El problema de las Torres de Hanoi y recursividad.

La palabra recursividad es utilizada para describir un proceso definido sobre sí mismo. Es decir, un proceso que espera determinados valores de entrada, y es capaz de procesarlas y producir una salida haciendo uso de su propia definición capaz de procesar con éxito un tamaño **n** de entrada, basándose en que la definición garantiza la solución para un tamaño (**n – 1)** de la misma.

Es posible ilustrar esta idea de recursividad estudiando la solución del juego Las Torres de Hanoi.

1. Se quiere trasladar la **n** piezas del poste origen al poste destino. La pieza **n** al principio está debajo de las **n-1** piezas restantes. Por regla (una pieza mayor no puede estar sobre una pieza menor), la única manera de pasar la pieza **n** a destino sería pasar la **n–1** piezas al poste auxiliar y luego la pieza **n** a destino.
2. Al querer mover las **n-1** piezas a auxiliar, se tiene que nuevamente se trata de una nueva torre de tamaño **n-1,** y se presenta el mismo problema al momento de pasar la pieza **n-1** al poste destino. La única manera de pasar la pieza **n-1** a destino, sería pasar las **n-2** piezas al poste auxiliar y luego la pieza **n-1** a destino.
3. El mismo problema se presentara *recursivamente* hasta que nos encontremos con el caso en el cual trasladar una torre es equivalente a trasladar un **solo disco.**

En base a estas consideraciones, es posible escribir el siguiente algoritmo recursivo que resuelve el juego de las torres de Hanoi, teniendo un problema inicial de tamaño **n**, que puede ser definido para el tamaño **n-1** sucesivamente hasta que se presente un caso base fácil de resolver, y que inductivamente garantiza la solución de todos llamados anteriores:

Algoritmo para trasladar la torre 1...n del poste origen al poste *destino,* usando el poste *intermedio* como poste auxiliar:

1. Si n = 1, lleve el disco 1 de *origen* a *intermedio* y termine.
2. Traslade la torre 1...n-1 usando este mismo algoritmo, *de origen* a *intermedio* usando *destino* como auxiliar*.*
3. Lleve el disco n de *origen* a *destino*.
4. Traslada la torre 1...*n*–1 usando este mismo algoritmo, de *destino* a *intermedio*, usando como auxiliar *origen*.

Este algoritmo se ejecuta en el menor número de pasos posibles para resolver el juego de las torres de Hanoi para *n* piezas, esto es:

$$F(n) = 2^n - 1$$

Pregunta 1) ¿Es posible solucionar el juego de las torres de Hanoi sin usar recursividad?

Sí, existe una manera de resolver el juego de las torres de Hanoi usando un algoritmo iterativo. Existen dos algoritmos para ello, dependiendo de el número de piezas de la torre y si este es par o impar.

1) Número de piezas es impar:
    1. Si inicialmente se tiene un número impar de discos, el primer movimiento debe ser colocar el disco más pequeño en la pila *destino*, y en cada paso impar se le mueve a la siguiente pila a su izquierda (o a la pila *destino* si está en la pila *origen*).

2) Número de piezas es par:
    1. Si se tiene inicialmente un número par de discos, el primer movimiento debe ser colocar el disco más pequeño en la pila *intermedia*, y en cada paso impar se le mueve a la siguiente pila a su derecha (o a la pila *origen* si está en la pila *destino*).

Pregunta 2) ¿Si el juego se encuentra en una posición cualquiera de los discos, se puede determinar si es una posición válida, es decir, una posición intermedia para llegar a la solución del juego?

Es posible determinar si una jugada hecha por quien juega es una jugada de la serie de jugadas intermedias para resolver el juego (las piezas se enumeran de menor a mayor radio).

1. El número total de piezas en la torre es impar:
    - Todas las piezas impares deben moverse en el siguiente patrón: *origen → destino → intermedio → origen → destino …*
    - Todas las piezas pares deben moverse en el siguiente patrón: *origen → intermedia → destino → origen → intermedia …*
2. El número total de piezas en la torre es par:
    - Todas las piezas impares deben moverse en el siguiente patrón: *origen → intermedia → destino → origen → intermedia …*
    - Todas las piezas pares deben moverse en el siguiente patrón: *origen → destino → intermedio → origen → destino …*