Huffman Encoding:

Step 1: Pick letters x & y from the alphabet A with the smallest frequencies and make a subtree that has these two characters as leaves. Label the root of this subtree as z.

Step 2: Set frequency $f(z) = f(x) + f(y)$. Remove x, y and z creating new alphabet $A' = A \cup \{z\} - \{x, y\}$. $|A'| = |A| - 1$.

This procedure (step 1 & step 2) is called merge.
Repeat till only one symbol is left
Resulting tree is the Huffman code.

Problem: Given alphabet $A = \{a_1, \cdots, a_n\}$ with frequency distribution $f(a_i)$, find a binary prefix code C for A that minimizes the number of bits.

$$B(C) = \sum_{a=1}^{n} f(a_i)\, L(c(a_i)) \qquad \text{[frequency} \times \text{code length]}$$

$$= \text{total number of bits.}$$

In the Huffman tree, code word is the path taken to reach that node from length. So $L(c(a_i)) = d(a_i)$ in that tree

Lemma: Consider the two letters x and y with smallest frequencies. Then there optimal code tree in which these two letters are sibling leaves in the in the tree in the lowest level.

Proof: Let T be an optimum prefix code tree, and b & c be two siblings at the maximum depth of the tree (must exist because T is full). Assume WLOG $f(b) \le f(c)$, and $f(x) \le f(y)$. Since x & y have the lowest frequencies, $f(x) \le f(b)$ and $f(y) \le f(c)$. And since b & c are at the deepest level $d(b) \ge d(x)$, and $d(c) \ge d(y)$.

Let us switch b & x in T to get T'

$$\text{cost}(T') = \text{cost}(T) + f(b)d(x) + f(x)d(b) - f(x)d(x) - f(b)d(b)$$

$$= \text{cost}(T) - [f(b) - f(x)][d(b) - d(x)] \le \text{cost}(T).$$

$\therefore \text{cost}(T') \le \text{cost}(T)$ and T' is also optimal.

Similarly we can swap c & y in T' to get T'' which is also optimal and satisfies the lemma.

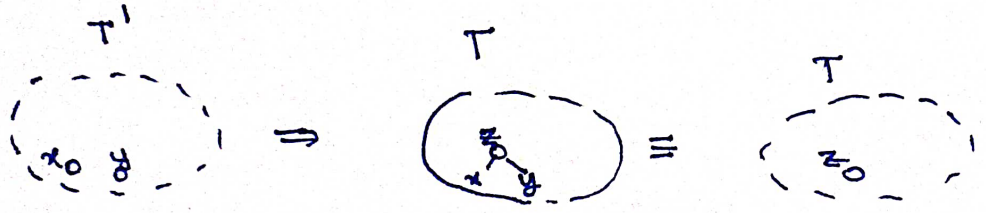
Lemma: The tree for any optimal prefix code must be full.

Proof: If some internal node has only one child, simply replace it with its child to obtain a shorter code length.

Theorem: Huffman's algorithm produces an optimum prefix code tree.

BS: When $n=2$, it clearly holds

IS: Assum the theorm holds on an alphabet with fewer than n letters. Consider an alphabet A' of n letters. Let T' be an optimum tree for A' with the two letters of lowest frequency x & y as sibling leaves. Let T be the coding tree for $A = A' \cup \{z\} - \{x, y\}$ [$n-1$ leaves] obtained by removing x & y and replacing their parent by z.



$d(x) = d(y) = d(z) + 1$. $\quad f(z) = f(x) + f(y)$.

$$B(T) = B(T') - f(x)d(x) - f(y)d(y) + f(z)d(z)$$

$$= B(T') - f(x)d(x) - f(y)d(y) + [f(x)+f(y)][d(x)-1]$$

$$= B(T') - f(x)d(x) - f(y)d(y) + f(x)d(x) + f(y)d(y) - f(x) - f(y)$$

$$= B(T') - [f(x)+f(y)]$$

Let the optimal code tree for A be H_A. Let $H_{A'}$ be obtained by adding x & y as children of z in H_A. As in the calculations for T & T', $B(H_A) = B(H_{A'}) - [f(x)+f(y)]$

Also, $B(H_A) \le B(T)$ [by optimality] so

$B(H_{A'}) = B(H_A) + f(x) + f(y)$

$\le B(T) + f(x) + f(y) = B(T)$

$\therefore B(H_{A'}) \le B(T)$