Analiza el algoritmo del ejercicio anterior y calcula cuál sería su complejidad utilizando la notación Big-O. Debes indicar claramente el procedimiento que has utilizado y no sólo la solución.

Suponiendo que introducimos un número de discos de 3:

Al ser un algoritmo recursivo y según dice la notación Big-O, tenemos que sumar el orden de complejidad del caso base más el del caso recursivo.

Caso Base:

```
if (numDiscos == 1) {
        System.out.println("- Muevo disco " + numDiscos + " de " + torreOrigen +
" a " + torreDestino);
    }
```

- O(1) --> Constante. Tarda lo mismo independientemente del tamaño de la entrada.

Caso Recursivo:

```
      else {
        moverDiscos(numDiscos - 1, torreOrigen, torreApoyo, torreDestino);
        System.out.println("- Muevo disco " + numDiscos + " de " + torreOrigen +
" a " + torreDestino);
        moverDiscos(numDiscos - 1, torreApoyo, torreDestino, torreOrigen);
    }
```

- O(a^n) --> Exponencial. Es una de las peores complejidades algorítmicas. Sube demasiado a medida que crecen los datos de entrada.

Complejidad Caso:

- O(1) + O(1^n) + O(1^n) --> O(2^n)

La complejidad de las Torres de Hanoi es O(2^n), donde 'n' es el número de discos. Esto es debido a que es una función recursiva que llamamos dos veces. En este caso concreto en el que hay 3 discos sería O(8).