Complexidade Subset Sum recursivo

```
SUBSET-SUM-REC (p, n, c)
se n = 0
    então se c = 0
              então devolva 1
              senão devolva 0
    senão s ← SUBSET-SUM-REC (p, n−1, c)
          se s = 0 e p_n ≤ c
              então s ← SUBSET-SUM-REC (p, n−1, c−p_n)
          devolva s
```

Fórmula do subset sum recursivo sabendo que $n >= 1$:

$T(n) = 2 + T(n-1) + T(n-1)$

$T(n) = 2 + 2T(n-1)$

Ampliando a função:

$T(n-1) = 2 + 2T(n-2)$

$T(n-2) = 2 + 2T(n-3)$

$T(n-3) = 2 + 2T(n-4)$

...

Substituindo na fórmula original:

$T(n) \sim> \quad T(n) = 2 + 2T(n-1)$

$T(n-1) \sim> T(n) = 2 + 2(2 + 2T(n-2)) = 2 + 2^2 + 2^2T(n-2)$

$T(n-2) \sim> T(n) = 2 + 2^2 + 2^2(2 + 2T(n-3)) = 2 + 2^2 + 2^3 + 2^3T(n-3)$

$T(n-3) \sim> T(n) = 2 + 2^2 + 2^3 + 2^3(2 + 2T(n-4) = 2 + 2^2 + 2^3 + 2^4 + 2^4T(n-4)$

Tem-se um padrão:

$T(n) = 2 + 2^2 + ... + 2^{(k-2)} + 2^{(k-1)} + 2^k + (2^n) * T(n-k)$

$2 + 2^2 + ... + 2^{(k-2)} + 2^{(k-1)} + 2^k$ é uma sequência, portanto pode ser escrito como $2^k - 2$

Porque sendo a sequência dada acima igual a abaixo:

$S = 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + ... + 2^n$

$2S = 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + ... + 2^{(n+1)}$

$2S - S = -2 + 2^{(n+1)}$

$-2 + 2^{(n+1)}$ pode ser definido como $2^k - 2$

Portanto substituindo em T(n):

$T(n) = (2^n) * T(n-k) + (2^k-2)$

Se para n = 0 T(n) = 0 e para n>=1 o próprio T(n), assim n - k = 0, logo n=k. T(n-k) será uma constante.

Logo:

$T(n) = (2^n) * \text{constante} + (2^k-2)$ ~> a constante pode ser dispensada e n-k

$T(n) = (2^n) + (2^n - 2)$

$T(n) = 2^{(n-1)} - 2$

Assim, a complexidade do algoritmo do Subset Sum recursivo é:

$O(2^n)$