ROTULAÇÃO

Algoritmo Recursivo

Negar a imagem de entrada (Trocar 1 por -1)

Percorra a matriz da esquerda para a direita, de cima para baixo. Se um pixel **P** for igual a -1, chame a função rotula_componente(**P**, *label*)

Considere **C** o vizinho acima de **P**, **B** o vizinho abaixo de **P**, **E** o vizinho esquerdo e **D** o vizinho direito

```
rotula_componente(P, label) {
    P = label;
    if(C == -1)
        rotula_componente(C, label);
    if(B == -1)
        rotula_componente(B, label);
    if(E == -1)
        rotula_componente(E, label);
    if(D == -1)
        rotula_componente(D, label);
}
```

ROTULAÇÃO

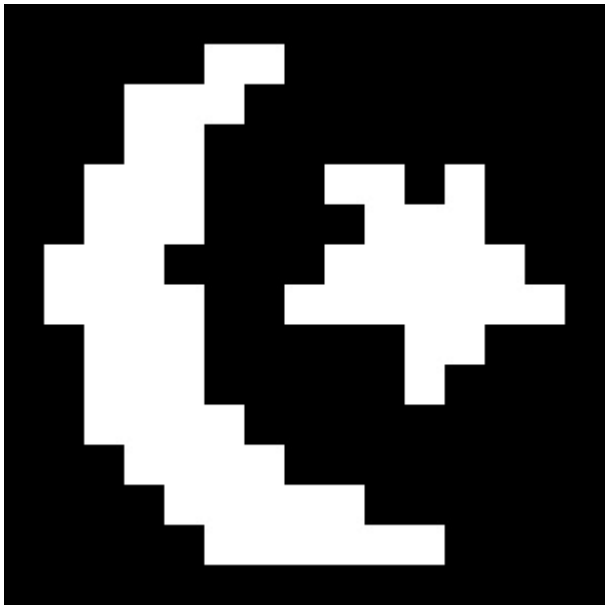
Algoritmo Iterativo

Percorra a matriz da esquerda para a direita, de cima para baixo. Para cada pixel **P**, considere **C** o vizinho acima e **E** o vizinho esquerdo.

- Se **P** for igual a 1
  - Se **C** e **E** forem iguais a 0, **P** recebe um novo rótulo
  - Se apenas um entre **C** e **E** for igual a 1, **P** recebe o rótulo deste vizinho
  - Se **C** e **E** forem iguais a 1 e tiverem o mesmo rótulo, **P** recebe este rótulo
  - Se **C** e **E** forem iguais a 1, mas tiverem rótulos diferentes, **P** recebe um dos rótulos, e os dois rótulos são marcados como equivalentes.

No final atribuir um mesmo rótulo aos rótulos equivalentes.

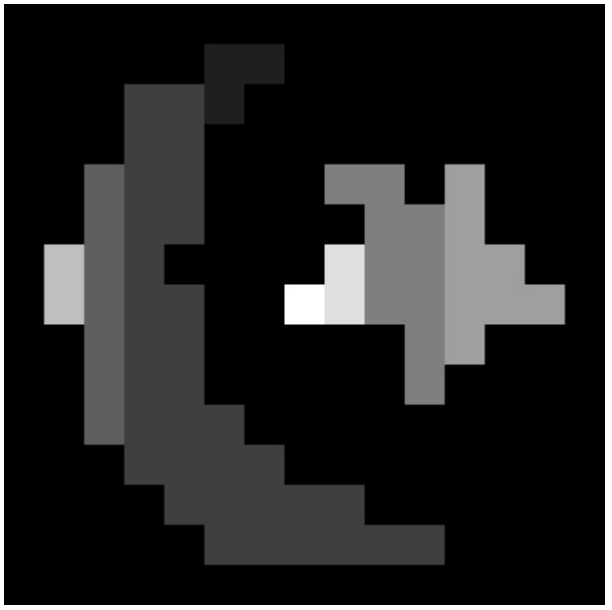
ROTULAÇÃO



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

Imagem de entrada.

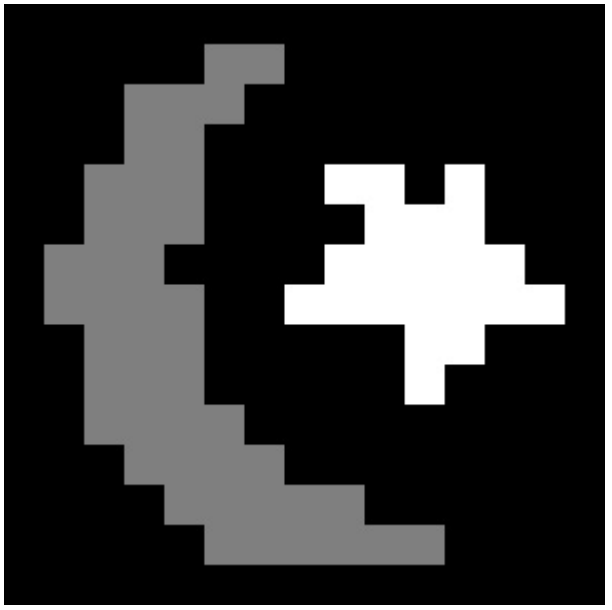
ROTULAÇÃO



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	2	2	1	0	0	0	0	0	0	0	0	0
0	0	0	2	2	0	0	0	0	0	0	0	0	0	0
0	0	3	2	2	0	0	0	4	4	0	5	0	0	0
0	0	3	2	2	0	0	0	0	4	4	5	0	0	0
0	6	3	2	0	0	0	0	7	4	4	5	5	0	0
0	6	3	2	2	0	0	8	7	4	4	5	5	5	0
0	0	3	2	2	0	0	0	0	0	4	5	0	0	0
0	0	3	2	2	0	0	0	0	0	4	0	0	0	0
0	0	3	2	2	2	0	0	0	0	0	0	0	0	0
0	0	0	2	2	2	2	0	0	0	0	0	0	0	0
0	0	0	0	2	2	2	2	2	0	0	0	0	0	0
0	0	0	0	0	2	2	2	2	2	2	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Imagem rotulada sem acerto de rótulos.

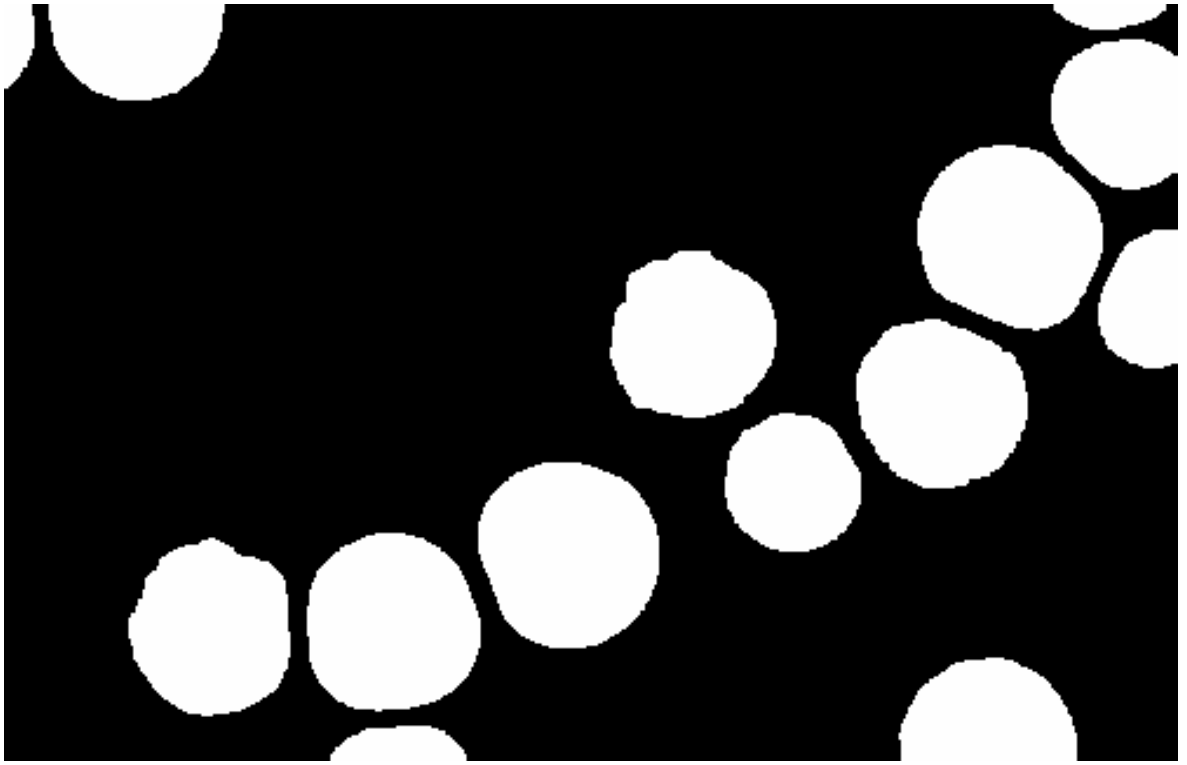
ROTULAÇÃO



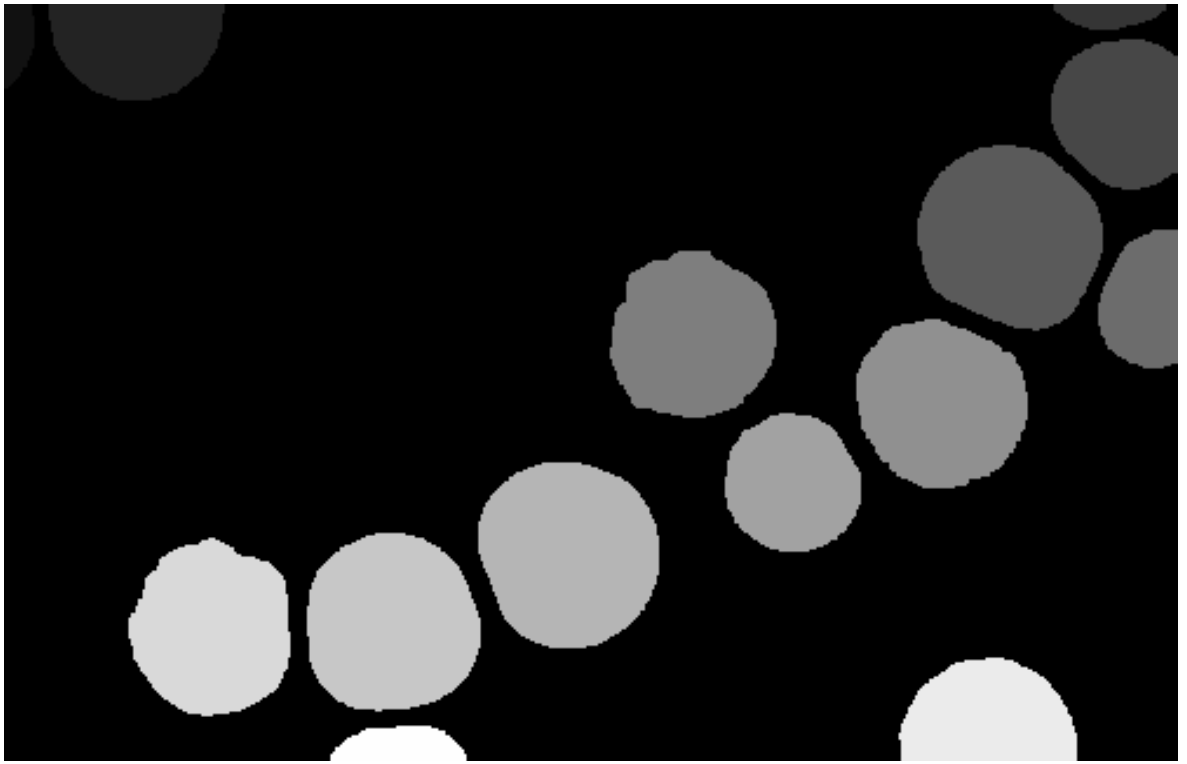
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	2	2	0	2	0	0	0
0	0	1	1	1	0	0	0	0	2	2	2	0	0	0
0	1	1	1	0	0	0	0	2	2	2	2	2	0	0
0	1	1	1	1	0	0	2	2	2	2	2	2	2	0
0	0	1	1	1	0	0	0	0	0	2	2	0	0	0
0	0	1	1	1	0	0	0	0	0	2	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Imagem rotulada
final.

ROTULAÇÃO



ROTULAÇÃO

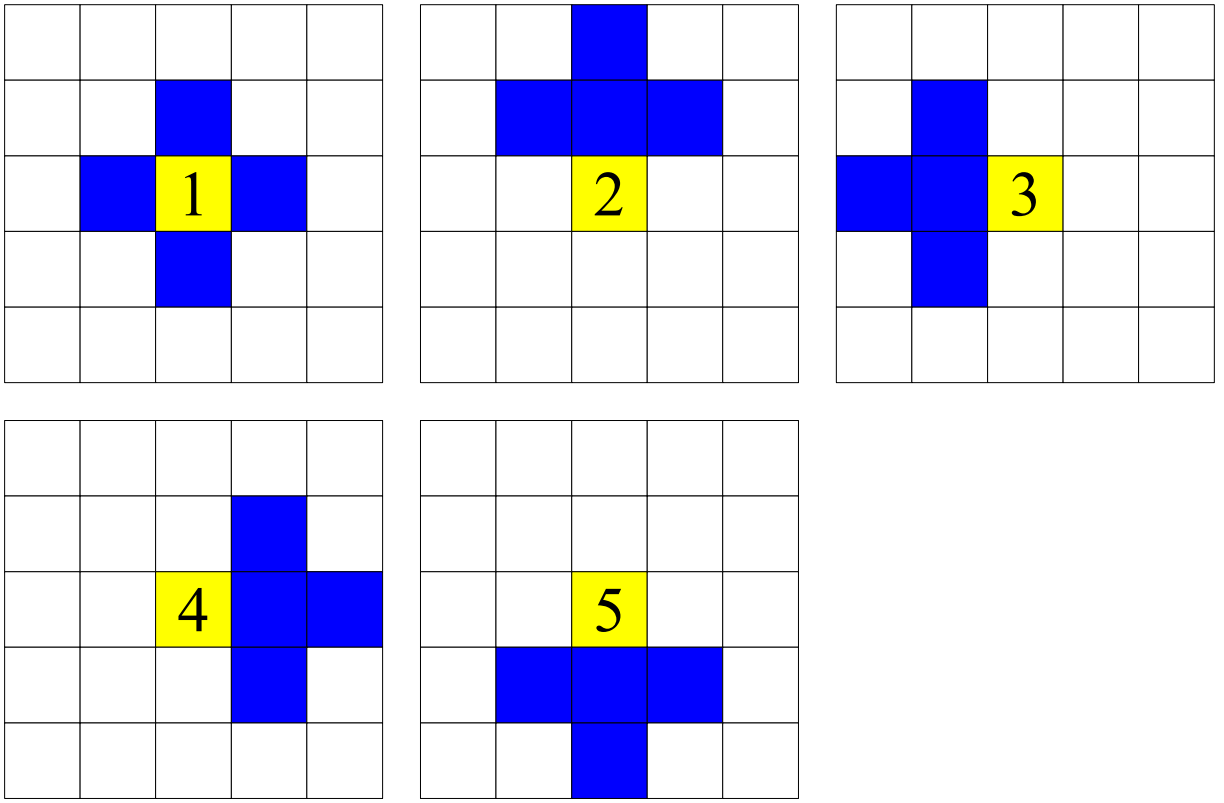


EXERCÍCIOS

A rotulação pode ser implementada de duas maneiras diferentes: recursivamente e iterativamente. Considerando que a imagem contém apenas uma componente conexa, um quadrado de 100x100 pixels, qual a quantidade de acessos a cada pixel que cada abordagem realiza?

EXERCÍCIOS

Algoritmo Recursivo = 5 acessos



EXERCÍCIOS

Algoritmo Iterativo = 3 acessos