This proof is a little different than the one in the textbook (pp. 433–435) and is drawn from Algorithms from P to NP, Vol. 1, Moret and Shapiro, Benjamin/Cummings Publishing Company, 1991.

**Definition:** Let $C$ be a set of characters and $f(c)$ be the number of occurrences of each character $c \in C$. Consider a particular code for $C$ and denote by $\ell(c)$ the length of the code for each character $c \in C$. The *cost* of the code is

$$\sum_{c \in C} f(c) \cdot \ell(c).$$

I.e., the cost of the code is the total number of bits needed to represent the encoded version of the entire text, when each character $c$ occurs $f(c)$ times and is coded using $\ell(c)$ bits.

**Definition:** A code is *optimal* if the cost of the code is the smallest possible.

Each code can be represented by a binary tree, with each edge labeled 0 or 1. Thus the length of the code for character $c$, denoted $\ell(c)$, is equal to the depth of $c$ in the tree.

**Theorem:** The Huffman code is an optimal code.

**Proof:** Use induction on $n$, the number of characters in $C$.

**Basis:** $n = 2$. Let $C = \{a, b\}$. The algorithm gives code 0 to $a$ and code 1 to $b$ or vice versa, depending on which character occurs less frequently. Clearly a 1-bit code is the smallest possible.

**Induction:** Suppose the Huffman code is optimal for $n - 1$ characters. We must show it is optimal for $n$ characters.

Let $C$ be a set of $n$ characters, with occurrences given by $f$. Let $T$ be the tree constructed by the Huffman algorithm. Consider characters $y$ and $z$ with the second fewest and fewest occurrences. By the way the algorithm works, $y$ and $z$ are leaves and are siblings in $T$.

**Claim:** In some optimal tree (i.e., tree corresponding to an optimal code) for $C$, $y$ and $z$ are also leaves and siblings.

**Proof of Claim:**

1. $z$ must be at the greatest depth in some optimal tree. Otherwise swapping $z$ with another character at greater depth would produce a better code.
2. $z$ must have a sibling. Otherwise merging $z$ with its parent would produce a better code that doesn’t waste a bit on the code for $z$.
3. $z$’s sibling is also a leaf, because $z$ has greatest depth.
4. $y$ is also at greatest depth. Otherwise swapping $y$ and $z$’s sibling would produce a better code.
5. $y$ has a sibling (same argument as for $z$).
6. If $y$ and $z$ are not siblings, then swap $y$ with $z$’s sibling and get another optimal code.

So the claim is true. Let $T_{opt}$ be an optimal tree for $C$ in which $y$ and $z$ are leaves and siblings.

Let $C' = C - \{y, z\} \cup \{x\}$ where $x$ is a new character with $f(x) = f(y) + f(z)$. That is, remove $y$ and $z$, and replace them with a new character whose number of occurrences is the sum of those of $y$ and $z$.

Note that $C'$ has $n - 1$ characters in it, so we can apply the inductive hypothesis to $C'$: i.e., the Huffman algorithm produces an optimal code for $C'$. 

1
Let $T'$ be the tree produced by the Huffman algorithm on $C'$. By the way the algorithm works, $T'$ is the same as $T$ (the Huffman tree for $C$) except that the leaves for $y$ and $z$, together with their parent, are replaced by the single node $x$, which is a leaf.

Let $T'_{opt}$ be the result of doing the same replacement (of $y$, $z$ and their parent with $x$) to $T_{opt}$.

We want to show that $\text{cost}(T) \leq \text{cost}(T_{opt})$, i.e., that the cost of the tree produced by the Huffman algorithm for $C$ is at least as small as the optimal cost for $C$.

\[
\text{cost}(T) = \text{cost}(T') + f(y) + f(z) \quad \text{see (a) below}
\leq \text{cost}(T'_{opt}) + f(y) + f(z) \quad \text{by the INDUCTIVE HYPOTHESIS $T'$ is optimal}
= \text{cost}(T_{opt}) \quad \text{see (b) below}
\]

So if we can prove (a) and (b), we have shown that the cost of the tree $T$ made by the Huffman algorithm for $C$ is at most the cost of the optimal tree, $T_{opt}$, and thus $T$ is optimal also.

Now prove (a). Let $d$ be the cost (i.e., depth) of $y$ and $z$ in $T$. Then the cost (i.e., depth) of $x$ is $d - 1$ in $T'$. So we have:

\[
\text{cost}(T') = \text{cost}(T) - \text{cost}(y) - \text{cost}(z) + \text{cost}(x)
= \text{cost}(T) - f(y) \cdot d - f(z) \cdot d + (f(y) + f(z)) \cdot (d - 1)
= \text{cost}(T) - f(y) - f(z).
\]

Now rearrange to get $\text{cost}(T) = \text{cost}(T') + f(y) + f(z)$.

Now prove (b). We want to show that

\[
\text{cost}(T_{opt}) = \text{cost}(T'_{opt}) + f(y) + f(z).
\]

This is done the same way as (a).