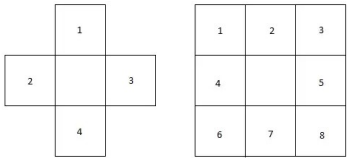
# LABELING REGIONS IN IMAGES

ATANDA ABDULLAHI ADEWALE - SCENE SEGMENTATION AND INTERPRETATION
MSc. Computer Vision, University of Bourgogne

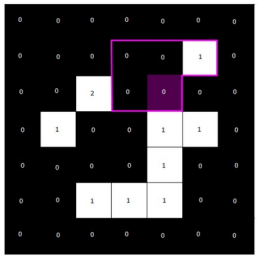
**IMAGE REGION LABELING:** used in blob extraction, is an image processing technique for finding connected components in a binary image, where connected components are defined as groups of pixels that are all connected to each other. The output is typically an image with the same size as the input image, where each connected component is labeled with a unique integer value and colored differently.

**DIFFERENT CONNECTIVITY**

There are two common ways of defining whether or not a component is connected. One is stating that a pixel only has 4 neighbours (sometimes called 4-connectivity). The other is stating that a pixel has 8 neighbours.




Using this mask



In this case, we are considering the 8 connectivity meaning diagonal connection inclusive.
Where mask W=(i-1,j-1),(i-1,j),(i-1,j+1),(i,j-i) for all i,j in binary image B(i,j)

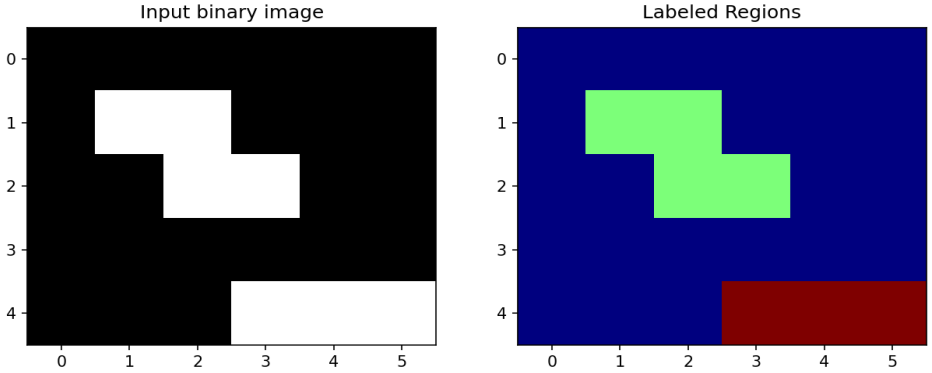
**METHOD**

- Import or generate a binary image
- Add zero padding around the borders to the image array: make computation easier, the mask consider at each pixel can stay of consistent size i.e. only pixel left of it, and the three above it.
- Consider 8-connectivity of B(i,j), Define mask configuration

```
if x > 0:
    neighbors.append(labels[y, x-1])
if y > 0:
    neighbors.append(labels[y-1, x])
if y > 0 and x < width-1:
    neighbors.append(labels[y-1, x+1])
if y > 0 and x > 0:
    neighbors.append(labels[y-1, x-1])
```

- Set a Label counter array, with initial value of 0
- For each pixel i,j in B(i,j)
  For each non-zero pixel, we check its neighbours.
  If it has zero neighbours, we know it is a new region, so we give it a new label.
  If it has two non-zero neighbour, these pixels are connected, we give it the same label as the neighbour.
- Increment Label counter by 4
- set the corresponding pixel as the label value for coloring
- continue loop
- while traversing other remaining pixels, again if mask has one two non-zero neighbour, these pixels are connected, we give it the same label as the neighbour.

**RESULT:**



**OBSERVATION:**
The label counter was initiated at 2 (0 is for the background, 1 is for the first region).

**APPENDIX:**

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

def connected_component_labeling(binary_image):
    height, width = binary_image.shape
    labels = np.zeros_like(binary_image, dtype=np.int32)
    label = 2

    for y in range(height):
        for x in range(width):
            if binary_image[y, x] != 0:
                neighbors = []

                if x > 0:
                    neighbors.append(labels[y, x-1])
                if y > 0:
                    neighbors.append(labels[y-1, x])
                if y > 0 and x < width-1:
                    neighbors.append(labels[y-1, x+1])
                if y > 0 and x > 0:
                    neighbors.append(labels[y-1, x-1])

                neighbors = [n for n in neighbors if n != 0]

                if len(neighbors) == 0:
                    labels[y, x] = label
                    label += 1
                else:
                    labels[y, x] = min(neighbors)
                    for i, n in enumerate(neighbors):
                        if n != labels[y, x]:
                            labels[labels == n] = labels[y, x]

    unique_labels = np.unique(labels)
    relabeled_labels = np.zeros_like(labels, dtype=np.int32)
    for i, l in enumerate(unique_labels):
        relabeled_labels[labels == l] = i

    return relabeled_labels

binary_image = np.array([[0, 0, 0, 0, 0, 0],
                         [0, 1, 1, 0, 0, 0],
                         [0, 0, 1, 1, 0, 0],
                         [0, 0, 0, 0, 0, 0],
                         [0, 0, 0, 1, 1, 1]], dtype=np.uint8)

labels = connected_component_labeling(binary_image)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
ax1.imshow(binary_image, cmap='gray')
ax2.imshow(labels, cmap='jet')
plt.show()
```