Laboratorio 1

Competencias a desarrollar

En este laboratorio el estudiante pondrá en práctica los conceptos básicos de procesamiento de imágenes aprendidos hasta ahora (visualización, almacenamiento, operadores morfológicos, técnicas de binarización, kernel, suavizado). Podrá utilizar como base el código ya desarrollado en el módulo `plot.py` donde lo considere necesario e implementará una variante particular del algoritmo **Connected components labeling** propuesta por *Lifeng He; Yuyan Chao; Suzuki, K* [1]

Instrucciones

Desarrolle la función que se describe en cada inciso. Entregue su trabajo en un archivo llamado `laboratorio_1.py` por medio de la actividad correspondiente en **MiU** antes del lunes 11 de septiembre a las 13:00 horas. El archivo debe contener su nombre comentado en la primera línea para facilitar la identificación.

Asegúrese que el código de su solución se ejecute utilizando el siguiente comando desde una terminal: `laboratorio_1.py input_image output_filename`

Su código se calificará ejecutando: `laboratorio_1.py fprint3.pgm fprint3_ccl.png`

Referencias útiles

Referencia sobre Connected component labeling:

- https://en.wikipedia.org/wiki/Connected-component_labeling (Contiene descripción del algoritmo a implementar)
- https://en.wikipedia.org/wiki/Disjoint-set_data_structure (**Importante**)
- http://aishack.in/tutorials/labelling-connected-components-example/

PARTE I: Funciones a desarrollar

1. img_pad

a. Escriba una función con firma ***def*** `imgpad(img, r)` que haga un padding de ceros de `r` pixeles alrededor de una imagen binaria `img` y devuelva dicha imagen.

Comportamiento esperado de la `imgpad(img, 1)`:

Entrada:

```
[[  0   0   0   0   0   0 255   0]
 [  0   0   0 255 255 255   0   0]
 [  0   0   0 255 255 255   0   0]
 [  0   0   0 255 255 255   0   0]
 [  0   0   0 255 255 255   0   0]
 [  0 255   0   0 255   0   0   0]
 [  0 255   0   0   0   0   0 255]
 [  0 255   0   0   0   0 255 255]]
```

Salida:

```
[[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,   0,   0,   0, 255,   0,   0],
 [  0,   0,   0,   0, 255, 255, 255,   0,   0,   0],
 [  0,   0,   0,   0, 255, 255, 255,   0,   0,   0],
 [  0,   0,   0,   0, 255, 255, 255,   0,   0,   0],
 [  0,   0,   0,   0, 255, 255, 255,   0,   0,   0],
 [  0,   0, 255,   0,   0, 255,   0,   0,   0,   0],
 [  0,   0, 255,   0,   0,   0,   0,   0, 255,   0],
 [  0,   0, 255,   0,   0,   0,   0, 255, 255,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0]]
```

[1]. Lifeng He; Yuyan Chao; Suzuki, K. (1 May 2008). "A Run-Based Two-Scan Labeling Algorithm". *IEEE Transactions on Image Processing*. **17** (5): 749–756

2. connected_c

a. Escriba una función con la firma `def connected_c(img)` que etiquete los componentes conectados de la imagen binaria `img`. La variante **two-pass** del algoritmo a implementar está descrita en el sitio de ***Connected Componet labeling*** (CCL)de wikipedia presentado en las referencias y se copia a continuación:

```
On the first pass:

    1. Iterate through each element of the data by column, then by row (Raster Scanning)
    2. If the element is not the background
            1. Get the neighboring elements of the current element
            2. If there are no neighbors, uniquely label the current element and continue
            3. Otherwise, find the neighbor with the smallest label and assign it to the current
                element
            4. Store the equivalence between neighboring labels

On the second pass:

    1. Iterate through each element of the data by column, then by row
    2. If the element is not the background
            1. Relabel the element with the lowest equivalent label
```

3. labelview

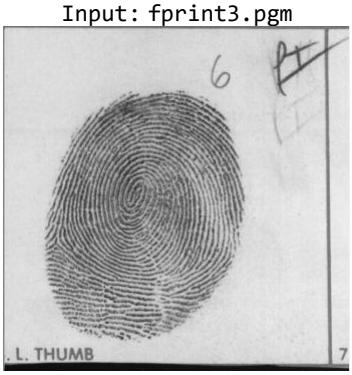
1. Escriba una función con la firma `def labelview(labels)` que realiza las siguientes operaciones:
    a. Asigna un color a cada etiqueta de la matriz `labels` de tipo `np.array` (imagen) que ha sido etiquetada por un algoritmo **CCL**.
    b. Hace uso de la función `plot.imgview` del módulo `plot.py` para visualizar dichos colores asignados.

PARTE II: Análisis de huellas dactilares

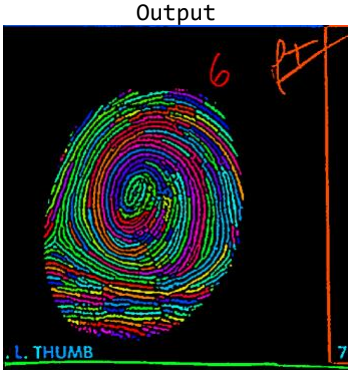
Utilizando las funciones mencionadas previamente y aplicando las técnicas expuestas en clase que se consideren necesarias, genere una imagen en donde se han asignado colores aleatorios, con la función `labelview()`, a los segmentos conectados detectados por la función `connected_c()` de la huella dactilar `fprint3.pgm` (adjunta a este documento).

Los resultados esperados son:

Input: fprint3.pgm



Output



NOTA: Los colores pueden variar en su solución.

Punteo

15% Implementación y funcionamiento adecuado de `pad()`
15% Implementación y funcionamiento adecuado de `labelview()`
25% Implementación de CCL en la función `connected_c()`
40% Generación correcta del output esperado al ejecutar `laboratorio_1.py fprint3.pgm fprint3_ccl.png`
5% Docstrings adecuados y con estilo correcto en cada función.