Actividad 1

Nicolás A. Cevallos

TIC, Universidad Internacional del Ecuador

Bloque B: Estructura de Datos

Ing. Richard F. Armijo

Julio 7, 2024

Implementación del Triángulo de Sierpinski

Introducción

El propósito de esta actividad es implementar el Triángulo de Sierpinski utilizando técnicas de programación recursiva. Esta tarea no solo nos permite explorar conceptos avanzados de programación, sino que también nos introduce en el fascinante mundo de los fractales.

La recursividad es un concepto fundamental en programación que permite resolver problemas complejos dividiéndolos en instancias más pequeñas del mismo problema. Es particularmente útil en la creación de fractales como el Triángulo de Sierpinski, donde un patrón se repite a diferentes escalas.

Descripción Teórica

Este fascinante fractal, que debe su nombre al matemático polaco Wacław Sierpiński, fue descrito en 1915. Es un ejemplo notable de conjunto con autosemejanza exacta, en el que el patrón se repite idénticamente a diferentes escalas. Su construcción comienza con un triángulo equilátero sólido, que se divide en cuatro triángulos congruentes más pequeños conectando los puntos medios de cada lado. El triángulo central resultante se elimina, dejando tres triángulos sólidos. Este proceso se repite recursivamente para cada triángulo restante, teóricamente hasta el infinito, aunque en la práctica se detiene en un nivel predeterminado.

La naturaleza recursiva de esta figura geométrica se refleja en sus intrigantes propiedades matemáticas. Su dimensión fractal es de aproximadamente 1,585, lo que indica que es más "densa" que una línea unidimensional, sin llegar a llenar por completo un plano bidimensional. Curiosamente, su área tiende a cero a medida que aumentan las iteraciones, mientras que su perímetro tiende a infinito, creando una fascinante paradoja.

Este fractal tiene numerosas aplicaciones y conexiones con otros campos. En la teoría del caos, el atractor del juego del caos forma esta misma estructura. En teoría de números, el patrón de números impares del Triángulo de Pascal se asemeja a esta figura. Además, en la naturaleza aparecen estructuras similares, como en ciertos tipos de brócoli romanesco o en algunos cristales. Estas conexiones interdisciplinarias lo convierten no sólo en un objeto matemático cautivador, sino también en una valiosa herramienta para comprender patrones complejos en diversos campos científicos.

Proceso de Implementación

La implementación del Triángulo de Sierpinski se realizó utilizando Python, aprovechando la biblioteca matplotlib para la visualización gráfica. El corazón de la implementación reside en una función recursiva llamada 'sierpinski', que encapsula la lógica fundamental del fractal. Esta función toma como parámetros los tres puntos que definen el triángulo actual y el grado de recursividad, lo que permite un control preciso de la complejidad del fractal generado. La elección de un enfoque recursivo no es arbitraria; refleja directamente la naturaleza autosimilar del triángulo de Sierpinski, donde cada subtriángulo es una versión reducida del conjunto. Dentro de la función, se implementa una lógica condicional crucial: si el grado de recursividad es mayor que cero, la función calcula los puntos medios de los lados del triángulo actual y se llama a sí misma recursivamente para cada uno de los tres subtriángulos resultantes, disminuyendo el grado en cada llamada. Este proceso de subdivisión y recursión continúa hasta que se alcanza el caso base, en el que el grado es cero. En este punto, matplotlib se utiliza para dibujar el triángulo más pequeño, contribuyendo a la construcción visual del fractal completo.

La implementación también incluye una función auxiliar 'get_midpoint' para calcular los puntos medios, favoreciendo la modularidad y la legibilidad del código. Además, se han

incorporado consideraciones prácticas, como la configuración inicial de los puntos del triángulo base y la definición del grado máximo de recursión, permitiendo ajustar fácilmente la complejidad y el detalle del fractal generado. Finalmente, se utilizan funciones de matplotlib para configurar y guardar la figura resultante, asegurando una visualización clara y de alta calidad del Triángulo de Sierpinski. Esta implementación no sólo demuestra la elegancia de la recursividad en programación, sino que también ilustra cómo conceptos matemáticos complejos pueden traducirse eficientemente en código, proporcionando una potente herramienta para la exploración y visualización de fractales.

```
import matplotlib.pyplot as plt
import numpy as np

def sierpinski(points, degree):
    def get_midpoint(p1, p2):
        return ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2)

    if degree > 0:
        p1, p2, p3 = points

        # Calcula los puntos medios
        m1 = get_midpoint(p1, p2)
        m2 = get_midpoint(p2, p3)
        m3 = get_midpoint(p3, p1)

        # Llama recursivamente para cada subtriángulo
        sierpinski([p1, m1, m3], degree-1)
        sierpinski([m1, p2, m2], degree-1)
        sierpinski([m3, m2, p3], degree-1)
    else:
        # Dibuja el triángulo
        plt.fill(*zip(*points), "k")

# Configuración inicial
points = np.array([(0, 0), (0.5, np.sqrt(3)/2), (1, 0)])
degree = 6

# Crear la figura
plt.figure(figsize=(10, 10))
sierpinski(points, degree)
plt.axis('off')
plt.tight_layout()
```

```
plt.savefig('sierpinski_triangle.png', dpi=300, bbox_inches='tight')
plt.show()
```

Pruebas realizadas

Se realizaron pruebas a distintos niveles de profundidad para evaluar el rendimiento y la precisión visual de nuestra aplicación del triángulo de Sierpinski. En concreto, se probaron los siguientes niveles de profundidad: 0, 1, 3, 3, 6 y 8.

Nivel 0: Este nivel produce un triángulo sólido, que es el caso base de nuestro algoritmo recursivo.

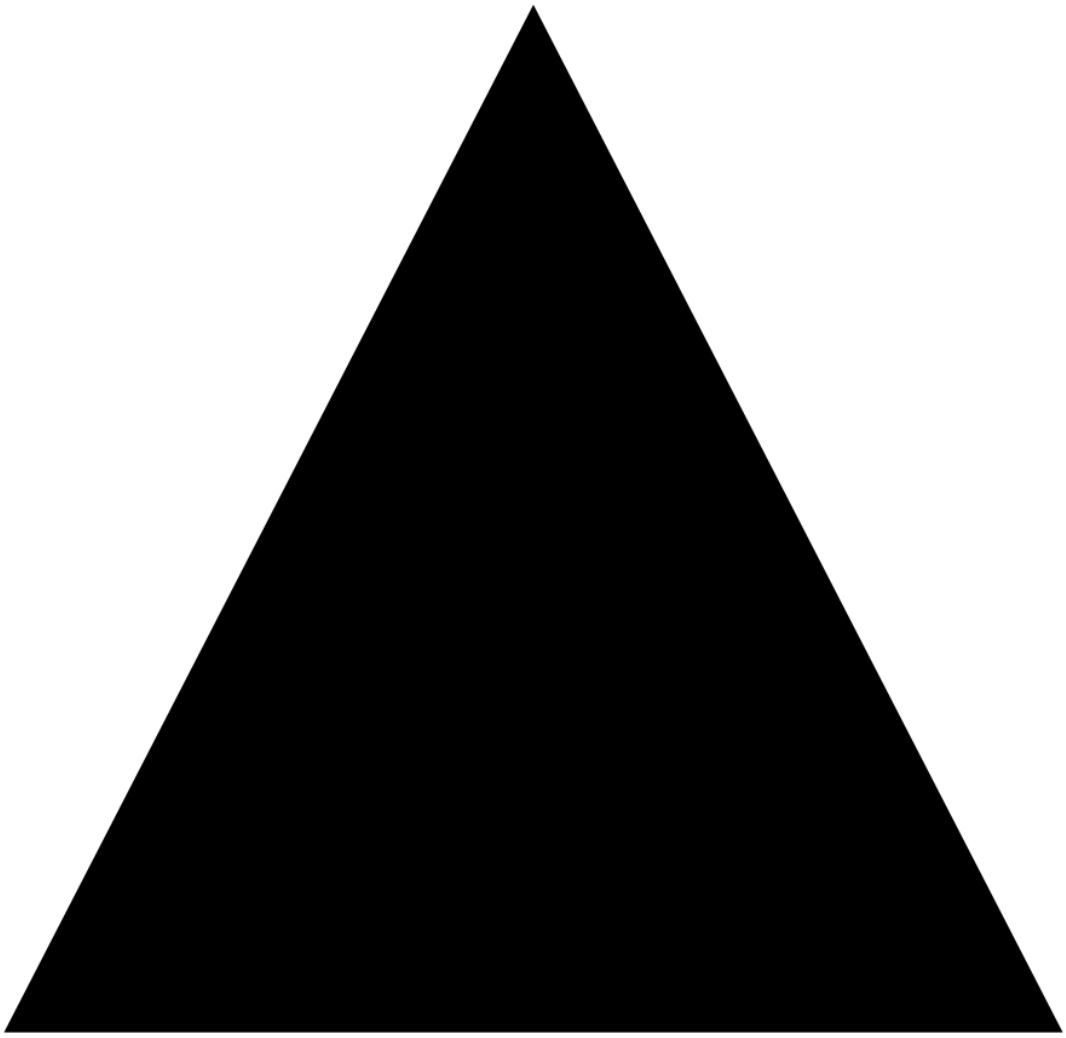
Nivel 1: Muestra la primera iteración del fractal, con un triángulo invertido en el centro.

Nivel 3: Empieza a mostrar la estructura fractal característica del Triángulo de Sierpinski.

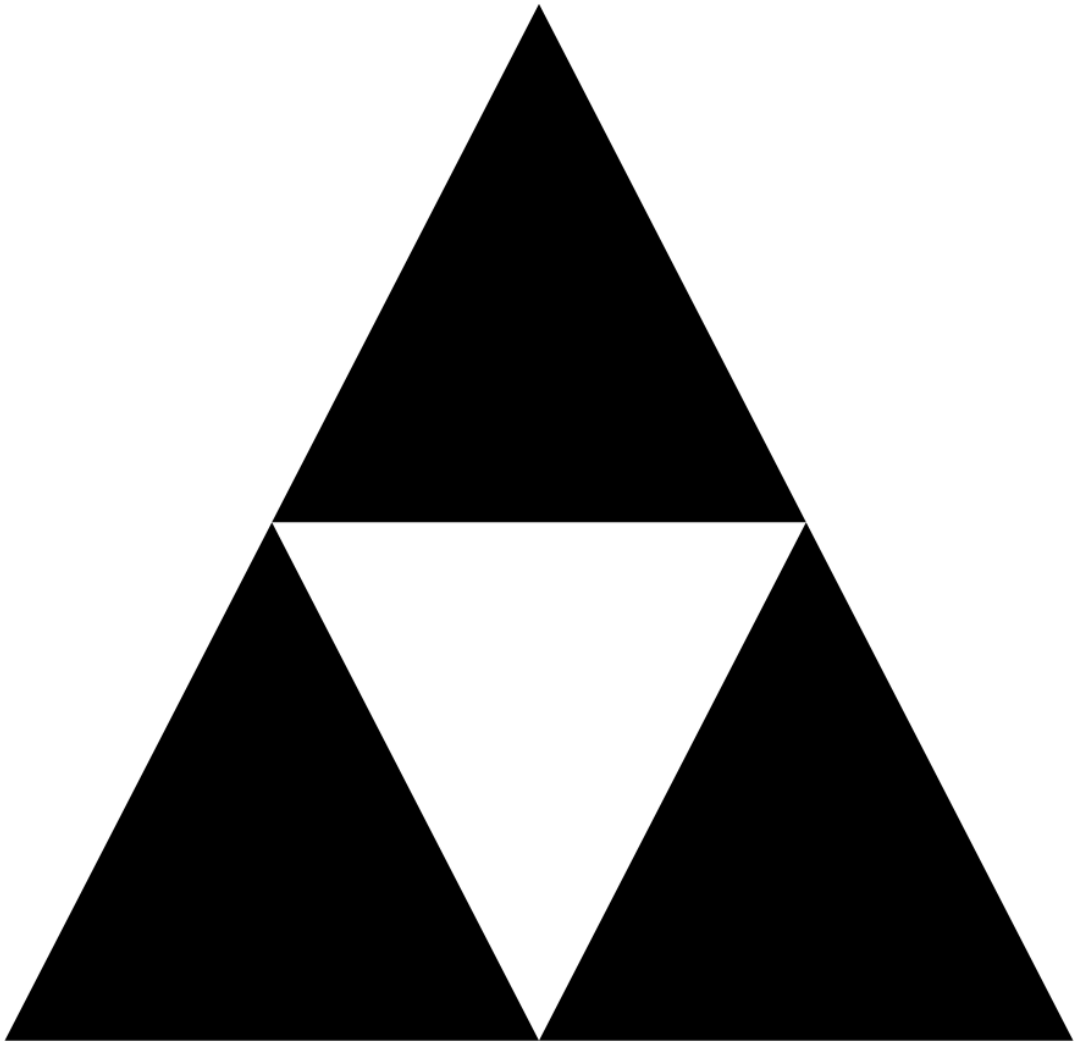
Nivel 6: Proporciona un buen equilibrio entre detalle y tiempo de ejecución, siendo nuestro ajuste por defecto.

Nivel 8: Proporciona un alto nivel de detalle, pero con un tiempo de ejecución notablemente mayor.

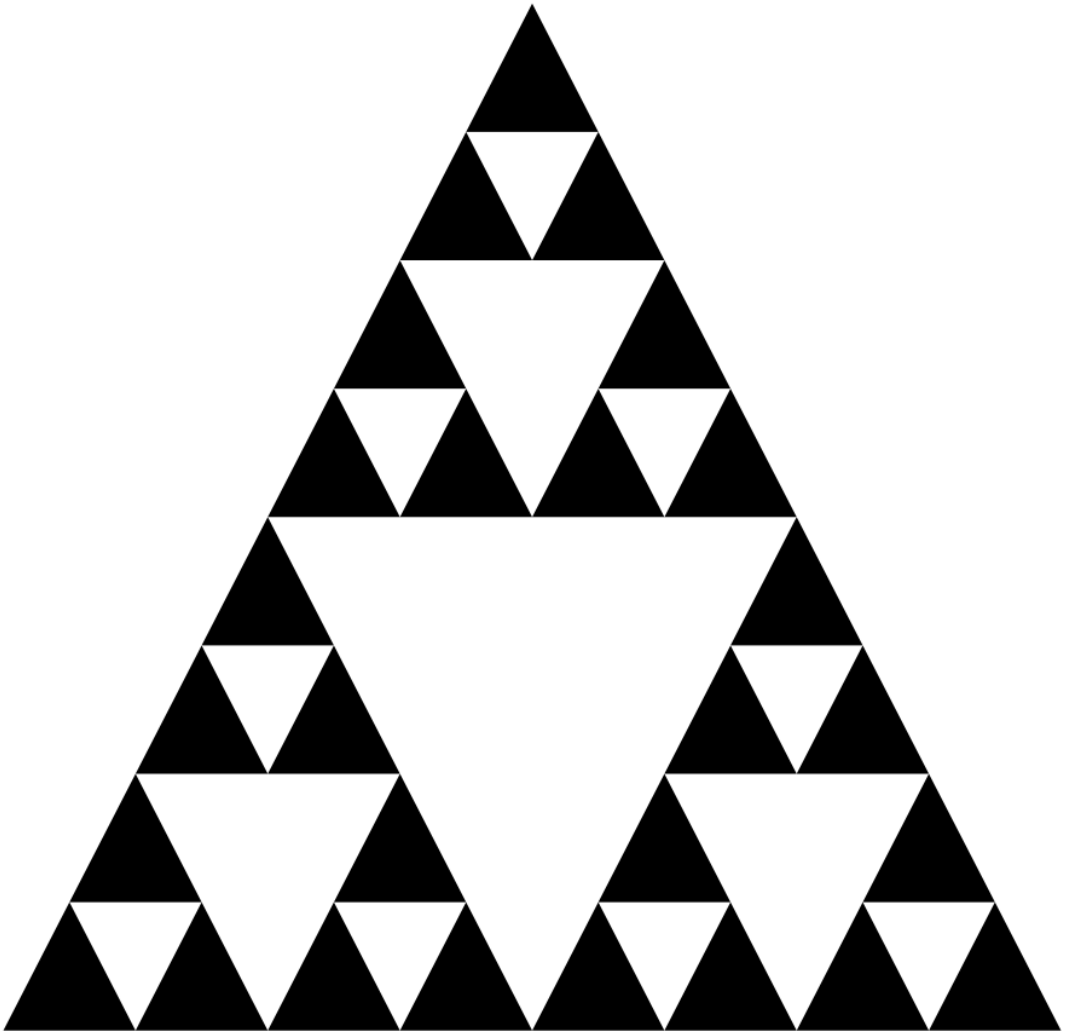
Triángulo de Sierpinski - Nivel 0



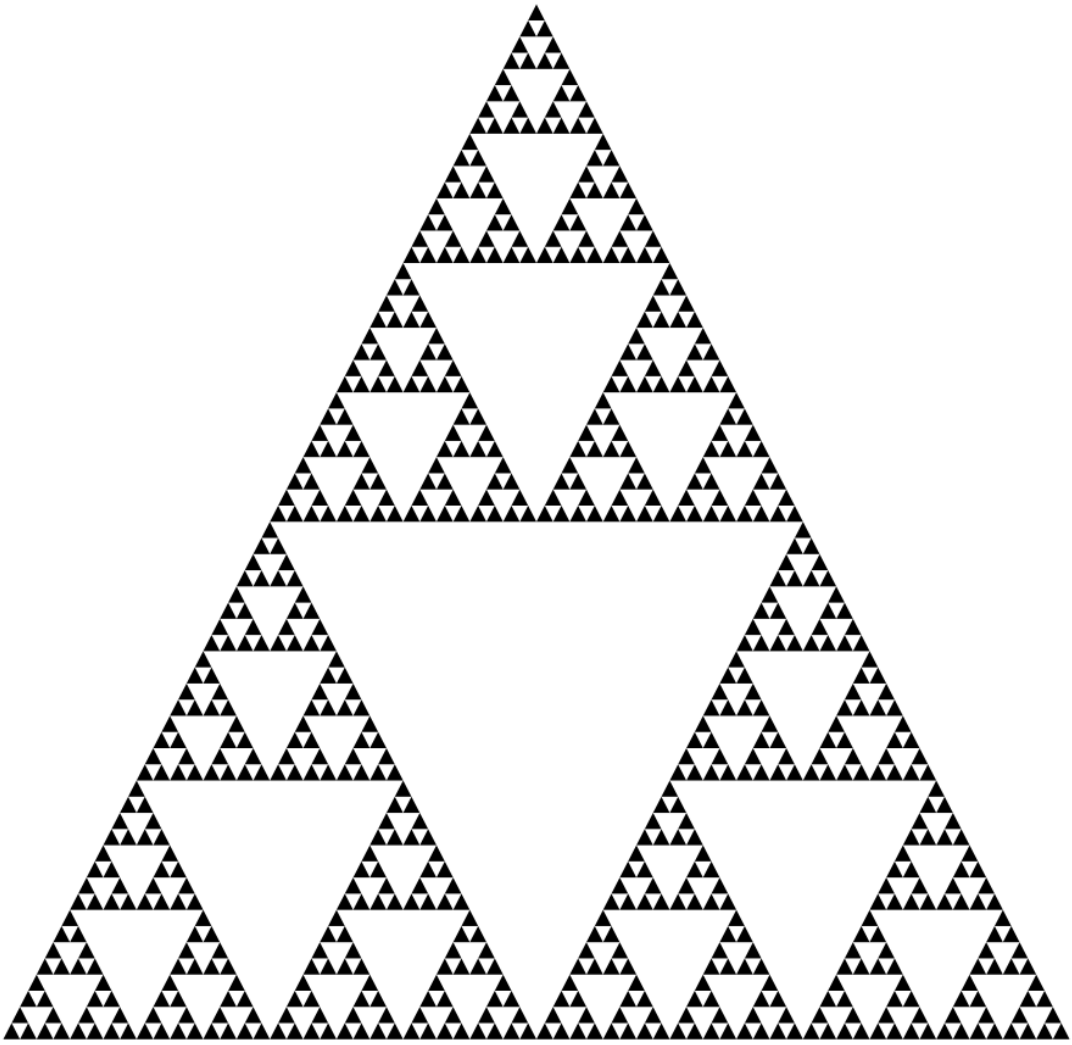
Triángulo de Sierpinski - Nivel 1

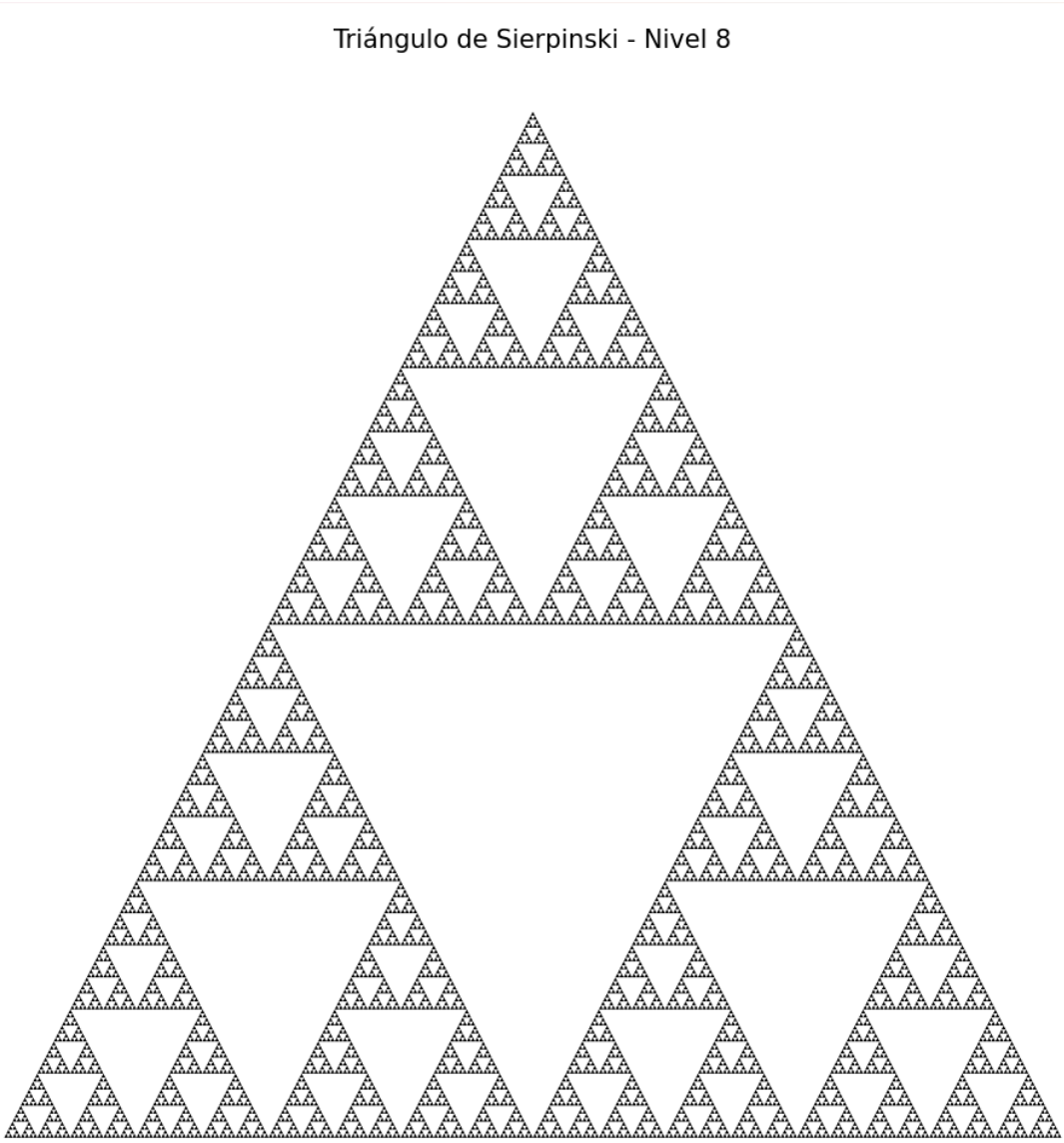


Triángulo de Sierpinski - Nivel 3



Triángulo de Sierpinski - Nivel 6





Observamos que a medida que aumenta el nivel de profundidad, el patrón fractal se vuelve más detallado y complejo. Sin embargo, también observamos que el tiempo de ejecución aumenta significativamente para niveles superiores a 6, debido a la naturaleza exponencial de la recursividad.

Conclusiones

La implementación del Triángulo de Sierpinski mediante programación recursiva ha demostrado ser un valioso ejercicio que ilustra varios conceptos clave de la informática y las matemáticas.

Puntos clave del informe:

- La recursión es una técnica poderosa para implementar estructuras fractales como el Triángulo de Sierpinski.
- La biblioteca matplotlib de Python proporciona herramientas eficaces para visualizar estructuras matemáticas complejas.
- El nivel de profundidad de la recursión tiene un impacto significativo tanto en el detalle visual como en el tiempo de ejecución del programa.

Durante la implementación, nos enfrentamos a algunas dificultades:

- Manejo de la recursividad: inicialmente, fue un reto conceptualizar cómo la función recursiva debía llamarse a sí misma para generar los subtriángulos. Esto se resolvió mediante un diseño cuidadoso de la función "sierpinski" y una comprensión clara de los casos base y recursivo.
- Rendimiento: Para niveles de profundidad elevados, el programa se volvía lento debido a la naturaleza exponencial de la recursividad. Para solucionar este problema, limitamos el nivel máximo de profundidad a 8 en nuestras pruebas y recomendamos un nivel de 6 para lograr un equilibrio óptimo entre detalle y rendimiento.
- Precisión del dibujo: Para garantizar que los triángulos se dibujaran con precisión, fue necesaria una cuidadosa implementación de la función "get_midpoint". El uso de operaciones aritméticas precisas fue crucial para evitar errores acumulativos.

En conclusión, este proyecto ha supuesto una exploración exhaustiva de la confluencia entre matemáticas, computación y visualización, profundizando

significativamente en nuestra comprensión de la recursividad, los fractales y la complejidad computacional. La aplicación ha establecido una base sólida para futuras investigaciones en estos campos, demostrando la eficacia de la programación como herramienta para la exploración y representación de conceptos matemáticos complejos. Las habilidades desarrolladas y los conocimientos adquiridos durante este proceso serán de gran relevancia en futuros estudios y aplicaciones en informática y disciplinas afines.