5 Oct 2022

1. Formulate Recursively
   (a) Specification: WHAT are you solving?
   (b) Solution: clear recursive formula or pseudocode
2. Build Sol'n from Bottom up
   (a) Identify + count the DISTINCT sub problems
   (b) Choose memoization data structure
   (c) Identify dependencies: make Dag — nodes: distinct probs, arrows: dependency
   (d) Find a good eval order
   (e) Analyze space/time
   (f) Write Down algo.
   (g) Improve!

Making Change → goal: given amt to make and set of denominations, find smallest # coin

- W/ US currency, greedy strategy works.
  $0.83 → $0.50 coin is biggest that "fits". use that, then make 33¢
  → next, use quarter then make 8¢
  → one nickle, then make 3¢
  → 3 1¢ coins
  Strategy: start w/ biggest that "fits" use as many as you can, then go down to next ~~lower~~ largest denomination.
- example: Make 20¢ and 34¢
  Denominations 1¢, 10¢, 15¢, 14¢
  → greedy: 6 coins: 1×15¢ + 5×1¢
  → optimal: 2 coins: 2×10¢

Task: Create a DP for making change (smallest # coins)

- input: amt $\in \mathbb{N}$

  D, an array of pos. integers (the values of the coins)

- output: the minimum # of coins needed to make amt.

```
MakeChange(amt, D)    //assuming the amt can be made
    if amt = 0
    | return 0
    endif
    best ← ∞
    for coin ∈ D
    |   if coin > amt
    |   | cur ← MakeChange(amt - coin
    |   |
    |   if cur < best
    |   | best ← cur
    |   endif
    |   endif
    end for
    return best
```