**Four Color Cat**

**Background and Introduction**

The color theorem or four-color map theorem (FCT) states “that no more than four colors are required to color the regions of any map so that no two adjacent regions have the same color.” Some regarded the proof of this theorem as one of the three most challenging problems of modern-day Mathematics. The question was proposed in 1852 and solved in 1976; it was a longstanding problem that lasted over a century and one of the first significant math problems solved by a computer. It is easy to see its significance in both Math and Computer Sciences.

It also happens that my family owns two hairless cats. I recently learned that cats can see only a portion of the visible spectrum; they cannot see the red to orange wavelengths. I, therefore, would like to write a filter that colors a given map in four colors (colors visible for cats) following the rules of the FCT for cats to be able to read maps and distinguish boundaries.

**Research**

Graph Representation:

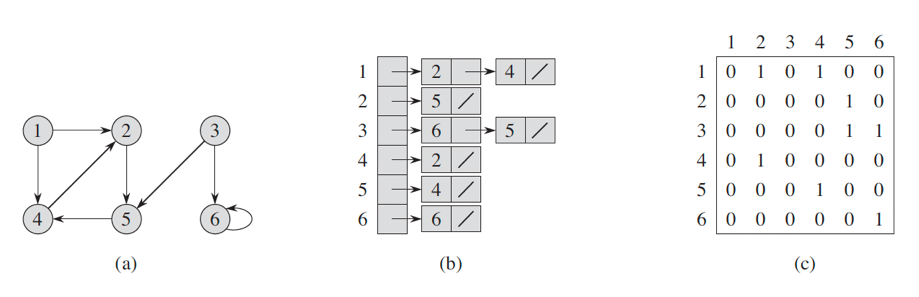
A graph G = (V, E) with V resembling a collection of vertices and E as edges/connections in a graph can be represented in a program through either an adjacency list or an adjacency matrix. Each vertex v ∈ G.V can be given a list of edges e to represent a connection between one vertex to another. This representation will be able to help me with differentiating contours in an image with the list of edges representing the adjacent/connected contours.

Figure Graph Representation With Adjacency List/Matrix implementation

Cat’s Vision:

Cats are only able to see blue-violet, yellow-green wavelengths of light, which gives me the four colors to work with in my project.

Blob Detection by Daniel Shiffman:

A simple algorithm that locates pixels of a certain color in an image, encapsulating the “blob” in a square. This was an inaccurate method of recognizing contours but was able to process each image very quickly. The algorithm locates the position of nearby pixels that share the same color. It inspired me as it made me realize contours could be identified via BFS which searches all connected pixels, as well as being able to determine adjacent contours via estimation.

Canny Edge Detection:

Canny’s algorithm converts a photograph into a black-and-white image to analyze the edges of any given image. I realized that black-and-white images are easier to work with and would allow me to construct a graph based on graph representation from basic graph theory.

Figure Canny Edge Detection Effect

**Design**

Dividing to contours:

While generating the image, I will apply the threshold filter with 0.5 as the default parameter. The filter will return a black-and-white version of the image with black lines resembling the boundaries of each contour. I could use a higher threshold, like 0.8, when working with an image with a darker theme.

Recognizing contours:

Use basic graph analysis algorithm BFS. First, generate adjacency lists for each pixel in the image (surrounding eight pixels – black colored pixels).

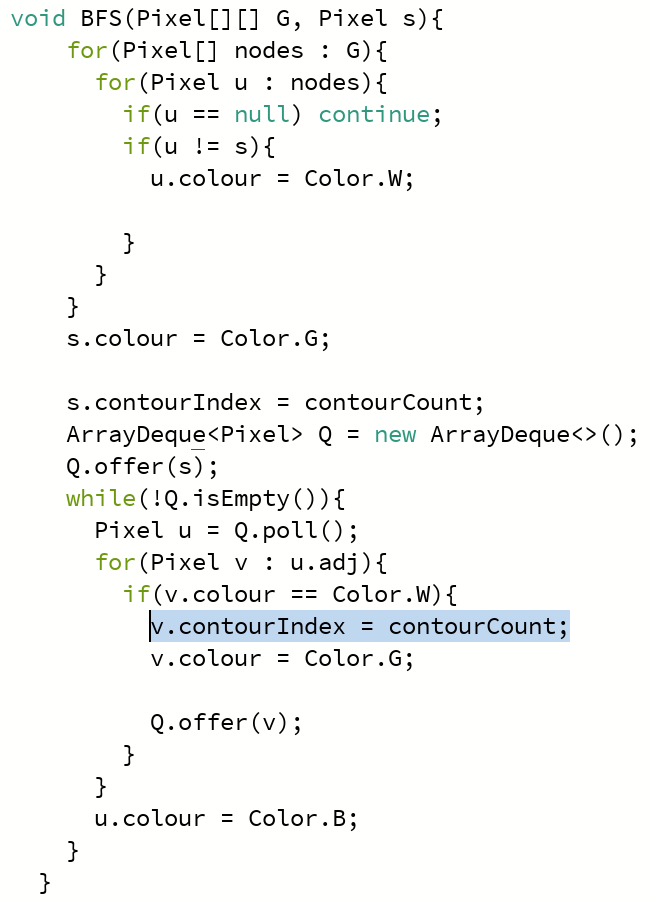
Create a 2D array resembling pixels of the image, placing objects containing necessary information.

Figure BFS Implementation, Marking Pixel to Contour in Highlighted Line

Iterate through all pixels in the image; if a pixel is white and not marked as part of a contour, run BFS on that pixel (the original pixel). BFS will find all other pixels that are reachable by some path containing the original pixel. (Marking black as boundaries/not eligible to be a part of a path) Mark all such pixels as part of a contour associated with the original pixel.

By the end of this process, I will end up with a 2D array marking each pixel as either a boundary (black pixel) or as a pixel part of a contour, as well as an indicator of which contour it belongs to.

Identifying Adjacency:

This project assumes the borders/boundaries to be of similar thickness; generalizing this algorithm may take more research and thought.

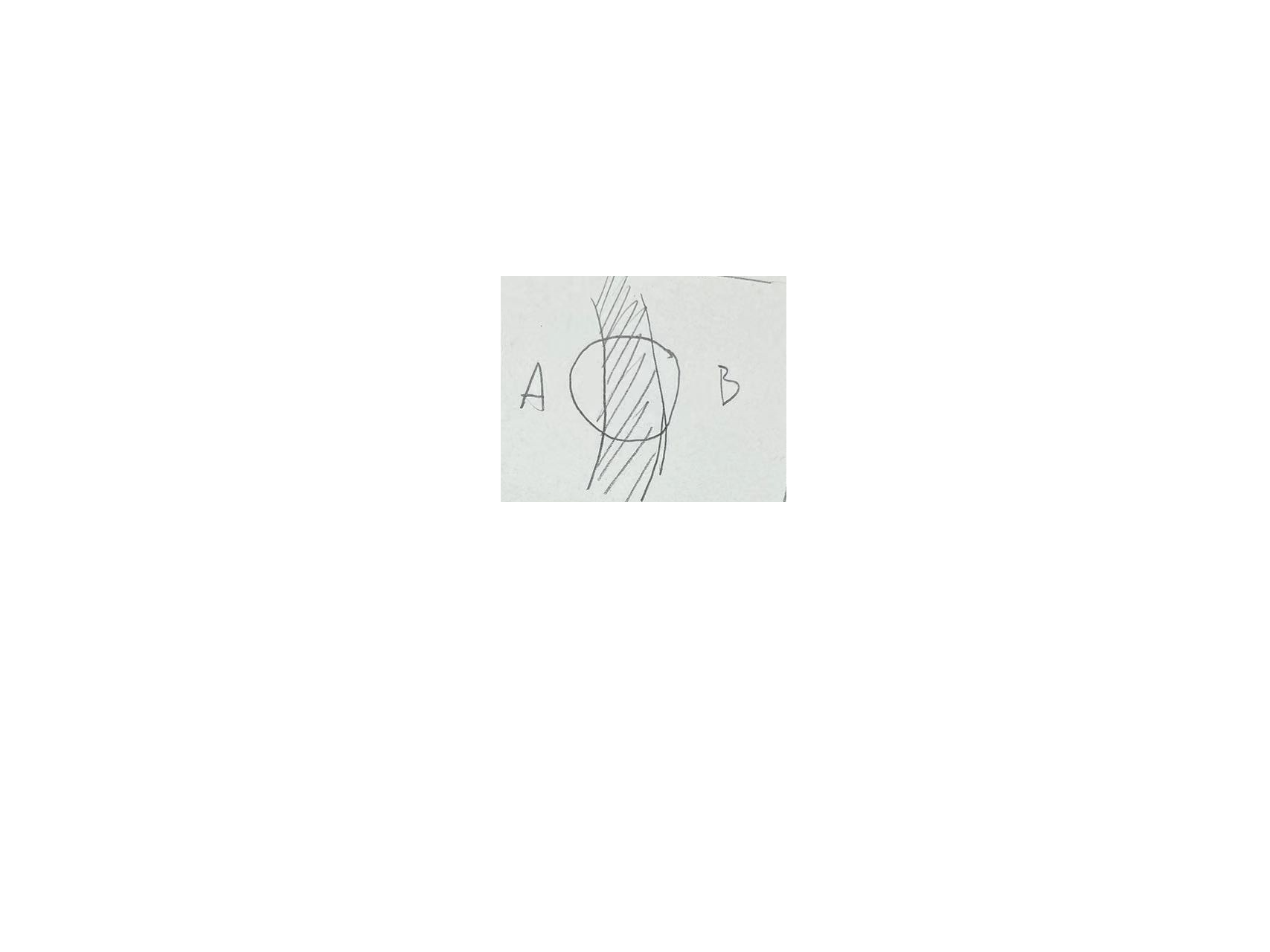
First, estimate the thickness of the boundaries, then create a circular region (CR) with a diameter slightly longer than the estimated thickness. The CR is a detection device used like a magnifying glass. The center of the CR is iterated throughout the entire image; during each iteration, the pixels within the CR are assessed. When two pixels of color are white and are of different contours, they are recognized as adjacent contours and recorded.

Figure CR Detecting White Pixels in Contour A and B

Formation of Graph:

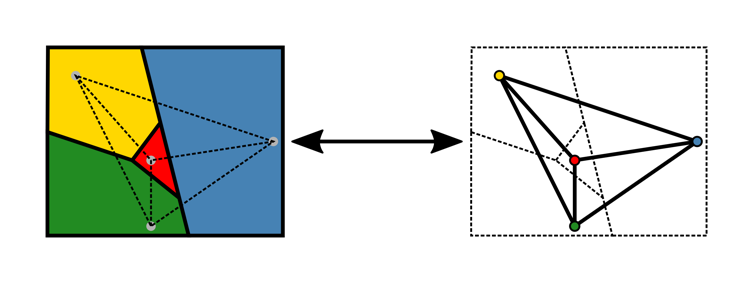
The data structure used to implement the relationship between contours is the basis of graph theory. Each contour is represented as vertices in a graph with edges implemented using ArrayLists to represent adjacency relations. The data structure can be seen as an undirected and unweighted graph. This structure comes in handy when implementing the algorithm to fill in the colors based on the rules of FCT.

Figure Graph Representation of Contours

Coloring with Welsh-Powell’s algorithm:

This algorithm was developed to color a graph such that connected vertices of a given graph do not share the same color. This algorithm qualifies as an elementary algorithm to solve the FCT problem in simple maps. It works as follows:

1. Find the degree/number of edges of each vertex.
2. List the vertices in order of descending degrees.
3. Color the first vertex with color 1.
4. Move down the list and color all the vertices not connected to the colored vertex with the same color.
5. Repeat step 4 on all uncolored vertices with a new color, in descending order of degrees, until all the vertices are colored.

Figure Welsh-Powell Implementation

My algorithm implementation has a complexity of O(V2) and uses the default sorting algorithm from the Arrays class, defining a custom comparator with a lambda function.

As each pixel is already marked to associate with some contour, I need to mark the color of each contour in this algorithm.

The coloring method iterates through all pixels in the image, determines the contour the pixel is associated with, and further checks the color that the contour is given, as calculated in Welsh-Powell’s algorithm, setting the pixel to said color.

**Challenges**

There are three difficulties in this project.

Contour Recognition:

Contour recognition is complex, as image boundaries are challenging to identify. As images become more complex, the boundaries of contours become more challenging to recognize; the task becomes even more complex when dealing with images with color gradients. As written in the Design portion of this journal, I had the image go through a black-white filter, thereby setting the regions of contours to white and boundaries to black. However, this filter would not work against an image like a photograph, with lights, shadows, and gradients without explicit boundaries. This becomes a limitation to my program as it could only analyze images with a simple layout of shapes with explicit boundaries and plain colors (without gradients).

Adjacency Detection:

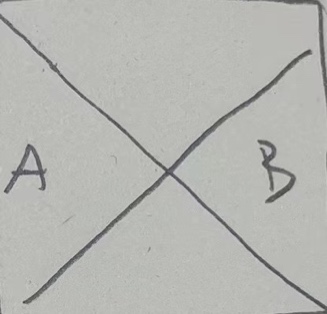
The difficulty again boils down to the inability to identify the borders of each contour's borders and the particular case of vertical contours as shown on the right. As borders in an image are typically interconnected, it is not easy to locate where a border begins and where it ends. Therefore, it is difficult to tell whether two contours share a particular border. The vertical contours also pose a problem because even if I were to be able to find a way to tell the start and end of a border, exceptional cases like such vertical contours technically also count towards sharing a border, thereby making it strangely challenging to locate the adjacent contours of any given contour.

Figure Vertical Contours A and B

Ultimately, I resorted to using the distance between contours to determine adjacency, which has limitations.

Coloring based on FCT:

At first glance, it may seem easy to color a map such that adjacent contours do not share the same color. However, this is quite a non-trivial task. Coloring a map based on FCT is like solving a sudoku puzzle; if each region was not colored strategically but through trial and error, it would sometimes take forever to complete. Luckily, there were existing algorithms to work with; an algorithm like Welsh-Powell's did save me much time in coming up with my own, potentially slower, algorithm.

**Discussion**

The limitation of this project is the need for more generalization.

Image Processing and Conversion:

My approach to recognizing contours in an image was to first convert the image to a black-and-white image with borders being black. This only sometimes works, even if the image has no gradients. The user of this program may still need to modify the threshold to achieve the desired effect. One thing to note is that this approach is quite similar to Canny edge detection, which converts a photograph to a black-and-white image. It is possible to use Canny edge detection in this project to generalize the program, allowing some limitations to be lifted regarding image selections. However, it may take time to implement Canny's as it seems complex.

Adjacency detection using CR:

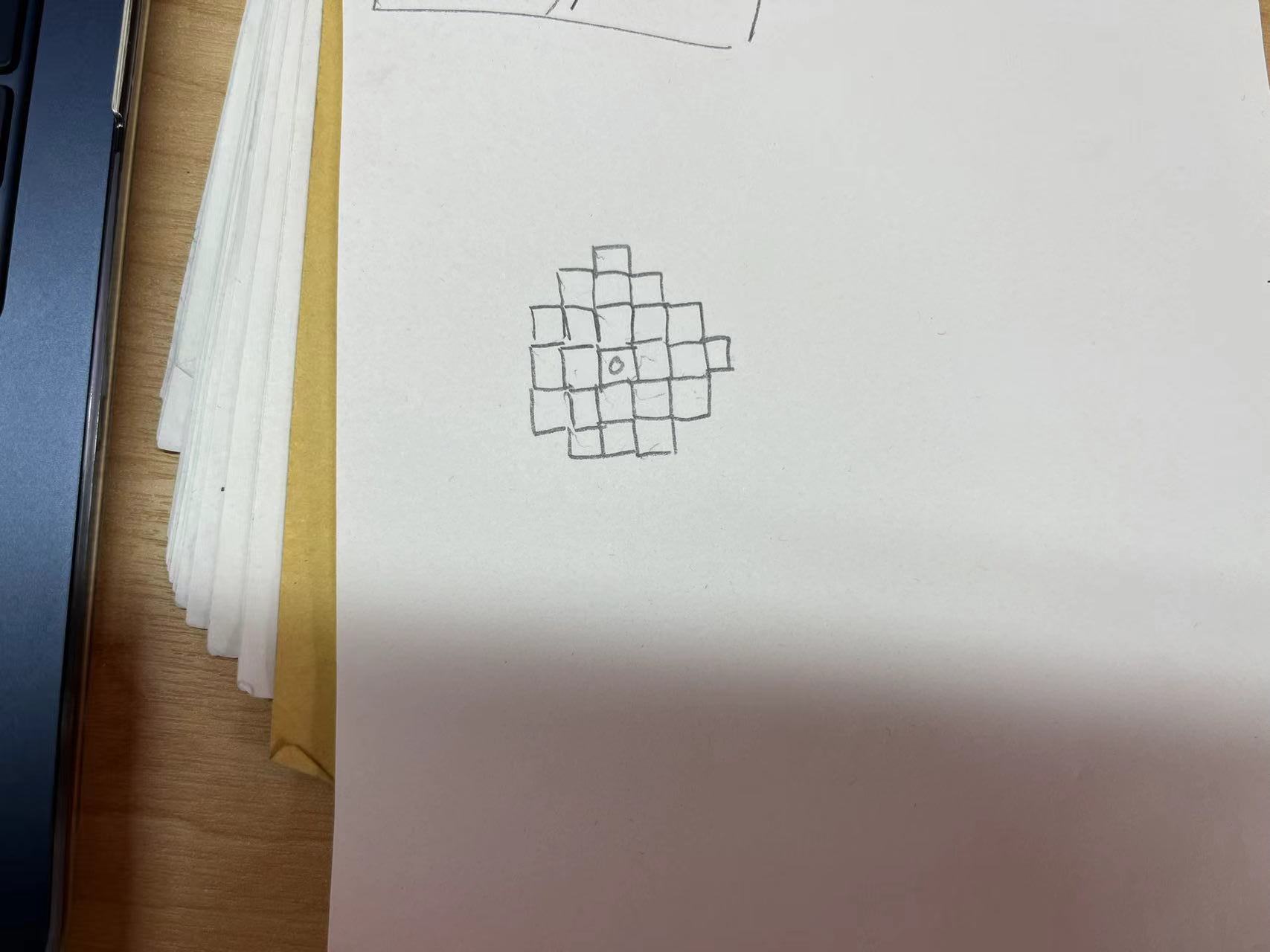
This approach is an estimate rather than an accurate method of determining the adjacency; the diameter of the CR is eyeballed, and the circle is imperfect, with the region taking the form as represented in figure 8. This implementation means that the image to be processed must have borders with uniform thickness, and the user will have to change the diameter manually by eyeballing the thickness represented in the number of pixels. Improvements could be found, given more time, that determine adjacency not via estimation but through more accurate means.

Figure CR Program Representation

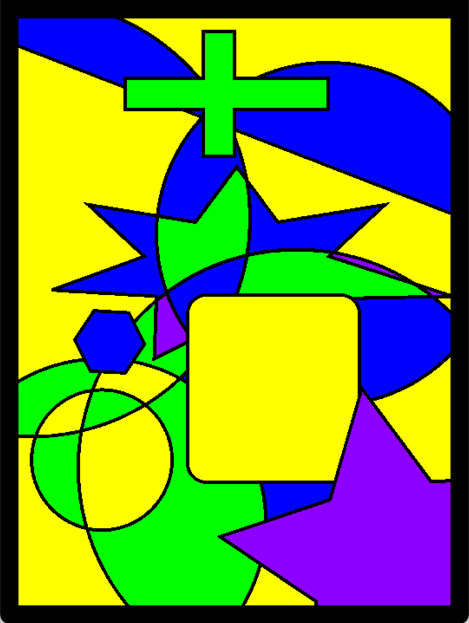
**Product**

Figure After Coloring Based on FCT

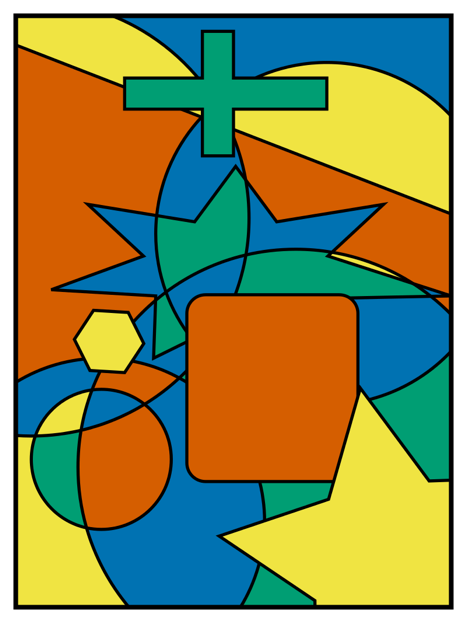
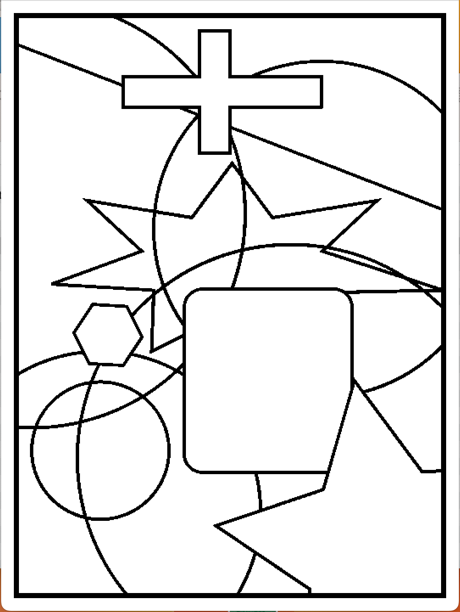


Figure After Preprocessing

Figure Original Image

Bibliography

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. Mit Press.

GeeksforGeeks. (2019, October 29). *Welsh Powell graph colouring algorithm*. GeeksforGeeks. <https://www.geeksforgeeks.org/welsh-powell-graph-colouring-algorithm/>

Haylee Bergeland, K.-C. (2020, November 18). *Can cats see color?*. Daily Paws. <https://www.dailypaws.com/cats-kittens/health-care/cat-conditions/can-cats-see-color>

Shiffman, Daniel. (2016). *11.7: Computer Vision: Blob Detection - Processing Tutorial*. *YouTube*. Retrieved September 11, 2023, from <https://www.youtube.com/watch?v=ce-2l2wRqO8&t=1133s>.

Wikimedia Foundation. (2023a, August 10). *Edge detection*. Wikipedia. <https://en.wikipedia.org/wiki/Edge_detection>

Wikimedia Foundation. (2023, August 18). *Four color theorem*. Wikipedia. <https://en.wikipedia.org/wiki/Four_color_theorem#Generalizations>