

## Bloqueio de Tela

A ideia primariamente deve ser a de construir um algoritmo que partindo de um ponto e sabendo quais pontos já foram usados, calcula-se pra quais pontos é possível ir. Com o tamanho máximo de 4 por 5, é possível determinar 6 retas que podem ser usadas para se achar os pontos:  $R = \{(0, 1), (1, 1), (1, 2), (1, 3), (1, 4), (2, 3)\}$ . Essas retas precisam ser espelhadas horizontalmente, verticalmente e também dobradas para cobrir todo o plano, e elas originam dos pares  $(x, y)$  onde  $\gcd(x, y) = 1$  respeitando  $x \leq 4$  e  $y \leq 4$ . Para cada uma, se calcula o primeiro ponto que não foi utilizado ainda.

Tendo esse algoritmo, é questão de criar uma função usando programação dinâmica tendo o primeiro argumento o ponto de partida, e o segundo argumento a utilização dos pontos em bitmask. São 20 pontos possíveis, então temos um vetor de tamanho  $20 \times 2^{20}$ , que do tipo de dados inteiro de 4 bytes, cabe no limite de memória.

Para o cálculo, lemos da entrada os “multiplicadores” para cada ponto. Um multiplicador de 1 significa que o ponto só admite uma combinação possível, enquanto que com 2, são admitidas duas combinações. Como somos limitados em  $[A, B]$ , se forem utilizados  $B$  pontos na bitmask, retornamos o multiplicador do ponto atual, senão, multiplicamos o nosso multiplicador pela soma da combinação de todos os pontos atingíveis (recursivamente) apenas se forem utilizados pelo menos  $A$  pontos.

Em todos esses cálculos, é necessário lembrar de aplicar o módulo. Para evitar problemas, é tudo feito usando long long, mas a memória da programação dinâmica não necessita, pois o módulo cabe em 4 bytes.

Outro problema é que tanto  $M$  quanto  $N$  podem ser 5, enquanto que geralmente na nossa implementação, geralmente apenas uma combinação é válida. Nesse caso, simplesmente transpomos os multiplicadores e trocamos  $N$  por  $M$ .