Problem Statement

- Description: Histogram equalization is one of the effective approaches to enhance the image quality. This assignment is to implement the histogram equalization algorithm to process images for better visual quality. The image is given below *("guide.bmp" or "guide.tif");*. It is required to print out the images and the corresponding histograms before and after the histogram equalization.

- Implementation: By expanding on the prewritten example program, I decided to use the same variables and pointers already declared.

Algorithm design

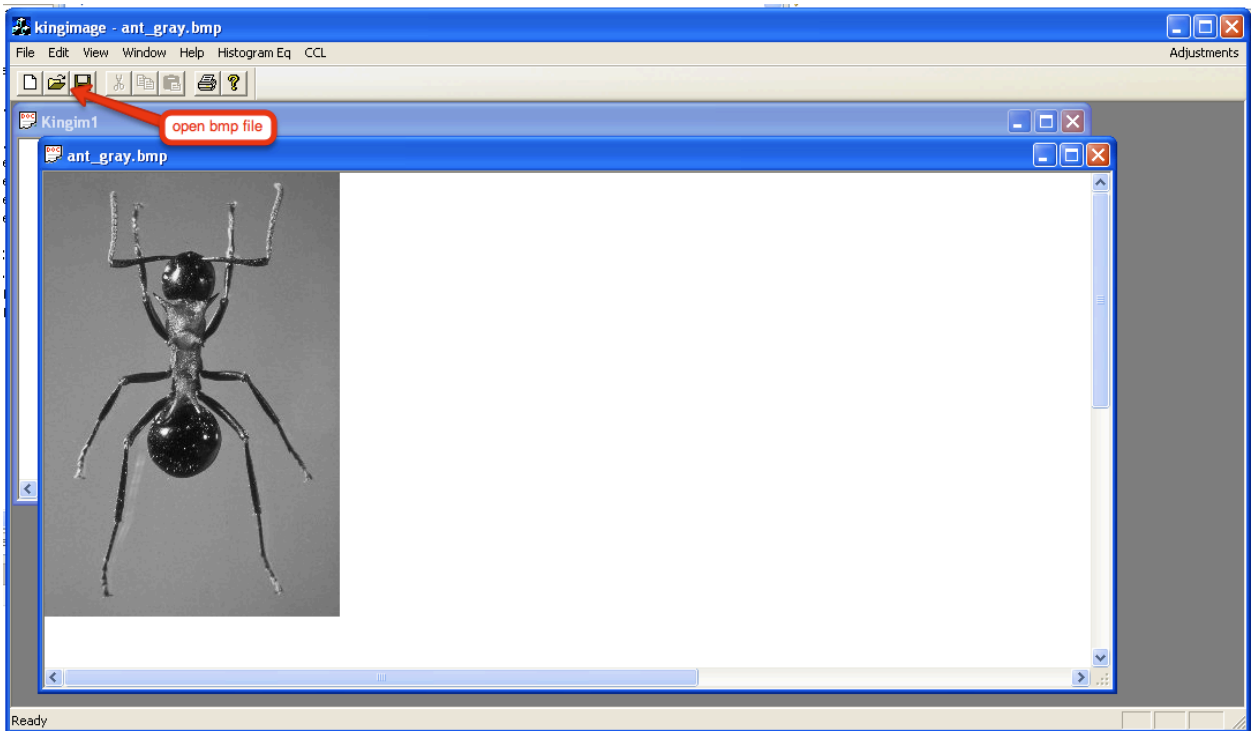
For this first part, I decided to expand on the already written example functions. I noticed that the array was already in place, and was in multiples of 8. This was due to the fact that for the gray level pictures, each pixel was contained in 8 bits. Also, according to the email sent by the TAs, I corrected the sizes of bmp files in case the rows were not multiples of 4. I did this by grabbing the width of the image and modding it by 4, then I saved that result and added it to the variable iWidth.
As for the actual algorithm of the histogram, first I created an array of 256 elements, each containing a value of gray. I initialized all of the elements to 0 and then I proceeded to count the frequency of all gray levels and storing them in this array called 'picture'. With this array I formed the cumulative image histogram by dividing each frequency of gray by the size of the image and adding that result to the previous value. Finally, I completed the formula by multiplying each of the newly updated values by 255, so the histogram would now look like a ramp.
Here is the formula I used:

```
for (int i=0;i<256;i++) pic2[i] = picture[i]/(double)size;
picture[0] = pic2[0];
for(int i=1; i<256; i++) picture[i] = picture[i-1] + pic2[i];

for(int i=0; i<iHeight; i++)
{
        for(int j=0; j<iWidth; j++)
        {
                plmg[i*iWidth+j] = (255*picture[plmg[i*iWidth+j]]);
        }
}
```

Manual

This first part of the program works by opening a gray valued image through the toolbar menu and then clicking on the menu bar 'Histogram Eq'.


kingimage - ant_gray.bmp
File Edit View Window Help Histogram Eq CCL
Adjustments
Kingim1
open bmp file
ant_gray.bmp
Ready

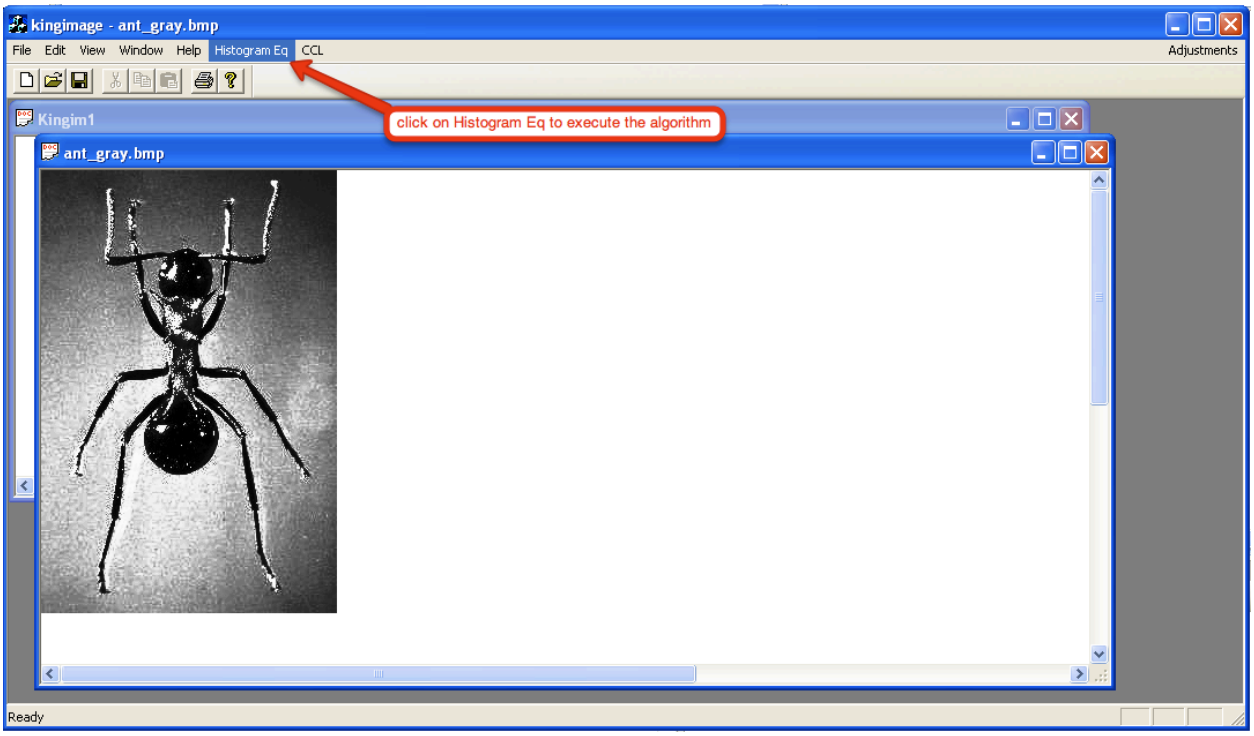

kingimage - ant_gray.bmp
File Edit View Window Help Histogram Eq CCL
Adjustments
Kingim1
click on Histogram Eq to execute the algorithm
ant_gray.bmp
Ready

Image Enhancement by Level Adjustment

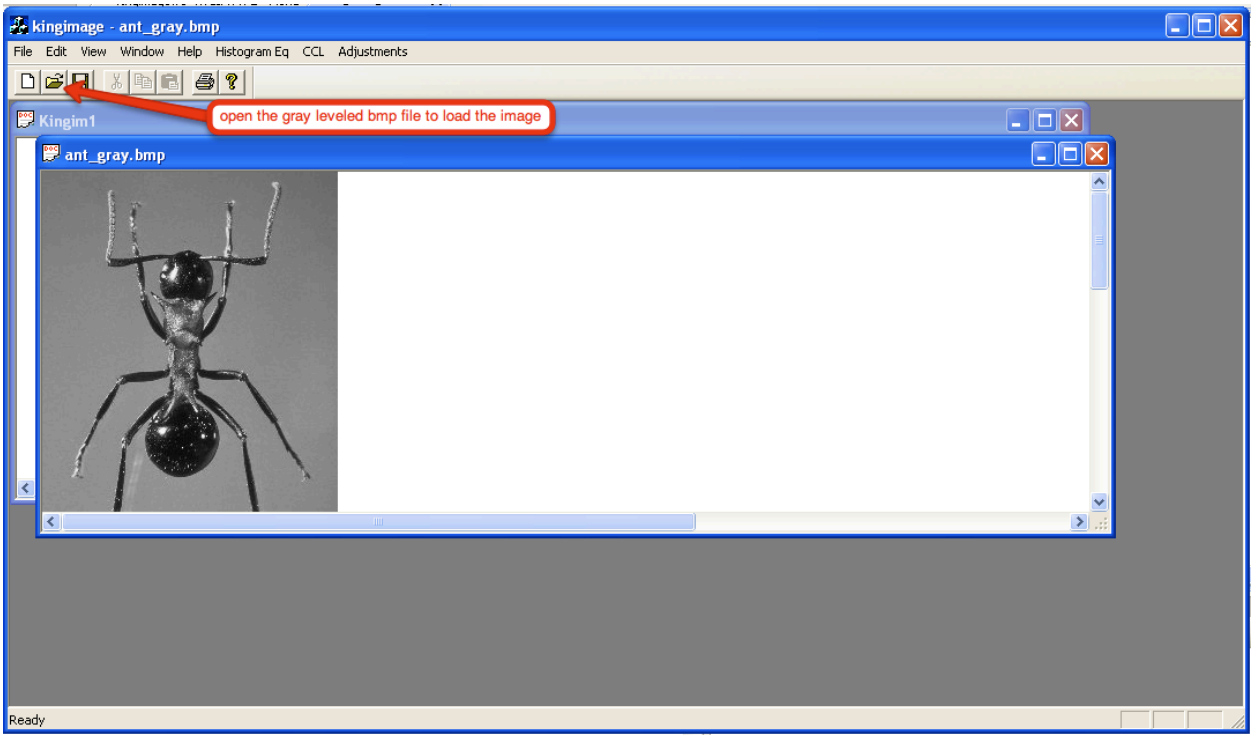
- Window/Level adjustment to enhance certain range of the gray scale is an efficient way to control the image display. You are going to realize such functionality in this assignment. Given the image below *("guide.bmp" or "guide.tif")*, it is required to show the images after the level adjustment (e.g., 50; 127; 200) with the Window size of 30 and 100.

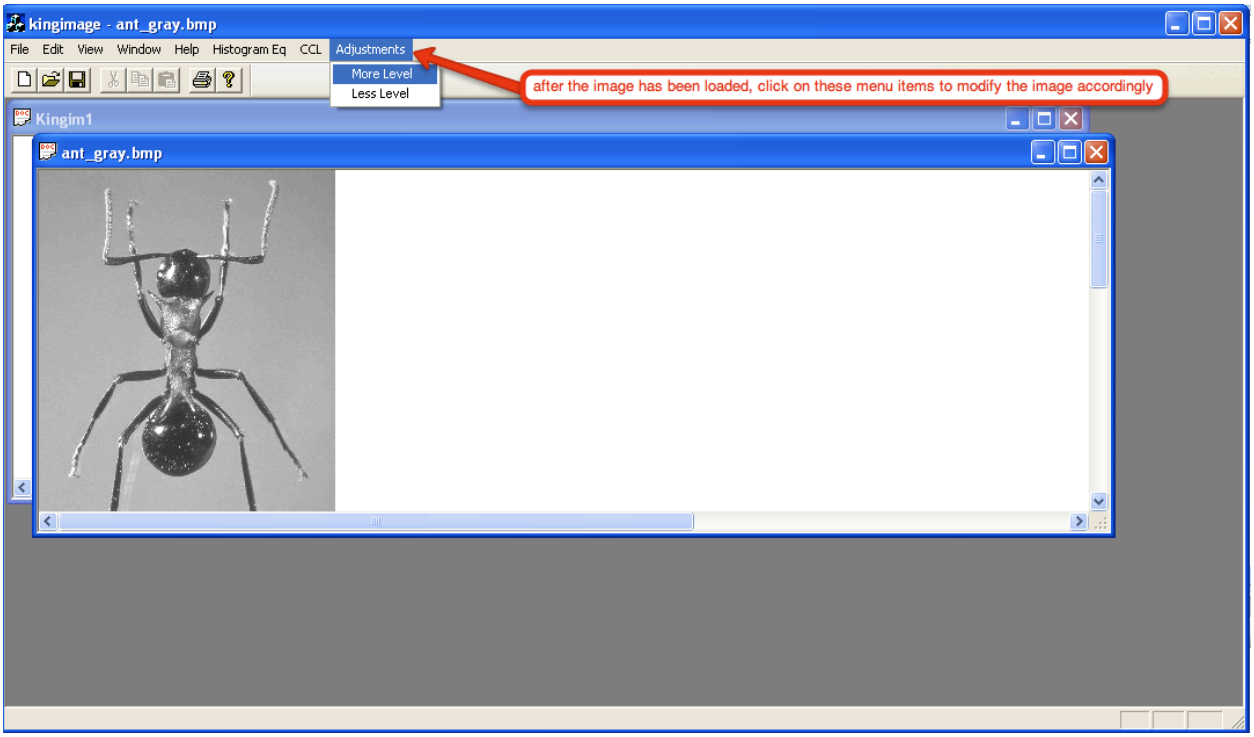
Algorithm Design

I decided to create a new function for these leveling instructions that adjust the level. By creating these both functions (more level adjustment and less level adjustment) I had to assign a specific value to increase the pixel values for. This value is 10, so for every time the image increases level, the gray values will increase/decrease by values of 10. I created the functions 'OnAdjustmentMoreBrightness' and 'OnAdjustmentLessBrightness' to modify the levels of the image. I did this by creating a constant value and modifying each pixel to be modified according to this value

Manual

Load the image by opening the bmp file. Click on the menu of Adjustments and click on either the more levels or less levels. This will increase or decrease the values accordingly.





Connected Component Labeling

- **Description:** Connected component is to describe the basic relationship among the neighbor pixels in an image. Pixels that are connected can be labeled as a same region. In this assignment you are to implement this algorithm incorporating with the Histogram Calculation and image Thresholding Operations. Given a gray scale image, after converting this image into binary image by using thresholding algorithm, the binary image can be labeled on several connected component regions. The resulted image will show the separate regions detected. This is a good chance for you to implement three concepts in one assignment.
- **Your implementation:**

  (1) Determine the optimal threshold T for further thresholding based on the image histogram. (8%) (5 extra points for automatic determination of the threshold and pre-processing the image).
  (2) Apply T to converting the original image into binary image f1. (10%)
  (3) Apply the 4-connected component algorithm to detect the regions of each character in image f1.
  (4) Labeling the connected component that you detected with different gray levels (e.g., 50, 100, 150, 200, 50, 100, …), or different color (e.g., red, green, blue, red, green, …), obtain the labeled image f2. (20%)
  (5) Print out your images f1and f2.
  (6) Print out the threshold value T.

Algorithm Design

For this last section, I decided to reuse some of the code given before. I used a similar algorithm as for the histogram equalizer section and decided to apply these gray levels into automatically

detect the optimal threshold T for my threshold algorithm. I did this by obtaining the average gray levels used in order to display the picture.

<EXTRA>
I added all the gray levels of each pixel into an integer called 'sum', then I divided that sum by the size of the picture (all pixels) in order to obtain an average. One other alternative was to obtain the median instead of the average but it required more computation for a likely same result. Then I proceeded to apply the threshold algorithm inside a function I created named OnThreshold. This was supposed to be an extra option on the menu bar, but I thought it would be pointless to create the option since it would require the user to perform 2 steps instead of 1.
</EXTRA>

For the threshold algorithm, once I found the average value, anything above that value would be considered 255 (the maximum gray value) and everything below would be 0 (the minimum), therefore creating a binary picture with values of 1 (255) and 0.
After the threshold had be calculated, the connected-component labeling algorithm would take place. For this section, I chose the two pass algorithm. It consists of two passes through every pixel: once for identifying all the white pixels and assigning it a label value, which in my case was 1; and second, to see if that pixel was adjacent to any 4-connected pixel that would also have the same value. If they were connected, then you would have to modify your labels accordingly and keep track of the equivalent table. This equivalent table makes sure that for all neighboring pixels you must have the same label applied to each one, so they belong to the same section. I did this by doing 5 simple if statements:

```
if (pImg[i*iWidth+j] == 255)
{
        if(pImg[i*iWidth+j-1] == 255) labels[i][j] = labels[i][j-1];
        else if (pImg[i*iWidth+j-1] == pImg[(i-1)*iWidth+j] && labels[i][j-1] != labels[i-1][j])
        {
                labels[i][j] = min(labels[i][j-1], labels[i-1][j]);
                labels[i][j-1] = labels[i][j];
                labels[i-1][j] = labels[i][j];
        }
        else if (pImg[i*iWidth+j-1] != 255 && pImg[(i-1)*iWidth+j] == 255) labels[i][j] = labels[i-1][j];
        else if (pImg[i*iWidth+j-1] != pImg[(i-1)*iWidth+j]) { counter++; labels[i][j] = counter; }
}
```

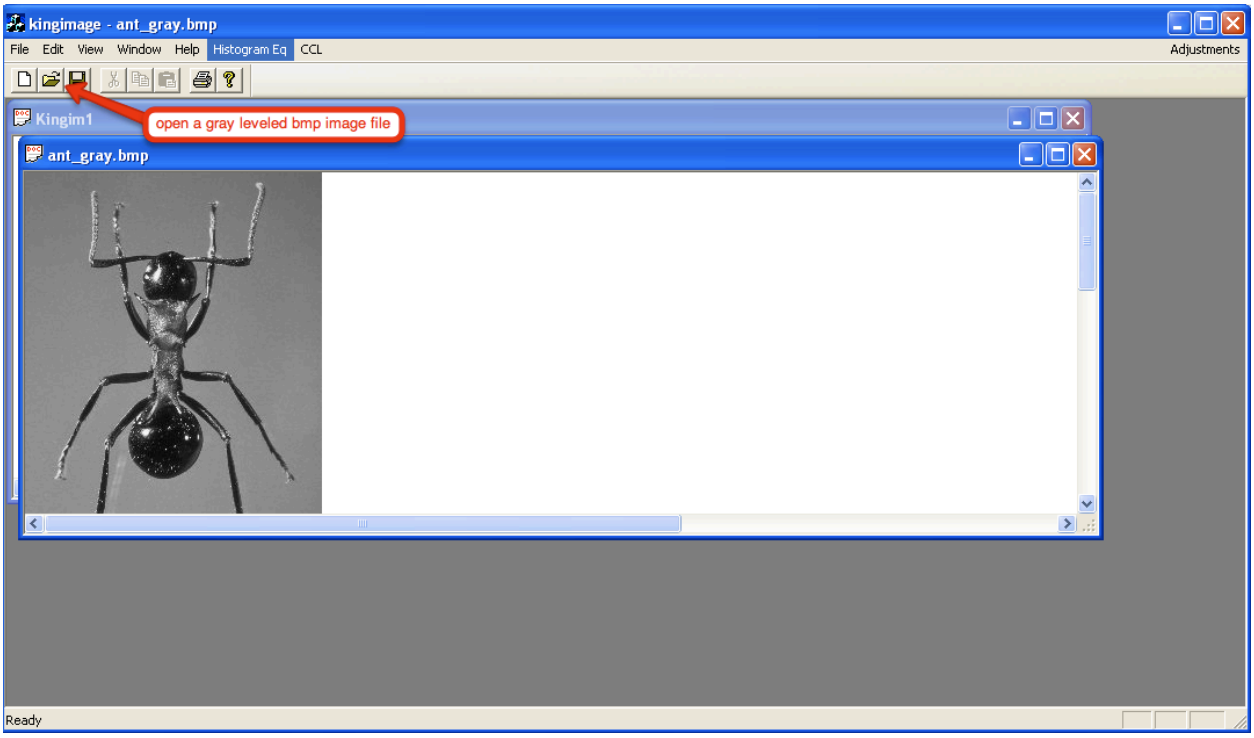
basically this:
1. Is the pixel white?
if yes:
   a. Is the pixel west to the current pixel not background? if yes apply the same label as west pixel.
   b. Is the pixel west to the current pixel and the pixel north to the current pixel the same value but different label? if yes select the smaller label and set all labels to be the same.

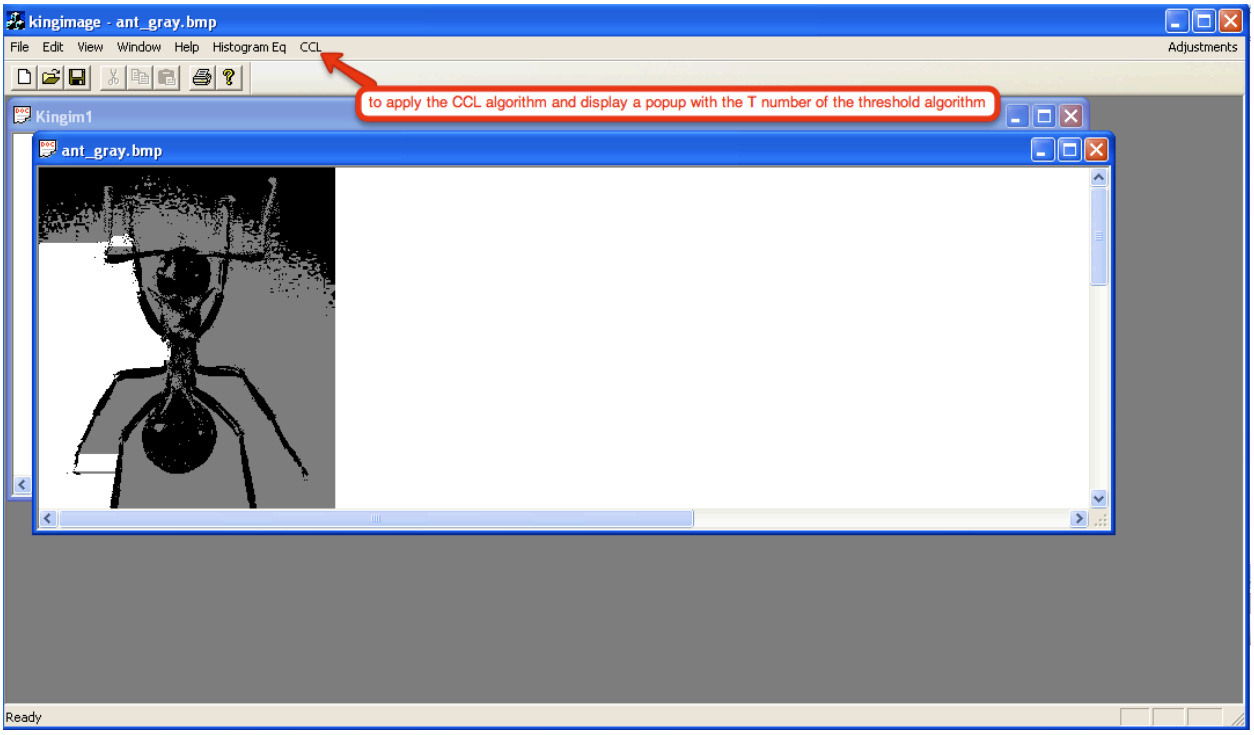
c. Is the pixel west to the current pixel a different value and north to the current pixel same color? if yes apply the same label as north pixel.
d. Are the pixels north of the current pixel and west of the current pixel non background? if yes, assign a new label to the current pixel.

This algorithm uses Union-find data structure which provides excellent performance for keeping track of equivalence relationships. For this section I must admit my coding style was not at its best. The performance is not bad, but the code a little too messy for my taste and could be done better if I had not had 3 test this past week.
Finally I recolored the bmp picture according to these newly created labels. There is a bug that I could not figure out on time. For this last procedure I decided to shade in different colors of gray each label identified by multiplying the number label times 255/amount of labels. That way each value from 1 to number of labels will be evenly distributed. However, for some images all of the values resulted in being 0 or 1 because the resulting image was completely black. So I added an if statement that would avoid the issue but not very conveniently.

Manual

Open an image file (gray scale bmp format) and it will load on the program. Later click on the menu button CCL which will apply both the threshold and the CCL algorithm.




kingimage - ant_gray.bmp
File Edit View Window Help Histogram Eq CCL
Adjustments
to apply the CCL algorithm and display a popup with the T number of the threshold algorithm
Kingim1
ant_gray.bmp
Ready