# Programación dinámica

...

Técnica que se usa para encontrar soluciones óptimas a problemas, y para contar el número de soluciones.

# Conteo de monedas

Tenemos un conjunto de valores de monedas *coin_values = {c1, c2, c3, … , cn}*, y un monto objetivo de dinero *x*. Obtén la cantidad *x* utilizando la menor cantidad de monedas con valores de entre los que están en *coin_values.* No hay restricción en cuanto al número de veces que se usa un valor de moneda.

coin_values = { 1, 2, 5}

x = 12

coins = { 5, 5, 2 }

# Greedy Algorithms

Un algoritmo que toma la mejor solución inmediata, o local, al buscar una respuesta. Encuentran la solución óptima en algunos casos, pero típicamente encuentran soluciones no óptimas para algunas instancias del problema.

# Conteo de monedas

```
def coins_greedy(coin_values, target):
  coins = []
  index = -1

  while target> 0:
    max_value = coin_values[index]

    if target - max_value >= 0:
      target -= max_value
      coins.append(max_value)
    else:
      index -= 1

  return coins
```

coin_values = { 1, 3, 4}

x = 6

En programación dinámica, se formula el problema recursivamente, para calcular la solución a partir de subproblemas más pequeños.

# Conteo de monedas

solve(0) = 0
solve(1) = 1
solve(2) = 2
solve(3) = 1
solve(4) = 1
solve(5) = 2
solve(6) = 2
solve(7) = 2
solve(8) = 2
solve(9) = 3
solve(10) = 3

solve(10) = 3

3 + 3 + 4 = 10

# Conteo de monedas

solve(x) = min( solve(x-1) + 1, solve(x-3) + 1, solve(x-4) + 1)

solve(10) = solve(7) + 1 = solve(4) + 2 = solve(0) + 3