

# 1 Código em C++

## 2 Demonstração

**Lema 2.1.** Cada solução ordenada  $(s_1, s_2, \dots, s_k)$  é unicamente representada por um vetor  $(a_1, \dots, a_n)$ , em que  $a_i$  é a quantidade de vezes que a moeda  $c[i]$  foi utilizada.

**Lema 2.2.** Denote por  $C(i, j)$  a quantidade de combinações  $(a_1, \dots, a_n)$  com soma  $j$  que usam apenas as moedas  $c[0], \dots, c[i]$ , isto é, com  $a_{i+1} = \dots = a_n = 0$ , e (extendendo essa definição) deixe  $C(-1, j)$  denotar a quantidade de combinações vazias com soma  $j$ .

Então, após a  $i$ -ésima iteração,  $dp[j] = C(i, j)$ .

*Prova.* Primeiro, provamos a seguinte sub-invariante de loop:

**Lema 2.3. Assuma** a hipótese de indução do lema principal, isto é, que temos inicialmente  $dp[j] = C(i, j)$  para todo  $0 \leq j \leq x$ . Então, durante a  $(i + 1)$ -ésima iteração do loop externo, o seguinte é válido: para todo  $0 \leq w \leq x$  temos que antes da iteração  $w = w_0$  do loop interno,  $dp[j]$  representa quantas combinações de soma  $j$  há usando apenas as moedas  $c[0], \dots, c[i + 1]$  para todo  $0 \leq j < w_0$ .

*Prova.* Antes da primeira iteração  $w = c[i + 1]$  do loop interno, temos que  $j < w \implies j < c[i + 1]$ , o que implica evidentemente que não há combinação com essa soma que use  $c[i + 1]$ . Dessa forma, temos que  $C(i + 1, j) = C(i, j) = dp[j]$ , pela hipótese indutiva.

Por fim, suponha válido para  $w = w_0$ . Toda combinação que usa as moedas  $c[0], \dots, c[i + 1]$  é de duas formas:

- $a_{i+1} = 0$  ( $C(i, w_0 + 1)$  possibilidades)
- $a_{i+1} \geq 1$ , e portanto a combinação obtida retirando uma repetição de  $c[i + 1]$  é uma combinação válida de soma  $w_0 + 1 - c[i + 1]$  usando as moedas  $c[0], \dots, c[i + 1]$ . Isso totaliza  $C(i + 1, w_0 + 1 - c[i + 1])$ .

Assim,

$$C(i + 1, w_0 + 1) = C(i, w_0 + 1) + C(i + 1, w_0 + 1 - c[i + 1])$$

Uma vez que  $w_0 + 1 - c[i + 1] \leq w_0$ , pela hipótese indutiva desse sub-lema temos que nesse instante  $C(i + 1, w_0 + 1 - c[i + 1]) = dp[w_0 + 1 - c[i + 1]]$ . Por outro lado, temos que  $dp[w_0 + 1]$  não foi alterada até agora nesse loop interno, e portanto pela hipótese indutiva do lema principal  $dp[w_0 + 1] = C(i, w_0 + 1)$ . Portanto, a ação do algoritmo é

$$dp[w_0 + 1] \leftarrow (dp[w_0 + 1] + dp[w_0 + 1 - c[i + 1]] = C(i, w_0 + 1) + C(i + 1, w_0 + 1 - c[i + 1]))$$

garantindo a manutenção da invariante descrita nesse sub-lema. □

Dessa forma, a invariante principal (i.e, do loop externo) segue com um simples argumento indutivo. A inicialização de  $dp[0] \leftarrow 1$  e  $dp[j] \leftarrow 0$  para todo  $j > 0$  garante que  $dp[j] = C(-1, j)$  para todo  $0 \leq j \leq x$ .

Assim, aplicando o lema anterior para  $i = -1$ , demonstramos o caso base: a validade ao final da iteração  $i = 0$ . Analogamente, o passo indutivo também segue da aplicação do lema anterior. □