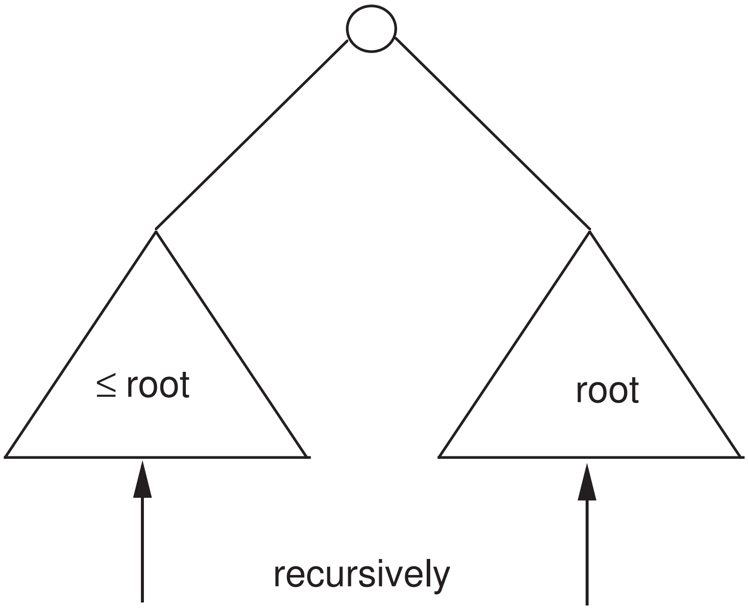
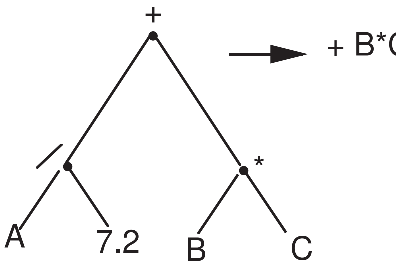
Labelled Trees

Binary Search tree: labelled tree such that for each node the labels in its left subtree are smaller than that of the node and labels in its right subtree are greater.



Parse tree of expression $A/7.2+B*C$.



Application of Binary Tree, Data Compression

Given an essemble of n **messages** or **source symbols**, we need $\lceil \log n \rceil$ bits to code each message (code length all equal)

Example: To encode 26 English letters,

- It needs $\lceil \log 26 \rceil$ bits to code each letter.
- To send a message "SOS": needs $3 \times \lceil \log 26 \rceil$ bits
- But if we code "S" with 01, "O" with 10, "SOS" uses 6 bits only
- We should *code highly likely letter short!*

Assume that

P_i is the probability of transmitting the ith source symbol, and $\sum p_i = 1$; and

t_i is the codeword length for ith symbol.

Then the **expected** (or average) coding length for a source symbol is $\sum_i p_i t_i$.

Example. Morse Code:

A: $\bullet -$

B: $- \bullet \bullet\bullet$

C: $- \bullet - \bullet$

D: $- \bullet \bullet$

E: $\bullet$

$\vdots$

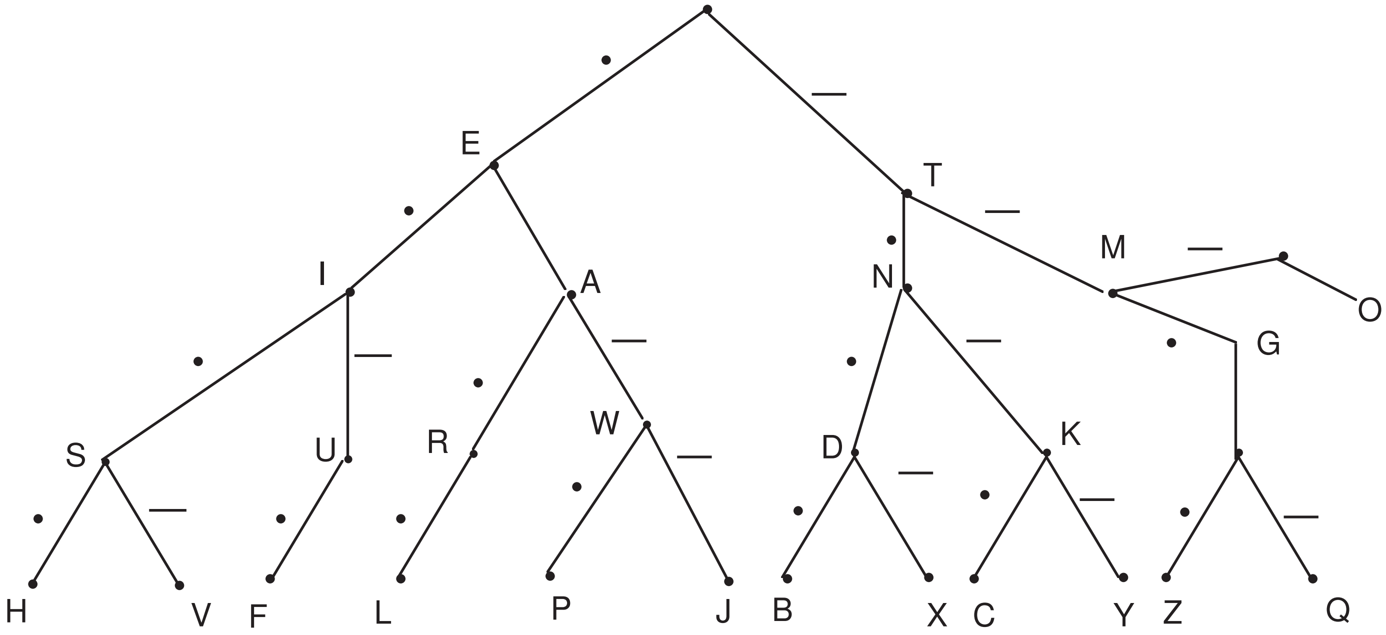
T: $-$

$\vdots$

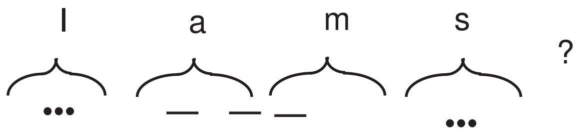
Z: $- - \bullet\bullet$

Formally,

- Source alphabet for Morse Code is: $\{a, b, \ldots z\}$.
- Codeword alphabet is: $\bullet, -$
- Morse code is not self-delimiting. (A code is self-delimiting is no code of a symbol is a prefix of another.)



Example:



Example: Code for SOS is

• • •   — — —   • • •

But if it read as:

• •   • —   — —   • • •

can also be deciphered as “I am S”.

We want a real binary prefix-free encoding of source symbols. That is, no codeword is a prefix of another codeword.

How well can we hope to do this?

Given an essemble of n objects (source symbols), let p_i be the probability the ith source symbol appears.

Note, we always have $\sum_i p_i = 1$

Shannon Entropy: $-\sum p_i \log p_i$. Information theory tells us we cannot do better than this—can't have a shorter encoding on average. But we can try to approach to this lower bound. The idea is to encode the letters which appears more often with shorter codes.

Example:

	Probabilities	A possible prefix-free coding
A	.30	00
B	.25	10
C	.08	0110
D	.20	11
E	.05	00111
F	.12	010

```
                  1
                 / \
                /   \
            0.55    0.45
            /  \     /  \
           /    \   /    \
       A(.3)  .25  B(.25) D(.2)
              /  \
             /    \
         F(.12)  .13
                 /  \
                /    \
            C(.08) E(.05)
```

Expected Cost is

$$\begin{aligned}\sum p_i t_i &= .3*2+.12*3+.08*4+.05*4+.25*2+.2*2\\ &= 2.38\text{bits.}\end{aligned}$$

Huffman's Algorithm to Design Codes

Initially, we have a graph of k nodes ($k = |\text{source alphabet}|$) labelled with source symbols, and corresponding probabilities.

We will convert this forest into an oriented labelled tree.

While there are more than 1 tree do

> Add new node, with 2 roots of least weight as its children. Label one edge 0, the other edge 1. Give root the weight = sum of two child weights

- Code word: path from root to sourceword
- To encode: build a table
- To decode: use tree

Huffman code is the best possible under the following reasonable (but not perfect) model: Given n message words $\{M_i\}$ and their independent probabilities $\{p_i\}$, produce codewords $\{c_i\}$ to minimize

$$\sum_{i=1}^{n} p_i |c_i|$$

Theorem: Under above model, Huffman's algorithm gives the shortest average code.

Proof Sketch: By contradiction.

Assume there is an n and p_i, for which some other optimal code is better than Huffman's. Fix minimal such n. Consider this new coding scheme.

- Consider 2 messages (letters) with lowest probabilities;
- Question: *Do they have longest codewords?*

If "yes":

- Permute codeword-message pairing so they differ only in last digit. *No change in cost.*

- Generate code for $n-1$ messages by replacing 2 lowest probability messages by 1 with probability $p_i + p_j$ and the code-word is their common prefix.

Claim: If the old version (Huffman) is not optimal, neither is this (for $n-1$ messages).

Therefore Not smallest n, contradiction.

If "No": we permute assignments so low probability codewords do have longest codewords— this reduces $\sum p_i|c_i|$. Therefore the given code is not optimal, contradiction. $\square$

But: we <u>assumed</u>

1. Probabilities are independent.

```
For example, in English:  qu
                          th
                          ch
```

If "q" appears, "u" becomes more likely to be the next. Same for "th" and "ch".

2. We assign code to "individual" source word. This is not necessarily always good.

Example. Let Prob(a) $=$ 0.9, Prob(b) $=$ 0.1

```
Huffman      a --> 0
             b --> 1
             Cost/letter = 1

New Coding   aa --> 0
             a  --> 10
             b  --> 11

Cost/letter = 1/2 x 0.81 + 2 x 0.19 = 0.785
```