**• ¿Cuál es la complejidad temporal del algoritmo que has implementado para convertir un número decimal a binario utilizando una pila? Explica por qué**

La **complejidad temporal** del algoritmo es **O(logb ​n)**, donde:

* n es el número decimal a convertir.
* b es la base a la que se realiza la conversión (en este caso, binario, por lo tanto b=2).

**Razón**: El algoritmo realiza divisiones sucesivas de n por la base b, reduciendo el tamaño del número en cada iteración hasta que el cociente sea 0. Esto significa que el número de iteraciones es proporcional a **O(logb ​n)**, ya que el número decimal se reduce en cada paso a la fracción correspondiente de la base.

Para la conversión a binario, específicamente, la complejidad es **O(log2 ​n)**. Esto es porque en cada iteración dividimos entre 2, lo que efectivamente recorta en la mitad el número de dígitos necesarios.

**•¿Qué espacio adicional utiliza tu algoritmo para almacenar los residuos en la pila? ¿Cómo afecta esto la complejidad espacial del programa?**

El espacio adicional necesario para almacenar los residuos en la pila es también **O(logb ​n).** Esto se debe a que la cantidad de residuos almacenados es igual al número de divisiones necesarias para reducir el número decimal a 0.

**Impacto en la complejidad espacial**:  
El espacio adicional utilizado es directamente proporcional a la cantidad de residuos que debemos almacenar, lo cual es igual a la cantidad de iteraciones del algoritmo. Cada residuo se almacena en la pila hasta que se alcance el cociente cero, lo que afecta la complejidad espacial en función del tamaño del número inicial. Para la conversión a binario, la complejidad espacial es **O(log2 ​n)**.

**• Si decidieras implementar la conversión sin utilizar una pila, ¿cómo cambiaría la complejidad temporal y espacial? Justifica tu respuesta**

Si decidimos **no usar una pila**, el algoritmo aún podría realizar las divisiones sucesivas. Sin embargo, al no usar una pila, tendríamos que almacenar los residuos en otro formato, probablemente en una lista o simplemente construyendo la cadena final directamente desde los residuos.

**Cambio en la complejidad temporal**:  
La complejidad temporal seguiría siendo **O(logb ​n)**, ya que las divisiones sucesivas seguirían siendo necesarias y no cambiaría el número de operaciones que realiza el algoritmo.

**Cambio en la complejidad espacial**:  
La complejidad espacial podría ser ligeramente reducida si evitamos almacenar los residuos en una pila y en su lugar construimos directamente el número convertido. Esto eliminaría la sobrecarga de la estructura de la pila, pero aún necesitaríamos espacio proporcional a **logb ​n** para almacenar el resultado final. Por lo tanto, la complejidad espacial también seguiría siendo **O(logb ​n)**, aunque posiblemente se reduciría la constante de proporcionalidad.

**• Compara la complejidad de tu algoritmo de conversión con otros algoritmos que realicen tareas similares (por ejemplo, conversión entre bases sin pilas). ¿Qué ventajas o desventajas encuentras?**

* **Ventajas de usar pilas**:
  + **Claridad y organización**: La pila proporciona una estructura clara para almacenar los residuos y facilita la construcción del número convertido en orden inverso.
  + **Uso de estructuras de datos conocidas**: El uso de pilas es un enfoque clásico para este tipo de problema, especialmente en entornos educativos y problemas de algoritmos.
  + **Modularidad**: Se pueden aplicar de manera eficiente otras operaciones de la pila como pop() y push(), haciendo el código más limpio y reutilizable.
* **Desventajas de usar pilas**:
  + **Sobrecarga espacial**: Las pilas requieren espacio adicional para almacenar los residuos antes de generar el número final. Esto introduce una sobrecarga adicional en comparación con otras técnicas que podrían construir directamente el número convertido.
  + **Operaciones adicionales**: Cada residuo se almacena primero en la pila, y luego se recorre la pila para generar el resultado final, lo que puede introducir un pequeño costo adicional en el número de operaciones, aunque este es generalmente insignificante.

**Comparación con otros métodos (sin pila):**

* **Método directo (sin pila)**: Se puede generar directamente el número convertido a medida que se realizan las divisiones, construyendo la cadena en lugar de almacenarla en una pila. Esto reduce la sobrecarga espacial, pero hace que el código sea menos modular y más difícil de entender.

En general, la **complejidad temporal y espacial** para ambos enfoques es la misma: **O(logb ​n)**, pero el uso de pilas proporciona mayor claridad y modularidad en el código, a costa de una pequeña sobrecarga espacial adicional.