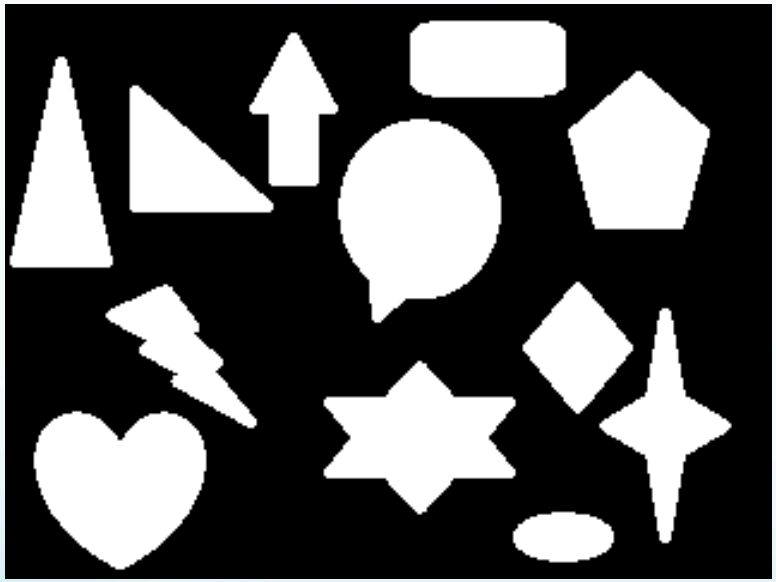
Labeling
Assignment 9

Gwenaelle Cunha Sergio
ABR Lab – BEP
KNU 2014.1

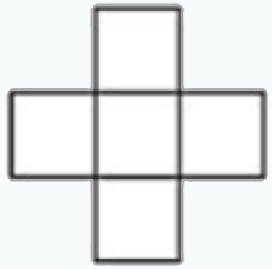
Labeling



Grass fire

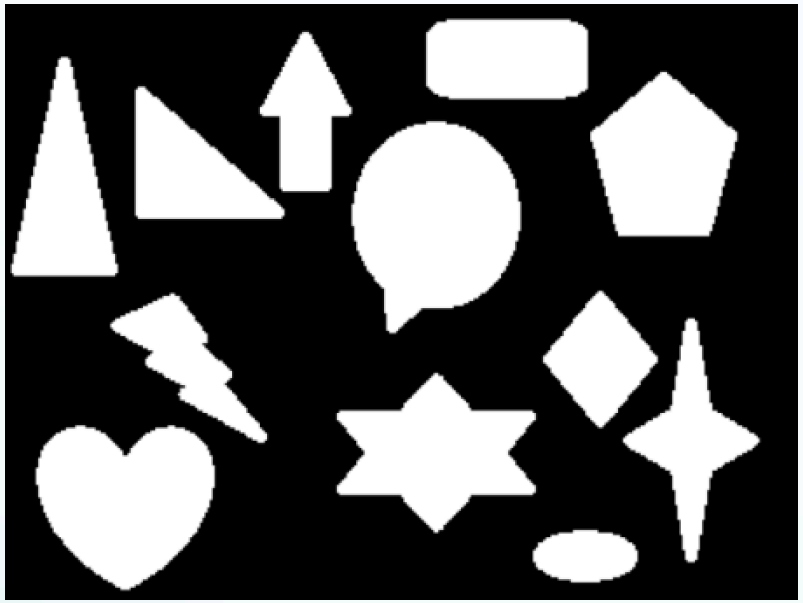
- Search the image until a pixel with a color different than that of the background is found and push it in the stack
- Look for neighbors belonging to that blob and also add them to the stack
- Label all pixels in that group with a specific number
- Return the new colored image

Grass fire - Code

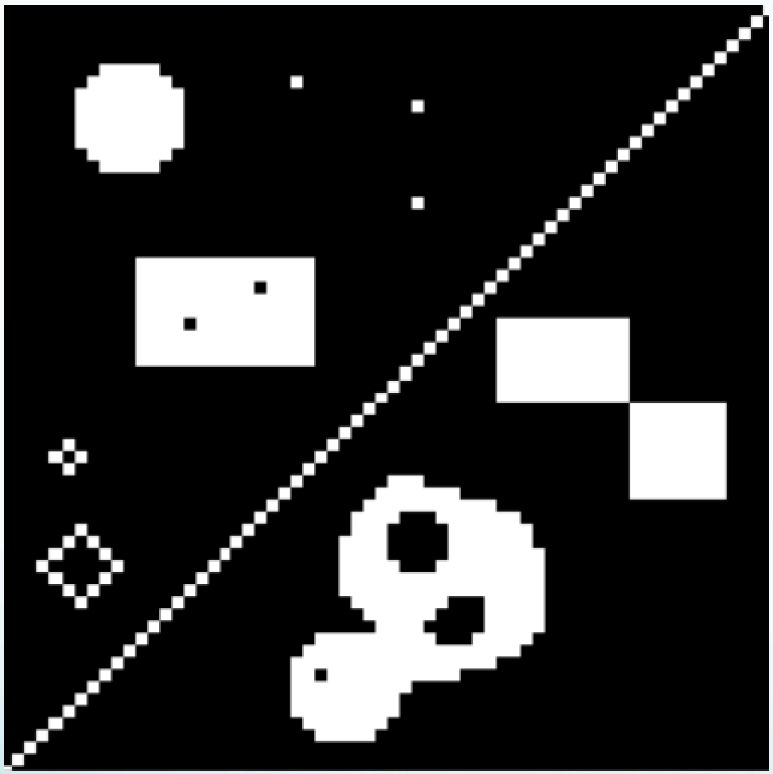


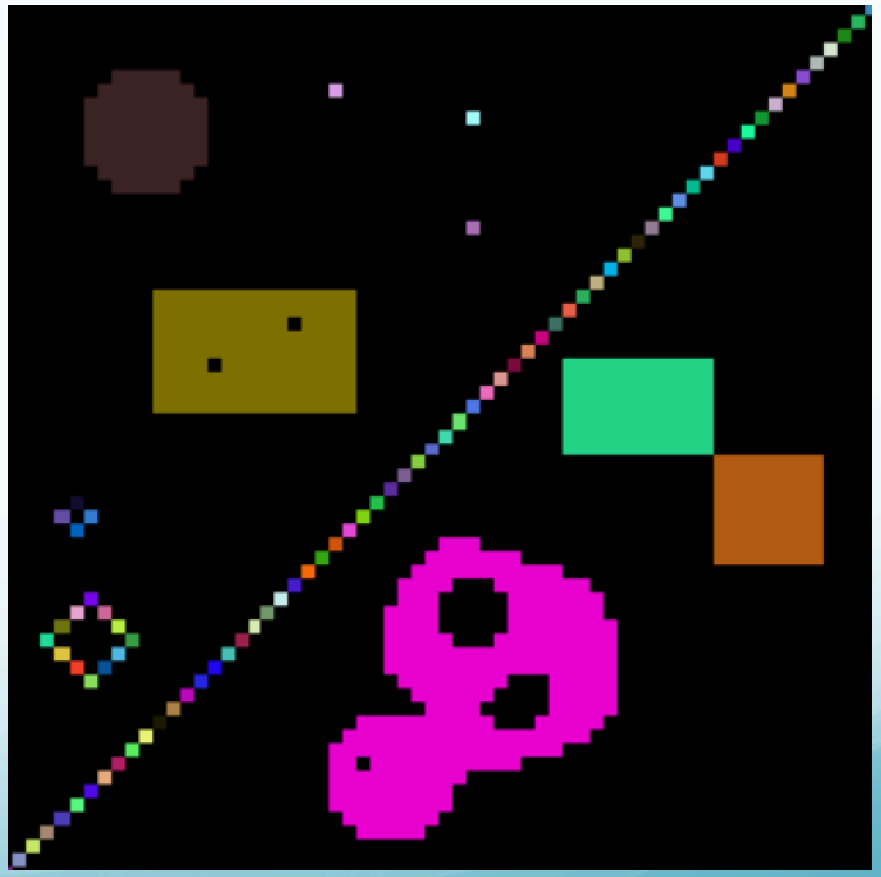
```
img4 = setAllPixelsToZero(img4);
int group = 0; Istack s;
for (int i = 0; i < mat.rows; i++) {
    for (int j = 0; j < mat.cols; j++) {
        if (mt.at<uchar>(i,j) != 0 & img4.at<uchar>(i,j) == 0) {
            group++;
            s.push(i, j, group);
            int itmp = i, jtmp = j;
            while (!s.isEmpty()) {
                    position pos = s.pop();
                    itmp = pos.i; jtmp = pos.j;
                    img4.at<uchar>(itmp,jtmp) = group;
                    if (itmp-1 >= 0) { //up
                        if (img4.at<uchar>(itmp-1,jtmp) == 0 & mat.at<uchar>(itmp-1,jtmp) != 0)
                            s.push(itmp-1, jtmp, group);
                    }
                    if (itmp+1 < mat.rows) { //down
                        if (img4.at<uchar>(itmp+1,jtmp) == 0 & mat.at<uchar>(itmp+1,jtmp) != 0)
                            s.push(itmp+1, jtmp, group);
                    }
                    if (jtmp-1 >= 0) { //right
                        if (img4.at<uchar>(itmp,jtmp-1) == 0 & mat.at<uchar>(itmp,jtmp-1) != 0)
                            s.push(itmp, jtmp-1, group);
                    }
                    if (jtmp+1 < mat.cols) { //left
                        if (img4.at<uchar>(itmp,jtmp+1) == 0 & mat.at<uchar>(itmp,jtmp+1) != 0)
                            s.push(itmp, jtmp+1, group);
                    }
                }
            }
        }
    }
}
return coloredLabels(img4, group); //Show detected classes with colored labels
```

Grass fire - Result



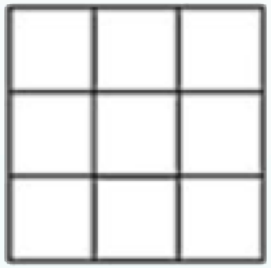
Grass fire - Result





Grass fire: 8-neighbors

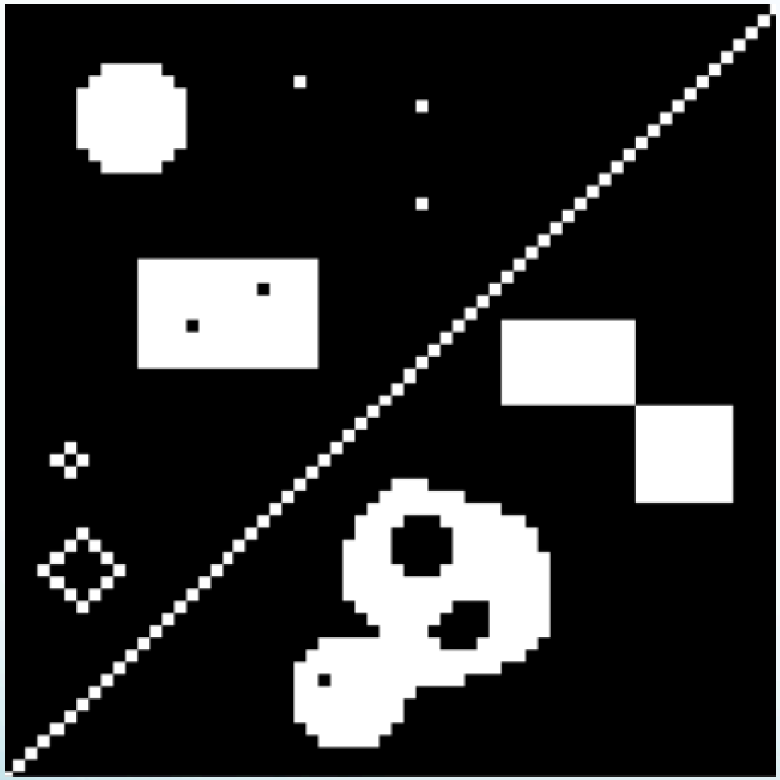
- Difference: mask

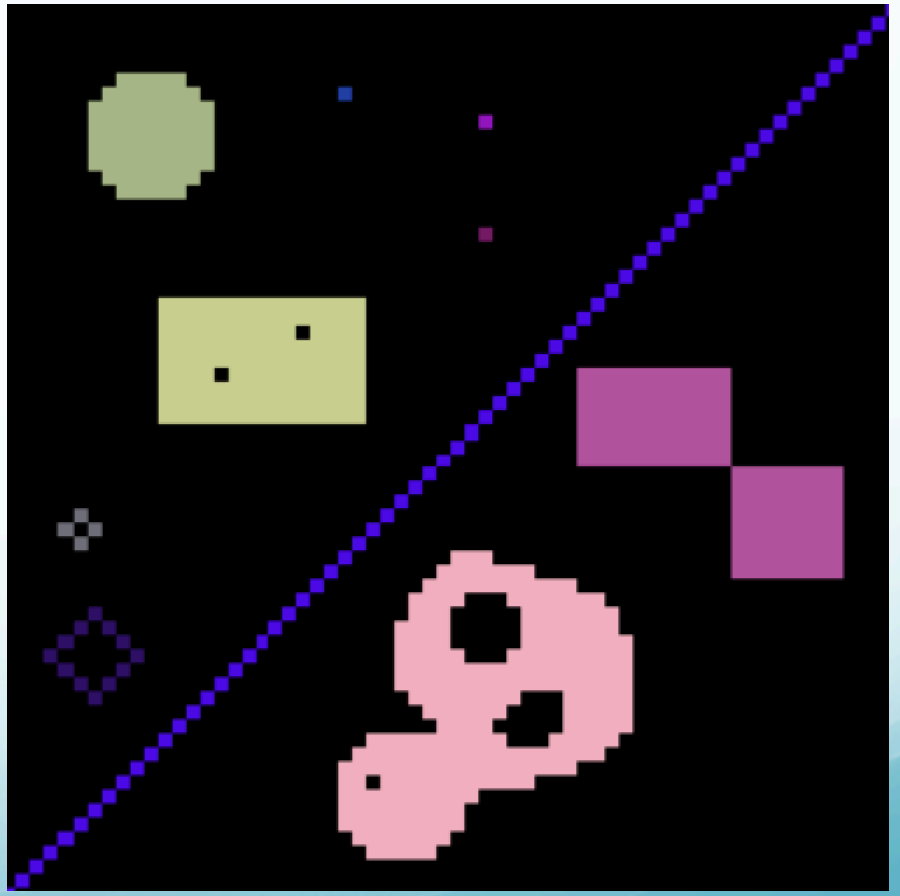


Grass fire - Code

```
img4 = setAllPixelsToZero(img4);
int group = 0;
Istack s;
for (int i = 0; i < mat.rows; i++) {
    for (int j = 0; j < mat.cols; j++) {
        if (mat.at<uchar>(i,j) != 0 & img8.at<uchar>(i,j) == 0) {
            group++;
            s.push(i, j, group);
            int itmp = i, jtmp = j;
            while (!s.isEmpty()) {
                position pos = s.pop();
                itmp = pos.i;
                jtmp = pos.j;
                img8.at<uchar>(itmp,jtmp) = group;
                for (int itmp = pos.i-1; itmp <= pos.i+1; itmp++) {
                    for (int jtmp = pos.j-1; jtmp <= pos.j+1; jtmp++) {
                        if (itmp >= 0 & itmp < mat.rows & jtmp >= 0 & jtmp < mat.cols) {
                            if (img8.at<uchar>(itmp,jtmp) == 0 & mat.at<uchar>(itmp,jtmp) != 0) {
                                s.push(itmp, jtmp, group);
                            }
                        }
                    }
                }
            }
        }
    }
}
return coloredLabels(img8, group); //Show detected classes with colored labels
```

Grass fire - Result





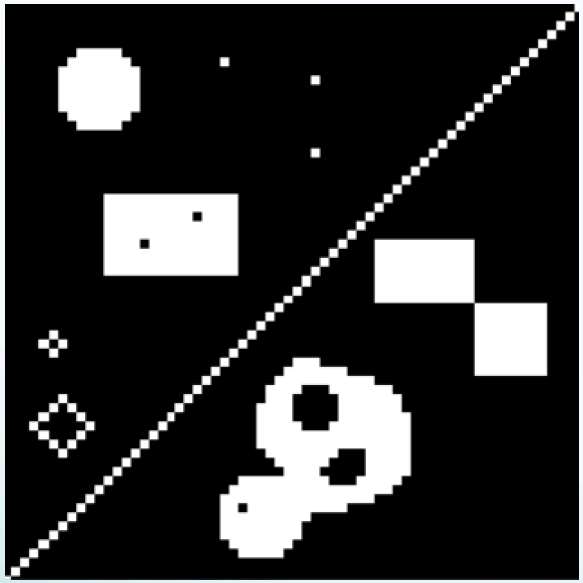
2-pass

- On the **first** pass:
  - Iterate through each element of the data by column, then by row (Raster Scanning)
  - If the element is not the background
    - Get the neighboring elements of the current element
    - If there are no neighbors, uniquely label the current element and **continue**
    - Otherwise, find the neighbor with the smallest label and assign it to the current element
    - Store the equivalence between neighboring labels

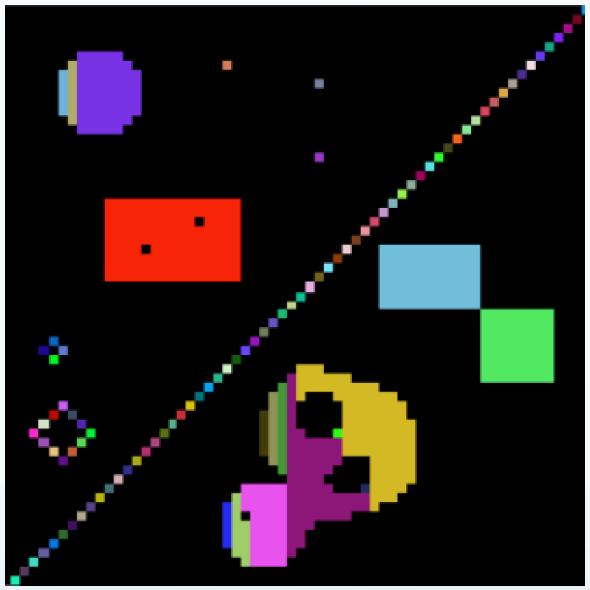
2-pass

- On the **second** pass:
  - Iterate through each element of the data by column, then by row
  - If the element is not the background
    - Relabel the element with the lowest equivalent label

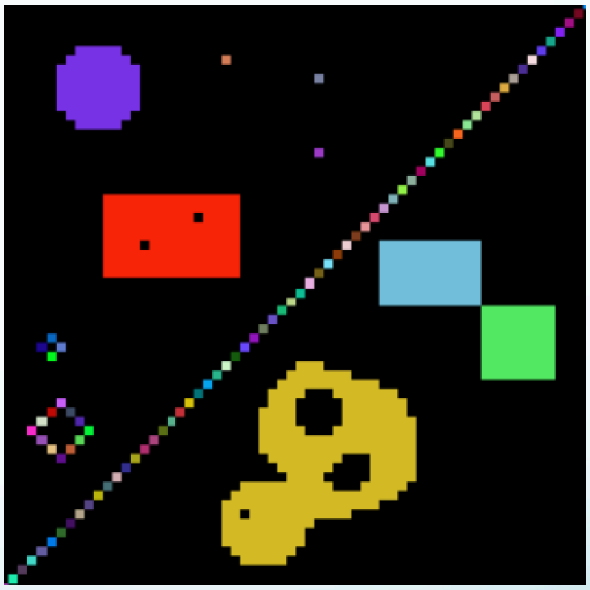
2-pass 4 - Result



Original

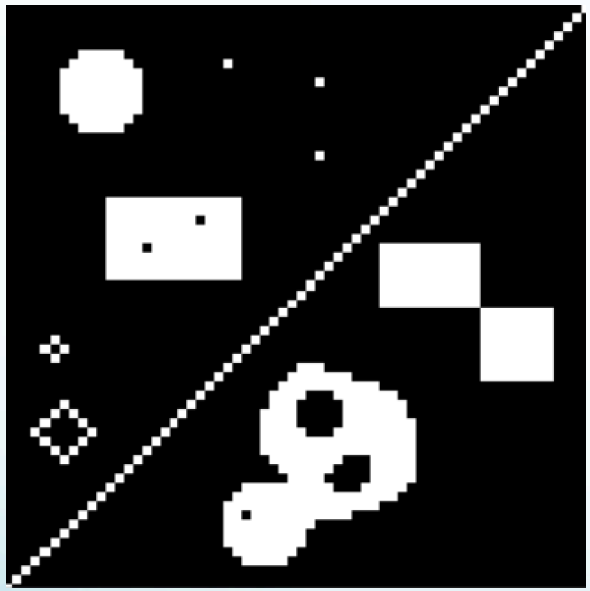


Phase 1

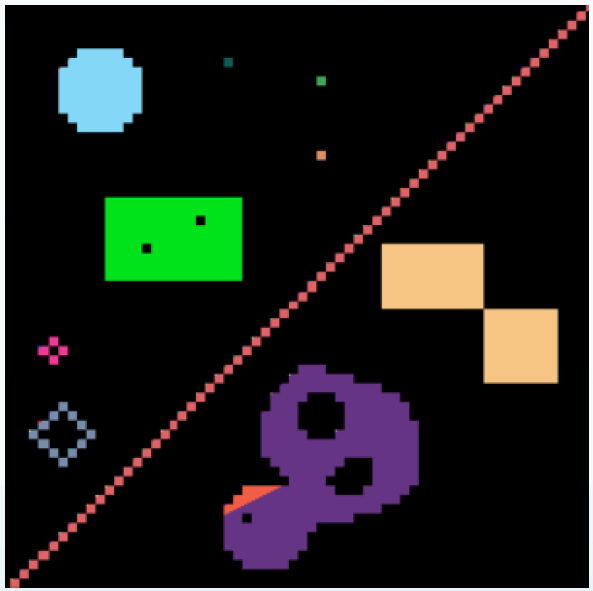


Phase 2

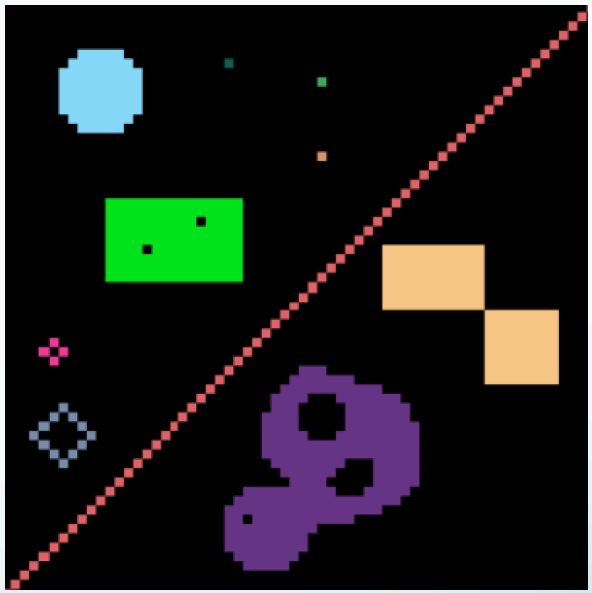
2-pass 8 - Result



Original



Phase 1



Phase 2

Stack Overflow

- What is it?
  - “In software, a **stack overflow** occurs when the stack pointer exceeds the stack bound.”
- How to solve it?
  - Increasing stack size
  - Project -> Properties -> Configuration Properties -> Linker -> System -> Stack Reserve Size
  - Do NOT use /F <num>
  - USE /stack option
  - In linux, the maximum is 8 000 000 (~8 MByte)