Reto 1: Números del revés

Descripción del reto

Un número del revés es un número entero que aparece igual cuando se gira 180 grados, por ejemplo:

1961 → **1961** ✅
88 → **88** ✅
66 → **99** ❌
101 → 101 ✅

Debes crear una función capaz de sacar los números de un rango que sean reversibles.

Input:

La función recibirá dos valores. Estos dos valores representarán los límites superior e inferior de un rango.

Output:

La función debe devolver los números y la cantidad total de números invertidos válidos dentro del rango de los dos argumentos de entrada, incluidos los límites superior e inferior.

Detalles importantes de la prueba

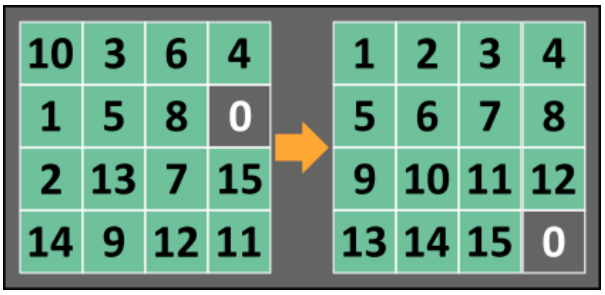
- El primer argumento siempre será menor que el segundo argumento (es decir, el rango siempre deberá ser válido).

Reto 2: El rompecabezas

Descripción del reto

Un rompecabezas deslizante es un rompecabezas combinado que desafía al jugador a deslizar piezas (con frecuencia planas) a lo largo de ciertas rutas (generalmente en un tablero) para establecer una configuración final determinada.

Su objetivo para este reto es escribir una función que produzca una secuencia de movimientos de fichas que resuelva el rompecabezas.



Input del algoritmo

Una matriz/lista n x n compuesta por valores enteros que van de 0 a n^2 - 1 (inclusive), que representa una cuadrícula cuadrada.

Ten en cuenta que siempre habrá una ficha vacía (representada por 0) para permitir el movimiento de fichas adyacentes.

Output del algoritmo

Una matriz/lista compuesta por cualquiera (pero no necesariamente todos) de los números enteros de 1 a n^2 - 1, inclusive.

Esto representa la secuencia de movimientos de fichas para una transición exitosa del estado inicial sin resolver al estado resuelto. Si el rompecabezas no se puede resolver, devolverá None (Python).

Comportamiento esperado

Input del algoritmo:

```
input_ejemplo = [
    [ 1, 2, 3, 4],
    [ 5, 0, 6, 8],
    [ 9,10, 7,11],
    [13,14,15,12]
]
```

Output del algoritmo:

```
# Transciones a realizar para resolver el rompecabezas:
'''

[ 1, 2, 3, 4]    [ 1, 2, 3, 4]    [ 1, 2, 3, 4]    [ 1, 2, 3, 4]    [ 1, 2, 3, 4]
[ 5, 0, 6, 8]    [ 5, 6, 0, 8]    [ 5, 6, 7, 8]    [ 5, 6, 7, 8]    [ 5, 6, 7, 8]
[ 9,10, 7,11] -> [ 9,10, 7,11] -> [ 9,10, 0,11] -> [ 9,10,11, 0] -> [ 9,10,11,12]
[13,14,15,12]    [13,14,15,12]    [13,14,15,12]    [13,14,15,12]    [13,14,15, 0]

'''
```

Detalles importantes para la prueba

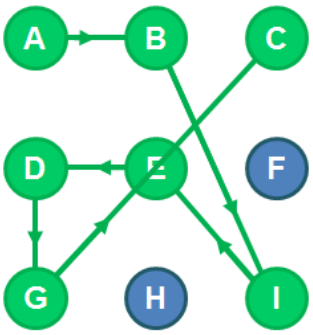
- El input debe ser válido.
- El rango de valores para n es: 10 >= n >= 3

Reto 3: Patrones de desbloqueo

Descripción del reto

Este reto está inspirado en la funcionalidad de desbloqueo de dispositivos Android a través de las combinaciones disponibles en la pantalla de bloqueo conectando los puntos sin levantar el dedo de la pantalla

Debes desarrollar una función capaz de encontrar el número de combinaciones posibles conectando los puntos disponibles.



La imagen anterior muestra un patrón de 7 puntos conectados: **(A -> B -> I -> E -> D -> G -> C)**

Objetivo:

Deberás implementar una función que devuelva la cantidad de patrones posibles a partir de un primer punto dado, que tienen una longitud determinada.

Los puntos a conectar serían:

A	B	C
D	E	F
G	H	I

La función contará con un primer parámetro de tipo caracter correspondiente al punto de la cuadrícula (A, B, C, D, E, F, G, H o I).

La función contará con un segundo parámetro de tipo numérico correspondiente al número de puntos que debe tener cada patrón.

Por ejemplo:

Por ejemplo, pasar los parámetros “C" y 2, debería devolver el número de patrones que comienzan en 'C' y que tienen 2 conexión de dos puntos. El valor de retorno en este caso sería 5, porque hay 5 patrones posibles:

(C -> B), (C -> D), (C -> E), (C -> F) y (C -> H).

No debe devolver los patrones en sí, solo el número de patrones posibles.

Reglas

- En un patrón, los puntos/puntos no se pueden repetir: solo se pueden usar una vez, como máximo.
- En un patrón, dos puntos/puntos posteriores cualesquiera solo se pueden conectar con líneas rectas directas de cualquiera de estas formas:
    - **Horizontalmente**: como (A -> B).
    - **Verticalmente**: como (D -> G).
    - **Diagonalmente**: como (I -> E), así como (B -> I).
- Pasando por encima de un punto entre ellos que ya ha sido 'usado': como (G -> C) pasando por encima de E, en la imagen del patrón de ejemplo.
    - Esta es la regla más complicada. Normalmente, no podría conectar G con C, porque E está entre ellos, sin embargo, cuando E ya se ha utilizado como parte del patrón que está rastreando, puede conectar G con C pasando por E, porque E se ignora, al haber sido usado.