Connected Components Labeling

Objective:

To implement the sequential algorithm for Connected Component Labeling (CCL) and verify the algorithm with test images

Language used: Python

Running the code:

In a terminal, enter:

```
./ccl.py [image name] [threshold value (optional)]
```

Example usage:

```
./ccl.py test.bmp
./ccl.py gun.bmp 250
```

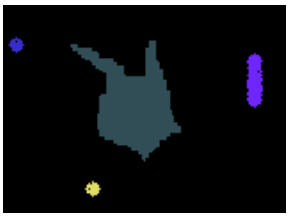
Description of sequential CCL algorithm implemented (4-connectivity was used):

1. Scan image top to bottom, left to right
2. If pixel is '0' or black, skip
3. If current pixel is '255' or white, then
    a. If upper neighbor has a label, copy that label to current pixel
    b. If left neighbor **also** has a label, then
        i. If both upper & left neighbors have same label, copy the label
        ii. If both have different labels, then copy the upper's label and enter the labels as equivalent labels in the equivalence table (this is achieved through a *union-find algorithm*[1] applied to the equivalence table)
    c. If only left neighbor has a label, copy that label
    d. Otherwise, assign a new label to current pixel and enter that label into equivalence table
4. Continue steps 2 & 3 until all pixels are scanned
5. Do a second pass:
    a. Find lowest label for each equivalent set in the equivalence table
    b. Replace each label (in the picture data) by the lowest label in its equivalent set

[1] A union-find algorithm performs two operations on a data structure:
   a. **Find**: Determine which subset a particular element is in. This can be used for determining if two elements are in the same subset.
   b. **Union/Merge**: Join two subsets into a single subset

Results Analysis:

The following table summarizes the results obtained with 4 test images:

Image No.	Input Image	Output Image	No. of components detected	Size Filter Threshold
1 (test)			1	None
2 (face)			6	None
3 (gun)			4	None
4 (gun with size filter)			1	250
5 (test 2)			12	10

All the input images produced the desired output images. The number of components detected in each of the images were accurate.

A size filter was applied to images 4 and 5 with different threshold values for each image. The 'size' of noise present in image 4 was larger than that in image 5. The size filter function was implemented in the following manner:

1. The number of pixels with various labels was determined
2. Those labels whose 'count' was less than the specified threshold value were excluded from the labels dataset