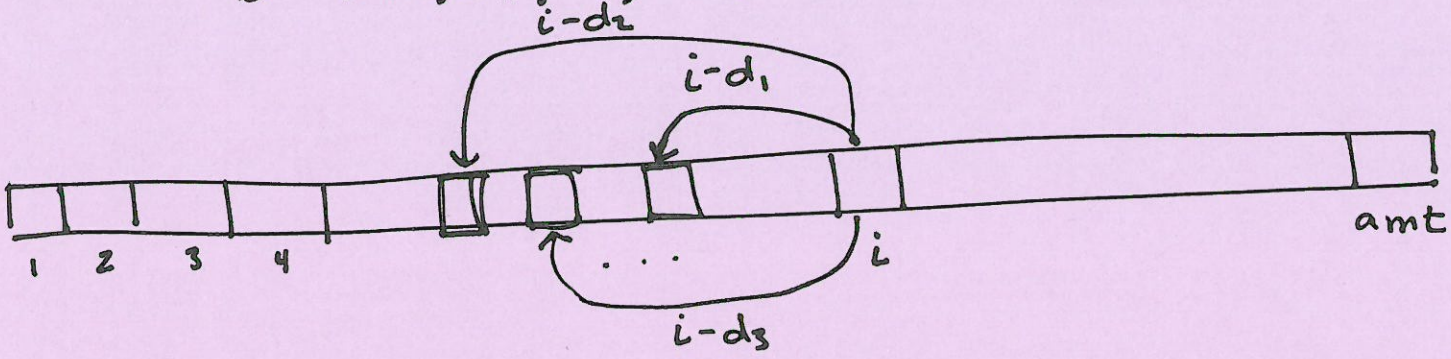
7 October 2022

Non-greedy Making Change

Given: amt, total we want to make

$D = \{d_1, d_2, \ldots, d_k\}$ denominations



The recurrence:

$$\text{NumCoins}(amt, D) = \begin{cases} 0, & amt = 0 \\ [\![\infty, & amt < 0]\!] \\ \min\limits_{\substack{d_j \in D \\ \text{s.t.} \\ amt \geq d_j}} \{\text{NumCoins}(amt - d_j, D) + 1\} \end{cases}$$

↑ Better not to use this recurrence in practice. So, Memoized DP!

For DP, we built the sol'n bottom-up.

Pseudocode:

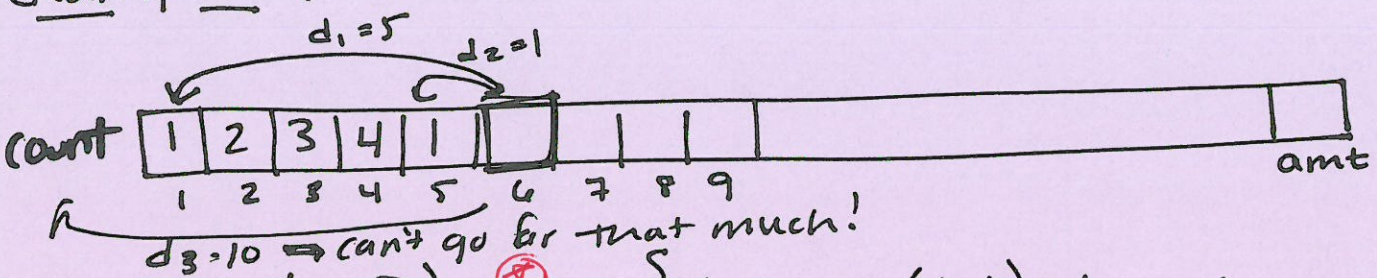
See rewrite

```
NumCoins (amt, D)
    count ← array of length amt+1, init ∞
    count(0) ← 0
    if amt=0
        return 0
    endif
    for i=1... amt
        for dj ∈ D
            if i-dj ≥ 0
                count(i) = min {count(i), count(i-dj)+1}
            endif
        end for
    end for
    return count(amt)
```

the current value

dj is last coin added

example: {1¢, 5¢, 10¢} = D



NumCoins(6, D) = min { NumCoins(1, D) + 1 , NumCoins(5, D) + 1
"adding nickle" "adding penny"

~~NumCoins~~ (6 − 10¢)

= min {2, 2} = 2

Note: this is using the recursive def'n.
To actually recurse could be costly...

= min { count(1, D) + 1, count(5, D) + 1 }

Rewrite!

```
NumCoins(amt, D)
    count ← array of length amt+1, init to ∞
    count(0) ← 0
    if amt = 0
    |  return 0
    end if
    for i = 1 ... amt
    |   for dj ∈ D
    |   |   if i - dj ≥ 0
    |   |   |   count(i) = min{count(i), count(i - dj) + 1}
    |   |   end if
    |   end for
    end for
    return count(amt)
```

count(i): current best

count(i − dj) + 1: trying using dj as the last coin

Runtime: $\Theta(amt \cdot D)$

Space: $\Theta(amt + D)$ in total

$\Theta(amt)$ additional/aux. structures needed.

Edit Distances

Given 2 words (sequence of letters), find the min # of edits to get between the words.

An edit can be one of the following:

- → insertion, cost = 1
- → deletion, cost = 1
- → replacement, cost = 1

e.g., BRANCH
BLANCH have edit distance 1

e.g., THIS
TWICE
two

Q: can we formulate this recursively?