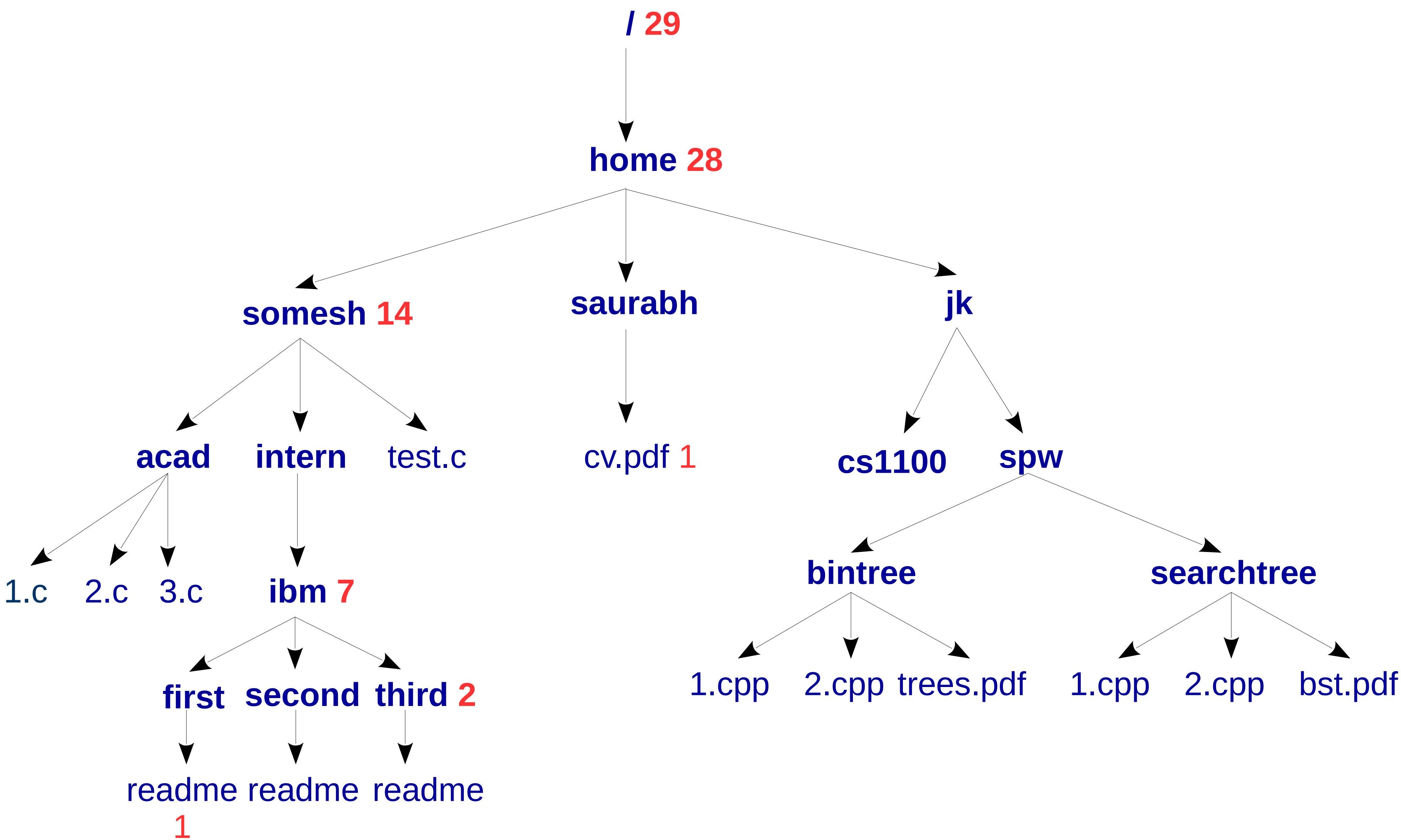
- What is *Triskaidekaphobia*?
- What is *paraskevidekatriaphobia*?

Stall numbers at Santa Anita Park progress from 12 to 12A to 14.

Numbers in a lift

Source: wikipedia

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;
};
```

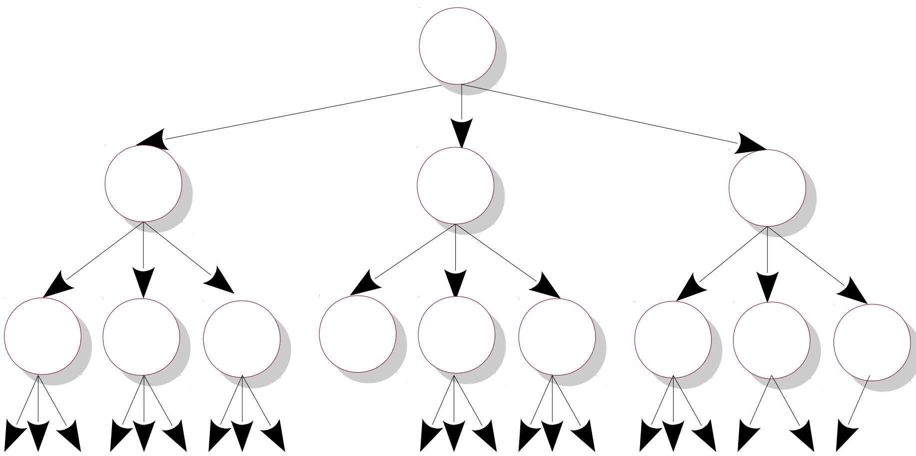
For a fixed K

```
typedef struct TreeNode *PtrToNode;

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

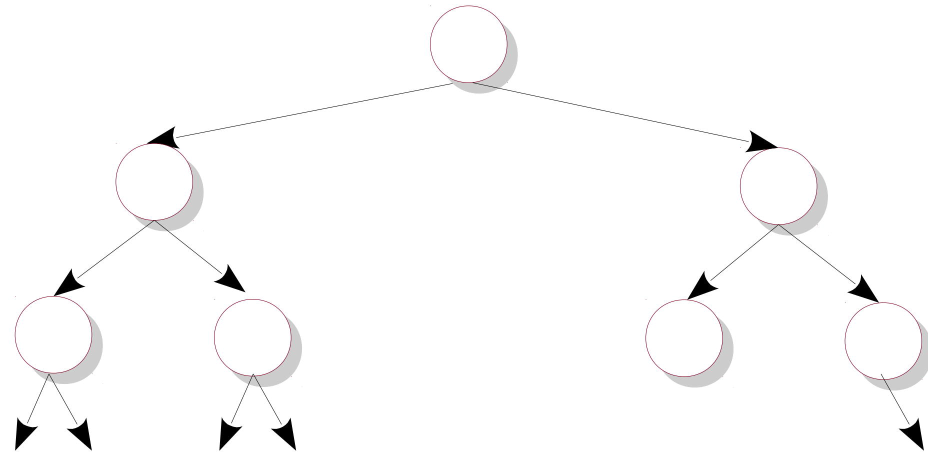
When K == 2

```
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

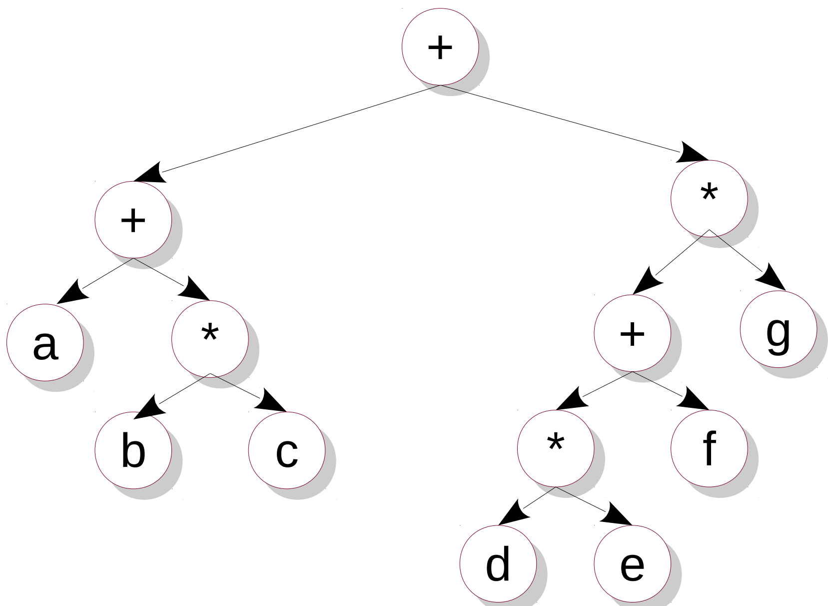
```
typedef struct TreeNode *PtrToNode;

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 / 2**
- 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 #full nodes + 1 == #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

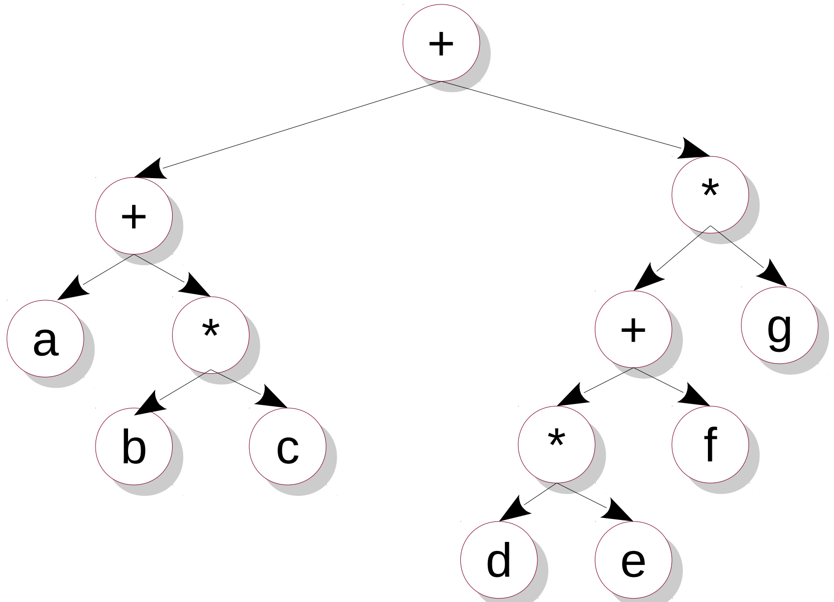
- preorder (NLR)
- postorder (LRN)
- inorder (LNR)

```
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

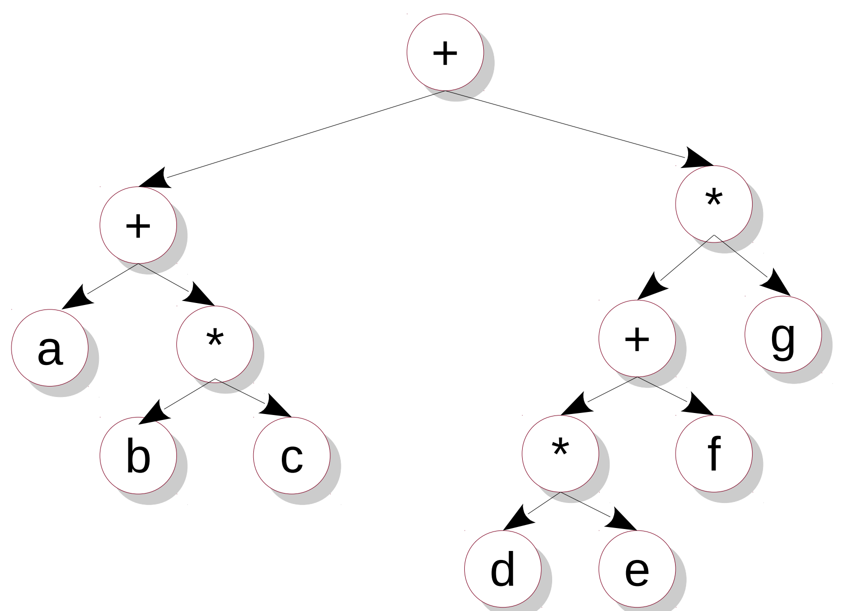
- preorder
- **Postorder** **(7.cpp)**
- inorder

```
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

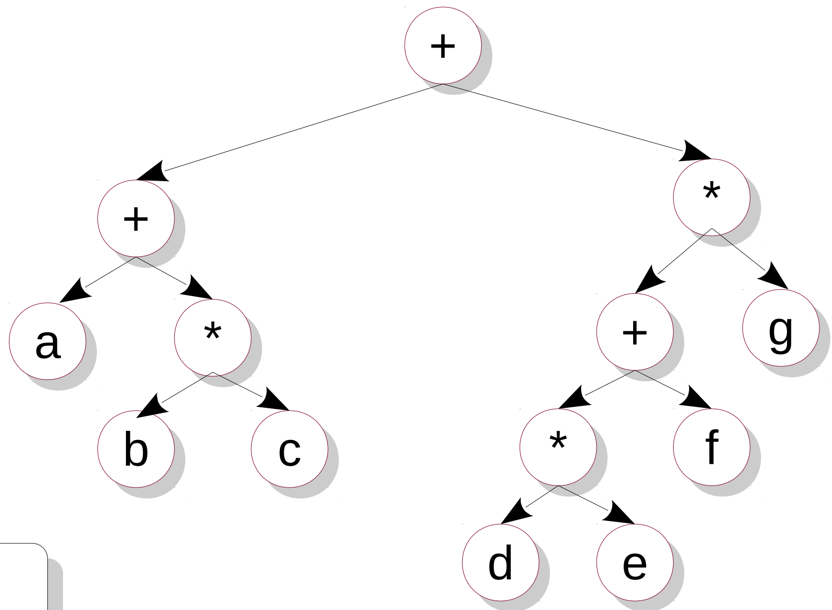
- Find the value of `5 1 2 3 * - 4 + 6 * -`.
- Write a program to evaluate a postfix expression.
  - Assume digits, +, –, *, /.

```
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.

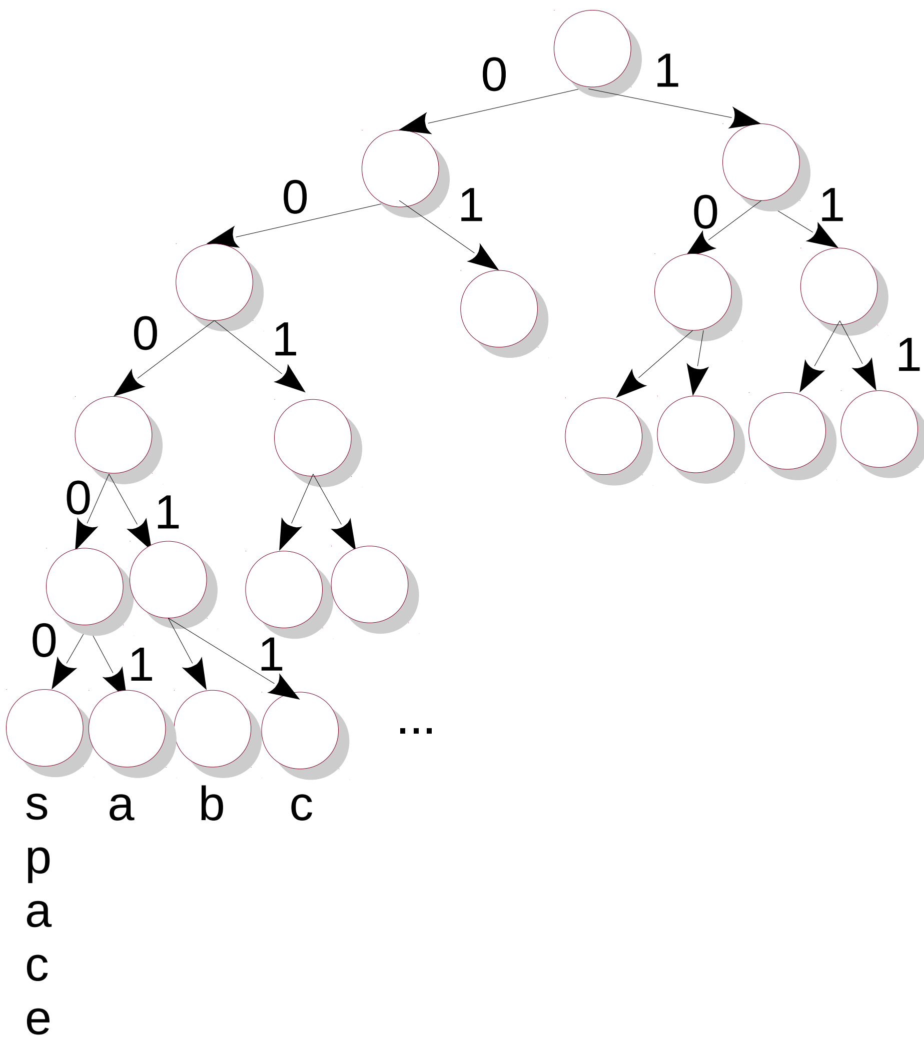
Some Questions?

- 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?

Coding (a little different)

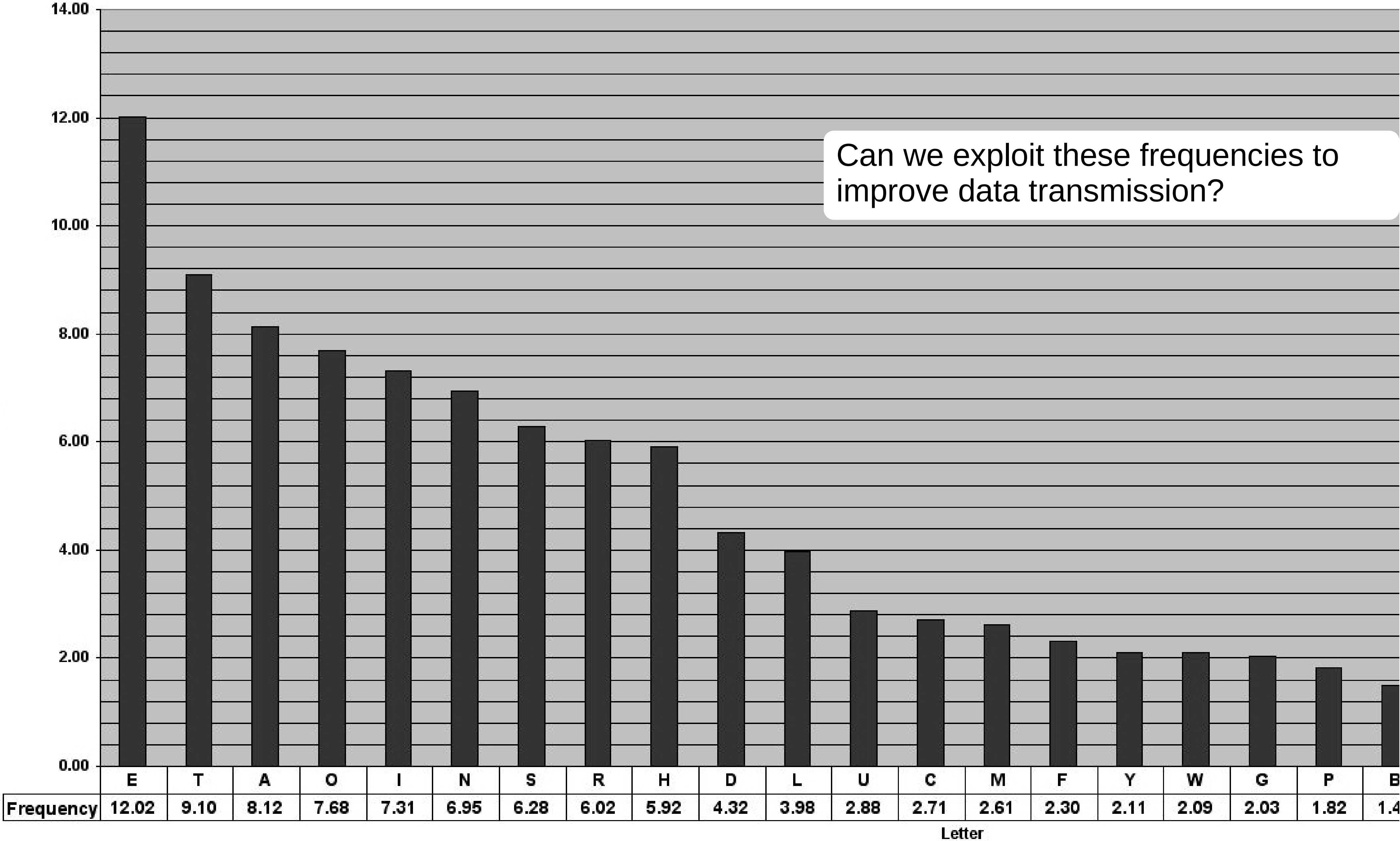
- I want to transmit some data.
- Data contains a-z and space.
- For these 27 characters, I need 5 bits.
  - For N characters, I need $\log_2(N)$ bits.
- Encoding pattern:
  - space = 00000, a = 00001, b = 00010, …, z = 11010
- Decoding:
  - Each 5-bit string represents a unique character (except the last five strings: 11011 to 11111).
  - What is 001001100101110000010110101111?

How is a code related to a tree?



- It is a binary tree.
- Tree is (almost) complete.
- Has height of 5, equal to the code length.
- Each character has a **unique** code, because each tree node has a unique path from the root.
- **Encoding**: Given a character, traverse back from its node toward the root, and we get reverse of its code.
- **Decoding**: Given a code, traverse the tree from the root, and the node we reach is the corresponding character.
- None of the interior nodes represents a character.

What is this plot?



Can we exploit these frequencies to improve data transmission?

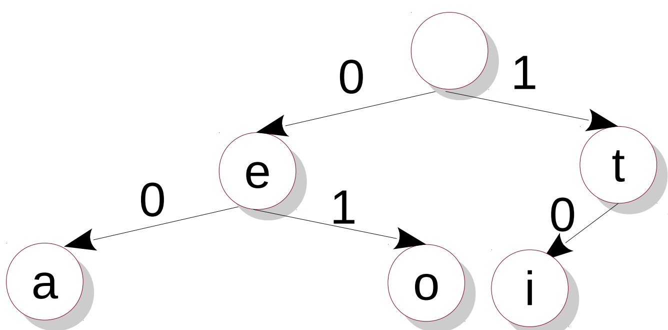
Source: http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

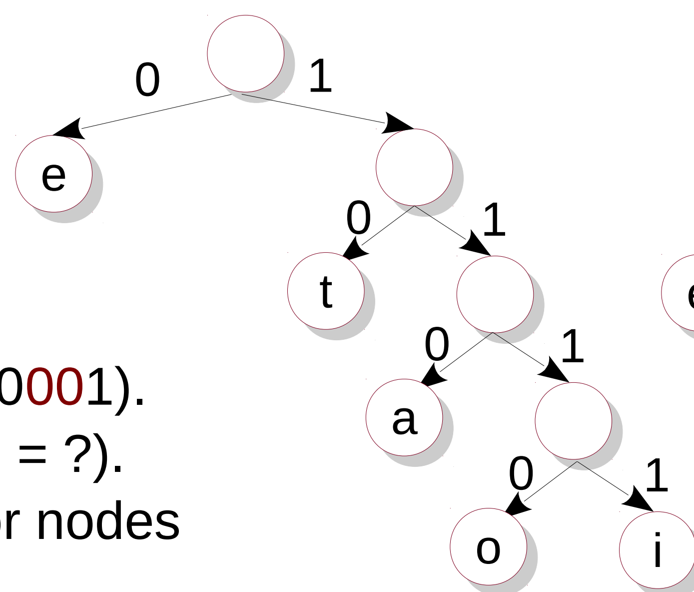
Make the common case faster!

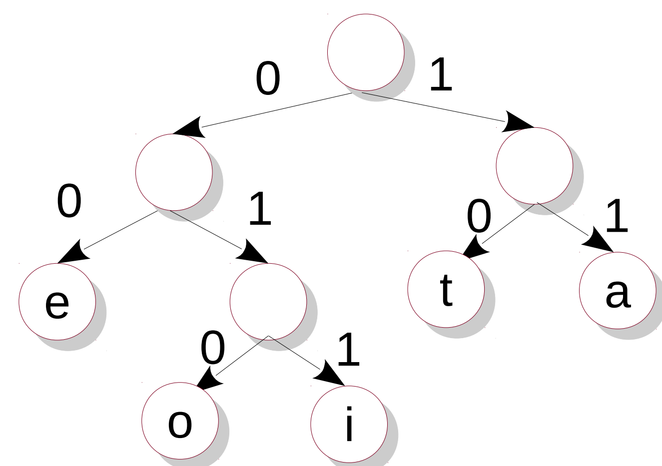
- If most people order vanilla ice-cream, keep it in front.
- If only a few students buy fish, keep it at a separate counter.
- If most people coming to the department use bicycle, bicycle parking should be prioritized.
- If most of the humans in the classroom stay in hostels, the classes should be held in hostels!
- If ‘e’ gets used more often, can we transmit it faster?

Shorter Codes

Character	Frequency	Code	Code 2	Code 3
e	12.02	0	0	00
t	9.10	1	10	10
a	8.12	00	110	11
o	7.68	01	1110	010
i	7.31	10	1111	011







Encoding is easy (e.g., eat = 0001).
Decoding is tough (e.g., 0001 = ?).
This happens because interior nodes also represent data.
We need data only at the leaves.

Called prefix codes.
We can 001110 easily now.

Prefix Codes

- Such codes were invented by Huffman.
  - as a term paper at MIT during his PhD.
  - had the habit of keeping poisonous snakes as pets.
- Prefix codes are easy to decode.
  - No ambiguous decoding possible.
- Faster transmission of frequent data.
  - In practice, close to 40-50% improvement
- We will study Huffman's algorithm during Heaps.

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.