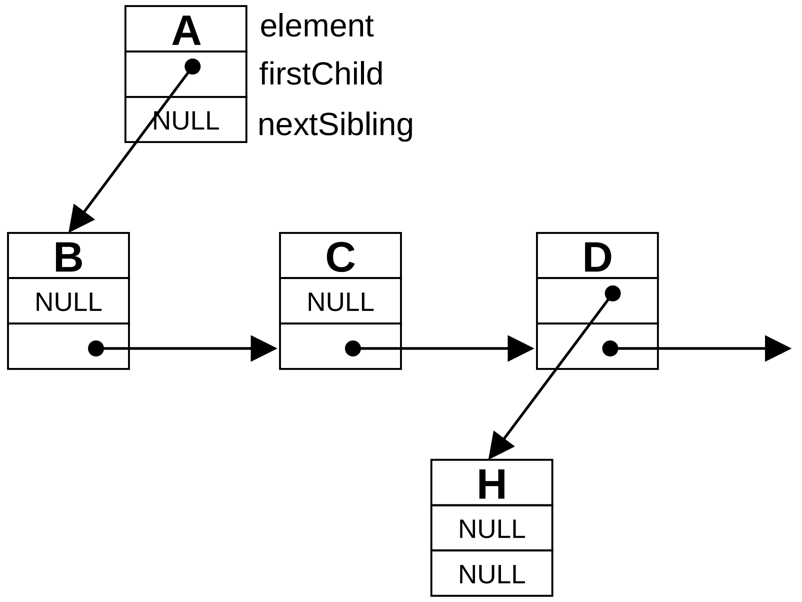
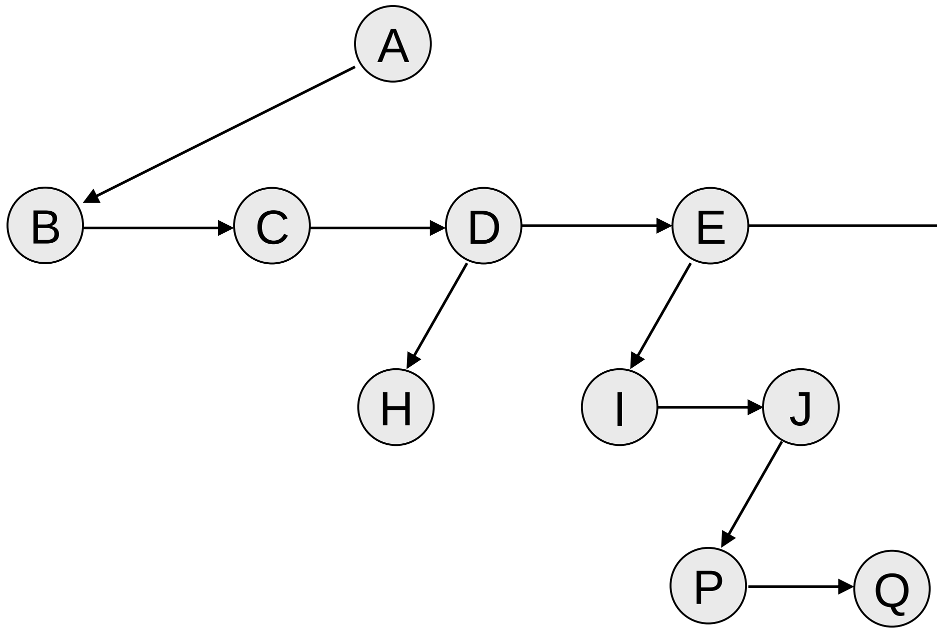
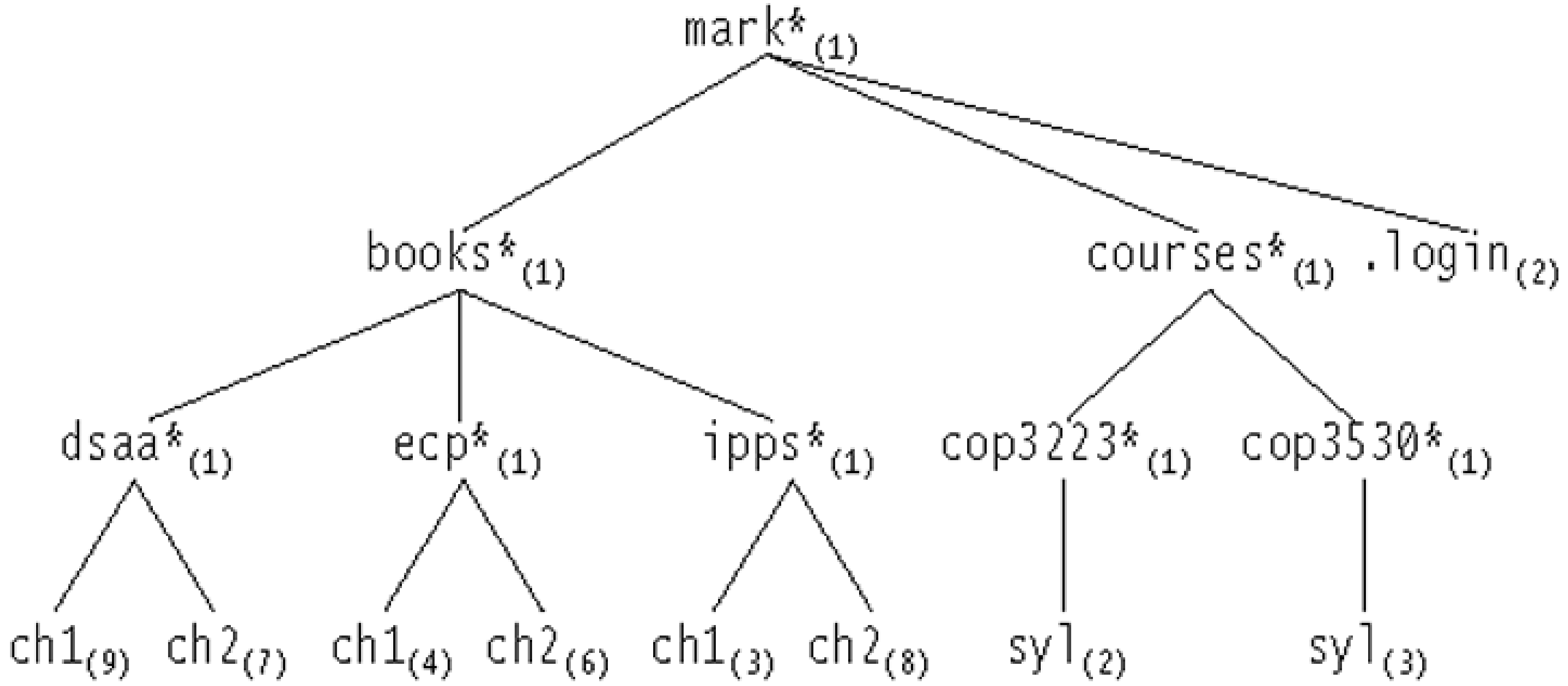
# Implementation of Trees

```
struct TreeNode {
    Object        element;
    struct TreeNode *firstChild;
    struct TreeNode *nextSibling;
};
```





## **Figure 2:** The Unix directory with file sizes

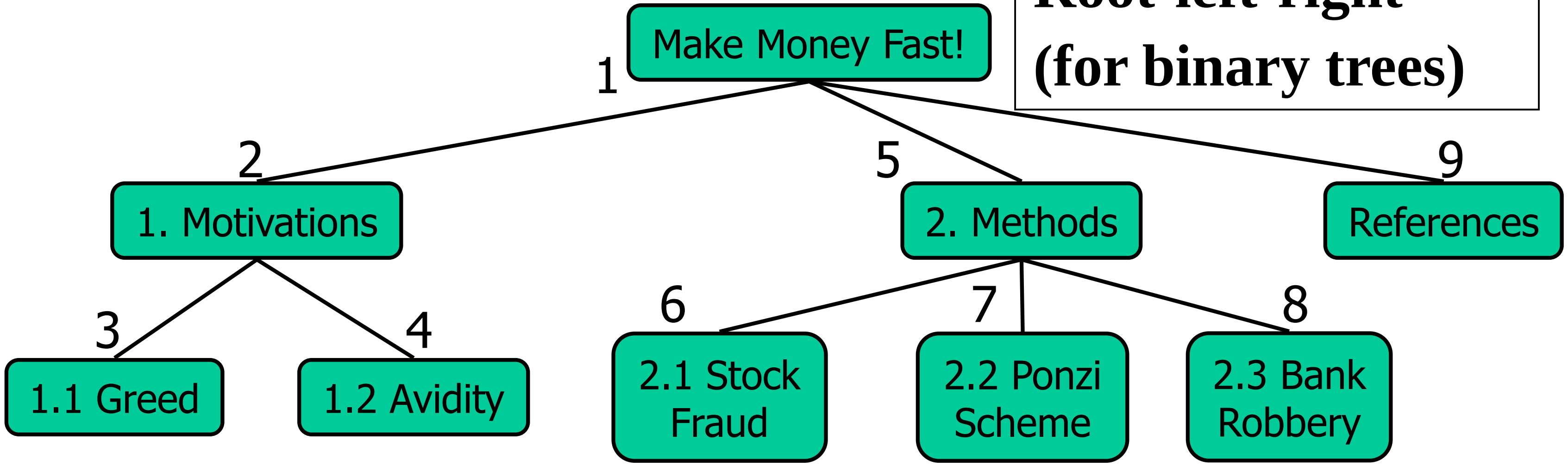


## Preorder Traversal

- A traversal visits the nodes of a tree in a systematic manner
- In a preorder traversal, a node is visited before its descendants
- Application: print a structured document

```
Algorithm preOrder(v)
    visit(v)
    for each child w of v
        preorder (w)
```

**Root-left-right**
**(for binary trees)**

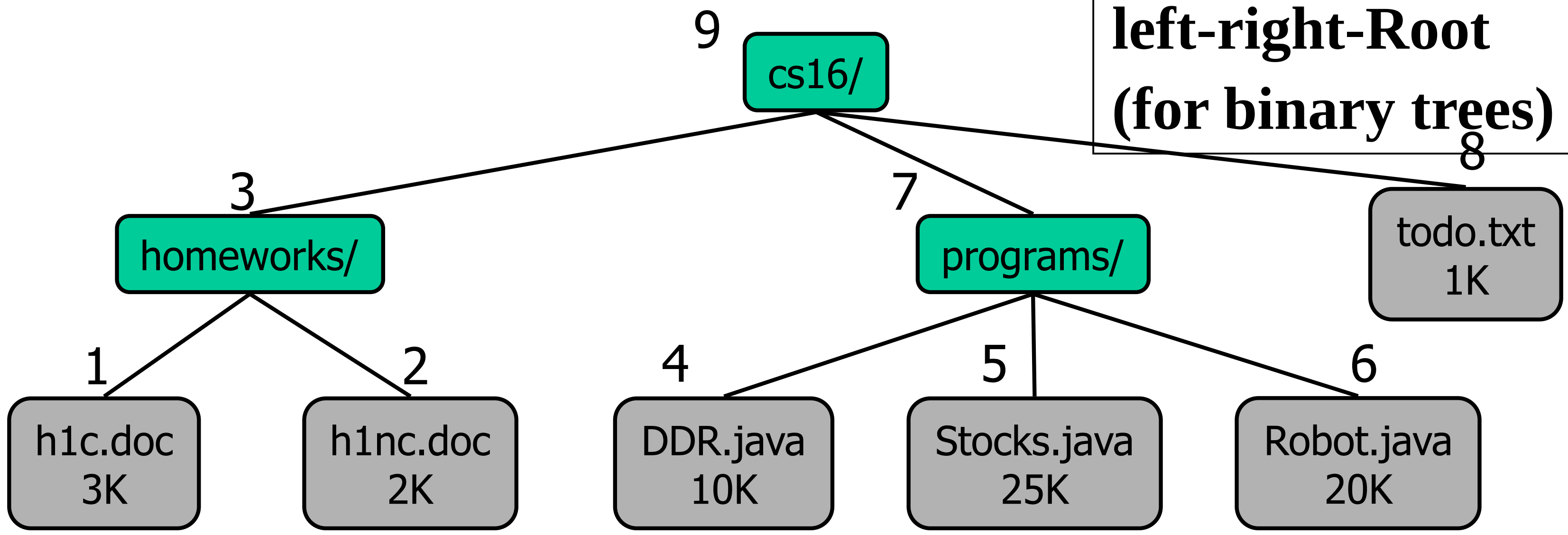


# Postorder Traversal

- In a postorder traversal, a node is visited after its descendants
- Application: compute space used by files in a directory and its subdirectories

```
Algorithm postOrder(v)
    for each child w of v
        postOrder (w)
    visit(v)
```

**left-right-Root**
**(for binary trees)**



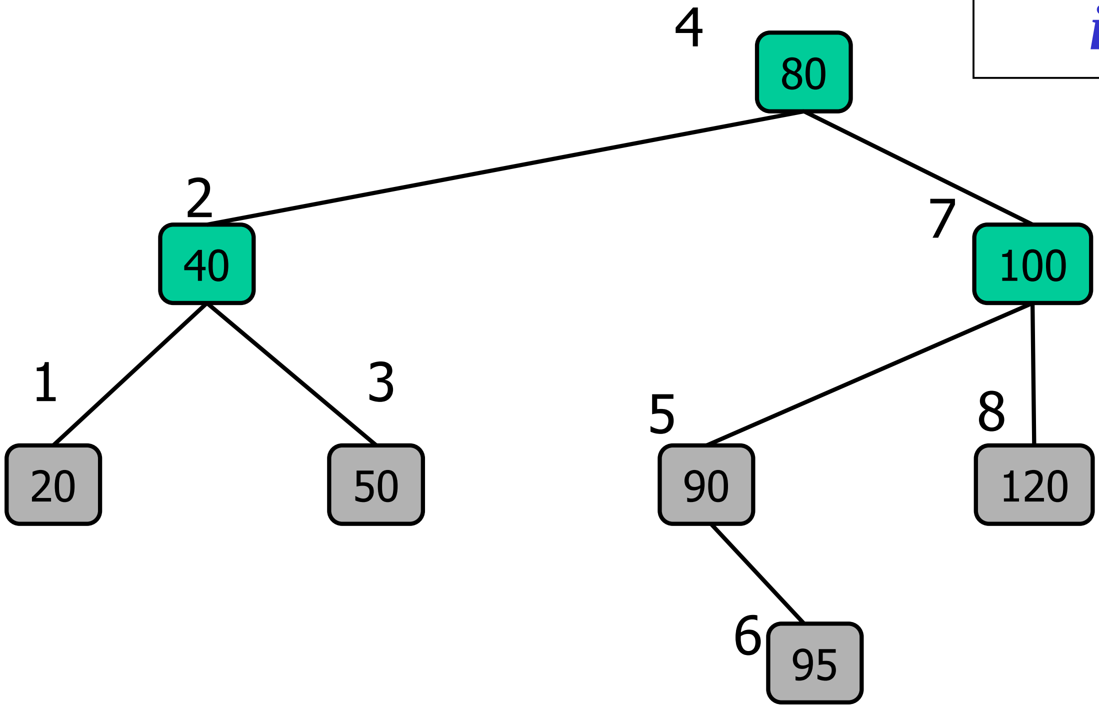
# Inorder Traversal

- In an inorder traversal, left node is visited followed by the Root followed by the right node (binary)
- Application: fast sorting on binary search trees

```
Algorithm inOrder(v)
        if(v == NULL) return
        inOrder (v.left)
        visit(v)
        inOrder (v.right)
```

**left-Root-right**

**(applicable to only binary trees only)**



4
80
2
40
7
100
1
20
3
50
5
90
8
120
6
95