The order of growth of the space used by this process as the amount to be changed increases is $\Theta(n)$ because the interpreter only needs to keep track of which nodes are above it in the tree during a given point of the computation, and how much it needs to keep track of grows linearly with the amount to change. The amount of space needed will be proportional to the maximum depth of the tree, which will be the branch under `(cc amount 1)` where `amount` is the original amount (this will make more sense after examining the order of growth of steps). In this branch, the depth of the tree increases by 1 for each decrement of `amount`.

To determine the order of growth of the number of steps of `count-change` we will start by calculating the number of steps involved with making change with 1 type of coin, and then increase the number of available coins from there.
When making change with 1 type of coin, there are $2n + 1$ steps involved. This is because there are two calls for each decrement of the amount, the first being `(cc amount 1)` and the second being `(cc amount 0)`. There are n such decrements, starting at the amount and ending at 1, with one additional call being `(cc 0 1)`, which requires no further expansion due to finding a way to make change (with `amount` many pennies). Thus, there are $2n + 1$ steps where n is the amount to make change for with one type of coin (a penny). This is confirmed by the output of the associated python script, and can be visualized by examining the furthest left branch of the tree file associated with this exercise. We can thus say that the order of growth for steps for the scenario where `count-change` uses only 1 denomination to be $\Theta(n)$.
When making change with 2 coins, every step involved with making change with 1 coin is involved, so the number of steps will be at least $2n + 1$. Starting with `(cc amount 2)`, where `amount` is the amount you are trying to change, there will be an additional `(cc amount 2)` call for each time you can subtract 5 from `amount`, until and including the first 0 or negative `amount`. For each of these calls, there will be an additional $2n + 1$ steps on the associated left-side `(cc amount 1)` branch.
Now we can deduce a formula based on this behavior. There are $\lceil \frac{n}{5} \rceil + 1$ calls of `(cc amount 2)`, with $\lceil \frac{n}{5} \rceil$ of those calls having a corresponding `(cc amount 1)` call, which itself has $(2n + 1)$ branches, where n is `amount`. We add together one equation considering all calls of the kind `(cc amount 2)` ($\lceil \frac{n}{5} \rceil + 1$) with another considering the branches of (and including the calls of) `(cc amount 1)`, of which there will be only $\frac{n}{5}$, since the last `(cc amount 2)` call will have an amount of 0 or negative, and thus not have any further steps beneath it. This can thus be represented by the equation $T(n, 2) = \lceil \frac{n}{5} \rceil + 1 + \sum_{i=0}^{\frac{n}{5}} (2(n - 5i) + 1)$ where $T(n, 2)$ represents the number of steps to count change of n amount with 2 types of coins. Notice that we can also represent $T(n, 2)$ as $\lceil \frac{n}{5} \rceil + 1 + \sum_{i=0}^{\frac{n}{5}} (T(n - 5i), 1)$ to reflect the recursive nature of these calculations. Because the bulk of the steps as n grows large will come from the calls in the summation, we can ignore the ceiling for the first $\frac{n}{5}$ and simplify to get

$$
\begin{aligned}
T(n,2) &= \frac{n}{5} + 1 + \sum_{i=0}^{\frac{n}{5}} (2n - 10i + 1) \\
&= \frac{n}{5} + 1 + \sum_{i=0}^{\frac{n}{5}} (2n) - \sum_{i=0}^{\frac{n}{5}} (10i) + \sum_{i=0}^{\frac{n}{5}} (1) \\
&= \frac{n}{5} + 1 + 2n\left(\frac{n}{5}\right) - 10\sum_{i=0}^{\frac{n}{5}} (i) + \frac{n}{5} \\
&= \frac{2n}{5} + 1 + \frac{2n^2}{5} - 10\left(\frac{\left(\frac{n}{5}\right)\left(\frac{n}{5}+1\right)}{2}\right) \\
&= \frac{2n^2}{5} + \frac{2n}{5} + 1 - 5\left(\frac{n^2}{25} + \frac{n}{5}\right) \\
&= \frac{2n^2}{5} + \frac{2n}{5} + 1 - \frac{n^2}{5} - \frac{5n}{5} \\
T(n,2) &= \frac{n^2}{5} - \frac{3n}{5} + 1.
\end{aligned}
$$

When compared to the output from the python script, this is a rough approximation, but useful enough to conclude that for 2 coins, the order of growth for steps is $\Theta(n^2)$. Continuing this process, we know how to calculate a given $T(n,k)$. This is because there will be $n/$`(first-denomination kinds-of-coins)` $+ 1$ calls of (`cc amount k`), with each of those having a branch that begins with (`cc amount k-1`), which has a number of steps that can be calculated by $T(n,(k-1))$, where n is `amount`. We are therefore able to provide a formula, that when expanded, would give the number of steps for counting change with 5 types of coins:

$$
T(n,5) = \frac{n}{50} + 1 + \sum_{i=0}^{\frac{n}{50}} (T((n-50i),4)).
$$

After expansion, a term of order n^5 should dominate for large n.
We are thus able to establish that the order of growth of the number of steps used by this process as the amount to be changed increases is $\Theta(n^5)$.