

# ALGORITMOS PARA PROCESADO DE IMÁGENES PGM

Programación Avanzada

Grupo Beta

Integrantes:

- Di Nicco Luis
- Antonioli Iván



# INTRODUCCIÓN

- Vamos a explicar el algoritmo de inversión de color aplicado a imágenes PGM.
- Vamos a explicar las dos posibles formas de aplicar la inversión.
- Analizaremos la complejidad computacional del algoritmo

The image features a light teal background with a large, dark teal horizontal bar across the middle. The word "DESARROLLO" is written in white, bold, uppercase letters within this bar. In the top right corner, there are two overlapping teal rectangles. In the bottom left corner, there are two overlapping teal triangles pointing towards the center.

# DESARROLLO

# ¿CÓMO FUNCIONA EL FORMATO PGM?

- Almacena imágenes en escala de grises.
- Cada píxel de la imagen se representa por un valor de brillo, donde 0 es negro y el valor máximo es blanco.



# TIPOS DE FORMATO

# TIPOS DE FORMATO PGM

## P2

- Formato ASCII
- Archivos mas grandes
- Mas facil de leer

## P5

- Formato binario
- Archivos mas compactos
- No se puede leer directamente

Ambos poseen un encabezado que almacena la informacion de la imagen en ASCII.

P5

274 184

255

Formato del archivo pgm

Ancho de la foto, Alto de la foto (en pixeles)

Valor maximo que puede tomar un pixel

# ¿CÓMO FUNCIONA EL ALGORITMO DE INVERSIÓN DE COLORES?

Algoritmo inversión de Colores mediante complemento con valor maximo

El valor del pixel invertido se calcula en base al valor maximo que puede tomar un pixel

Valor Invertido = Valor máximo – Valor Actual Pixel

Este proceso se aplica a cada píxel de la imagen para obtener la imagen con los colores invertidos.

# EJEMPLO:

Si el valor máximo es 255 y un píxel tiene un valor 100, el valor invertido será:

$$255 - 100 = 155$$

## CARACTERÍSTICAS DE LA INVERSIÓN:

- **UN PÍXEL NEGRO (VALOR 0) SE CONVIERTE EN BLANCO (VALOR 255).**
- **UN PÍXEL BLANCO (VALOR 255) SE CONVIERTE EN NEGRO (VALOR 0).**
- **UN PÍXEL CON UN VALOR INTERMEDIO SE CONVIERTE EN SU VALOR COMPLEMENTARIO EN LA ESCALA DE GRISES.**



# EJEMPLO:

Matriz Original:

<b>10</b>	<b>50</b>	<b>100</b>
<b>150</b>	<b>200</b>	<b>250</b>
<b>255</b>	<b>220</b>	<b>120</b>

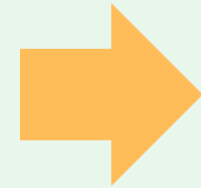
\*Valor máximo 255

Para realizar la inversion se debe recorrer cada elemento de la matriz de pixeles e ir aplicando la inversion a cada uno. Como resultado se obtiene una matriz que representa la imagen con los colores invertidos.


# EJEMPLO:

Matriz Original:

$i = 0$



$j = 0$



10	50	100
150	200	250
255	220	0

Valor actual del pixel = 10

Valor Maximo = 255

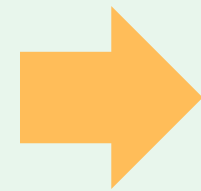
Valor máximo - Valor Actual Pixel = Valor Invertido

$$255 - 10 = 245$$

# EJEMPLO:

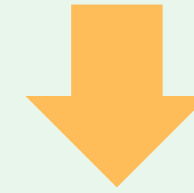
Matriz Original:

$i = 0$



<del>10</del>	50	100
150	200	250
255	220	0

$j = 1$



Valor actual del pixel = 50

Valor Maximo = 255

Valor máximo - Valor Actual Pixel = Valor Invertido

$$255 - 50 = 205$$

# EJEMPLO:

Matriz Original

<b>10</b>	<b>50</b>	<b>100</b>
<b>150</b>	<b>200</b>	<b>250</b>
<b>255</b>	<b>220</b>	<b>120</b>

Matriz Invertida

<b>245</b>	<b>205</b>	<b>155</b>
<b>105</b>	<b>55</b>	<b>5</b>
<b>0</b>	<b>35</b>	<b>135</b>

# Algoritmo inversión de Colores mediante complemento al valor maximo real

- El valor del pixel invertido se calcula en base al valor maximo que puede tomar un pixel

**Valor Escalado Pixel = (Valor máximo Archivo/Valor Maximo Real)\* Valor Actual Pixel**

**Valor Invertido = Valor máximo Archivo – Valor Escalado Pixel**

Si los valores de los pixeles de la imagen original son muy chicos, al aplicar el algoritmo anterior no se podria notar ningun cambio al realizar la inversion. Con esta alternativa del algoritmo permite poder diferenciar las pequeñas variaciones entre los pixeles.

# EJEMPLO:

Matriz Original:

1	2	3
4	5	6
7	8	9

\*Valor máximo 255

\*Valor máximo real 9

Para realizar la inversion se debe recorrer cada elemento de la matriz de pixeles e ir aplicando la inversion a cada uno. Como resultado se obtiene una matriz que representa la imagen con los colores invertidos en base al maximo real.

# EJEMPLO:

Matriz Original:

$j = 0$

$i = 0$

1	2	3
4	5	6
7	8	9

Valor actual del pixel = 1

Valor Maximo Real = 9

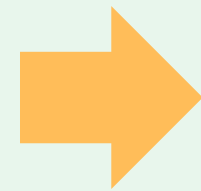
**VALOR ESCALADO PIXEL =  $(255 / 9) * 1 = 28,3$**

**VALOR INVERTIDO =  $255 - 28 = 227$**

# EJEMPLO:

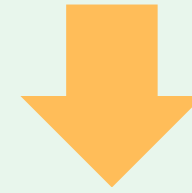
Matriz Original:

$i = 0$



<del>1</del>	2	3
4	5	6
7	8	9

$j = 1$



Valor actual del pixel = 2

Valor Maximo Real = 9

**VALOR ESCALADO PIXEL =  $(255 / 9) * 2 = 56,6 \approx 56$**

**VALOR INVERTIDO =  $255 - 56 = 199$**



# EJEMPLO:

Matriz Original

1	2	3
4	5	6
7	8	9

Matriz Invertida

227	199	171
143	115	87
59	31	3

# ANALISIS DE COMPLEJIDAD COMPUTACIONAL

# ANALISIS DE COMPLEJIDAD COMPUTACIONAL

```
58- public void invertirImagen() {  
59     for (int i = 0; i < getCantidadFilas(); i++) {  
60         for (int j = 0; j < getCantidadColumnas(); j++) {  
61             setImagen(i, j, maximoValor - getImagen(i, j));  
62         }  
63     }  
64 }  
65 }
```

$O(N*M)$

La complejidad computacional del metodo InvertirImagen es  $O(N*M)$  siendo N la cantidad de filas y M la cantidad de columnas de la imagen.

La formula para calcular el doble ciclo for anidado es  $T(N) = N * M * T(s)$   
Siendo S una operacion elemental que tiene una complejidad de  $O(1)$

# ANALISIS DE COMPLEJIDAD COMPUTACIONAL

```
5 public static void main(String[] args) {  
6  
7     ImagenPGM imagen= LeerPGM.LeerCabeceraPGM("Foto.pgm");    O(1)  
8  
9     LeerPGM.LeerImagenPGM(imagen);                            O(N * M)  
10  
11     imagen.invertirImagen();                                   O(N * M)  
12  
13     EscribirPGM.escribirImagenPGM(imagen, "FotoSalida.pgm");  O(N * M)  
14 }  
15 }
```

Finalmente la complejidad computacional de este algoritmo es:

$$O(1) + O(N * M) + O(N * M) + O(N * M)$$

Lo cual da como resultado una complejidad de  $O(N * M)$ .

Si consideramos a N como el máximo entre el ancho y el alto de la imagen, la complejidad computacional del algoritmo sería  $O(N^2)$ .

# PASO A DEMOSTRACIÓN

**MUCHAS GRACIAS**