When $n = 5$, plot the following list of pairs: ((A 1) (B 2) (C 4) (D 8) (E 16)). The Huffman tree will form in following fashion.
Initial leaves {(A 1) (B 2) (C 4) (D 8) (E 16)}

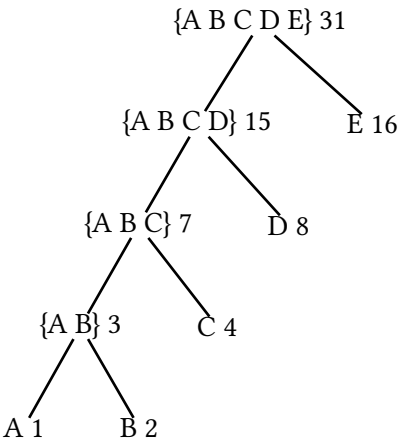
Merge {({A B} 3) (C 4) (D 8) (E 16)}

Merge {({A B C} 7) (D 8) (E 16)}

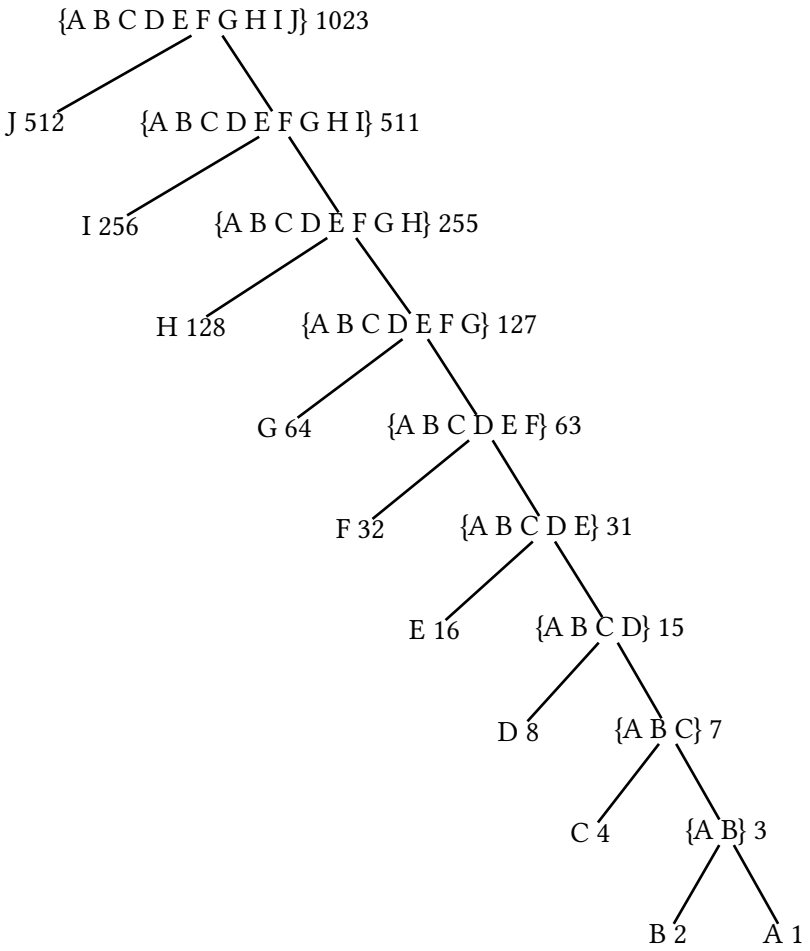
Merge {({A B C D} 15) (E 16)}

Final merge {({A B C D E} 31)}

As you can see, because $2^{n+2} > 2^{n+1} + 2^n$ for all $n > 0$, the first two elements will always be merged. This yields a tree that branches off to one side because the new node created by the merge is always the new smallest node. If the smallest node is always the left branch of the new merged node, and the merged node is always the new smallest node, then the non-leaf node will always be the left branch (except for the root and deepest leaf).



Below is the tree for $n = 10$ with alphabetic symbol-weight pairs following the same logic. This time the smallest node is the right branch of the new merged node.



In all such trees (for general n), only 1 bit is required to encode the most frequent symbol and $n - 1$ bits are required to encode the least frequent symbol. This is because each bit (that is 0) eliminates the next most frequent symbol, until the final bit which chooses between two symbols. Thus, there are $n - 2$ bits that could signal either a symbol or to continue along the tree, and 1 bit that signals two possible symbols, adding up to n symbols and $n - 1$ (bits to encode the least frequent symbol). The most frequent symbol is always one bit, as the first bit can signal reaching a leaf that contains the most frequent symbol, thus requiring no further bits and restarting the encoding process with the next symbol.