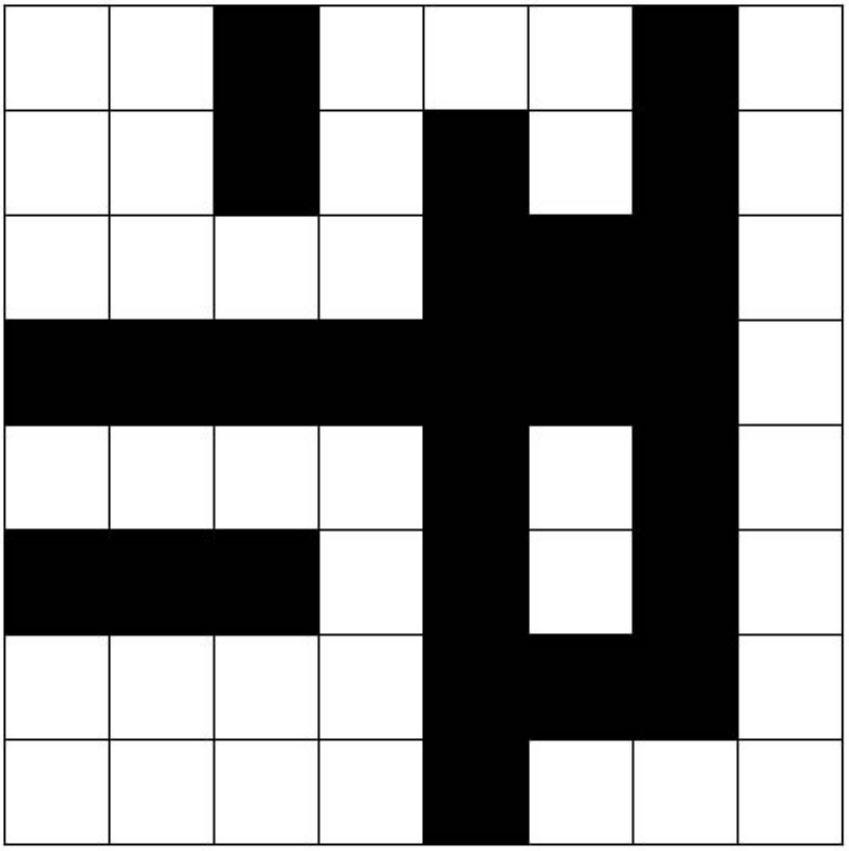
Labelling connected components – Example

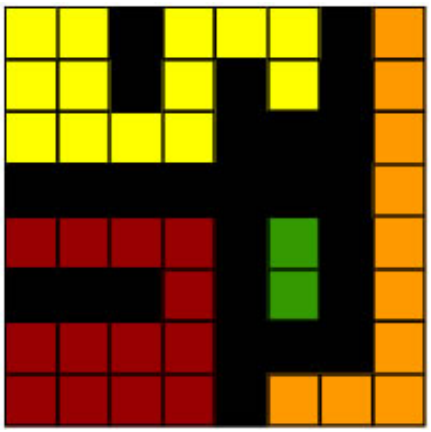
By Utkarsh | Published: March 21, 2010

We'll go through an example for Labelling connected components algorithm. I assume you know how the algorithm works (if not, check Labelling connected components) and also how the union-find data structure works. We'll work on a binary image to keep things simple. Suppose the binary image is the following:



The "blocks" represent a pixel. White means its a '1' and black means its a '0'. We're interested in labeling

the non-zero pixels. Ideally, you can just loop at the image and come up with a possible labeling scheme:



The actual labels are just 1, 2, 3, 4, etc. Here, the colors are just a visual representation of the actual labels.

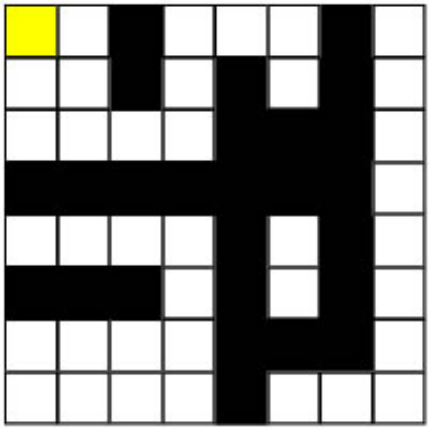
Now that you know what's about to happen,, lets get started.

The first pass

Here is the same image, represented as a matrix. We start off at the top left corner.

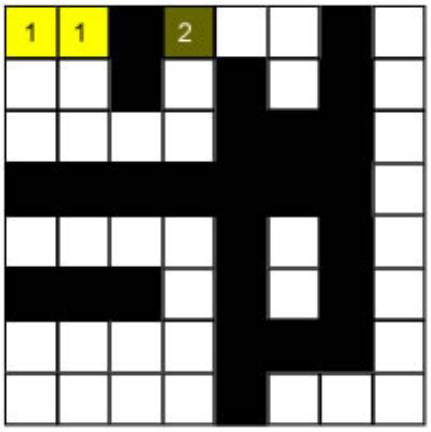
1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	1	1	0	0	0	1
1	1	1	1	0	1	1	1

We check the top and left pixels. They do not exist… so we need to create a new label. Thus, we set the label for the top left pixel as 1 (shown as yellow).



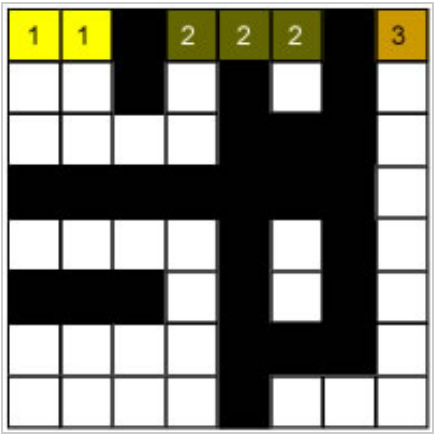
Next we check the pixel in row 1, column 2, or simply pixel (1, 2). It does has a pixel to its left. So we copy its label. The next pixel, (1, 3) is a background pixel. We're not interested in it. So we simply skip it and let it be black (a label of 0).

Next comes the pixel (1, 4). There are no pixels above it. But the pixel to its left is a background pixel. So we create a new label. So, we mark (1, 4) with label 2 (shown as dark yellow).



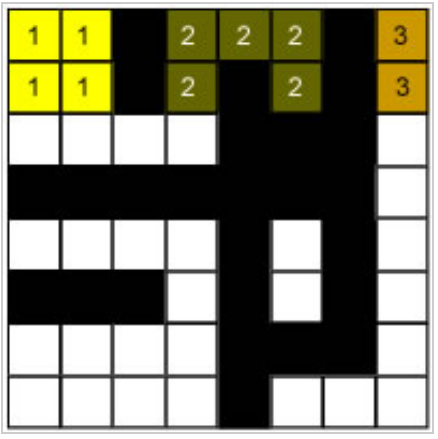
Now the next two pixels, (1, 5) and(1, 6) will have pixel to their left. So, they get the label 2 as well. (1, 7) is background so we ignore it. At (1,8) we'll have to create a new label again (pixel above does not exist, and pixel to left is a background pixel).

So, after completing row 1, we have a result that looks like this:



After row 1 is complete

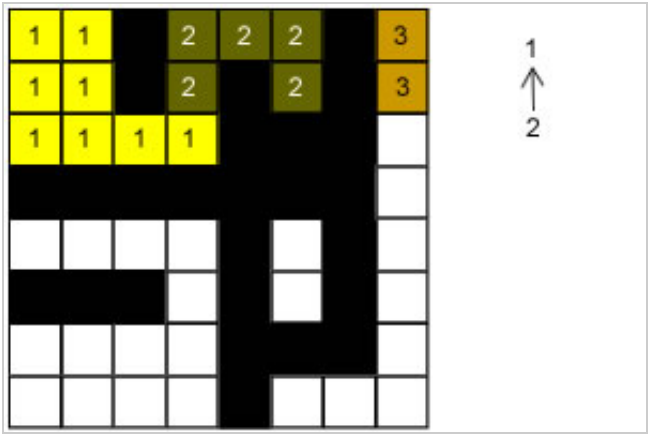
Now the pixel (2, 1) does not have anything to its left. But it does have a pixel just above it. Its label is '1', so we copy that. Similarly for (2, 2). Infact, the same thing holds for the entire row. All pixels in the second row have a pixel just above them. So the result looks like this:



Completion of row 2

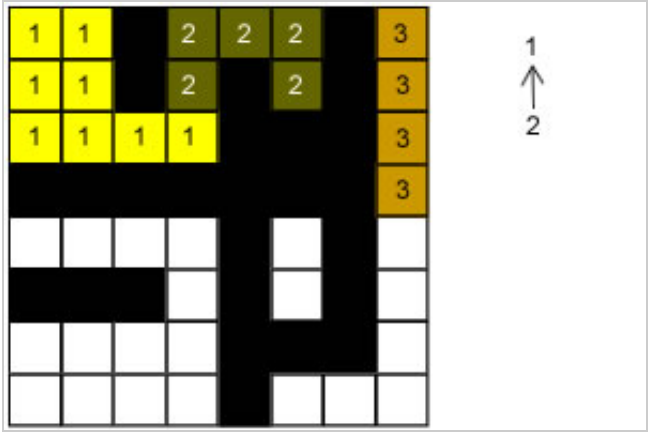
In the third row, pixels (3, 1) (3,2) and (3, 3) are quite straight forward. They get the label ’1’. But pixel (3, 4) is a tricky one. You have pixel both above and to the left. And both have different labels. Which one do you choose? How do you handle the sudden “realization” of the connectedness of labels 1 and 2.

Well, you take the smaller label (in this case ’1’) and put that on (3, 4). And, you also store that 2 (the numerically larger label) is a child of 1 (using the union-find data structure).

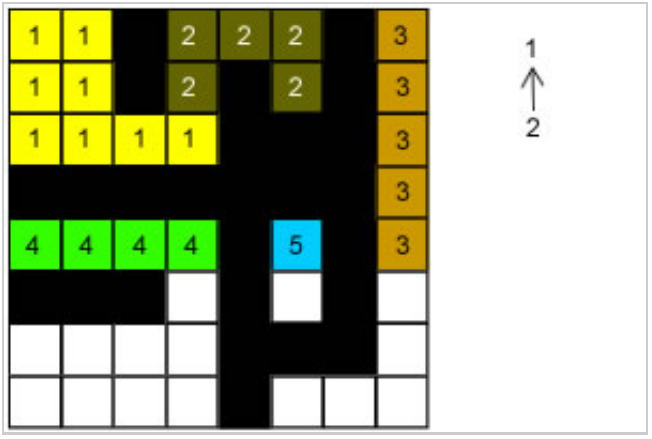


Resolving the dilemma at (3, 4)

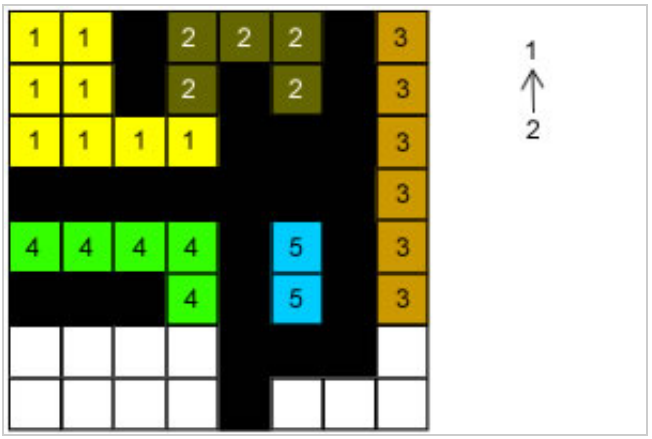
Continuing on, you should face no difficulties. Here’s a set of images to guide you along to the final result of pass 1.



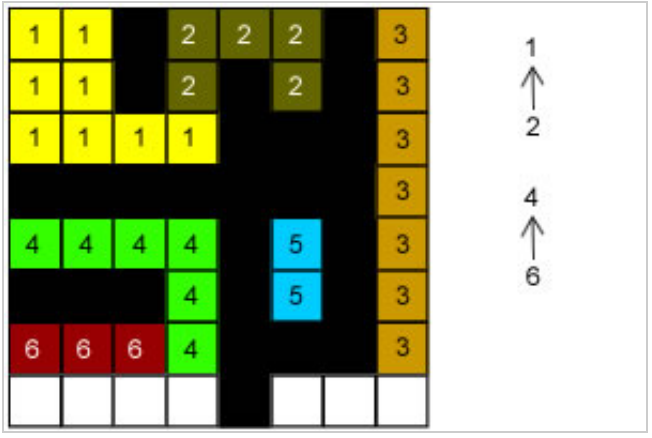
After 4 rows



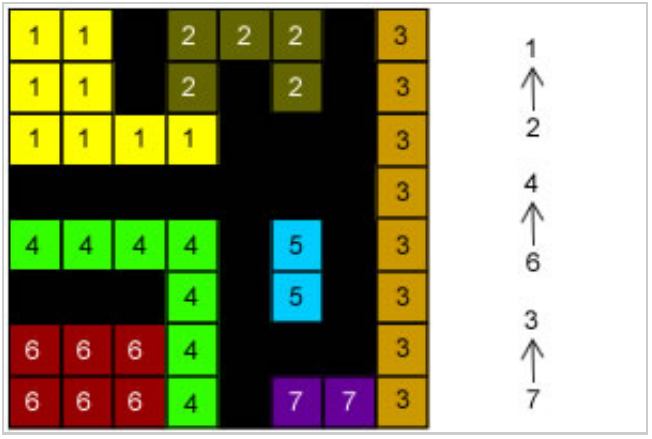
Completing 5 rows



Row 6, done



Completing row 7, note the new label



First pass complete. Note the new label.

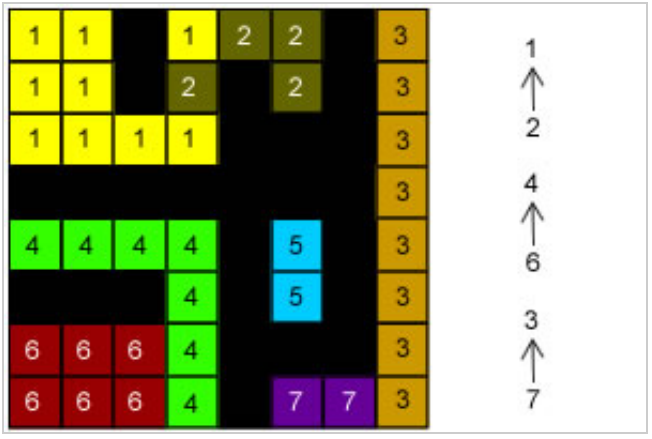
The second pass

Now, you can see this labeling technique has made quite a mess. Multiple labels for connected regions. We need to rectify this.

To do this, we again go through every pixel one by one. It starts with pixel (1, 1). It checks the union-find data structure for the label ’1’. On checking, it figures out that ’1’ is not a child of any other labels. It is a root itself. So it moves on.

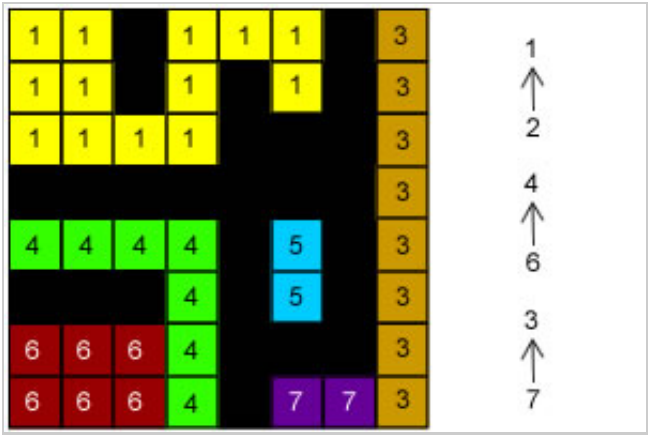
Similarly for (1, 2). For the pixel (1, 3) we’re not concerned because it is a background pixel.

Now on pixel (1,4) it checks the data structure for ’2’. It notices that ’2’ is a child of ’1’. Then, it checks for ’1’. It notices ’1’ is a root. So, it replaces the label of (1, 4) with ’1’:



Converting a '2' into a '1'

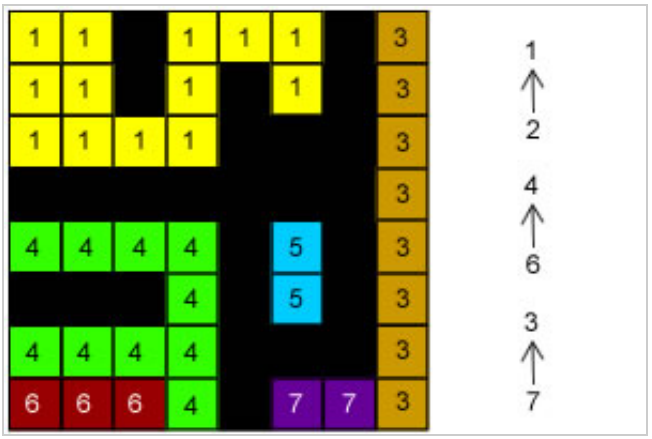
Similarly, the process goes on for the entire row. The result after completing 4 rows is:



Completing till row 4

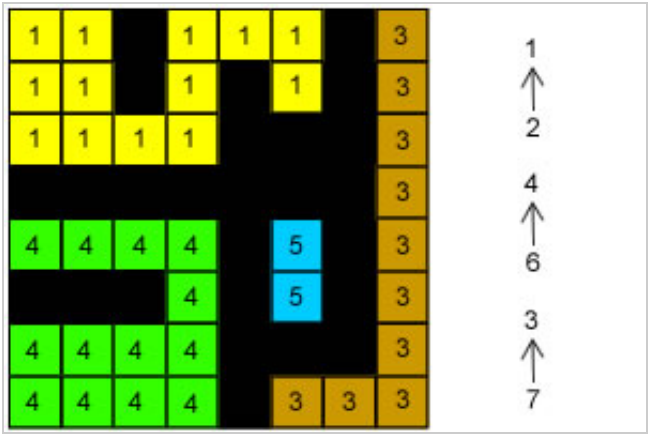
We've made progress! The connected region on the top right is fully cleaned up. Great! Now, row 5 remains unchanged. '4' is a root label. So is '5'. And '3' as well. Row 6 also remains unchanged.

Now, row 7 undergoes changes:



Completing row 7

And after doing row 8, here's the final result:



The final result

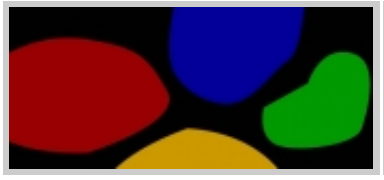
And thus you have the final result. And its the result we want. Every connected region has exactly one value.

Done!

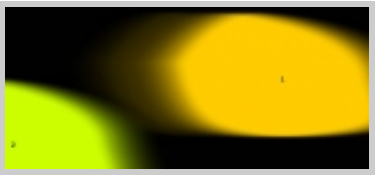
Well that was a pretty in depth example. Hope you learned a bit from this!

Issues? Suggestions? Visit the Github issue tracker for AI Shack

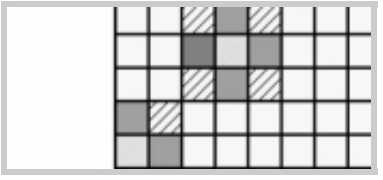
Back to top



Connected Component Labelling



Fast connected components labeling



Pixel neighbourhoods and connectedness

5 Comments

Ajusal
Posted June 7, 2010 at 8:20 pm | *Permalink*

Well explained man…
Gud for starters…especially to understand da basic fundas relating to blobs n connected components n how to detec n label dem…
Gud job

Reply

Bishant
Posted September 9, 2010 at 1:18 am | *Permalink*

Nice Explanations ! Loved to read it.

Keep it up. Really nice openCV tutorials.

Reply

Utkarsh
Posted September 20, 2010 at 12:03 am | *Permalink*

Hi Bishant! Glad you liked this!

Reply

Parisa
Posted June 27, 2011 at 12:59 pm | *Permalink*

Great Explanation! thanks

Reply

dora
Posted April 25, 2014 at 9:51 pm | *Permalink*

you make things easier to understand, thanks man

Reply

Post a Comment

Your email is *never* published nor shared. Required fields are marked *

Name *

Email *

Website

Comment

You may use these HTML tags and attributes `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`

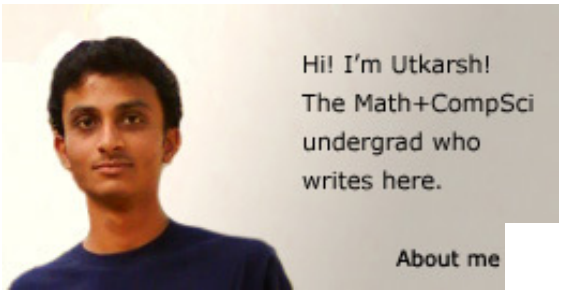
Post Comment

Free updates

Email Address

More info

1000+ subscribers!



Popular Posts

- Scanning QR Codes
- The Canny Edge Detector
- Implementing Canny Edges from scratch
- Image Moments
- Predator: Tracking + Learning

More on AI SHack

Popular **Beginners** **Utkarsh's favs**

About me

My name is Utkarsh Sinha, and I'm an undergraduate student, pursuing B.E. Computer Science + M.Sc. Mathematics. Here, I help you understand ideas in Artificial Intelligence, using a not so techy and mathematical language. And in the process, learn more about Artificial Intelligence myself.

Read more at the about page