Homework 4: Counting Objects in an Image

Deadline: December 9 Monday by Midnight

Instructions

For this homework you will be designing and implementing a system, in either C or C++, to label and count the number of objects in a given image. Each image consists of a number of pixels, and initially all pixels will have one of two labels, 0 and 1. 0 is considered a background pixel while a 1 represents a pixel that is part of an object. Your system will apply, over multiple iterations, the same label to pixels that are connected to form objects, and then count the number of objects for a given image.

Requirements

This assignment has two parts: **a design portion and an implementation portion.**

Design Document

For the design portion, you must generate documentation, in PDF format, describing your system and design process. The purpose of this is for you to explain not just what your system is doing, and how it is doing it, but why. You will need to justify your design decisions in a concise, informative manner. Justifications such as "I did this because it was easy" are not sufficient, as you should actually explain why a particular data structure or algorithm was more efficient, effective, or optimal. Additionally, commented code, while sometimes helpful in small examples, is not a sufficient explanation in and of itself. Your explanations and justifications are expected to be presented in prose and in paragraph format, i.e. not bulleted lists. Further, part of the evaluation of your design document is the apparent amount of thought and effort that went into its creation.

This document should be divided into four main parts, each with an appropriate header.

In the first part, you should describe your design process. Did you work out the algorithm on paper or a whiteboard before hand? Did you draw UML diagrams of the system? Did you create a small prototype? Did you simply start coding

away and then recode once or twice with newfound understanding? In a few paragraphs, describe in detail how you went about designing the system, and be sure to provide sufficient justification of your methodology.

For the second part, you should describe the data structures you used in your system. What, if any, objects or structs did you create to store data? How did you organize and manage them? What types of formal data structures did you make use of (trees, graphs, arrays, hashes, etc)? In a few paragraphs, describe in detail how you stored the various data elements in your system, and be sure to provide sufficient justification of your methodology.

For the third part, you should describe functionality of your system. How is data moved and transformed? How is it read in? How is it output? What are the various major functions you constructed and how do they work? In a few paragraphs, describe in detail how your system works, and be sure to provide sufficient justification of your methodology. You might also consider including diagrams to more easily visualize how all of the pieces fit together.

Finally, you will need to briefly describe what share of the work was performed by each group member. Be sure to be specific about tasks.

Implementation

Your program must provide the following functionality and adhere to the following constraints:

- Allow the user to **input the name of the file describing the image**
  - This file will be a comma delimited list of pixel labels
  - All pixels will be labeled 0 or 1
  - You will need to store this information to process the image
- Every pixel needs to be examined. One by one, starting at the top left corner, and moving linearly to the bottom right corner
  - **In the first pass:**
    - If a pixel is a background pixel (its label is 0), simply ignore it and move on to the next pixel
    - Otherwise, check the label of the pixel above and to the left of the current pixel if they exist. **There may be 3 possible cases:**
      - **Case 1**: If the top and left pixels are background pixels or do not exist, create a new label
      - **Case 2**: If one is a background pixel or does not exist and the other has a label, the current pixel will be labeled the same as the labeled pixel
      - **Case 3**: If both the above and left pixels have different labels, assign the current pixel the smallest label and store that there is a relationship between the two labels. This will be used in the second pass.
  - **In the second pass:**
    - Consider the information you saved during the first pass for case 3. Any time you find a pixel whose label has a relationship with another label, assign the current pixel the smaller value of the relationship.
  - After both passes are complete, count the number of unique labels. This represents the number of objects in the image
  - **Output the image with the new labels and the count of the number of objects in the image**

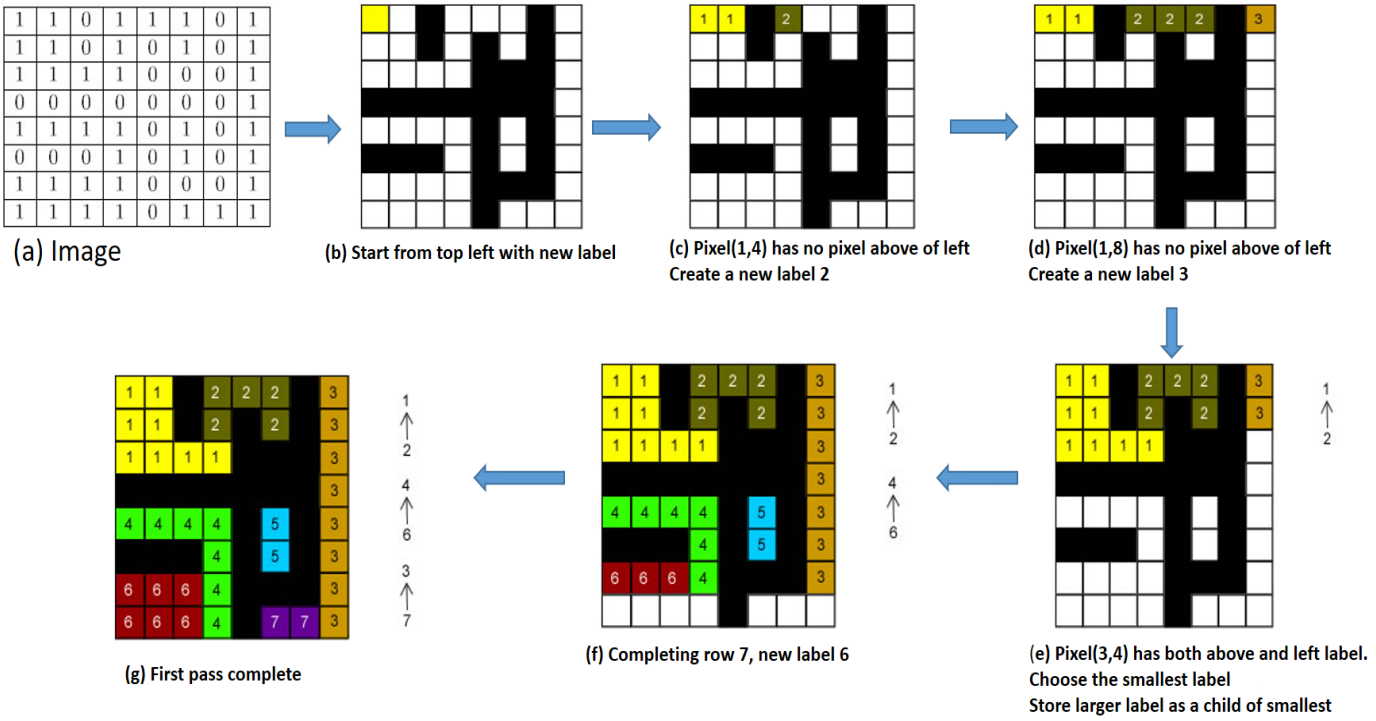
Example:

Input Image

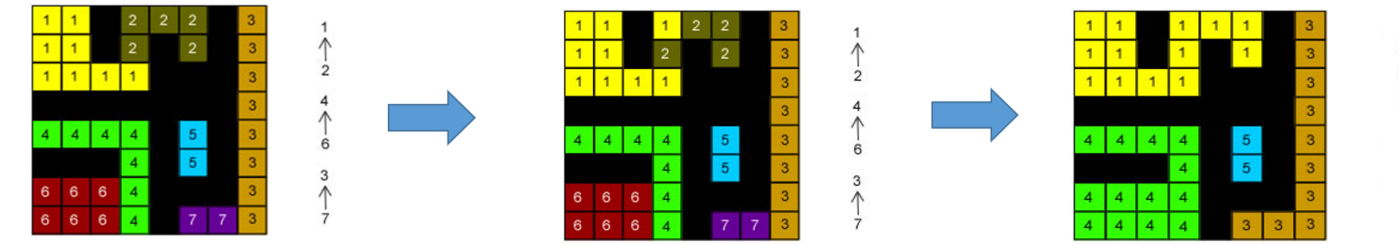
1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	1	1	0	0	0	1
1	1	1	1	0	1	1	1

- Label 0 indicates a background pixel
- Label 1 indicates a pixel that is part of an object
- Only consider pixels which have label 1

- First Pass



- Second Pass



(a) Label after First Pass

(c) The Final result

(b) **Converting a '2' into a '1'**

- Start again from top left with new label:
  - checks the data structure for the label '1'
  - '1' is not a child of any other labels
- **For pixel (1,4),checks the data structure for '2'**
  - It notices that '2' is a child of '1'
  - Then, it checks for '1' (root)
  - It replaces the label of (1, 4) with '1'

Input files

All input files will be a comma delimited matrix of 1 and 0s that represent the input image. The dimensions of the matrix will be **always square but size will vary.**

Suggestions

This project requires the use of a few different data structures to store and represent the label within the model. First, **you will need a way to store image information**. T**his is a good opportunity to use either vectors or dynamic arrays, as you do not know how big the image is until you have read the entire input file**.

Additionally, updating the label from given image to the final one could be done with two grids, one for the first pass and one for the second pass. Trying to update in place may cause problems. Thus, consider using a temporary grid for image.

Complete Homework Submission

Your program must provide all requested functionality. You must submit a .zip file containing the following:

1. All files necessary to compile and run your program
2. A README file explaining how to compile and run your program
3. Your design document in PDF format

Rubric

The entire assignment is worth 100 points. The breakdown of those points is as follows.

- 10 points: Design documentation
- 80 points: Code satisfies complete requirements
- 10 points: Professional coding style
    - 5 points: Adequate comments
    - 3 points: Modularity
    - 2 points: Readability
- If your code fails to compile on the Linux machines you may not receive credit for the programming portion of the assignment. I recommend not making changes to your code without checking for compilation before you submit.

Bonus

For 10 bonus points, give the length of the biggest objects in terms of pixel count. Example shown above, two objects have a length of 13.