Q2

```
CONTA-TROCOS(v, q, n, m)
  t[0..m]                      // Guarda os trocos
  t[0] = 1                     // Tem um jeito de dar 0 de troco
  para i <- 1 até m faça       // Inicialmente não tem;          O(m)
    t[i] <- 0                  // como dar outros trocos;        O(1)
  para i <- 1 até n faça       // Extende os conjuntos contados; O(n)
    para k <- 0 até m -1 faça  // Conta novos trocos possíveis;  O(m)
      l <- m - k               // Corrige o index
      se q = 0 então continua  // ignora quando não tem notas
      acc <- 0                 // Conta a quantidade de novos
      para x <- 1 até q[i] faça // trocos possíveis;            O(max
em q)
        y <- l - v[i] * x      // Encontra o índice do conjunto  O(1)
        se y >= 0              // E vê se ele é valido para extender
          então acc <- acc + q[y]
      t[l] <- t[l] + acc
  devolva t[m]
```

Esse algoritmo consome $O(n \cdot m \cdot \max(q))$.

A corretude se deve ao fato do algoritmo testar todas as configurações (ignorando os trocos maiores que m) usando todas as quatidades disponíveis de notas para cada valor de nota.