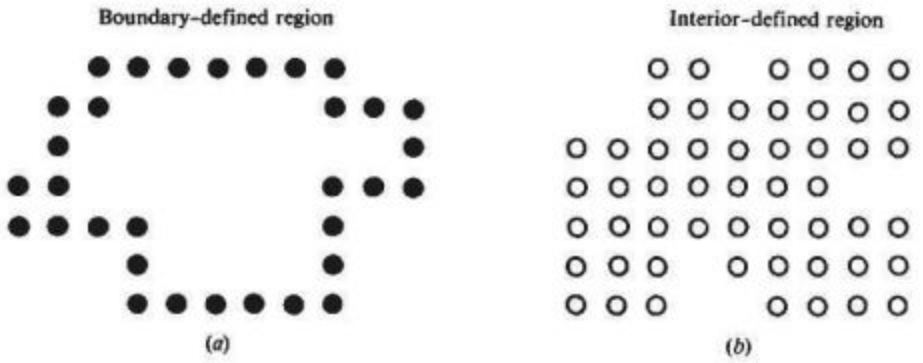
CSE - 321
Computer graphics
Lecture-6
(Region Filling, Character Display

Region Filling

- Region filling is the process of "coloring in" a definite image area or region.
- Regions may be defined at the pixel or geometric level.
- At the pixel level, we describe a region either in terms of the bounding pixels that outline it or as the totality of pixels that comprise it.
- In the first case the region is called boundary-defined and the collection of algorithms used for filling such a region are collectively called boundary-fill algorithms.
- The other type of region is called an interior-defined region and the accompanying algorithms are called flood-fill algorithms.

Region Filling



- At the geometric level a region is defined or enclosed by such abstract contouring elements as connected lines and curves.
- For example, a polygonal region, or a filled polygon, is defined by a closed polyline, which is a polyline (i.e., a series of sequentially connected lines) that has the end of the last line connected to the beginning of the first line.

Region Filling

- There are two ways in which pixels are considered connected to each other to form a "continuous" boundary.
- One method is called 4-connected, where a pixel may have up to four neighbors. Fig-a
- The other is called 8-connected, where a pixel may have up to eight neighbors . Fig-b

Fig-a

Fig-b

Boundary Fill Algorithm

- This is a recursive algorithm that begins with a starting pixel, called a seed, inside the region.
- The algorithm checks to see if
  - this pixel is a boundary pixel or
  - has already been filled.

If the answer is no, it fills the pixel and makes a recursive call to itself using each and every neighboring pixel as a new seed.

If the answer is yes, the algorithm simply returns to its caller

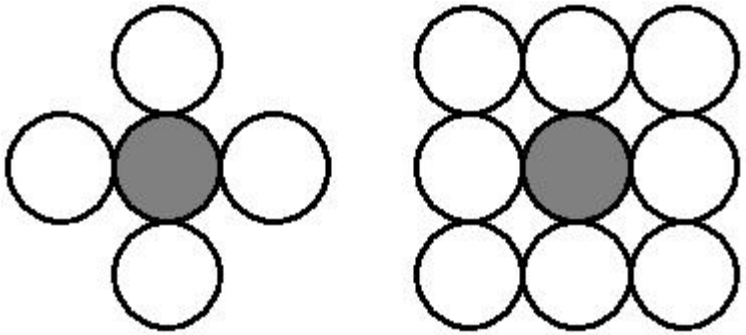
Boundary Fill Algorithm

The following steps illustrate the idea of the **recursive** boundary-fill algorithm:

1. Start from an interior point.
2. If the current pixel is **not already** filled and if it is not an edge point, then set the pixel with the fill color, and store its neighboring pixels (**4** or **8-connected**) in the stack for processing. Store only neighboring pixel that is **not already** filled and is not an edge point.
3. Select the next pixel from the stack, and continue with step **2**.

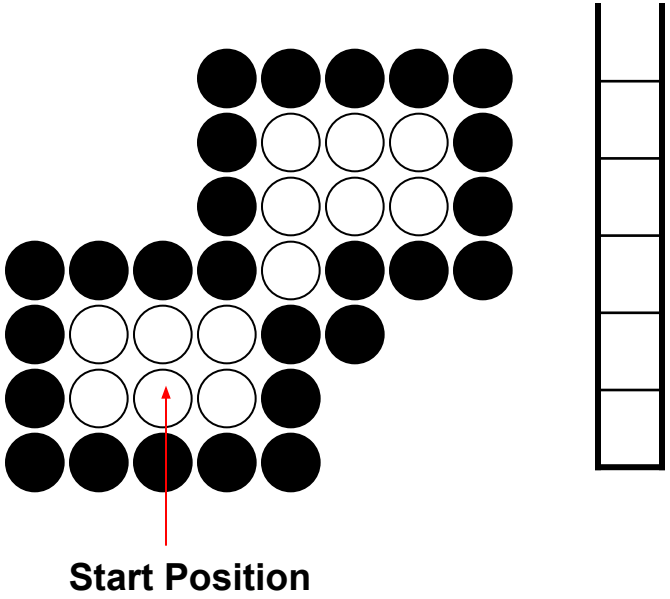
Boundary Fill Algorithm

The order of pixels that should be added to stack using **4-connected** is above, below, left, and right. For **8-connected** is above, below, left, right, above-left, above-right, below-left, and below-right.



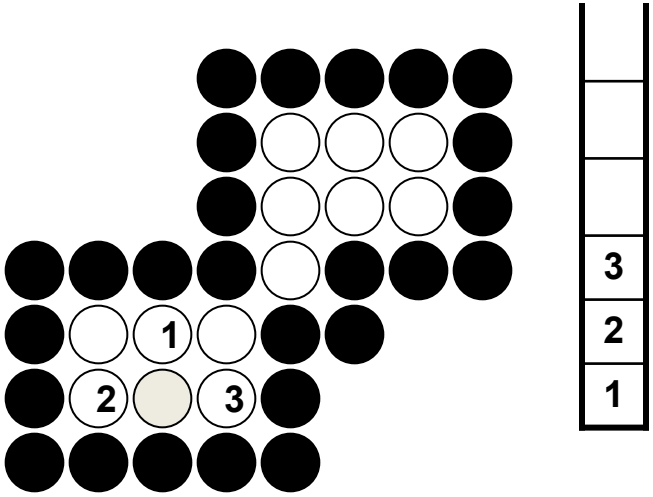
Boundary Fill Algorithm

4-connected (Example)



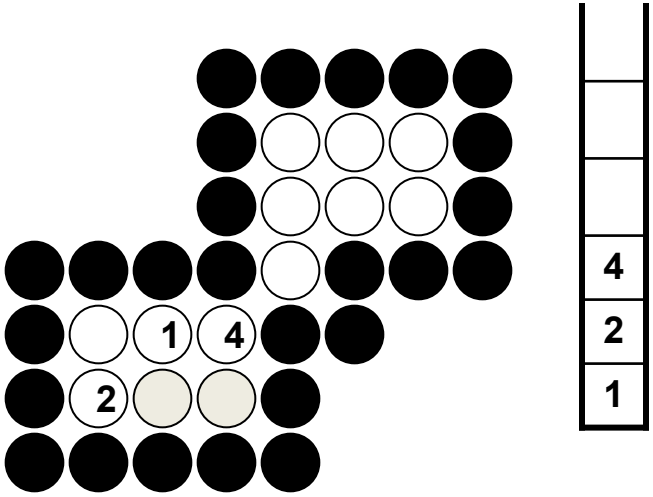
Boundary Fill Algorithm

4-connected (Example)



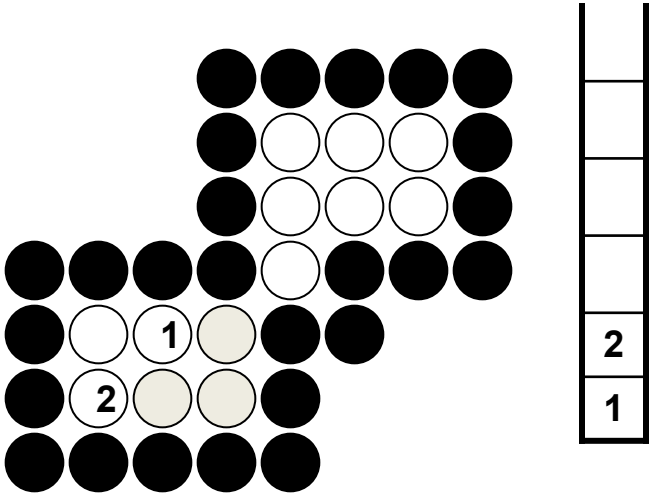
Boundary Fill Algorithm

4-connected (Example)



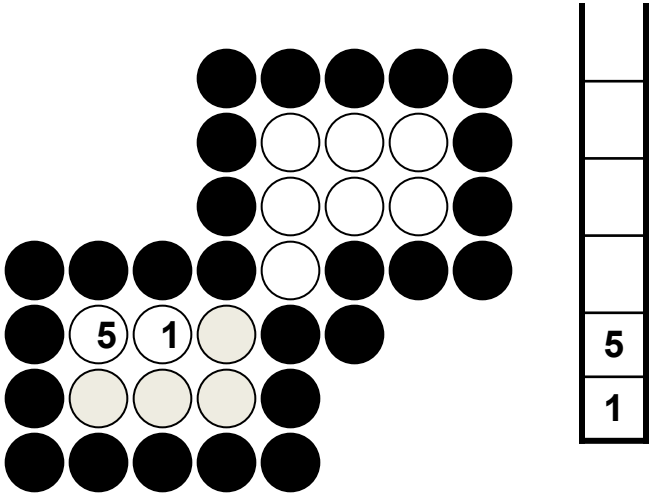
Boundary Fill Algorithm

4-connected (Example)



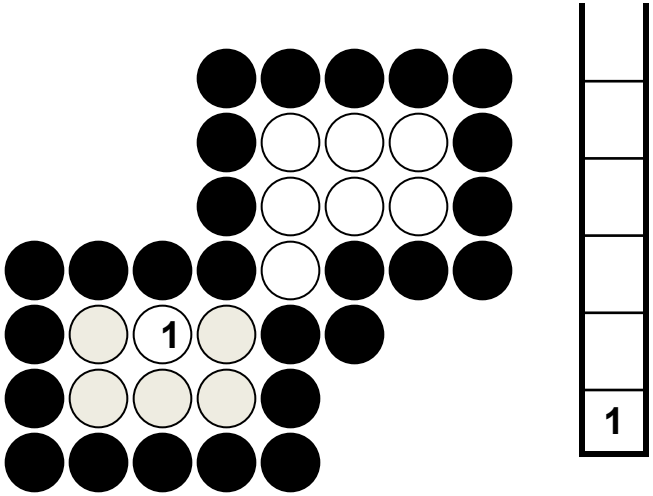
Boundary Fill Algorithm

4-connected (Example)



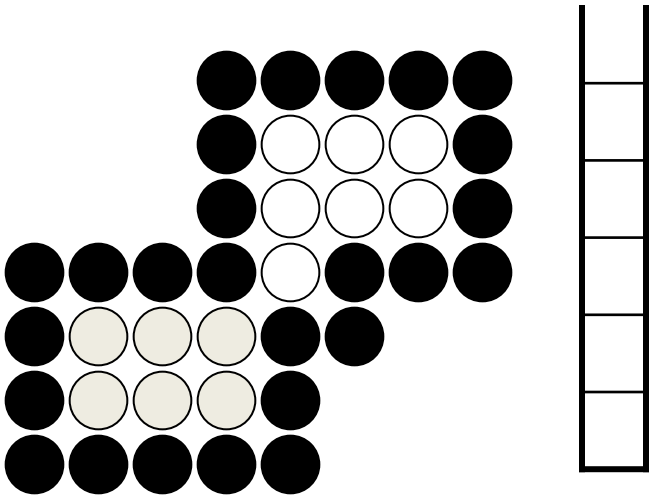
Boundary Fill Algorithm

4-connected (Example)



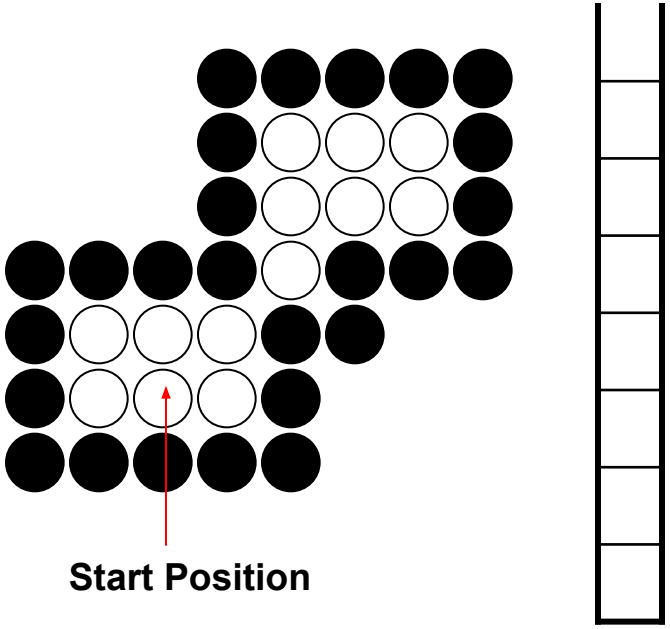
Boundary Fill Algorithm

4-connected (Example)



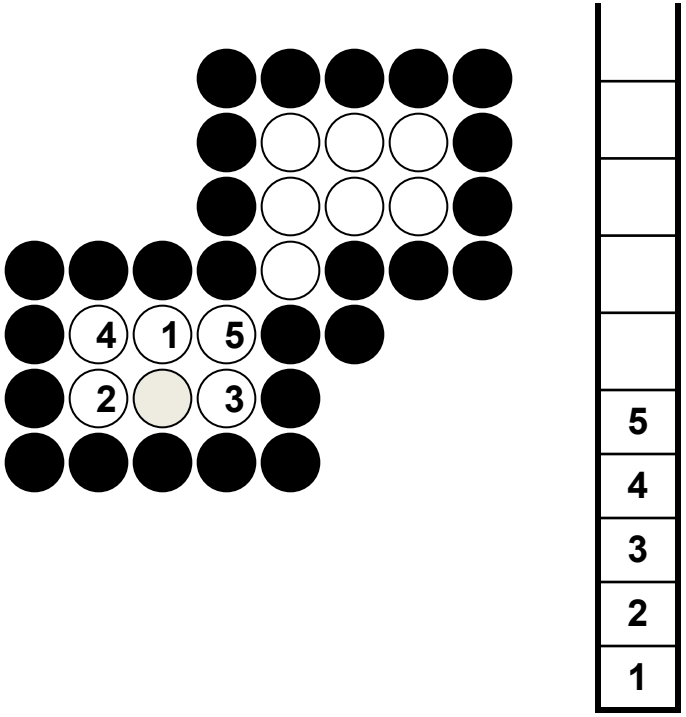
Boundary Fill Algorithm

8-connected (Example)



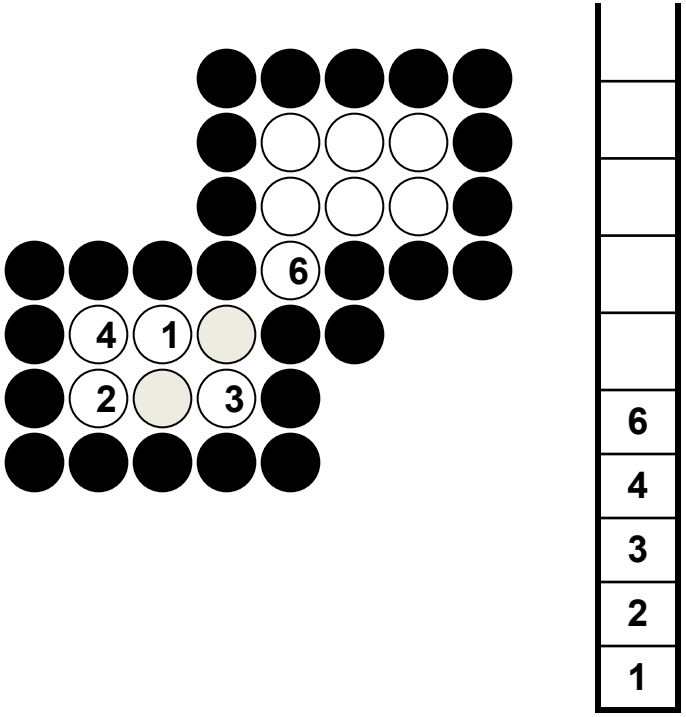
Boundary Fill Algorithm

8-connected (Example)



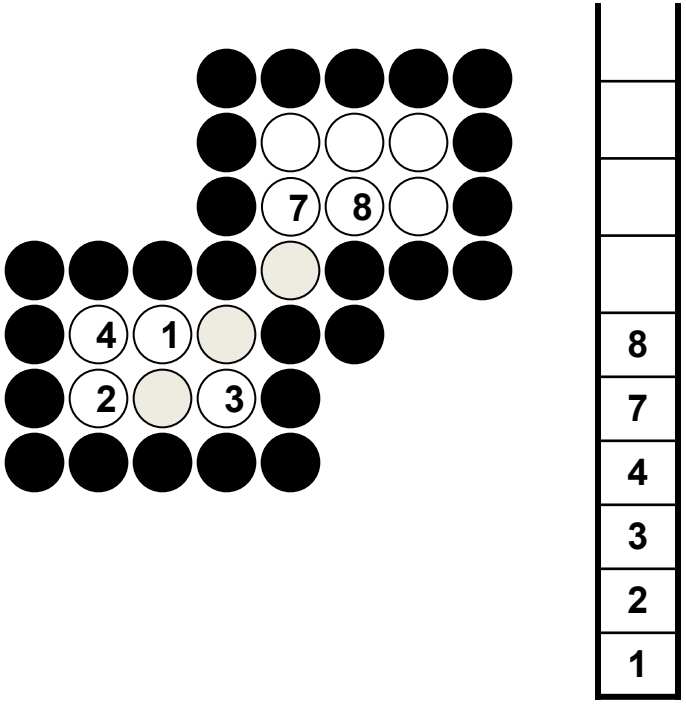
Boundary Fill Algorithm

8-connected (Example)



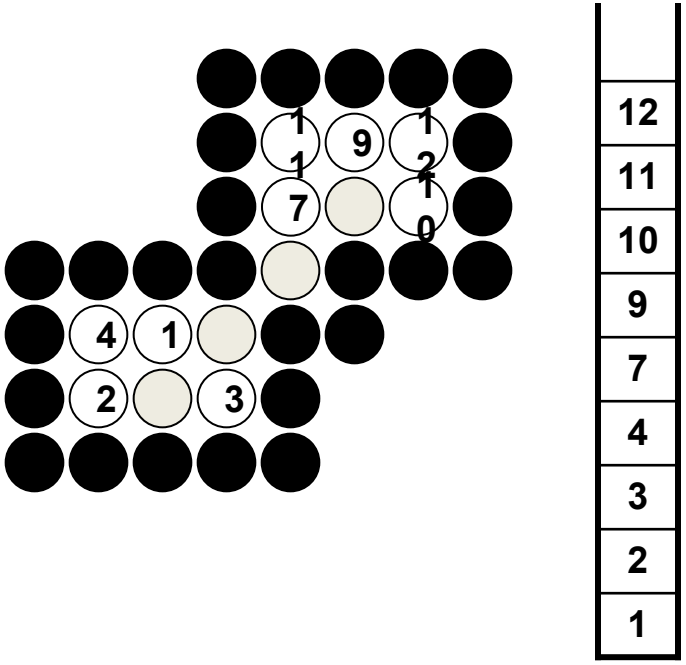
Boundary Fill Algorithm

8-connected (Example)



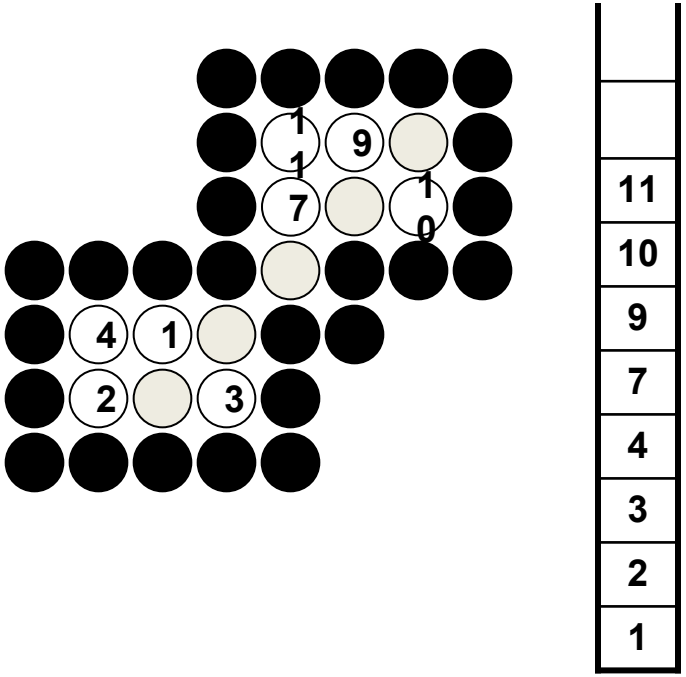
Boundary Fill Algorithm

8-connected (Example)



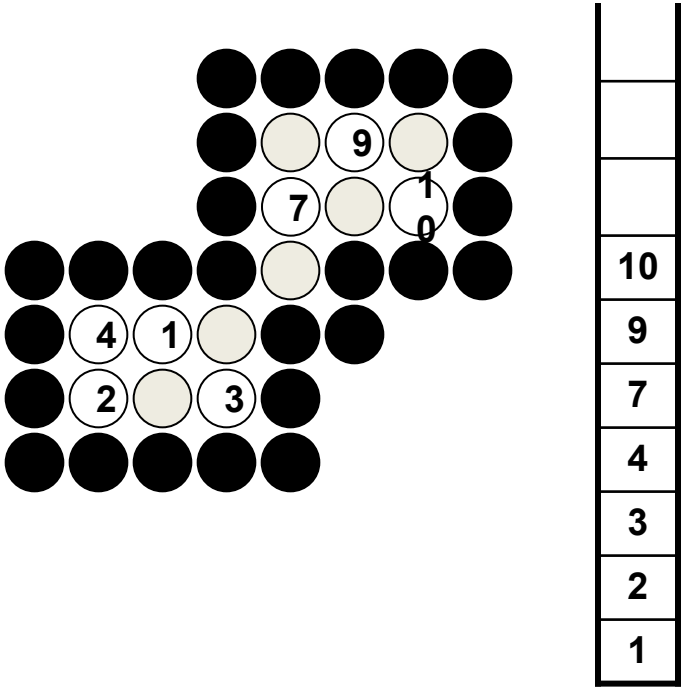
Boundary Fill Algorithm

8-connected (Example)



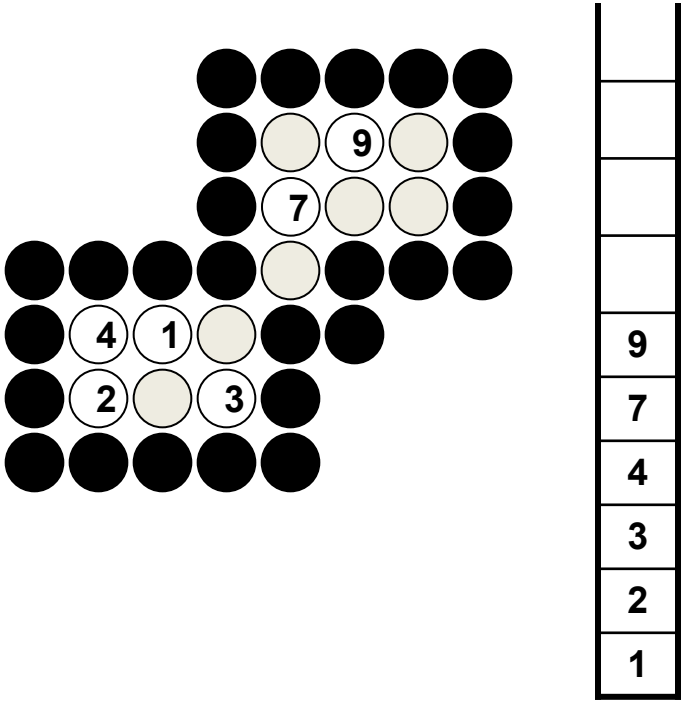
Boundary Fill Algorithm

8-connected (Example)



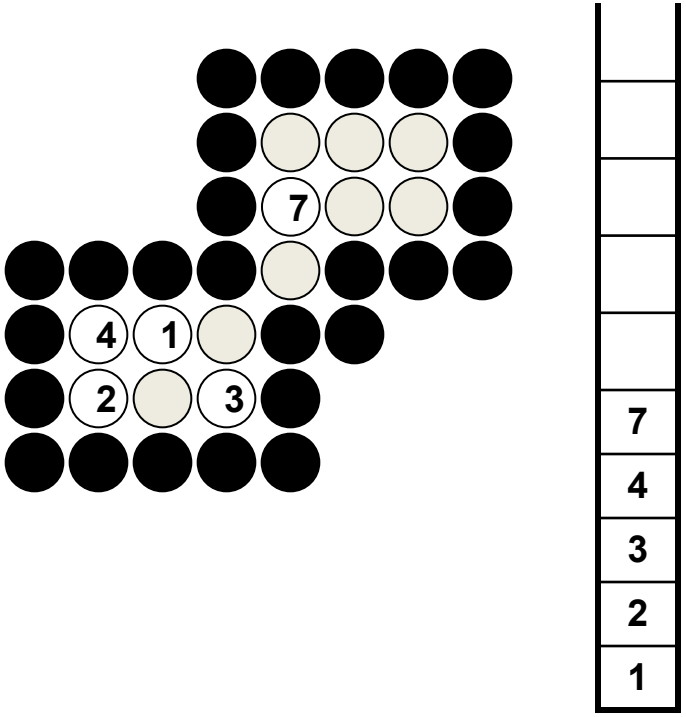
Boundary Fill Algorithm

8-connected (Example)



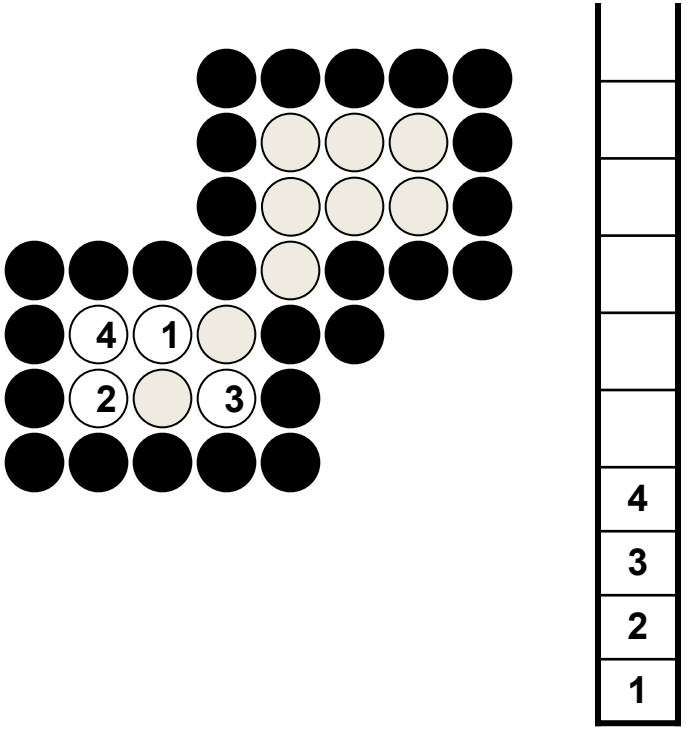
Boundary Fill Algorithm

8-connected (Example)



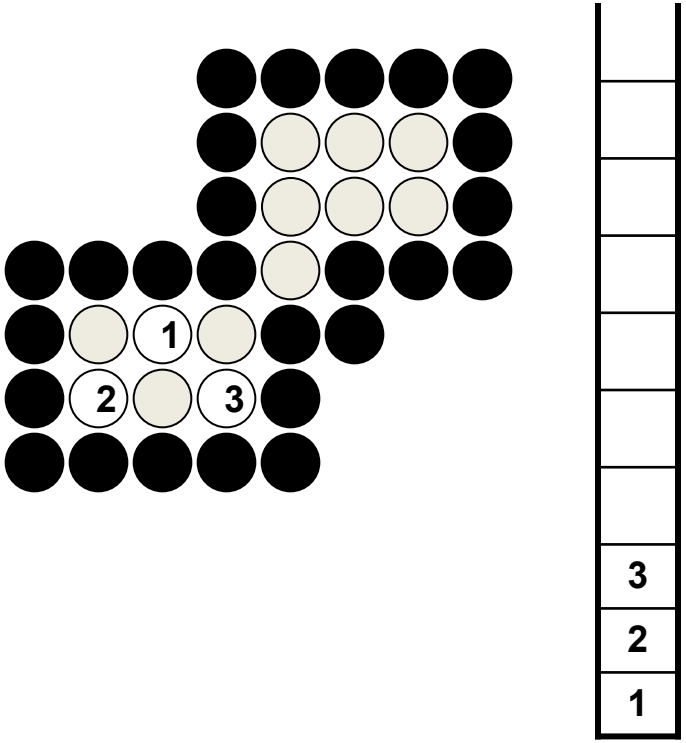
Boundary Fill Algorithm

8-connected (Example)



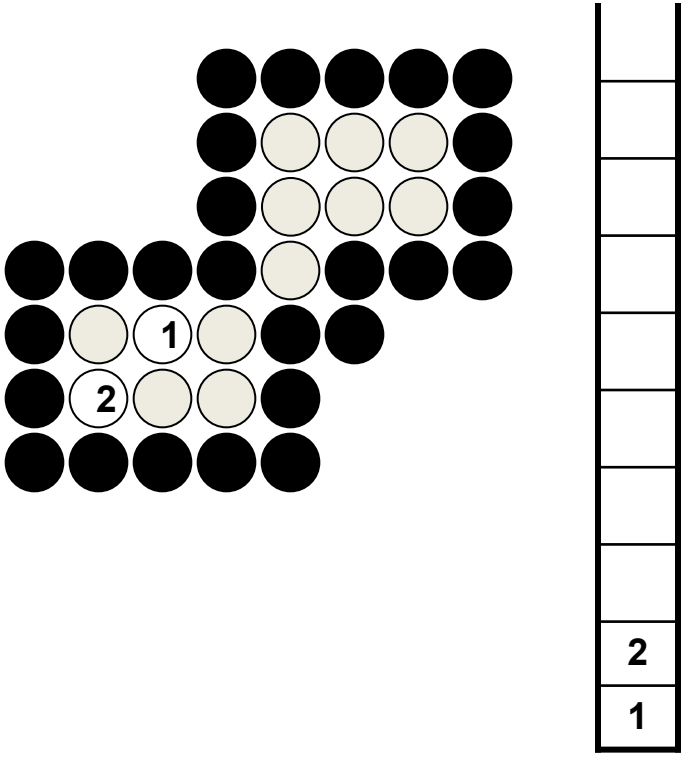
Boundary Fill Algorithm

8-connected (Example)



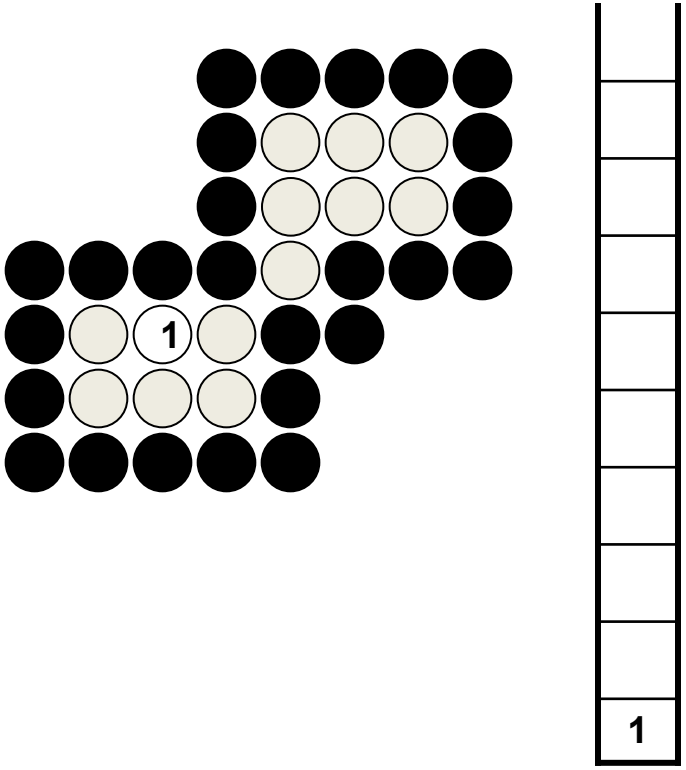
Boundary Fill Algorithm

8-connected (Example)



Boundary Fill Algorithm

8-connected (Example)

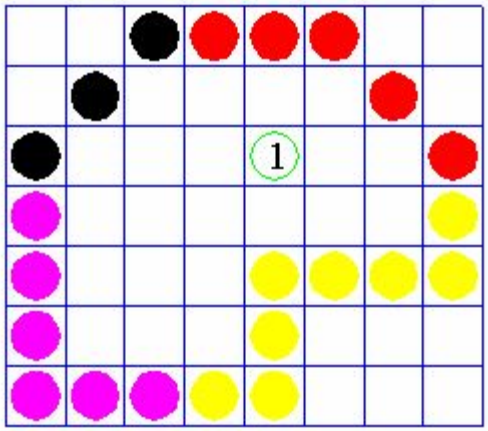


Flood Fill Algorithm

Sometimes we want to fill in (recolor) an area that is not defined within a single color boundary.

We paint such areas by replacing a specified interior color instead of searching for a boundary color value.

This approach is called a **flood-fill algorithm**.



Flood Fill Algorithm

We start from a specified interior pixel (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color.

If the area has **more than one** interior color, we can first **reassign pixel values** so that all interior pixels have the same color.

Using either **4-connected** or **8-connected** approach, we then step through pixel positions until all interior pixels have been repainted.

Character Display

1. Two methods: Bitmap Font and Outline Font

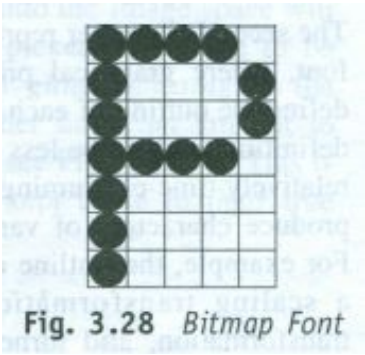


Fig. 3.28 *Bitmap Font*

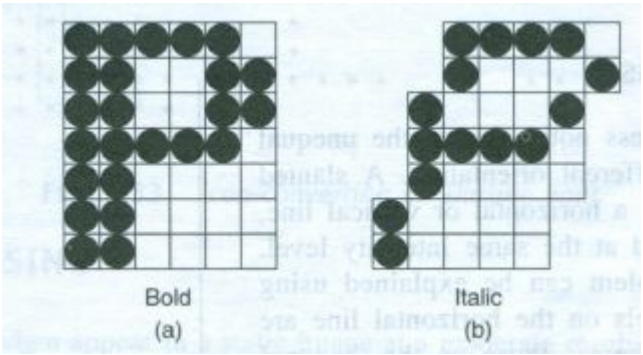


Fig. 3.30 *Outline Font*

Bitmap Font

- Character is represented by the On pixels in a bilevel pixel grid pattern called a bitmap.
- Characters are already in scan converted form.
- We may overlay the bitmap onto itself with a horizontal offset of one pixel to produce bold and shift rows of pixels to produce italic.
- Thus variation in appearance and size from one font, the overall results tends to be less than satisfactory.
- Furthermore, the size of a bitmap font is dependent on image resolution.

Outline Font

- The graphical primitives such as lines and arcs are used to define the outline of each character.
- Requires scan conversion operations.
- But it can be used to produce characters of varying size, appearance and even orientation.
- It can be resized through a scaling transformation, made into italic through a shearing transformation and turned around with respect to a reference point through a rotation transformation.