Text Compression

- Used for compression of texts.
- The general idea is to use:
  - A variable length code instead of fixed length code like 8-bit ASCII code.
  - Use a shorter code for frequently occuring character in text, and
  - A longer code for the characters that occur occasionally (with low frequency) in the text.

Prefix Property

- Prefix property: no code word is a prefix of any other code word.
- It implies the original symbols can be obtained from the coded text by repeatedly taking out a prefix that is a legal code word.
- For example, if you have a prefix with four ”1”s then it can either be a ”c” or a ”b”.
- The ambiguity is resolved simply by the next bit.
- In case of fixed length code, each symbol is obtained simply by taking 3 bits at a time.

Example

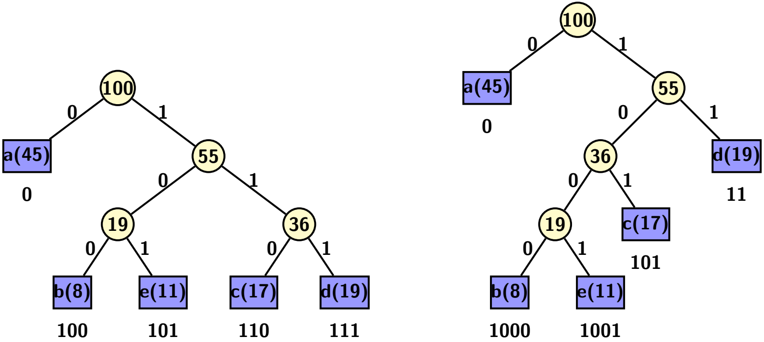
Suppose you have a text of 100,000 characters

	a	b	c	d	e
Occurrence in 10^3	45	8	17	19	11
Fixed length code	000	001	010	011	100
Variable length code 1	0	1000	101	11	1001
Variable length code 2	0	100	110	111	101

- # of characters in fixed length code = 3 million bits.
- # of characters in variable length code 1 = 45*1 + 8*4 + 17*3 + 19*2 + 11*4 = 2.1 million bits
- # of characters in variable length code 2 = 45*1 + 8*3 + 17*3 + 19*3 + 11*3 = 2.1 million bits

Hoffmann Code

It is a lossless compression based on statistical coding.



Theorem on Optimum Code

Main Lemma

Let x and y be two symbols with the smallest frequencies. Then there exists an optimal code tree in which x and y are siblings at the deepest level.

Proof

- Let T be an optimal code tree.
- Assume that x and y are not siblings in T
- Let a and b be the siblings at the deepest level in T.
- Assume $f(x) \leq f(y)$ (otherwise interchange x and y)
- Also assume that $f(a) \leq f(b)$ (otherwise interchange).
- In T, $d(x) \leq d(a)$ and $d(y) \leq d(b)$

Change T to T'

Proof

- Now switch positions of a and x in T.
- Denote the resulting tree by T'.
- Changes in path lengths contributed by a, x are as follows:

Symbol	Old value	New value
a	$f(a)d(a)$	$f(a)d(x)$
x	$f(x)d(x)$	$f(x)d(a)$

Change T to T'

Proof

- Consider Average Bits per Letter (ABL) is the sum over all symbols of its frequency times the number of bits.

$$ABL(code) = \sum_{x \in S} f(x)|code(x)|$$

- Since T is optimal, $ABL(T) \leq ABL(T')$.

Comparing ABL of T and T'

Proof

$$\begin{aligned} ABL(T) &\leq ABL(T') \\ &= ABL(T) - f(x)d(x) - f(a)d(a) + f(x)d(a) + f(a)d(x)) \\ &= ABL(T) - (f(a) - f(x))(d(a) - d(x)) \\ &\leq ABL(T) \end{aligned}$$

Therefore, $ABL(T) = ABL(T')$.

- Now change position of y and b to define a new tree T'' from T'.
- By the same argument you have $ABL(T') = ABL(T'') = ABL(T)$.

Algorithm

```
Huffman(A) {
    n = |A|;
    Q = A; \\ the future leaves
    for i = 1 to n-1 {
        z= newNode();
        left[z] = deleteMIN(Q);
        right[z] = deleteMIN(Q);
        f[z] = f[left[z]] + f[right[z]];
        Insert(Q,z);
    }
    return deleteMIN(Q); // root of the tree
}
```