8. ⋆Se tiene una grilla con $m \times n$ posiciones, cada una de las cuales tiene un número entero en $[0, k)$, para un $k \in \mathbb{N}$ dado. Dado un valor objetivo $w \in \mathbb{N}$ y una posición inicial (x_1, y_1), que tiene un valor inicial v_1, queremos determinar la mínima cantidad de movimientos horizontales y verticales que transformen v_1 en w, teniendo en cuenta que el i-ésimo movimiento transforma a v_i por $v_{i+1} = (v_i + z) \bmod k$, donde z es el valor que se encuentra en la casilla de destino del movimiento. Por ejemplo, para la siguiente grilla y $k = 10$, se puede transformar $v_1 = 1$ en $w = 0$ con tres movimientos $1 \to 6 \to 4 \to 9$, aunque la solución óptima es vía el camino $1 \to 3 \to 6$.

1	3	6
6	7	4
4	9	-3

Modelar este problema como un problema de grafos que se resuelva usando BFS en $O(kmn)$ tiempo.

Sea G un grafo conexo que vamos a ir construyendo a medida que buscamos un camino mínimo con BFS.

Los vértices de G se identifican como una tupla (x, y, s) donde x, y es la casilla de la grilla y s es la suma parcial mod K que llevamos acumulada hasta llegar a ese vértice.

Una arista $((x_1, y_1, v_1), (x_2, y_2, v_2))$ representa haberse movido desde x_1, y_1 hacia x_2, y_2 en la grilla, $v_2 = (v_1 + \text{grilla}[x_2][y_2]) \bmod K$. Solo vamos a generar aristas que representan movimientos válidos.

Cada vez que visitamos un vértice (x, y, s) durante el BFS, si $s = W$ cortamos y el nivel del árbol BFS el la cantidad mínima de movimientos para resolver el problema.

Caso contrario encolamos todos los vecinos alcanzables con movimientos válidos.

Al identificar los vértices de esta forma permitimos visitar una misma casilla x, y varias veces en función del valor de s. Esto es necesario porque si bien cada casilla puede aparecer una única vez por camino, pueden haber muchos caminos que llegan a la misma casilla con un acumulado s distinto. Si 2 caminos llegan con el mismo valor de s a una casilla, el resto de ese camino se explora una única vez.

Si terminamos el BFS y no cortamos antes entonces no hay solución.

La complejidad resulta $O(Knm)$ porque el grafo generado en el peor caso contiene todas las casillas que son $\Theta(nm)$ con todos los posibles valores de $s \in [0, K)$. Es decir $O(Knm)$ vértices, y como el grafo es un árbol BFS, tiene $O(Knm-1)$ aristas. Todas las operaciones realizadas al visitar un vértice son $O(1)$. BFS tiene complejidad $O(V+E) = O(Knm + Knm-1) = O(Knm)$.

Algoritmo

Input:

$M[1...n][1...m]$ con $M[x][y] \in [0, K)$ grilla

$K \in \mathbb{N}$ módulo

$W \in \mathbb{N}$ valor objetivo

$(x_I, y_I) \in \mathbb{N}^2$ posición inicial

```
visited[1...n][1...m][0...K) ← False
path_length ← 1
level_count ← 1
next_level_count ← 0
Q ← nueva cola vacía
enqueue(Q, (x_I, y_I, M[x_I][y_I]))

while Q ≠ ∅:
    (x1, y1, v1) ← dequeue(Q)
    if v1 = w: return path_length

    for (dx, dy) ∈ {(1,0), (-1,0), (0,1), (0,-1)}:
        (x2, y2) ← (x1 + dx, y2 + dy)
        if ¬(1 ≤ x2 ≤ n ∧ 1 ≤ y2 ≤ m): continue
        v2 ← (v1 + M[x2][y2]) mod K
        if visited[x2][y2][v2]: continue
        visited[x2][y2][v2] ← true
        next_level_count ← next_level_count + 1

    level_count ← level_count - 1
    if level_count = 0:
        level_count ← next_level_count
        next_level_count ← 0
        path_length ← path_length + 1

return -1
```