The order of the list `coin-values` does not affect the answer produced by `cc` because the order is not relevant to the method of counting change. The method works by splitting the problem into

1. finding the number of ways to count change of the same amount without one of the coins
2. finding the number of ways to count change using said coin
   - the amount remaining is the amount from before minus the value of said coin, since it must be used at least once.

As long as every coin that is an option to count change is in the list only once, it does not matter what order this operation takes place. This is because every possible combination will be exhausted by this method.

For example, the number of ways to make change of the entire amount (a dollar) with only pennies will be found regardless of the order. In the original order, the left-most node will eventually be `(cc 100 (1))`, whereas if the penny is the first value in the list, then the right-most node will `(cc 0 (1))`. The right branch of `(cc 100 (1))` will eventually also contain `(cc 0 (1))` as the amount is reduced by 1 over and over until it reaches 0. Thus the way to count change with only pennies will be covered regardless. This also is true if 1 is in the middle of the list, because the left-most branch of the `cc` tree will always include some node `(cc 100 (1 ...))` and then the right branch of that node will have a right-most node that is reducing the amount by the first denomination, which is 1, until 0 is reached.

Any possible combination will be covered by this method, the order of the list might just change the order in which that combination is found. So the combination of 3 quarters, 2 dimes, and a nickel might be found by trying to find change for 25 cents with only dimes and nickels, or trying to find change for 90 cents with only quarters, nickels, and pennies. In both scenarios, the 3Q,2D,1N combination will be one of the possible combinations found by the tree.