# Coin change algorithm using tabular approach

The assignment asks about writing an algorithm using tabular approach, that shows what’s the minimum number of coins is needed to exchange a given amount and given coins.

For example given coins 1 and 5:

The minimum nr of coins to exchange 10 is 2, because we need 2 x five coins to exchange 10. (10 = 2\*5)

The minimum nr of coins to exchange 14 is 6 because we need 2 x five coins and 4 x one coins. (14 = 2\*5 + 4\*1)

In the assignments the given coins are: 1, 5, 10 and 20.

The amount to exchange is 1,040,528 NOK (it’s not precised that it’s NOK, but we just assume it’s NOK).

The answer is: 52,030 coins

The coins to use for change are {1: 3, 5: 1, 10: 0, 20: 52,026} (I think the assignment didn’t ask for that, but I found it out anyway).

### The sub problems and the recurrent relations:

S is sum amount.

And the value of the coins is 1, 5, 10 or 20.

coin\_exchange(S) is a function that figures out the minimum number of coins needed to exchange S.

If we can find coin\_exchange(S-1), coin\_exchange(S-5), coin\_exchange(S-10), and coin\_exchange(S-20), and if we can figure out which of these four values found is the least, then coin\_exchange(S) is just minimum of those frou valus pluss one (that one is the last coin needed).

Also:

coin\_exchange(S) = minimum value between(coin\_exchange(S-1), coin\_exchange(S-5), coin\_exchange(S-10) and coin\_exchange(S-20)) + 1

That process is repeated on each of the four functions inside “minimum value between” function. Also e.g. in S-1:

Coin\_exchange(S-1) = minimum value between(coin\_exchange(S-(1+1)), coin\_exchange(S-(1+5)), coin\_exchange(S-(1+10)) and coin\_exchange(S-(1+20))) + 1

And we have set coin\_exchange(0) = 0. It’s obvious that if the amount is 0 NOK, then 0 coins needed to exchange it. That’s the value that ends the recursion, else the recursion may continue forever.

Another thing is, if S < any coin, like S = 19 NOK, then we may get negative values. So the minimum value between(coin\_exchange(S-1), coin\_exchange(S-5), coin\_exchange(S-10) and coin\_exchange(S-20)) will be coin\_exchange(S-20) because it’s -1 and. And that is wrong because 19 can’t be exchanged with a ‘20’ coin.

So in the algorithm we have to ignore the negative values and only choose the minimum among zero or positive values.

### Tabular approach:

In tabular approach we don’t use recursion. Rather we build and array. The size of the array is equal to S+1 also amount of money. The array start from index zero, and the index is the amount.

We set the index zero of the array equal to zero, since it’s obvious that 0 coin used to exchange 0 NOK.

Here’s the implementation in Python:

def coin\_changer(s, coins):

if s < 0:

print("the amount can't be negative integer")

return

coin\_change = list()

coin\_change.append(0)

rest\_amount = 0

# rest\_amount = s % min(coins)

for i in range(1, s+1):

min\_val = min([float('Inf') if (i - j < 0) else coin\_change[i - j] for j in coins])

min\_val = 0 if min\_val == float('Inf') else min\_val+1

coin\_change.append(min\_val)

return coin\_change[-1], rest\_amount

Let’s explain the code step by step:

if s < 0:

print("the amount can't be negative integer")

return

First, we want to avoid processing negative amounts, so it checks if it’s negative, and it prints some text and return. I could make it check if any coins are of negative values, but I just drop that.

coin\_change = list()

coin\_change.append(0)

rest\_amount = 0

#rest\_amount = s % min(coins)

We then initialize a table, an array. And we set the first value of it zero. In Python array index start from 0. The index show the amount, and the values that will be saved in there will show the minimum number of coins required to exchange that amount. If the amount is 0 NOK, we need 0 coins to exchange, that's obvious. It's from there we iterate and find out minimum number of coins for all amounts between 1 and S. So the size of array will be S + 1 (since we include zero too). The variable rest\_amount show the amount that can’t be exchanged. E.g. if we didn’t had coin valued 1, then we couldn’t exchange whole of 9 NOK. The rest amount would be 4 NOK. But I commented that out because we don’t need it in this case to save from execution time.

for i in range(1, s+1):

min\_val = min([float('Inf') if (i - j < 0) else coin\_change[i - j] for j in coins])

min\_val = 0 if min\_val == float('Inf') else min\_val+1

coin\_change.append(min\_val)

Then we iterate in a range from index 1 to index S +1 .

Inside “min()” we iterate through coins. The coins here are 1,5,10,20. We check the minimum number of coins to exchange s-1, s-5, s-10 etc. and we then choose the minimum value. If the value is negative, we put Inf so that it’s ignored when comparing. Also we want to get the minimum value that’s NOT negative.

But on the next line to we check if the value is Inf and turn it into 0 so that we don’t save any Inf value in the array (recall we get Inf only if the value is less than the minimum coin). That’s possible when we don’t have any ‘1’ coin. If we only have the coin ‘5’, then we need zero coin to exchange 4 NOK.

If it’s not Inf, then we add +1 to the value. If we need 5 coins to exchange S-5, then to exchange S we need 6 coins (assume we have coin valued 5 NOK), the 6th one is the last amount of 5 NOK. Thus we put +1 to the min\_val (minimum value).

return coin\_change[-1], rest\_amount

Then we return both value of last index and rest\_amount, and we print the result outside the function.

The minimum coins needed to exchange 1,040,528 is 52,030

when the avialable coins are 1, 5, 10, and 20 NOK.

The coins to use for change are {1: 3, 5: 1, 10: 0, 20: 52,026}

The function “which\_coins()” is used to find which coins to use to change the amount, which needs the return value from the first function. But since the assignment doesn’t ask about which coins to use to change the sum amount, I won’t describe that in detail.