Algorithms

Dana Shapira
Lesson #3: Greedy Algorithms
Part II

Huffman codes

Example

	a	b	c	d	e	f
Frequency	45	13	12	16	9	5
Fixed-length Codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

Which one is cheaper?

Prefix-free Codes

- Codes in which no codeword is also a prefix of some other codeword.
  - **Example:**

	a	b	c	d	e	f
Variable-length codeword	0	101	100	111	1101	1100

110001001101

- easy to encode and decode using prefix codes.
- **No Ambiguity !!**
- It is possible to show that the optimal data compression achievable by a character code can always be achieved with a prefix code.

Prefix-free Codes

	a	b	c	d	e	f
Variable-length codeword	0	101	100	111	~~1101~~	1100
Variable-length codeword	0	101	100	111	110	1100

FACE = 11000100110

- 1100 = 110 0 = “f"
- or
- 1100 = 110 + 0 = “ea"

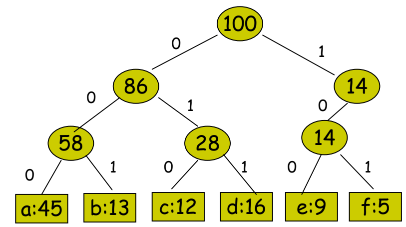
Prefix-free Codes

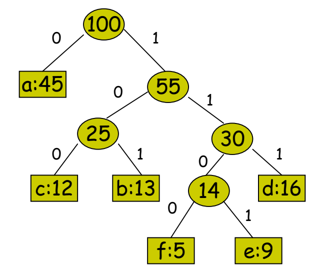
- Represented as a binary tree
  - each edge represents either 0 or 1
    - 0 means "go to the left child"
    - 1 means "go to the right child."
  - each leaf corresponds to the sequence of 0s and 1s traversed from the root to reach it, i.e. a particular code.
- Since no prefix is shared, all legal codes are at the leaves, and decoding a string means following edges, according to the sequence of 0s and 1s in the string, until a leaf is reached.

Tree Representation

	a	b	c	d	e	f
Frequency	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101

	a	b	c	d	e	f
Frequency	45	13	12	16	9	5
Variable-length codeword	0	101	100	111	1101	1100





Huffman Algorithm

- Greedy
  - The <u>two smallest nodes</u> are chosen at each step, and this local decision results in a globally optimal encoding tree.
- bottom-up manner
  - Starts with a set of $|\Sigma|$ leaves and performs a sequence of $|\Sigma|$ - 1 "merging" operations to create the final tree.

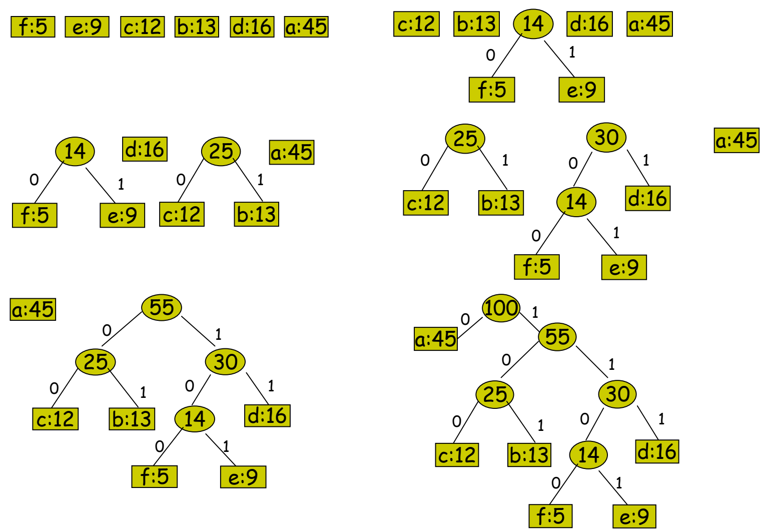
Professor David A. Huffman
(August 9, 1925 - October 7, 1999)

Huffman Algorithm

```
HUFFMAN(Σ)
n ←|Σ|
Q ← Σ
for i ←1 to n - 1
    do  ALLOCATE-NODE(z)
     left[z] ← x ← EXTRACT-MIN(Q)
     right[z] ← y ← EXTRACT-MIN(Q)
     w[z] ←w[x] + w[y]
     INSERT(Q, z)
return EXTRACT-MIN(Q)
```

What is the running time?

Huffman Algorithm



Optimality of Huffman Codes

Theorem:

Given weights $w_1,\dots,w_n$. Huffman Algorithms assigns code lengths $l_1,\dots,l_n$ such that $\sum_{i=1}^{n} w_i l_i$ is minimal.

Lemma 1:

An optimal code for a file is always represented by a ***full* binary tree**, in which every non-leaf node has two children.

Lemma 2:

In an optimal tree the two lowest weights w_{n-1} and w_n are in the lower level.

Optimality of Huffman Codes

Lemma 3:

In an optimal tree the two lowest weights w_{n-1} and w_n can be assumed to be brothers.