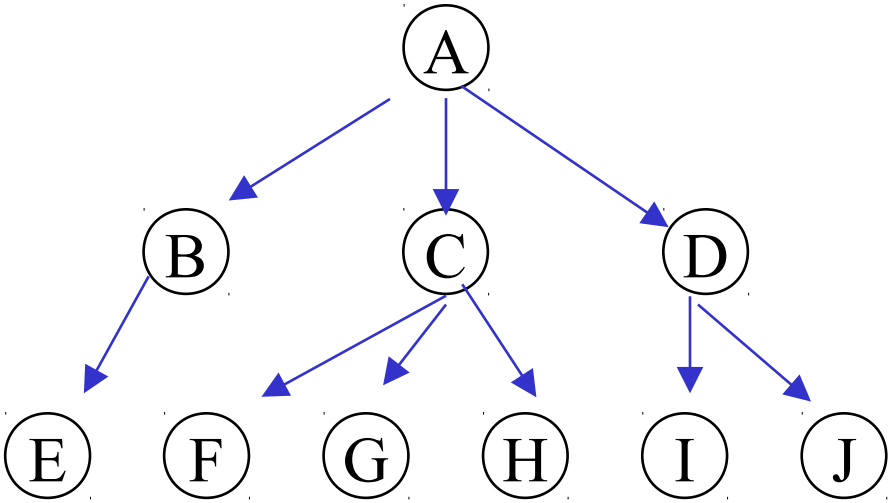
Computer Science 112
Data Structures

Lecture 15:
Huffman coding

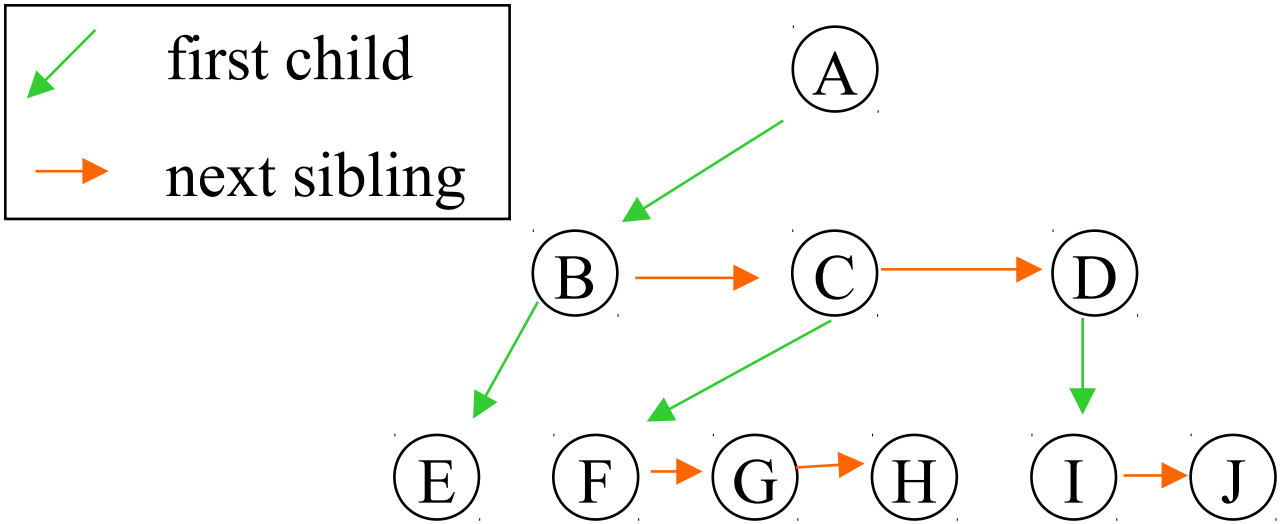
General Trees

- **How do you represent a node when it can have any number of children?**



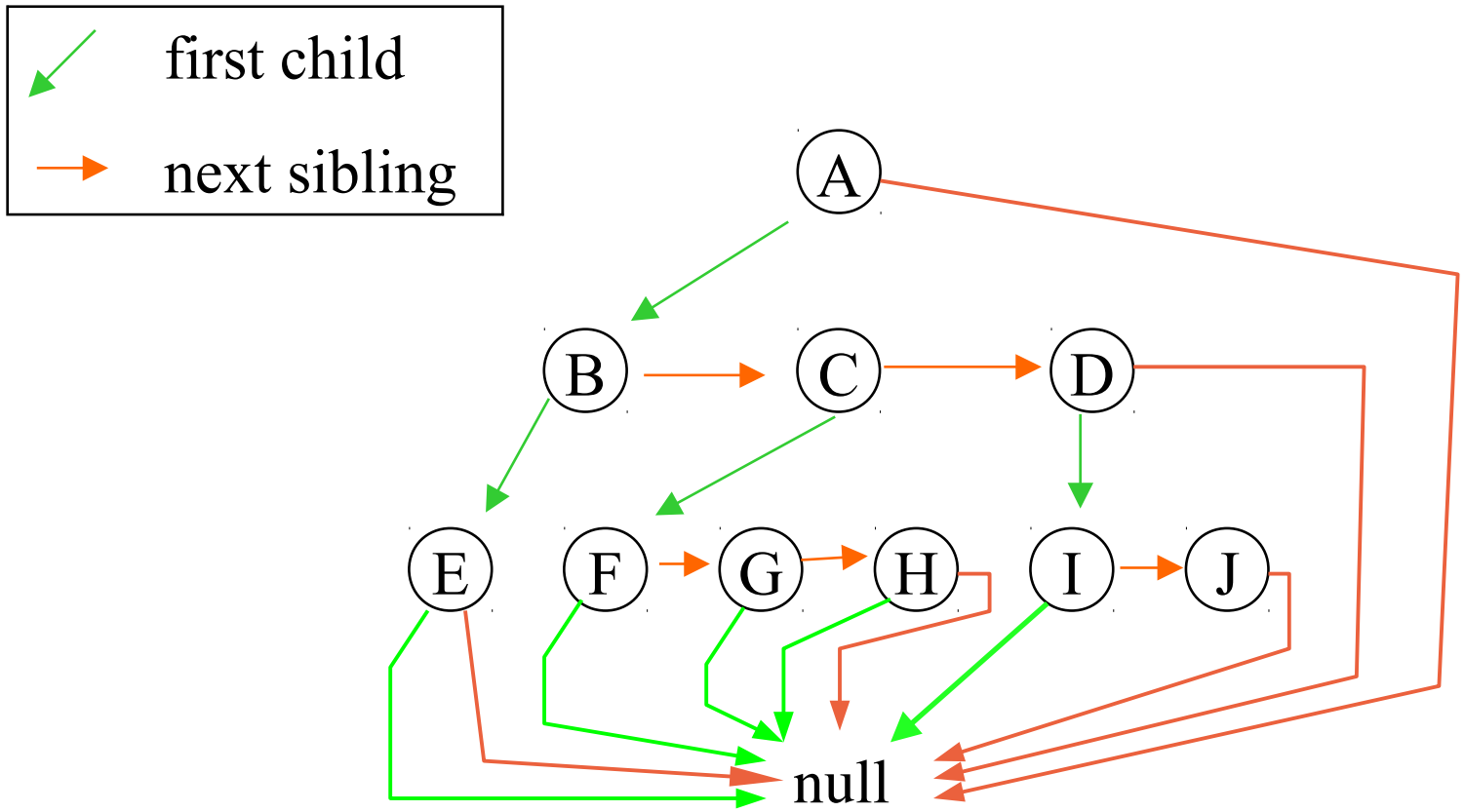
General Trees

- **Solution: linked list of children**



General Trees

- **Solution: linked list of children**



DOM tree in assignment

<html>

<body>

** **

this is a line

** **

** **

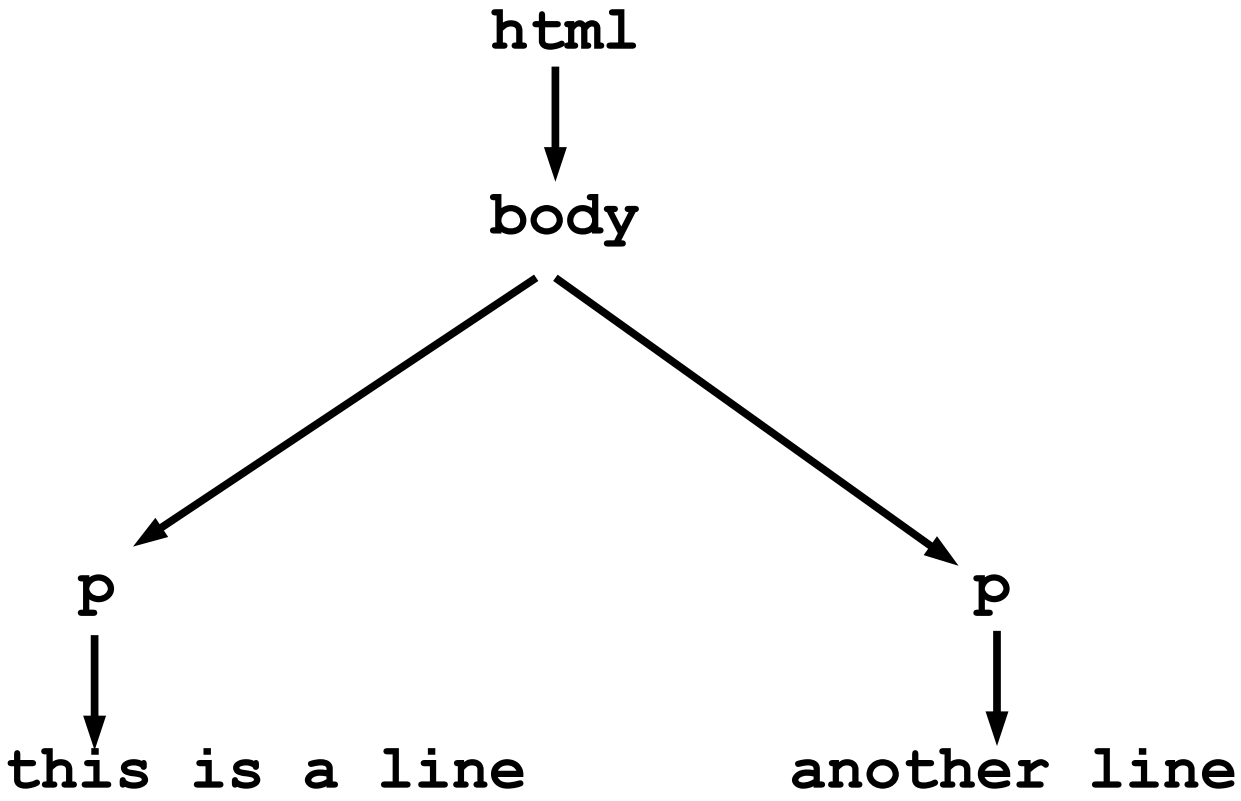
another line

** **

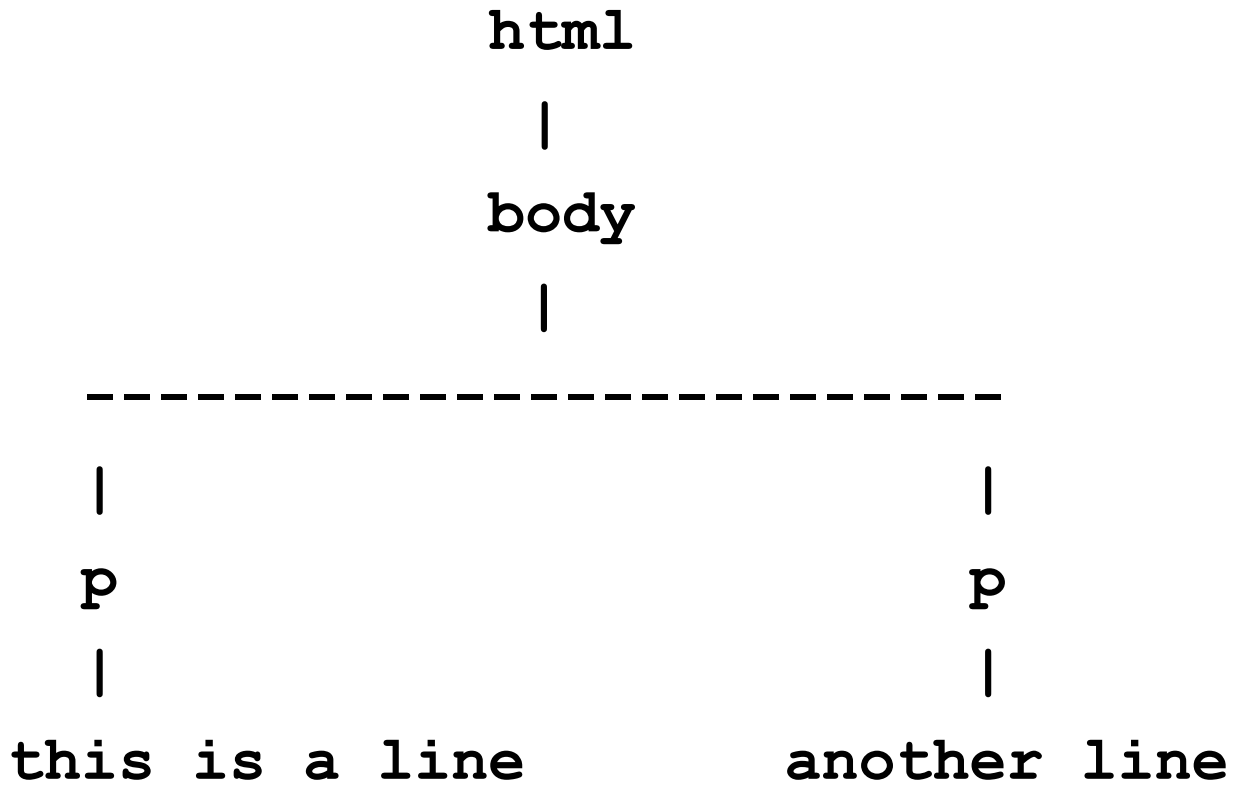
</body>

</html>

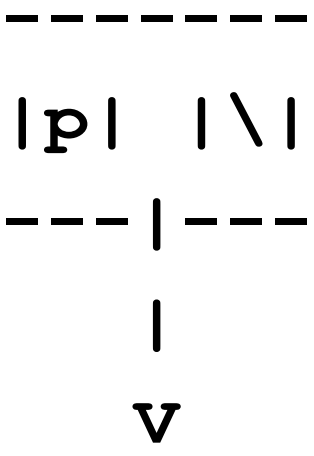
The tree as we think about it

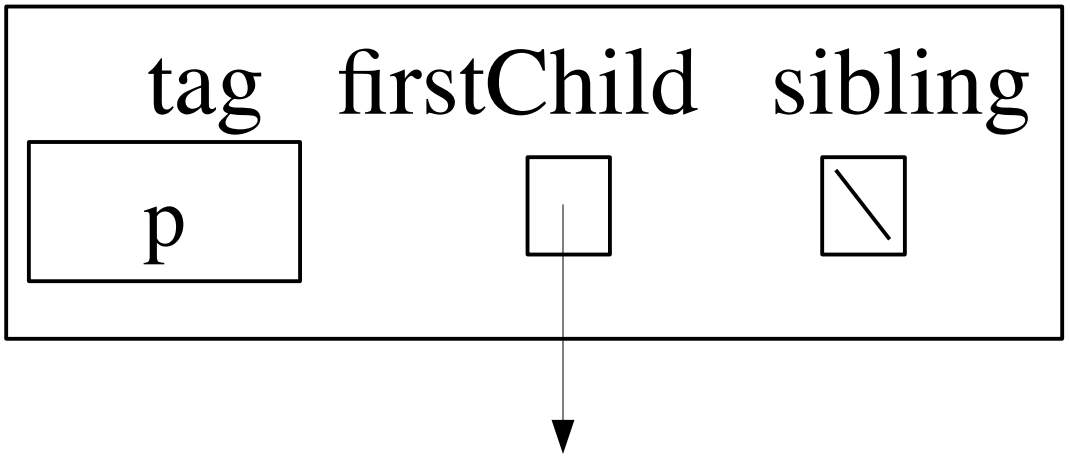


The tree as we drew it

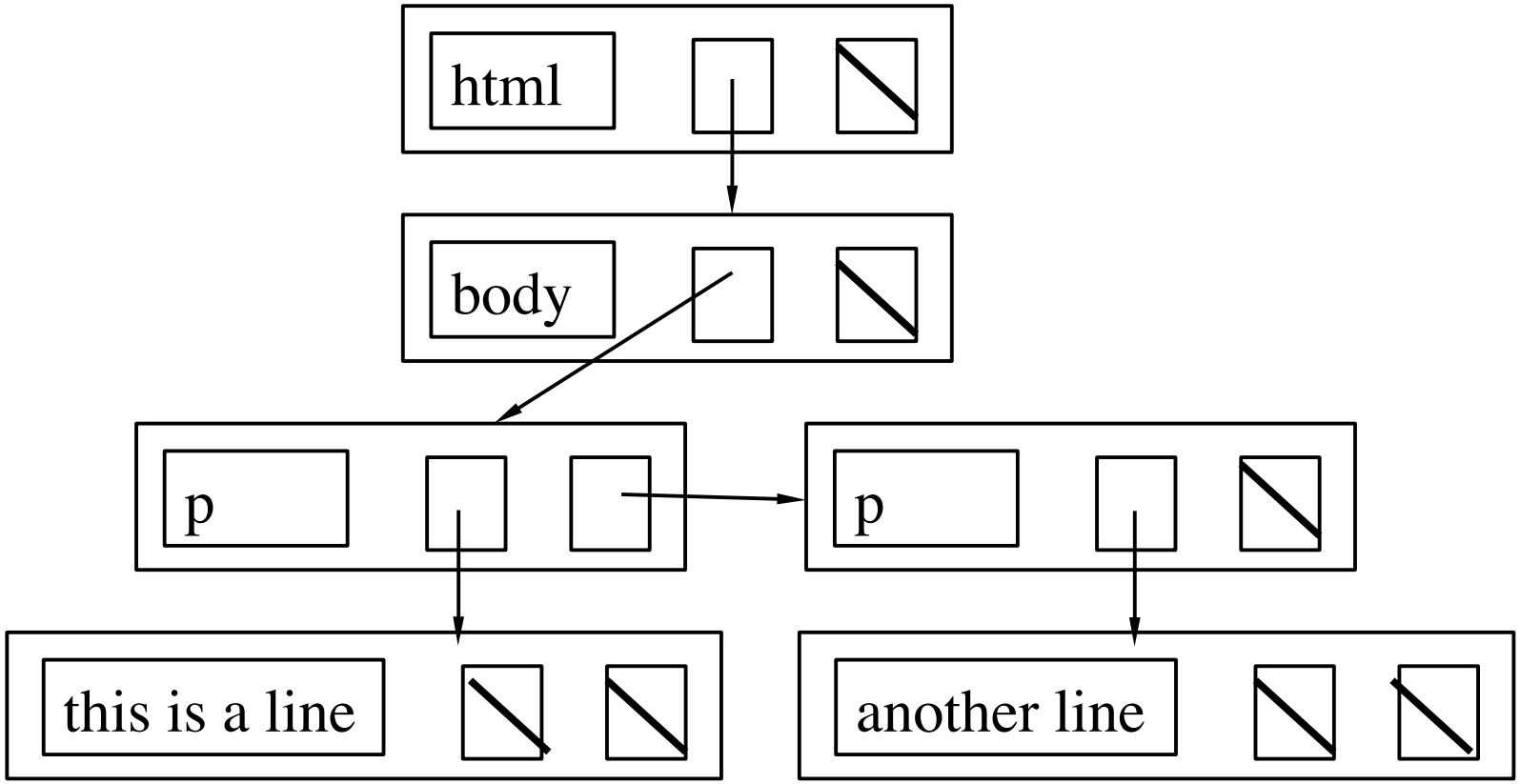


TagNodes





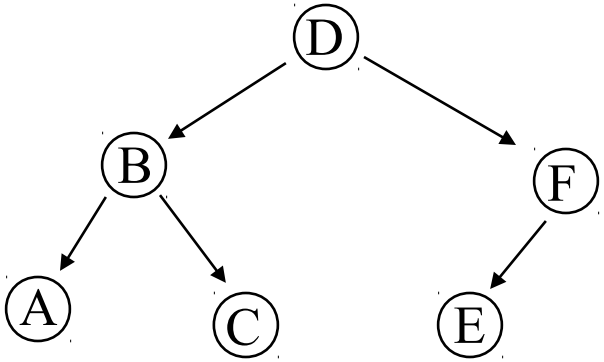
Tree as TagNodes

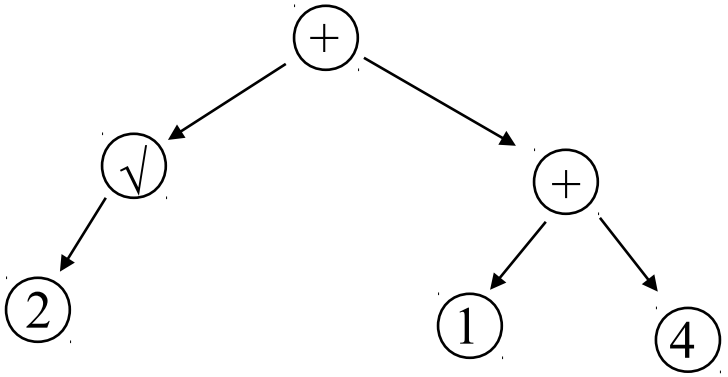


Tree Traversals

- **"Visit" each node in the tree**
  - **"visit" = do something, e.g. print**
- **In some systematic order**
  - **eg "inorder":**
    visit all nodes in left subtree
    then visit node
    then visit all nodes in right subtree

Traversals





InOrder A B C D E F

2 √ + 1 + 4

See TreeNode.java

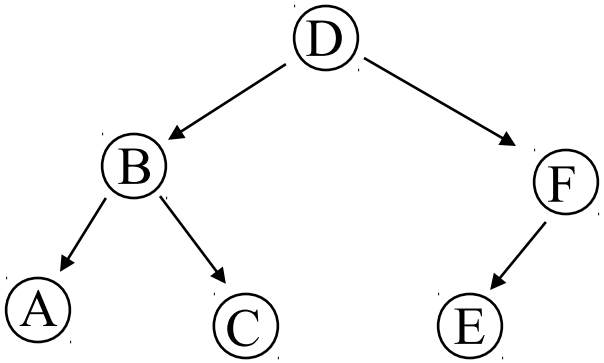
Recursive Traversals

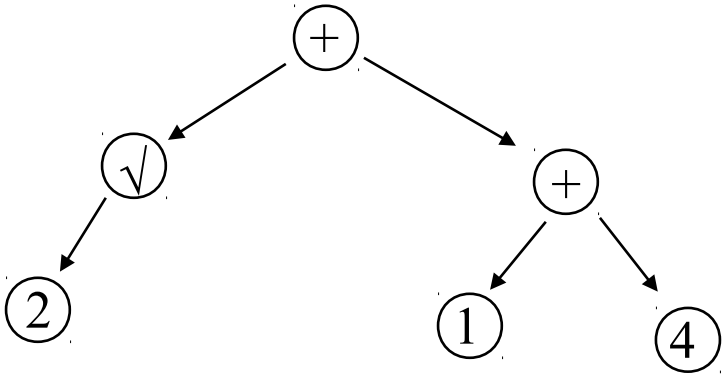
```
inOrderPrint(tree):
    if (tree == null)
       return
    inOrderPrint(tree.lst)
    print(tree.node)
    inOrderPrint(tree.rst)
```

```
preOrderPrint(tree):
    if (tree == null)
       return
    print(tree.node)
    preOrderPrint(tree.left)
    preOrderPrint(tree.right)
```

```
postOrderPrint(tree):
    if (tree == null)
       return
    postOrderPrint(tree.lst)
    postOrderPrint(tree.rst)
    print(tree.node)
```

Traversals





InOrder	A B C D E F	2 √ + 1 + 4
PreOrder	D B A C F E	+ √ 2 + 1 4
PostOrder	A C B E F D	2 √ 1 4 + +

Depth vs Breadth first

- **Depth first**
  - **when you start a node, do its whole subtree before anything else**
  - **recursion or stack**
- **Breadth first**
  - **do all at same level before anything else**
  - **queue**

New: Huffman Encoding

Data Compression:

- **In most data some symbols appear more often than others**
  - **Eg English text ‘e’ appears more often than ‘q’**
- **Can we make use of this to reduce the average number of bits in a message?**

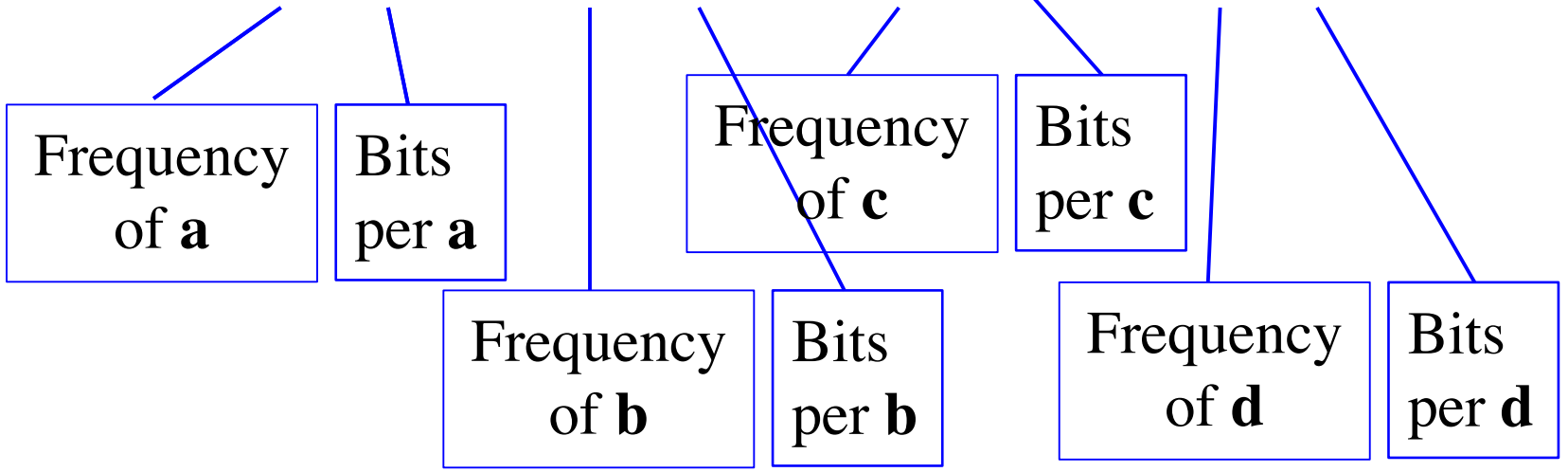
Fixed-length Code

- EG 4 symbols: a, b, c, d with frequencies:
  - a: 50%, b: 30%, c: 10%, d:10%
- Fixed length code: 2 bits / character
  - e.g: 00 = a, 01 = b, 10 = c, 11 = d
  - aabcbaa = 00000110010000
  - 14 bits / 7 characters = 2 bits/character
- Decode: 00010010 = abac

Fixed-length Code

- **EG 4 symbols: a, b, c, d with frequencies:**
  - **a: 50%, b: 30%, c: 10%, d:10%**
- **Fixed length code: 2 bits / character**
- **Average bits per character:**

0.5*2 + 0.3*2 + 0.1*2 + 0.1*2 = 2



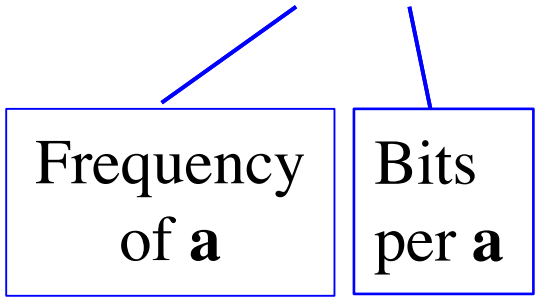
Variable Length Code

- **EG 4 symbols: a, b, c, d with different frequencies:**
  - **a: 50%, b: 30%, c: 10%, d:10%**
- **Variable length code: 1, 2, or 3 bits / character**
  - **e.g: 0 = a, 10 = b, 110 = c, 111 = d**
  - **aabcbaa = 00101101000**
  - **11 bits / 7 characters = 1.6 bits/character**
- **Decode: 0100110 = 'abac'**

Variable Length Code

- **EG 4 symbols: a, b, c, d with different frequencies:**
  - **a: 50%, b: 30%, c: 10%, d:10%**
- **Variable length code: 1, 2, or 3 bits / character**
  - **e.g: 0 = a, 10 = b, 110 = c, 111 = d**
- **Average bits per character:**

0.5*1 + 0.3*2 + 0.1*3 + 0.1*3 = 1.7



• • •

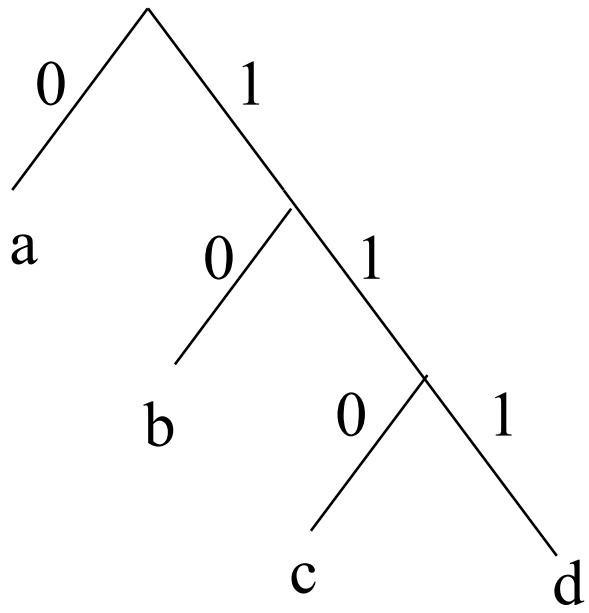
Variable Length Code

- **Suppose code was**

  1 = a, 10 = b, 110 = c, 111 = d

- **Decode 1111 as 'ad', as 'da', or as 'aaaa'?**
- **No character's code can be prefix of another**

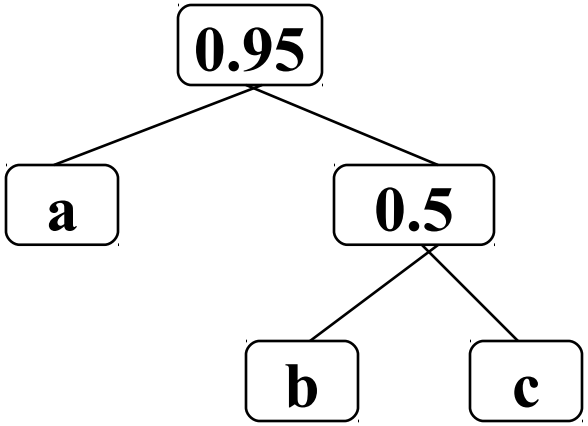
Huffman Code as a Tree



Symbols only at leaves

Algorithm

- 2 queues: S, T
- Contents of each queue: Tree
  - A leaf node stores (an index of) a symbol
  - A non-leaf mode stores total frequency of all symbols at leaves under this node
- E.g., for frequencies a: 0.45, b: 0.3, c: 0.2:



Algorithm

- **2 queues:**
  - **S initially holds 1-node trees for all symbols, least likely first**
  - **T empty**

while not (S empty and T length == 1)

find two least-weight trees in S, T and dequeue them

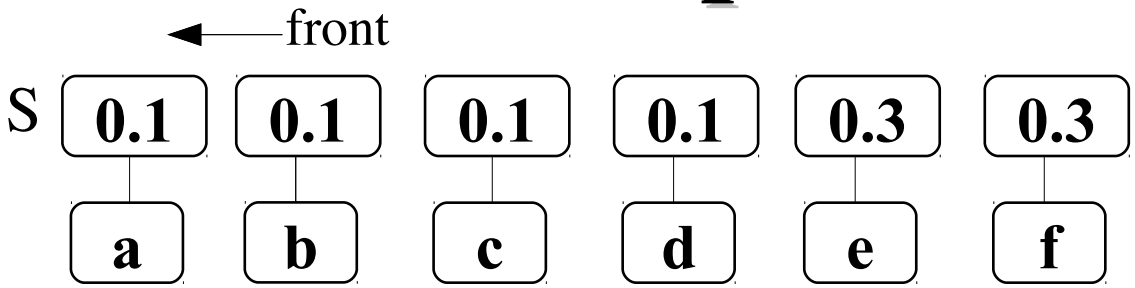
make a tree with these two as subtrees

enqueue this tree on T

Example

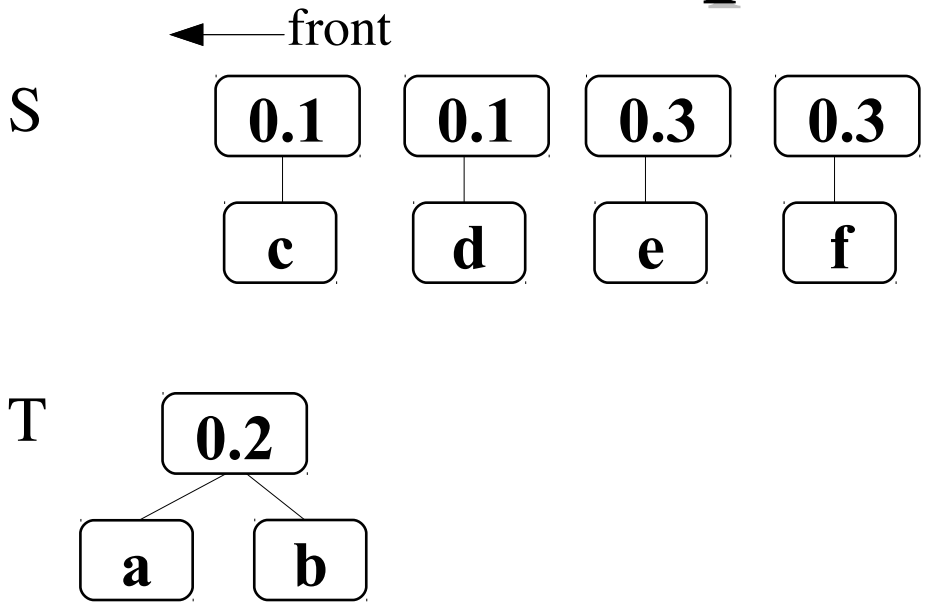
A	**.1**
B	**.1**
C	**.1**
D	**.1**
E	**.3**
F	**.3**

Example

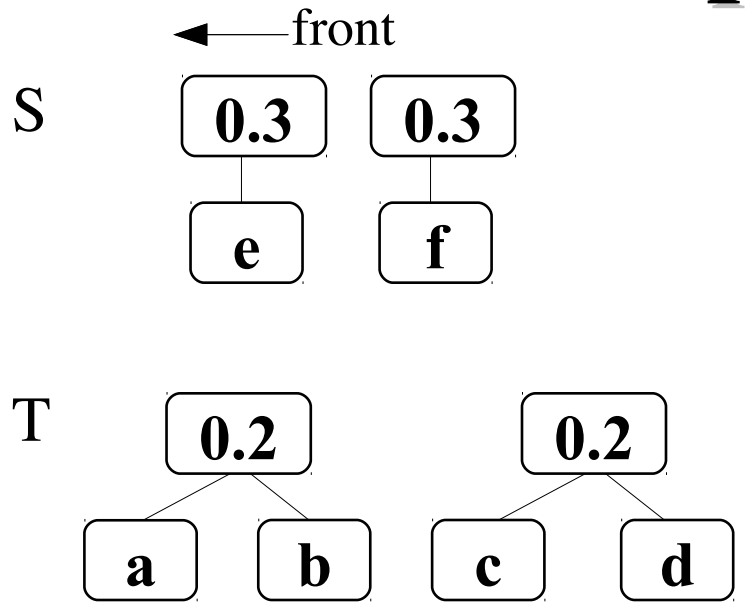


T

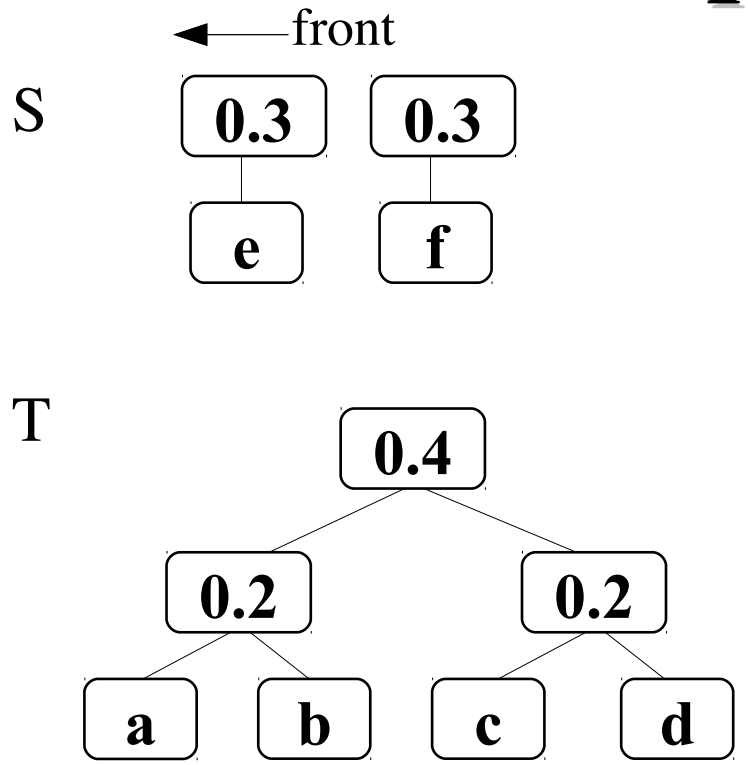
Example



Example



Example

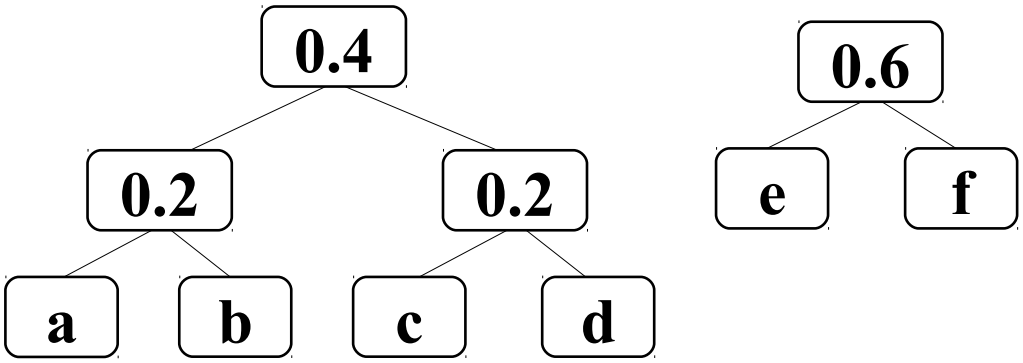


Example

front

S

T

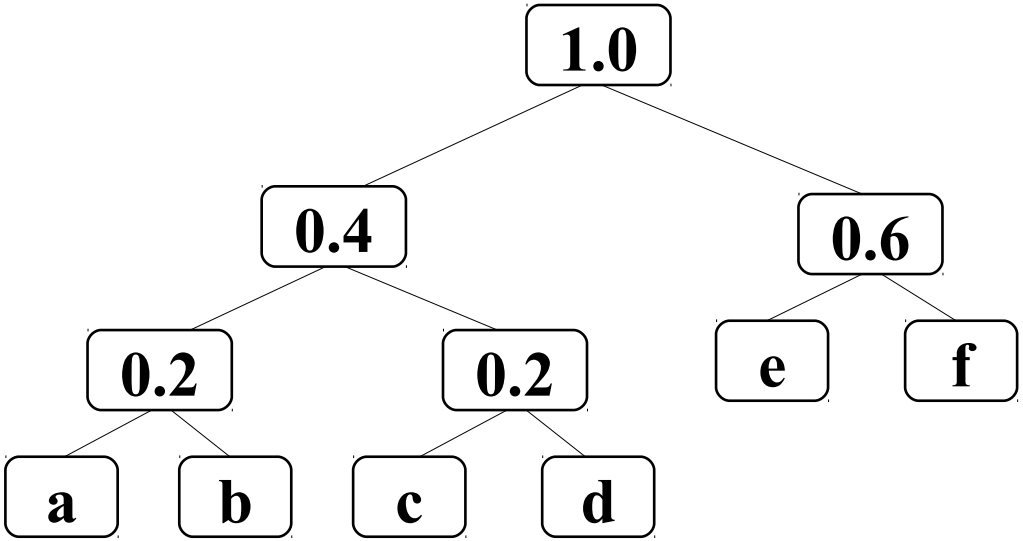


Example

front

S

T



Book code

- **See Huffman.java and HuffmanDriver.java in dsoi.progs.src.zip in apps/tree/**