Coding Problem

Coding: assignment of bit strings to alphabet characters
Codewords: bit strings assigned for characters of alphabet

Two types of codes:
- *fixed-length encoding* (e.g., ASCII)
- *variable-length encoding* (e,g., Morse code)

Prefix-free codes: no codeword is a prefix of another codeword

Problem: If frequencies of the character occurrences are known, what is the best binary prefix-free code?

Huffman codes

- Any binary tree with edges labeled with 0’s and 1’s yields a prefix-free code of characters assigned to its leaves
- Optimal binary tree minimizing the expected (weighted average) length of a codeword can be constructed using Huffman’s algorithm

Huffman’s algorithm

Initialize *n* one-node trees with alphabet characters and the tree weights with their frequencies.

Repeat the following step *n*-1 times: join two binary trees with smallest weights into one (as left and right subtrees) and make its weight equal the sum of the weights of the two trees.

Mark edges leading to left and right subtrees with 0’s and 1’s, respectively.

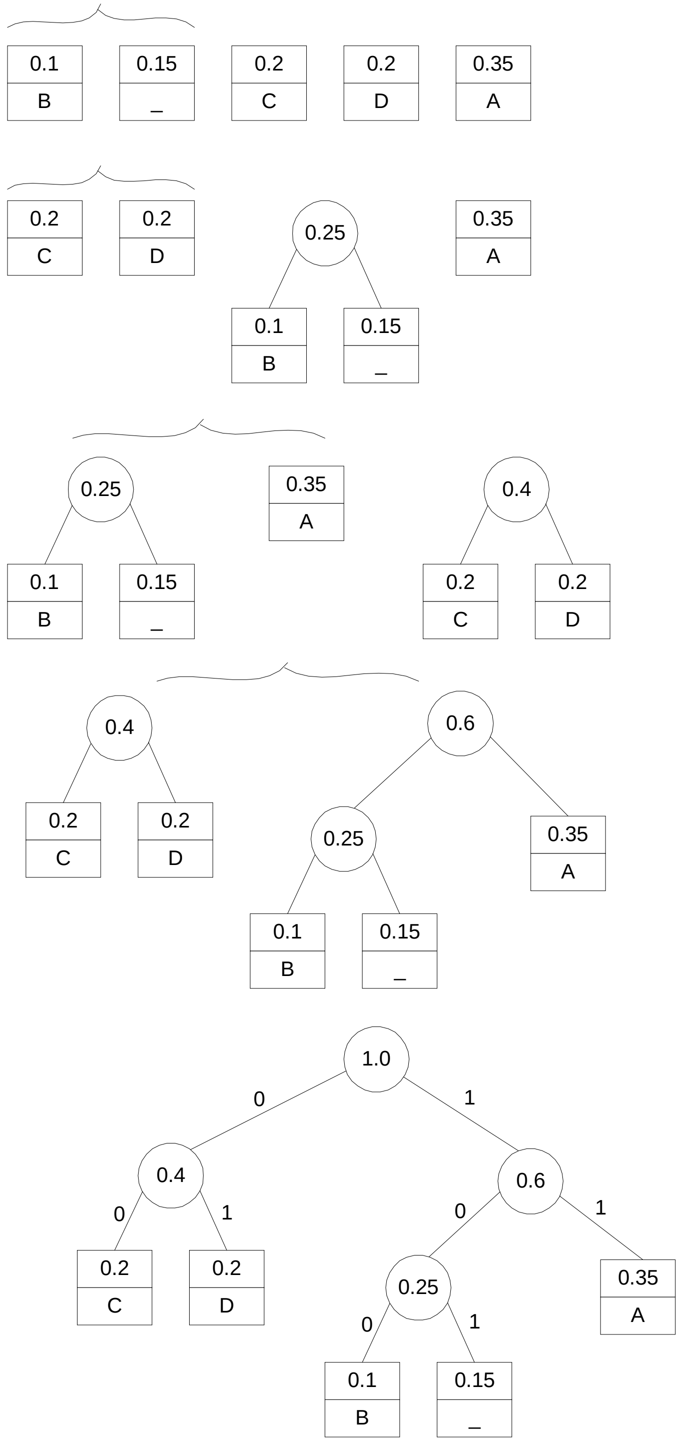
Example

character	A	B	C	D	_
frequency	0.35	0.1	0.2	0.2	0.15
codeword	11	100	00	01	101

average bits per character: 2.25

for fixed-length encoding: 3

compression ratio: (3-2.25)/3*100% = 25%



```
**ALGORITHM** $Huffman(C)$
// Assume $C$ is a set of $n$ characters, $c \in C$ is an object,
// $c.freq$ is its frequency, and $Q$ is a min-priority queue.
    $n = |C|$
    $Q = C$
    **for** $i = 1$ to $n - 1$
        Allocate a new node $z$
        $x = \text{Extract-min}(Q)$
        $z.left = x$
        $y = \text{Extract-min}(Q)$
        $z.right = y$
        $z.freq = x.freq + y.freq$
        Insert $(Q, z)$
    **return** $\text{Extract-min}(Q)$
```