Instituto Politécnico Nacional

Escuela Superior de Cómputo

Semestre 2023-1

PRACTICA 4

Grupo: 3CV11

Materia: Análisis de Algoritmos

Alumnos:

Isaac Sánchez Verdiguel

isanchezv1603@alumno.ipn.mx

Axel Treviño Palacios

atrevinop1500@alumno.ipn.mx

Maestro:

Benjamin Luna Benoso

14 Diciembre 2022

Índice general

Índice de figuras

1 | Introducción

1.1. Resumen

La práctica consta de implementar un algoritmo que pueda descomponer en matrices una imagen y con estas matrices rotar la imagen. La técnica a utilizar es **Divide y Vencerás** que será empleada en el análisis del algoritmo. El algoritmo fue codificado en el lenguaje de programación **Python** ejecutado en un entorno de desarrollo **Linux**.
Palabras Clave: Python, Algoritmo, Dividir, Matriz.

1.2. Introducción

Los algoritmos son una parte fundamental de la ciencia de la computación, ya que estos al ser computables pueden dar solución o una idea más concreta acerca de la solución de un problema.
Un algoritmo no siempre dará una solución correcta, lo cual jamás será malo, porque esto nos ayudará a poder minimizar su radio de error. Una característica casi obligatoria para el buen funcionamiento de un algoritmo es su **rendimiento y eficacia**. El rendimiento adecuado se encuentra en la solución más rápida y menos costosa [Cormen, 2009].
Una forma de poder atacar algoritmos complejos es por la técnica de Divide y Vencerás. Consiste en resolver la complejidad de un algoritmo mediante una resolución recursiva dividiendo el problema en dos o más subproblemas, repitiendo ese paso hasta que los subproblemas se vuelvan tan sencillos para que se puedan resolver directamente [Wikipedia, 2022].
Dicha técnica se utilizará para encontrar la complejidad del algoritmo realizado en esta práctica. El algoritmo de Rotación de Imagen. Una imagen puede ser descompuesta píxel a píxel, esto gracias a que un píxel RGB posee una matriz de valores (R,G,B), el propósito es almacenar y rotar estas matrices haciendo inspiración del algoritmo de Strassen. Dentro de la practica también se anexan ejercicios que resuelven problemas de complejidad.

2 | Desarrollo

2.1. Conceptos Básicos

La **complejidad temporal**, dentro del análisis de algoritmos, es el número de operaciones que ejecuta un algoritmo en cierto tiempo. Su denotación es T(n) y puede ser analizada mediante dos tipos de análisis:

- Análisis de priori: entrega una función que muestra el tiempo de cálculo de un algoritmo.
- Análisis a posteriori: es la prueba en tiempo real del algoritmo, midiendo su costo mediante valores de entrada.

El análisis de complejidad temporal define que un algoritmo alcanza su máximo potencial cuando los valores de entrada son mayores al tiempo estimado de ejecución, siendo que es factible poder completar sus ejecuciones en menor tiempo posible.
Algoritmo Strassen Es un algoritmo para la multiplicación de matrices. Es más rápido que el algoritmo estándar de multiplicación de matrices y es útil en la práctica para matrices grandes, pero sería más lento que los algoritmos más rápidos conocidos para matrices extremadamente grandes. Calcula el producto de dos matrices cuadradas de tamaño n [FrWIKI, 2022].
Teorema Maestro Es una técnica para las recurrencias de divide y vencerás proporciona un análisis asintótico para las relaciones de recurrencia de tipos que ocurren en el análisis de muchos algoritmos de divide y vencerás [Hmong, 2022].

2.2. Algoritmo Rotación Imagen

2.2.1. Algoritmo Rotación Imagen

Pseudocódigo Algoritmo Rotación Imagen

El algoritmo consta de rota una imagen 90 grados hacia la izquierda partiendo de dividir la imagen en sectores de pixeles, obteniendo sus matrices RGB. Aplicando Divide y Vencerás se parte el problema en pequeños subproblemas para rotar la imagen. El proceso del algoritmo es obtener el *width y height* (el ancho y largo respectivamente) del sector, hace un análisis, si ambas medidas son iguales a 1 hace la rotación de pixeles sin entrar en el proceso recursivo. En caso contrario, de no existir cuadrantes en la imagen, divide en 4 cuadrantes que serán rotados usando recursividad, la función recibe como parametro el sector y la rotación, esto en cada paso de la recursividad se realiza hasta que el sector sea lo suficientemente pequeño para rotar cada pixel de cada cuadrante. Al final intercambia cuadrantes para hacer el cambio de rotación, finalizando con la unión de los cuadrantes en la imagen final. En la figura 2.1 se muestra un ejemplo del resultado esperado.

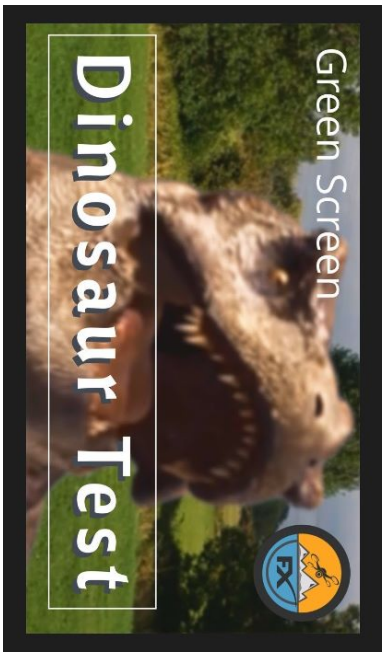

Figura 2.1: Figuras de Ejemplo

Algorithm 1: Rotación Imagen

Result: *imagenRotada*

```
if width == 1 then
    for i ← 0 to ceil(n/2) + 1 do
        invierte pixeles;
else
    continue;
if height == 1 then
    for i ← 0 to ceil(n/2) + 1 do
        invierte pixeles;
else
    image00 = crop((0, 0, floor(width / 2), floor(height / 2)));
    image01 = crop((floor(width / 2), 0, width, floor(height / 2)));
    image10 = crop((0, floor(height / 2), floor(width / 2), height));
    image11 = crop((floor(width / 2), floor(height / 2), width, height));
    image00 = rotate(image00, clkwise);
    image01 = rotate(image01, clkwise);
    image10 = rotate(image10, clkwise);
    image11 = rotate(image11, clkwise);
    if clkwise then
        image00 = image10;
        image10 = image11;
        image11 = image01;
        image01 = aux;
    else
        image00 = image01;
        image01 = image11;
        image11 = image10;
        image10 = aux;
result.paste(image00, box=(0, 0));
result.paste(image01, box=(image00.size[0], 0));
result.paste(image10, box=(0, image00.size[1]));
result.paste(image11, box=(image00.size[0], image00.size[1]));
return result;
```

3 | Experimentación y Resultados

Aquí se presentaran los resultados del **Análisis a Priori y Posteriori** del algoritmo de rotación de imagen.

3.1. Algoritmo: Rotación Imagen

Rotación de Imagen consta de descomponer una imagen en su matriz RGB, donde se almacena toda la información de la imagen. Con esto se puede rotar el archivo haciendo uso de una función que invierta la posición de de los elementos de la matriz. El algoritmo se ejecutó de manera correcta, siendo capaz de rotar imágenes dividendo los sectores. En la figura 3.1 se muestra una de las imágenes puestas de tamaño 100x100 a prueba por el algoritmo.

Figura 3.1: Imagen Rotada por Algoritmo

3.1.1. Análisis a Priori

La figura 3.2 presenta el análisis a priori realizado sobre el pseudocódigo del algoritmo de Rotación de Imágenes. Concluyendo que el algoritmo presenta $T(n) = 4T(\frac{n}{2}) + \theta(n)$

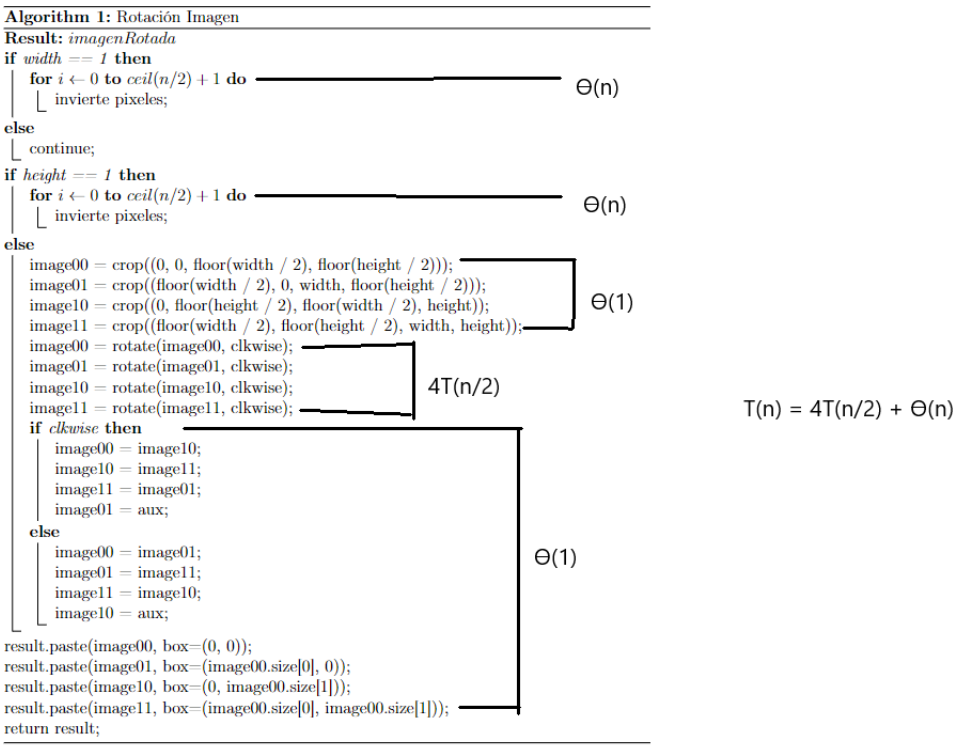

Figura 3.2: Análisis a Priori: Rotación Imagen

3.1.2. Desarrollo Análisis a Priori

Probando mediante el Teorema Maestro, resulta: Con $T(n) = 4T(\frac{n}{2}) + \theta(n)$ se tiene $a = 4, b = 2, f(n) = cn$
Por el caso (III) del Teorema Maestro se obtiene un orden de complejidad de:

$$\theta(n^{log_2 4})$$
$$\therefore T(n) \in \theta(n^2)$$

3.1.3. Análisis a Posteriori

En el análisis posteriori se verifica que el análisis a priori demostró que la complejidad del peor caso es $\theta(n^2)$. En la figura 3.3 se muestra la función que acota al peor caso junto con los puntos que demuestran la complejidad del algoritmo funcionando de manera aleatoria y el mejor caso en el eje de las x siendo constante.

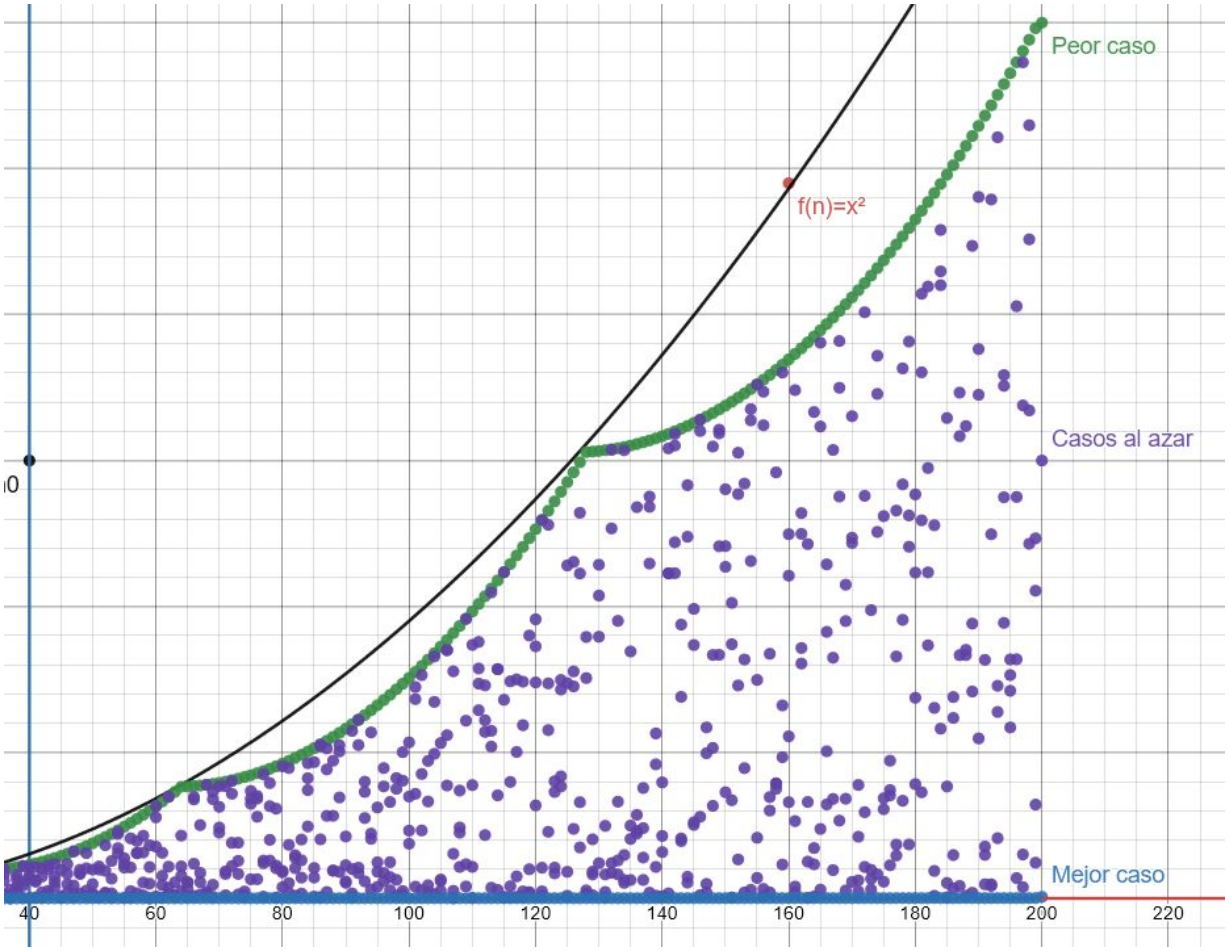

Figura 3.3: Análisis a Posteriori: Rotación Imagen

3.2. Pantallas de Ejecución del Algoritmo

Se muestra en la figura 3.4 la ejecución del algoritmo, haciendo uso de imágenes externas de diferente tamaños para medir el rendimiento tomando en cuenta diferentes imágenes de diferentes tamaños.

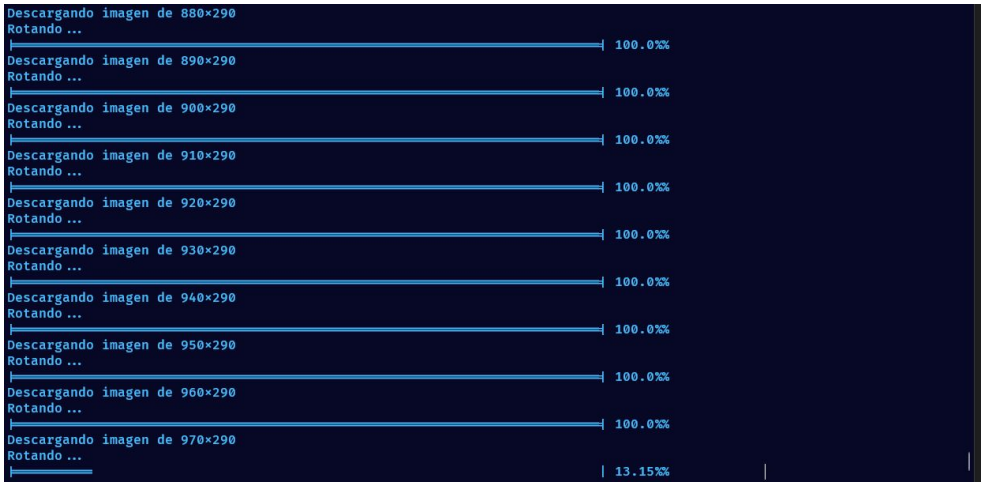

Figura 3.4: Ejecución de Rotación de Imagenes

4 | Conclusiones

4.1. Conclusiones Generales

La técnica de Divide y Vencerás se torno difícil para nosotros a la hora de aplicarla al algoritmo propuesto para esta práctica. Repasamos los temas y vimos que nuestro punto bajo era la aplicación de funciones que no habíamos trabajado en el lenguaje Python. Concluyendo que reforzar el tema de Divide y Vencerás no solamente ayuda al análisis de algoritmos sino que es un aprendizaje que se puede llevar a todos lados con su filosofía.

4.2. Isaac Sánchez - Conclusiones

Esta práctica me resultó confusa, ya que la codificación del algoritmo para cambiar de posición las matrices de la imagen jamás lo había visto. Graias al aprendizaje de divide y vencerás pude entrar en detalle para poder lograrlo. Presentamos problemáticas de tiempo para concretar los problemas anexos. De ahí en fuera, fue una práctica que reforzó mis conocimientos.

Figura 4.1: Isaac Sánchez

4.3. Axel Trevino - Conclusiones

Esta práctica es un buen ejemplo de los resultados de la recursividad adicionada al divide y vencerás, ya que cuando se piensa de manera normal, para rotar la imagen se tendrían que hacer dos cosas: cambiar los cuadrantes y rotarlos; pero al pensar que los cuatro cuadrantes igualmente son imágenes, sólo hay que aplicar el algoritmo ad infinitum y ya, todo se resuelve con el mismo proceso

Figura 4.2: Axel Treviño

5 | Anexo

5.1. Problemas y Ejercicios

En esta sección se aborda la solución de los problemas solicitados durante las clases previas a la práctica.

5.1.1. Ejercicio 1

Descripción: Probar mediante sustitución hacia atrás que $T(n) \in \Theta(n \log(n))$

$$T(n) \begin{cases} \Theta(1) si & n = 1 \\ 2T(\frac{n}{2}) + \Theta(n) \end{cases}$$

Resolviendo se tiene $n = 2^k$ entonces $T(2^k) = 2T(2^{k-1} + c2^k)$

$$\begin{gathered} T(2^k) = 2T(2^{k-1} + c2^k \\ T(2^k) = 2[2T(2^{k-2} + c2^{k-1}] + c2^k \\ T(2^k) = 2^2T(2^{k-2}) + 2c2^k \\ T(2^k) = 2^2[2T(2^{2k-3}) + c2^{k-2}] + 2c2^k \\ T(2^k) = 2^3T(2^{k-3}) + c2^k + 2c2^k \\ T(2^k) = 2^3T(2^{k-3}) + 3c2^k \\ T(i) = 2^iT(2^{k-i}) + ic2^k \end{gathered}$$

Teniendo que $k - 1 = 0$ y $k = 1$

$$\begin{gathered} = 2^kT(2^0 + k(2^k) \\ = 2^kc + kc2^k \\ = (c + kc)(2^k) \\ = (c + c\log_2 n)n \\ \therefore T(n) \in \theta(n \log n) \end{gathered}$$

5.1.2. Ejercicio 2

Descripción: Utilizando decremento por uno, pruebe que $T(n) \in O(n^2$
Teniendo que $T(n) = T(n-1) + c(n+1)$ resolvemos:

$$= T(0) + \sum_{j=1}^{n} f(j)$$

$$= T(0) + \sum_{j=1}^{n} c(j+1)$$

$$= T(0) + \sum_{j=1}^{n} c(2+3+4+...+(n+1)) = c(n+1)(n+2)$$

$$= cn^2 + c3n + 2c - 2c$$

$$= \frac{c(n^2+3n)^2}{2}$$

$$\therefore T(n) \in O(n^2)$$

5.1.3. Ejercicio 3

Descripción: Utilizando el teorema maestro, pruebe que $T(n) \in A = \Omega(n \log n)$
Teniendo que $T(n) = 2T(\frac{n}{2}) + \theta(n)$ es $= T(n) = 2T(\frac{n}{2}) + Cn$

Resolviendo se tiene que $a = 2, b = 2, f(n) = cn$

$$cn = \theta(n^{log_2 2})$$

$$\theta(n^{log_2 2} \log n) = \theta(n \log n)$$

$$\therefore T(n) \in \theta(n \log n)$$

5.1.4. Ejercicio 4

Descripción: Mediante Teorema Maestro, pruebe que $T(n) \in \theta(n^2)$
Con $T(n) = 4T(\frac{n}{2}) + \theta(n)$ se tiene $a = 4, b = 2, f(n) = cn$
Por el caso (III) del Teorema Maestro se obtiene:

$$\theta(n^{log_2 4})$$

$$\therefore T(n) \in \theta(n^2)$$

5.1.5. Ejercicio 5

Descripción: Mediante Teorema Maestro, pruebe que $T(n) \in \theta(n^{\log 3})$
Con $T(n) = 3T(\frac{n}{2}) + \theta(n)$ se tiene $a = 3, b = 2, f(n) = cn$
Por el caso (III) del Teorema Maestro se obtiene:

$$\theta(n^{log_2 3})$$
$$\therefore T(n) \in \theta(n^{log_2 3})$$

5.1.6. Ejercicio 6

Descripción: Mediante Teorema Maestro, pruebe que $T(n) \in \theta(n \log n)$
Con $T(n) = 2T(\frac{n}{2}) + \theta(n)$ se tiene $a = 2, b = 2, f(n) = cn$
Por el caso (II) del Teorema Maestro se obtiene:

$$\theta(n^{log_2 3})$$
$$\therefore T(n) \in \theta(n \log n)$$

Bibliografía

[Cormen, 2009] Cormen, T. H. (2009). *Introduction to Algorithms*. The MIT Press, London.

[FrWIKI, 2022] FrWIKI (2022). Algoritmo de strassen. https://es.frwiki.wiki/wiki/Algorithme_de_Strassen. [Online; accessed 14-December-2022].

[Hmong, 2022] Hmong (2022). Teorema maestro. https://hmong.es/wiki/Master_theorem_(analysis_of_algorithms). [Online; accessed 13-December-2022].

[Wikipedia, 2022] Wikipedia (2022). Algoritmo divide y vencerás. https://es.wikipedia.org/wiki/Algoritmo_divide_y_vencer%C3%A1s#:~:text=En%20las%20ciencias%20de%20la,de%20igual%20tipo%20o%20similar. [Online; accessed 11-December-2022].