

Huffman Codes

Compression

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
<i>Freq(ink)</i>	45	13	12	16	9	5
<i>3-bit fixed</i>	000	001	010	011	100	101
<i>variable</i>	0	101	100	111	1101	1110

100,000 characters in file.

Size using fixed 3-bit = 300,000 (no fixed spaces needed)

Size using variable = 224, 000

This is optimal for this file? (How ?)

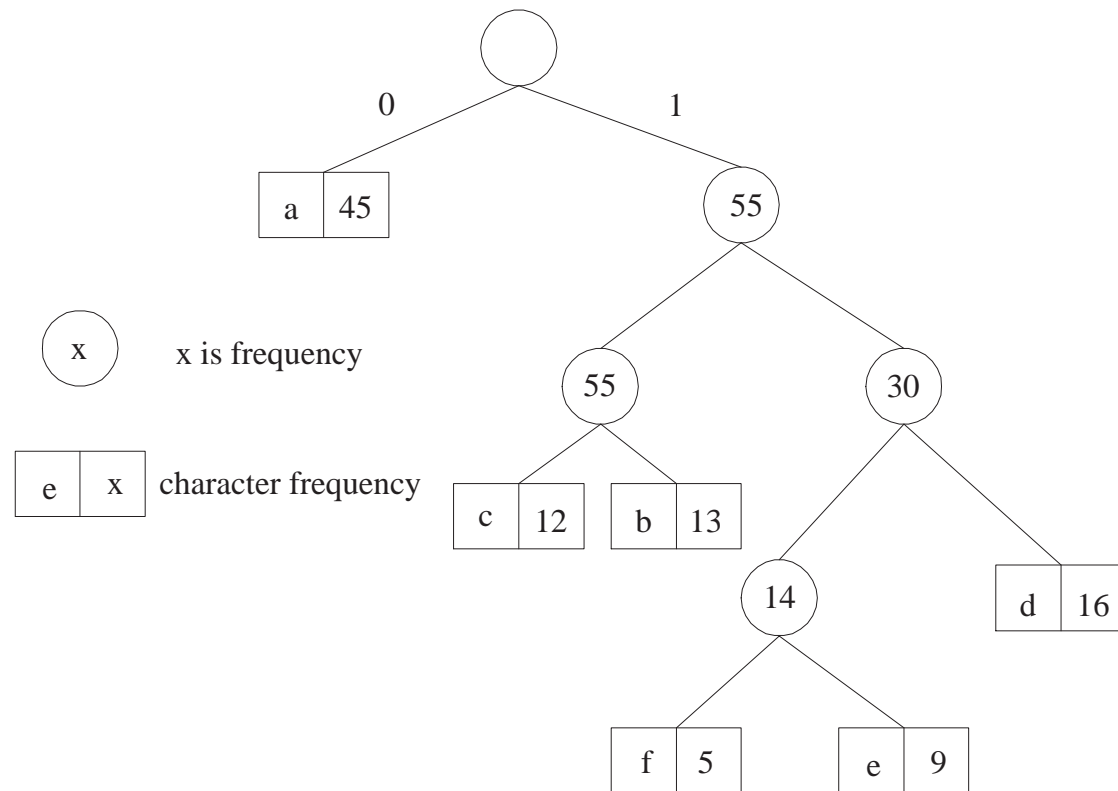
Prefix code

1. No code word is a prefix of another code word. An optimal compression can always be achieved using prefix codes.
2. Decoding is simple. The code word that starts the file is unambiguous. Recursive reduction:

0 0 101 1101 a a b e

As soon as a character is matched, it can be striped off.
No need for space.

3. Codes can be expressed as a binary tree.



Side notes: fixed codes should be full binary trees. The example is not.

Huffman code is a greedy algorithm that builds a tree.

It begins with $|C|$ leaves and does $|C| - 1$ merges to create the tree.

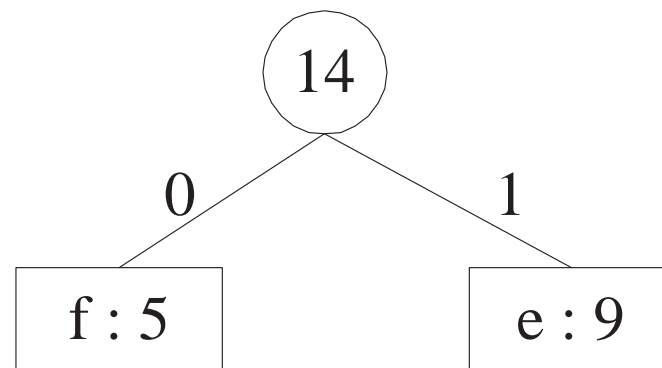
Assume C is sorted by frequencies (small to large).

Huffman (C)

1. $n \leftarrow |C|$
2. $Q \leftarrow C$
3. *for* $i \leftarrow 1$ *to* $n - 1$
4. *do* $z \leftarrow \text{Allocate} - \text{Node}()$
5. $x \leftarrow \text{left}[z] \leftarrow \text{Extract} - \text{Min}(Q)$
6. $y \leftarrow \text{right}[z] \leftarrow \text{Extract} - \text{Min}(Q)$
7. $f[z] \leftarrow f[x] + f[y]$
8. $\text{Insert}(Q, z)$
9. *return* $\text{Extract} - \text{Min}(Q)$

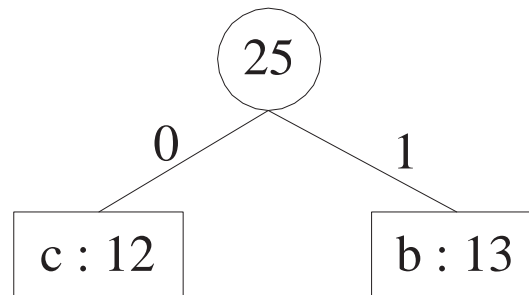
E.G. f:5 e:9 c:12 b:13 d:16 a:45

Allocate

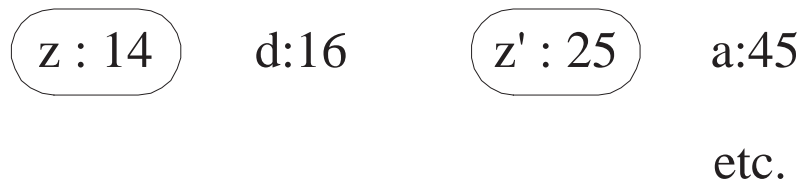


Insert(Q, z) c:12 b:13 z:14 d:16 a:45

Allocate



Insert



Complexity

Q is a binary heap

$$O(n \lg n) \quad n = |C|$$

- (1) Each character is touched once (n-1 actually)
- (2) Each insert is a heap insert $O(\lg n)$

So $O(n \lg n)$ using heap sort and heap.

Note: inserting in regular sorted list is $O(n)$.