Informal Proof

Any tree that differs from the one constructed by Huffman's algorithm can be converted into one that is equal to Huffman's tree without increasing its cost. In any optimal encoding scheme, minimum frequency characters exist as deepest leaf nodes. Let's assume x, y, z are the characters with minimum frequencies. Then, in an optimal code tree in which these three characters are siblings at the maximum depth in the tree. Also, let T be any optima prefix code tree, and let a, b, c be siblings at the maximum depth of the tree. Therefore, f (x) <= f (a), f (y) <= f (b) and f (z) <=f(c). If we swap (x, a), (y, b) and (z, c) to produce a new tree T^+, then our bit requirement will be less than the optimal. The cost of the optimal tree does not increase after swapping of nodes. Moreover, it cannot decrease either because then that would be a contradiction, i.e. T was not optimal in the first place. Since T was an optimal tree, T^+ is also an optimal tree. The above claims applies to only one pair of nodes with the lowest frequencies. We need to prove that the entire tree is optimal after swapping.

Let T_n be an optimal prefix-code tree which has x, y and z as siblings at the deepest level. Then, let T_{n-2} be the tree that results by replacing these three nodes with a single node z^+ whose frequency is equal to the sum of the frequencies of x, y, and z. Then, $T_{n} = T_{n-2} + f(z^+)$ which means that the two trees only differ by the fixed term $f(z^+)$.

Base Case: n =1.

If (n>=3), then

$T_{n-2} = T_n - \{x, y, z\} \cup \{z^+\}$

$f(z^+) = f(x) + f(y) + f(z)$

Cost of tree after swapping + $f(z^+)$ = cost of tree before swapping. Since, T_{n-2} is optimal and cost of swapping depends on a fixed amount $f(z^+)$, T_n is optimal as well.