Given a set of coin denominations $\mathbb{C}$ of size n, in how many ways can an amount A be changed using the coin denominations in $\mathbb{C}$?

A fairly straightforward solution to this is as follows, using the set of 5 coin denominations, $(1, 5, 10, 25, 50)$.

```
(define (count-change amount)
  (cc amount 5))

(define (nth xs n)
  (first (drop xs n)))

(define (denom n)
  (nth '(1 5 10 25 50) (- n 1)))

(define (cc amount kinds-of-coins)
  (cond ((= amount 0) 1)
        ((or (= kinds-of-coins 0)
             (< amount 0)) 0)
        (else (+ (cc amount (- kinds-of-coins 1))
                 (cc (- amount
                        (denom kinds-of-coins))
                     kinds-of-coins)))))
```

Drawing the call-tree of (count-change 11) is straightforward using the substitution method. The later part of Exercise 1.14 of SICP asks you to find the orders of growth for the space and time consumed by the procedure cc.

Space complexity
The space consumed by the recursive process generated by cc is going to be proportional to the maximum height of the recursion tree corresponding to an instance of cc, since at any given point in the recursive process, we must only keep track of the trail of nodes that leads to the root of the tree.

The maximum height of the call tree, for relatively larger amounts n, is going to be dominated by the subtree that contains successive recursive calls with the amount decreased by 1. Clearly, this means the maximum height is going to be linear in the amount n, or $\Theta\left(n\right)$.

Time complexity
Let us start with the call tree for changing some amount n, with just 1 kind of coin, i.e., the call tree for (cc n 1):

Call tree for (n, 1)

We are only allowed here to use one kind of coin, with value $\mathbb{C}_1 = 1$.

The red nodes are terminal nodes that yield 0, the green node is a terminal node that yields 1 (corresponding to the first condition in the code for cc). Each nonterminal node spawns two calls to cc, one (on the left) with

the same amount, but fewer kinds of coins, and the other (on the right) with the amount reduced by 1 and equal kinds of coins.

Excluding the root, each level has exactly 2 nodes, and there are n such levels. This means, the number of cc calls generated by a single (cc n 1) call (including the original call) is:

$$T\left(n,1\right)=2n+1=\Theta\left(n\right)$$

Next, we will look at the call tree of (cc n 2) to calculate $T\left(n,2\right)$:

Call tree for (n, 1)

Here, we are allowed to use two denominations of coins, viz. $\mathbb{C}_2=5$ and $\mathbb{C}_1=1$.

Each black node spawns a (cc m 1) subtree (blue), which we've already analyzed, and a (cc (- m 5) 2) subtree. The node colored in red and green is a terminal node, but yields 0 if the amount is less than zero and 1 if the amount is exactly zero. I've denoted this final amount as ϵ, which can be ≤ 0.

Excluding the root and and the last level in this tree which contains the red-green terminal node, there will be exactly $\lfloor\frac{n}{5}\rfloor$ levels. Now each of these levels contains a call to (cc m 1) (the blue nodes), each of which, in turn, is $\Theta\left(n\right)$ in time. So each of these levels contains $T\left(n,1\right)+1$ calls to cc. Therefore, the total number of nodes (including the terminal node and the root) in the call tree for (cc n 2) is:

$$T\left(n,2\right)=\lfloor\frac{n}{5}\rfloor\left(T\left(n,1\right)+1\right)+2=\lfloor\frac{n}{5}\rfloor\left(2n+2\right)+2=\Theta\left(n^2\right)$$

Moving ahead, let's take a look at the call tree of (cc n 3), i.e., we are now allowed to use three denominations of coins, the new addition being $\mathbb{C}_3=10$:

Call tree for (n, 1)

Here also, we see, similar to the previous case, that the total number of calls to cc will be

$$T\left(n,3\right)=\lfloor\frac{n}{10}\rfloor\left(T\left(n,2\right)+1\right)+2=\lfloor\frac{n}{10}\rfloor\times\Theta\left(n^2\right)+2=\Theta\left(n^3\right)$$

We can see a pattern here. For some k, $k>1$, we have,

$$T\left(n,k\right)=\lfloor\frac{n}{\mathbb{C}_k}\rfloor\left(T\left(n,k-1\right)+1\right)+2$$

Here, $\mathbb{C}_k$ is the k^{th} coin denomination. We can expand this further:

$$T\left(n,k\right)=\lfloor\frac{n}{\mathbb{C}_k}\rfloor\left(T\left(n,k-1\right)+1\right)+2=\lfloor\frac{n}{\mathbb{C}_k}\rfloor\left(\lfloor\frac{n}{\mathbb{C}_{k-1}}\rfloor\left(...\lfloor\frac{n}{\mathbb{C}_2}\rfloor\left(2n+1\right)...\right)\right)+2=\Theta\left(n^k\right)$$

Conclusion

In the preceding analysis of the recursive process generated by cc, we see that although it is an elegant and straighforward way of solving the problem, it is not particularly efficient in time and grows exponentially with the number of allowed denominations of coins, and polynomially with the amount to be changed when the number of denominations is fixed. Note that the actual values of the coin denominations have no effect on the order of growth of this process.