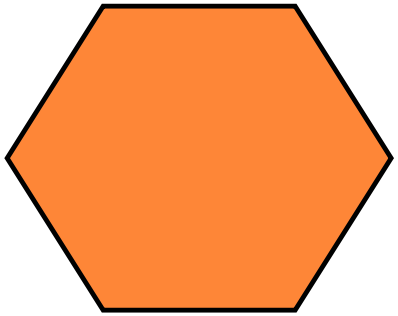
Polygon Filling

Polygon Filling

Types of filling

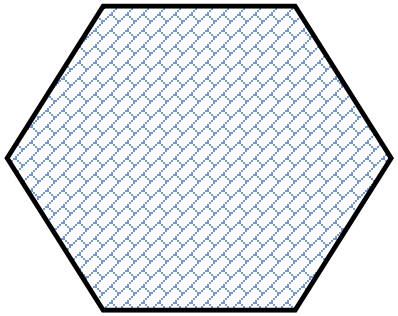
- **Solid-fill**

All the pixels inside the polygon's boundary are illuminated.



- **Pattern-fill**

the polygon is filled with an arbitrary predefined pattern.



Boundary Fill Algorithm

- Another approach to area filling is to start at a point inside a region and paint the interior outward toward the boundary.
- If the boundary is specified in a single color, the fill algorithm processed outward pixel by pixel until the boundary color is encountered.
- A boundary-fill procedure accepts as input the coordinate of the interior **point (x, y)**, a **fill color**, and a **boundary color**.

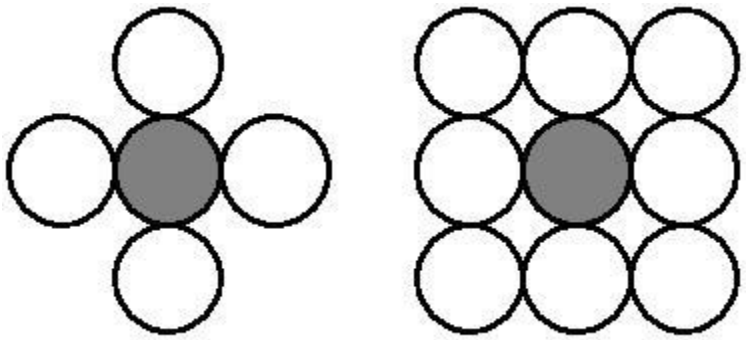
Boundary Fill Algorithm

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**.

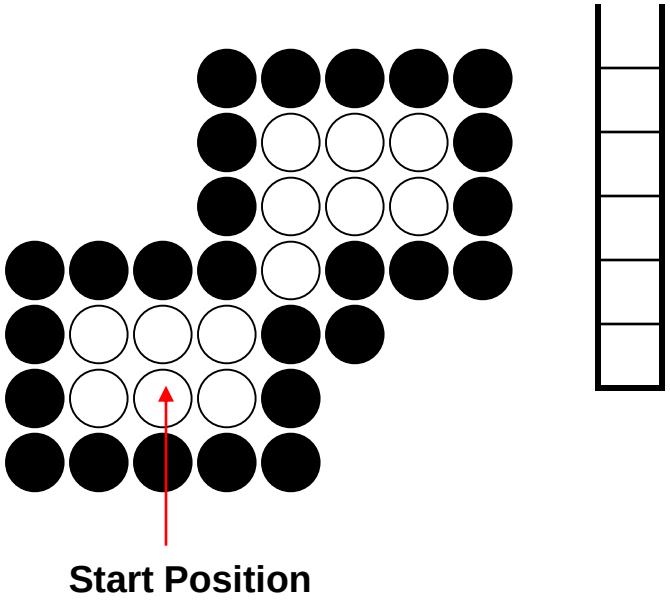
Boundary Fill Algorithm

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.



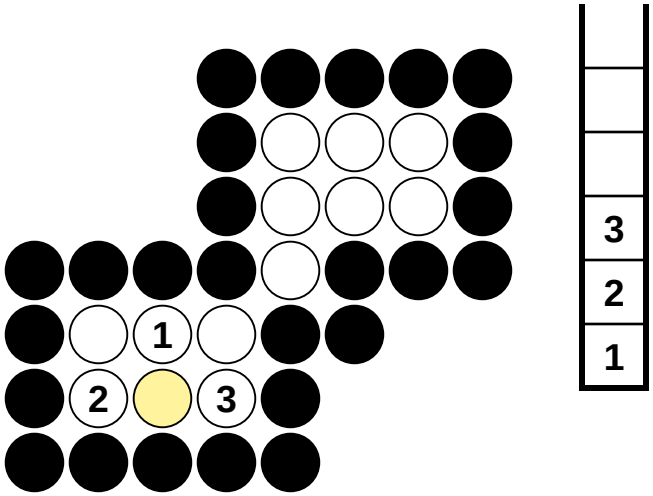
Boundary Fill Algorithm

4-connected (Example)



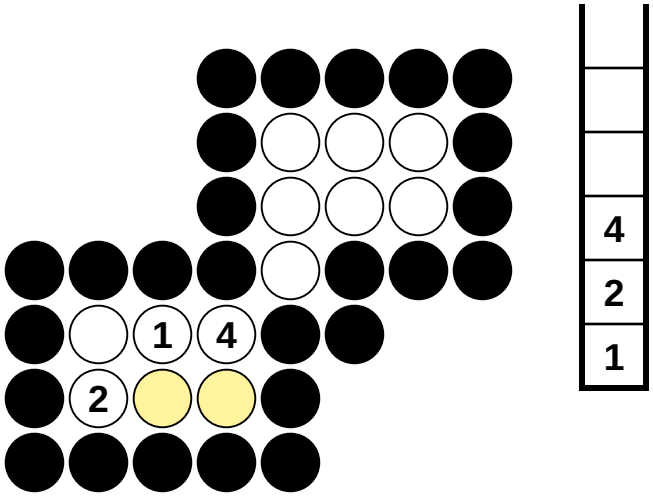
Boundary Fill Algorithm

4-connected (Example)



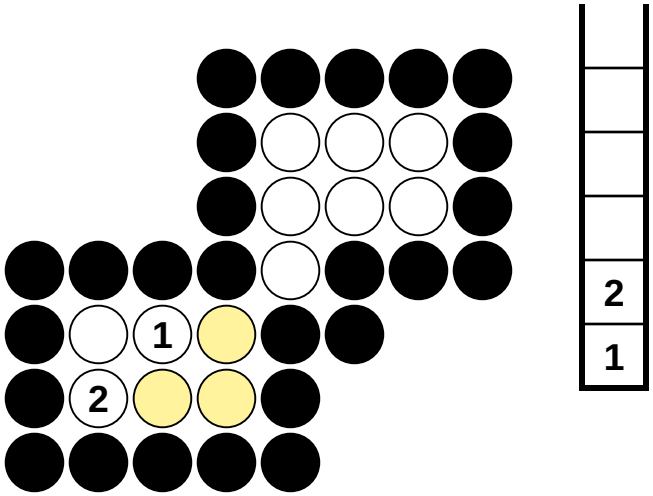
Boundary Fill Algorithm

4-connected (Example)



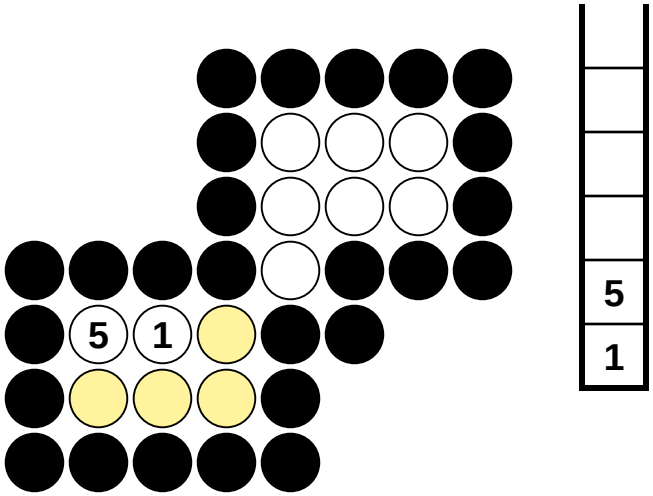
Boundary Fill Algorithm

4-connected (Example)



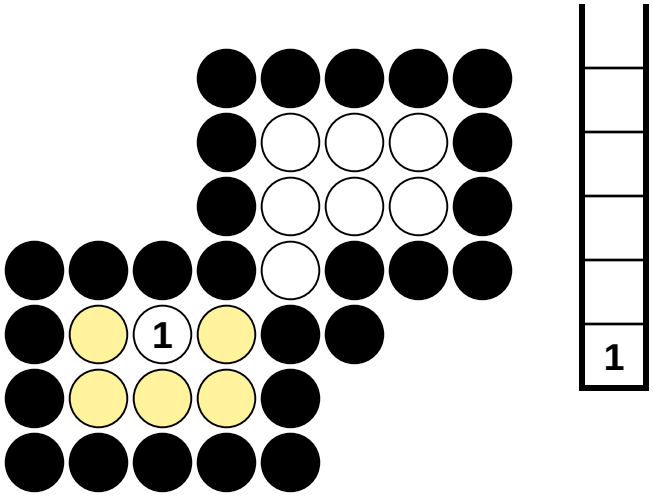
Boundary Fill Algorithm

4-connected (Example)



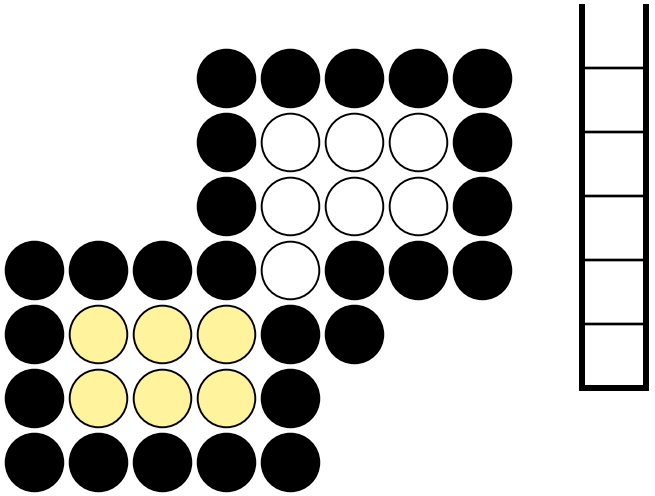
Boundary Fill Algorithm

4-connected (Example)



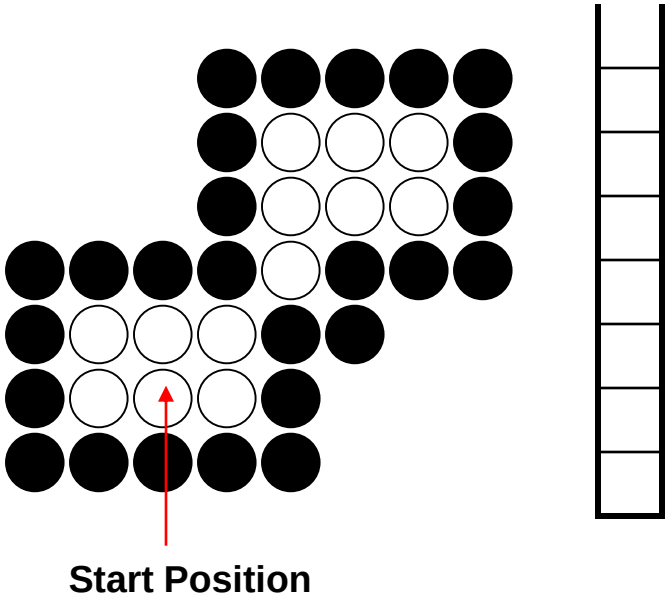
Boundary Fill Algorithm

4-connected (Example)



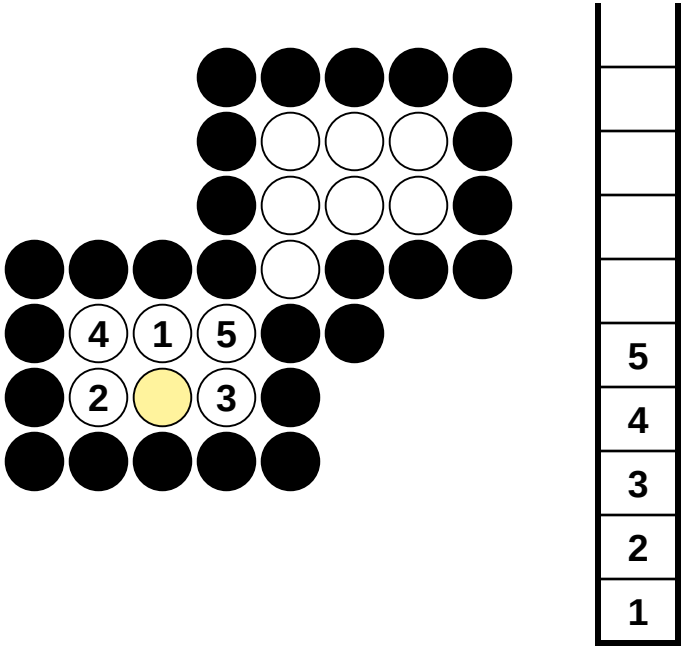
Boundary Fill Algorithm

8-connected (Example)



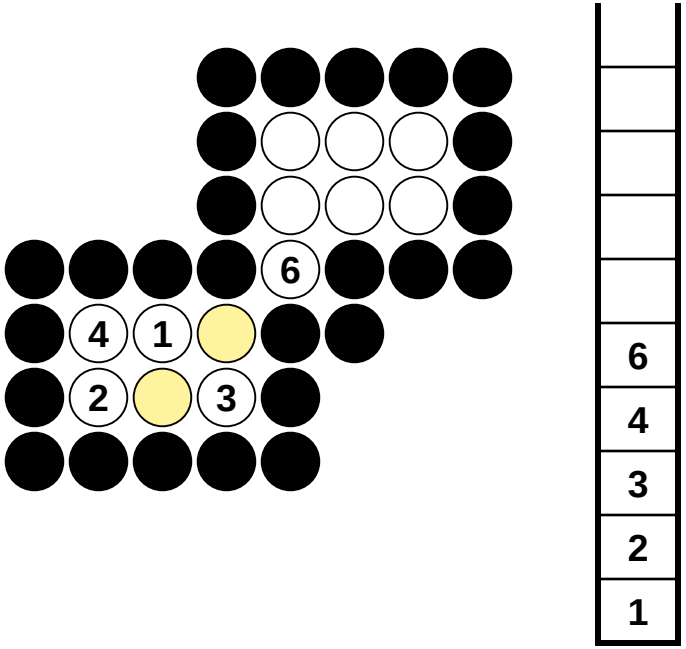
Boundary Fill Algorithm

8-connected (Example)



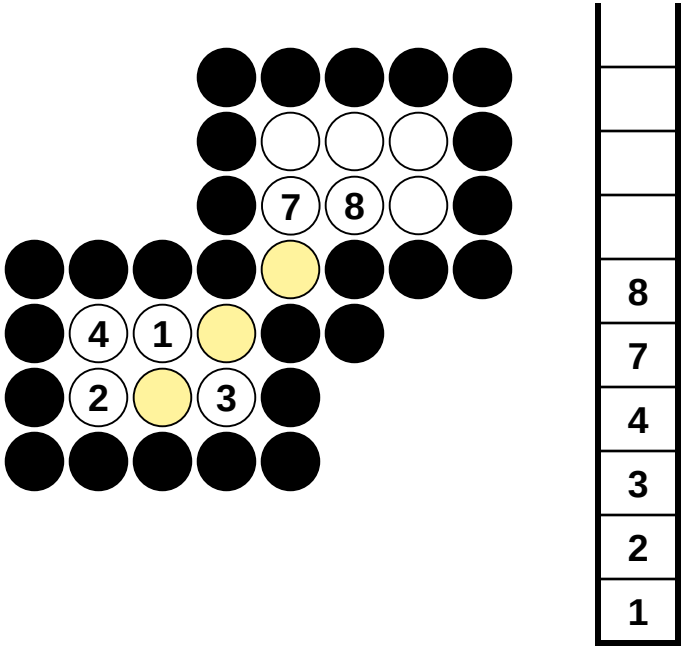
Boundary Fill Algorithm

8-connected (Example)



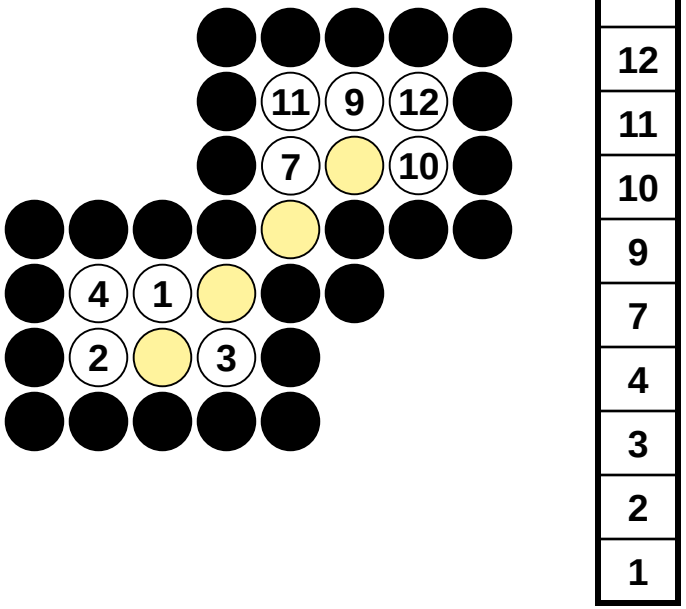
Boundary Fill Algorithm

8-connected (Example)



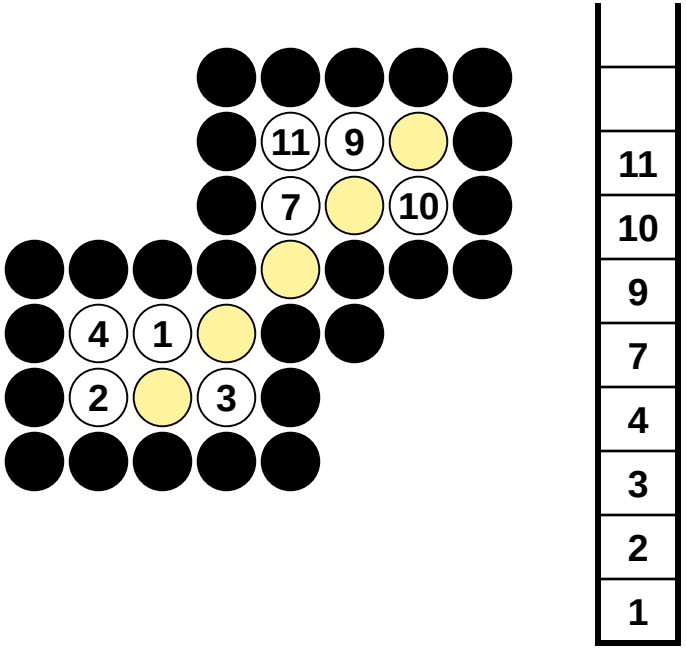
Boundary Fill Algorithm

8-connected (Example)



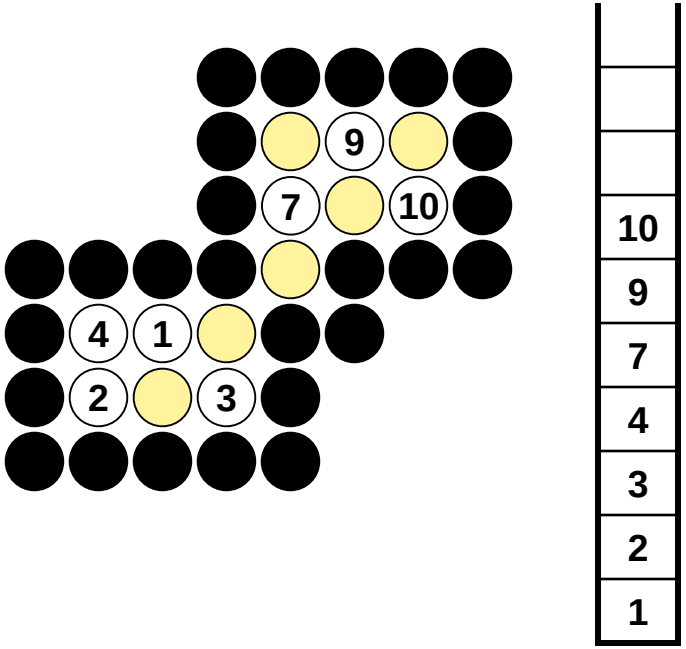
Boundary Fill Algorithm

8-connected (Example)



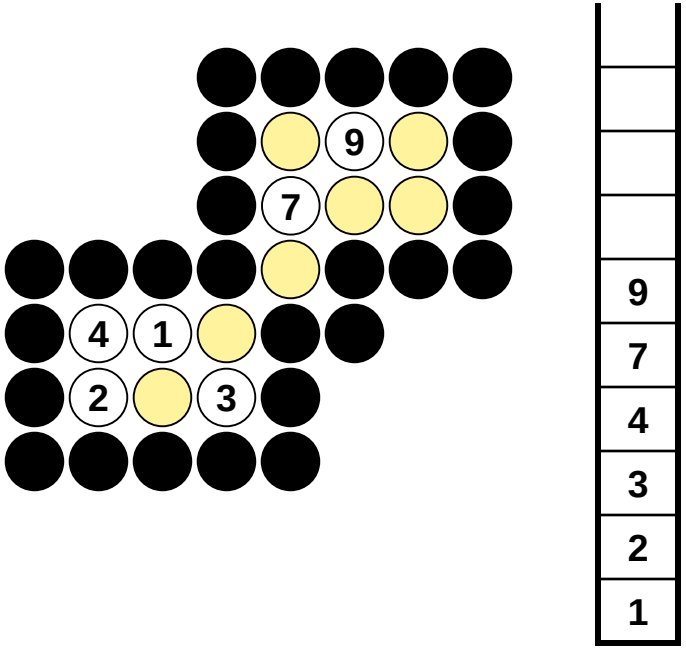
Boundary Fill Algorithm

8-connected (Example)



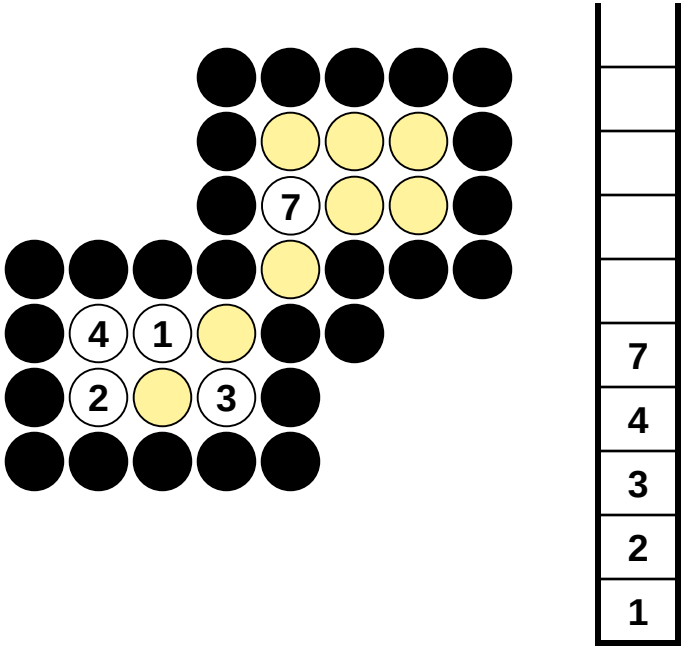
Boundary Fill Algorithm

8-connected (Example)



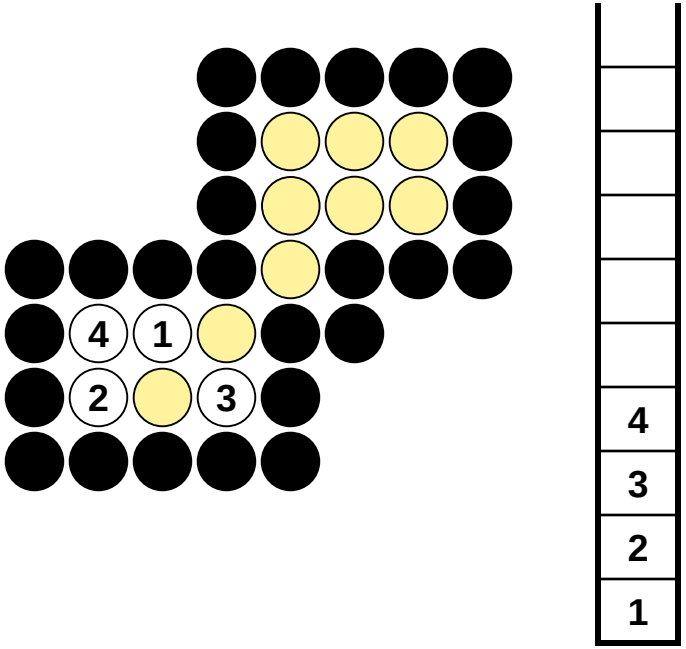
Boundary Fill Algorithm

8-connected (Example)



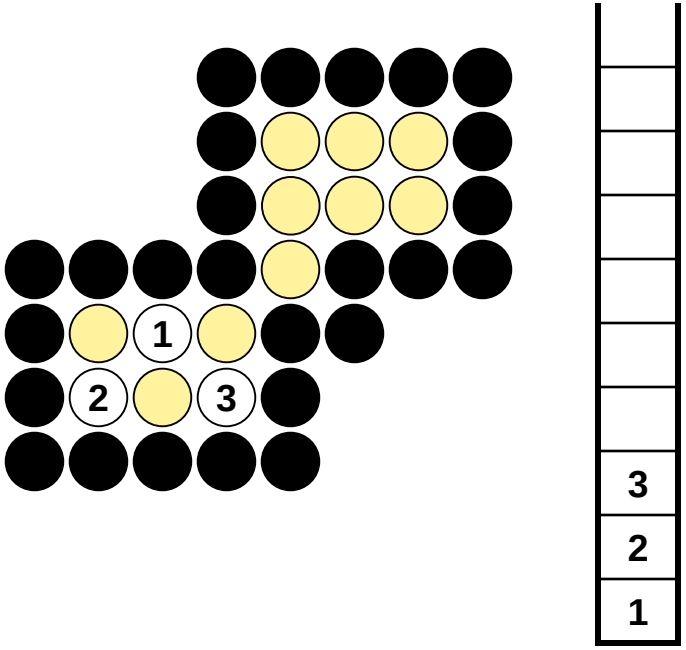
Boundary Fill Algorithm

8-connected (Example)



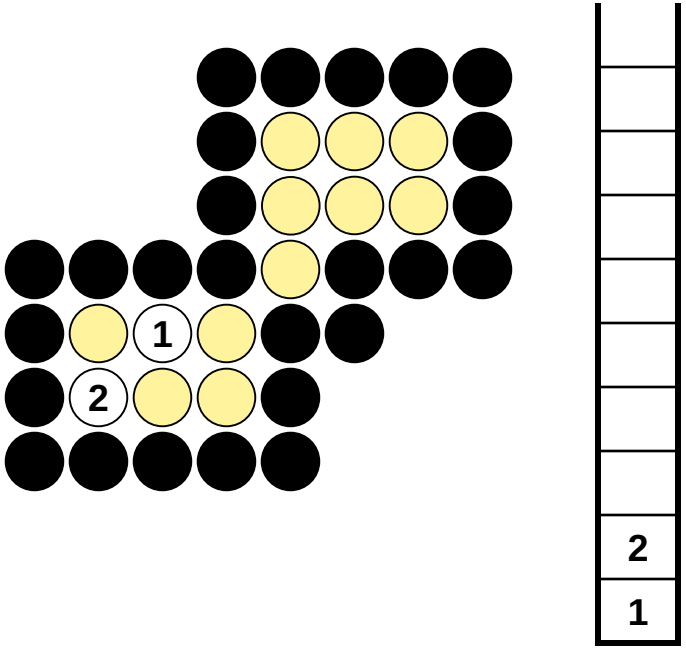
Boundary Fill Algorithm

8-connected (Example)



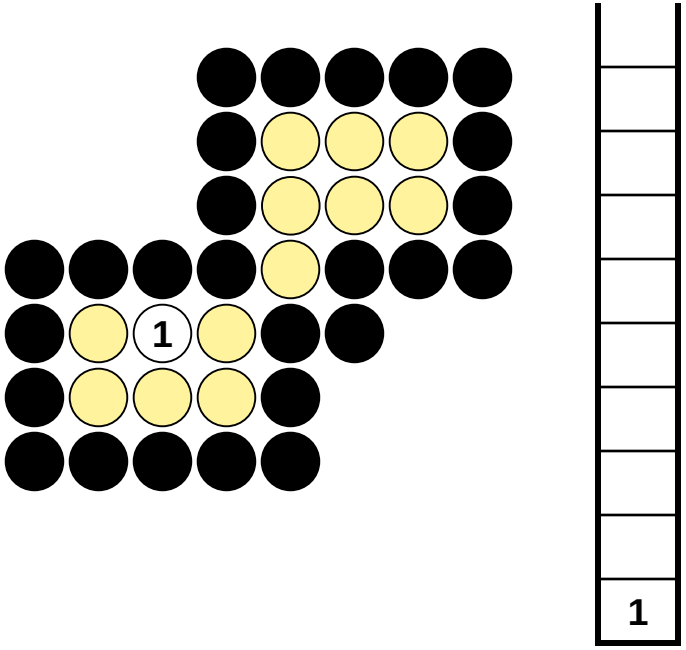
Boundary Fill Algorithm

8-connected (Example)



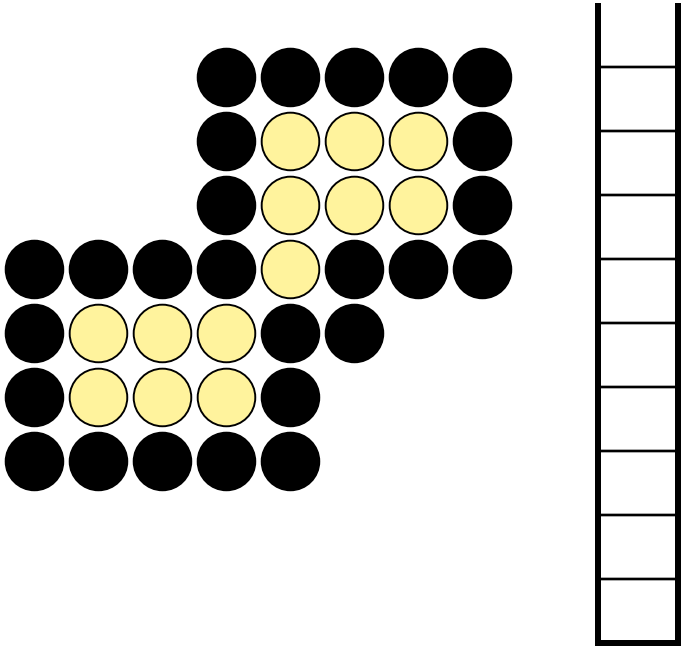
Boundary Fill Algorithm

8-connected (Example)



Boundary Fill Algorithm

8-connected (Example)



Boundary Fill Algorithm

Since the previous procedure requires considerable stacking of neighboring pixels, more **efficient methods** are generally employed.

These methods (**Span Flood-Fill**) **fill horizontal pixel spans across scan lines**, instead of proceeding to 4-connected or 8-connected neighboring pixels.

Then we need only stack a beginning position for each horizontal pixel spans, instead of stacking all unprocessed neighboring positions around the current position.

Span Flood-Fill Algorithm

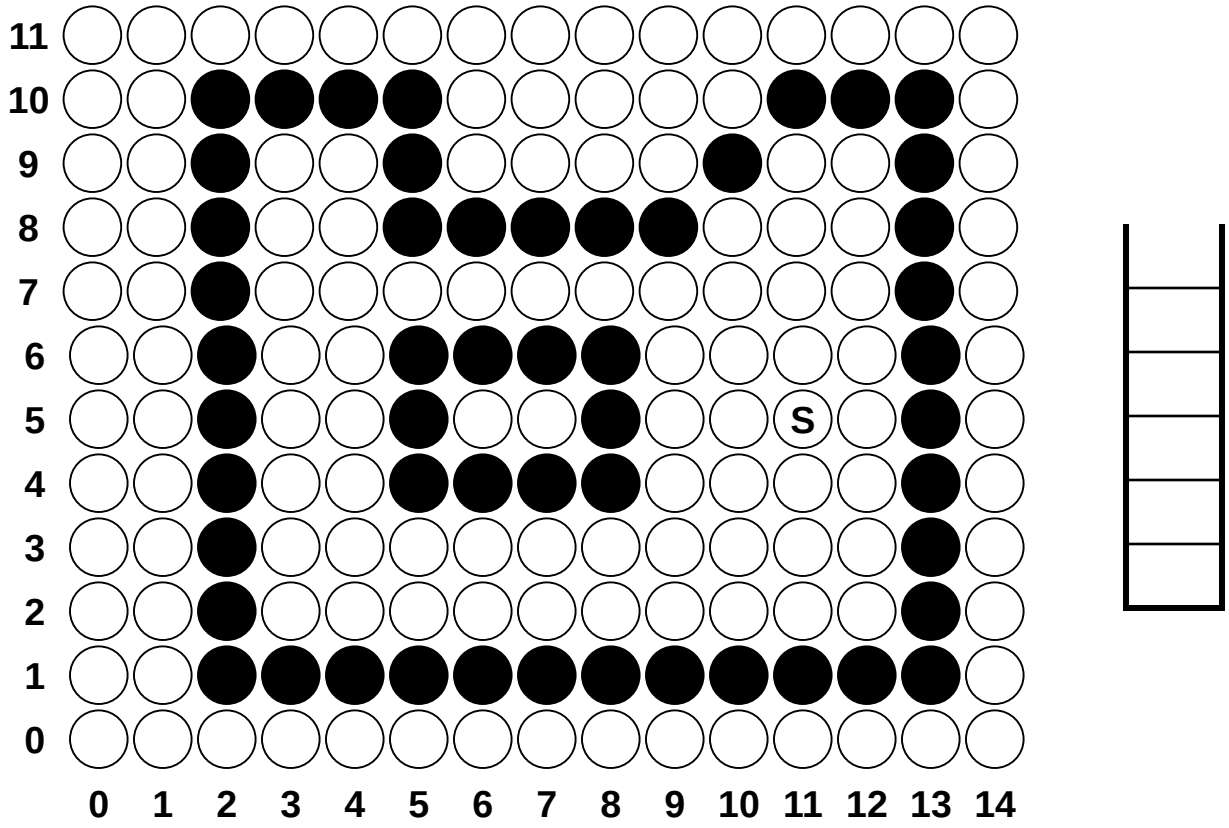
The algorithm is summarized as follows:

- **Starting** from the initial interior pixel, then fill in the contiguous span of pixels on this starting scan line.

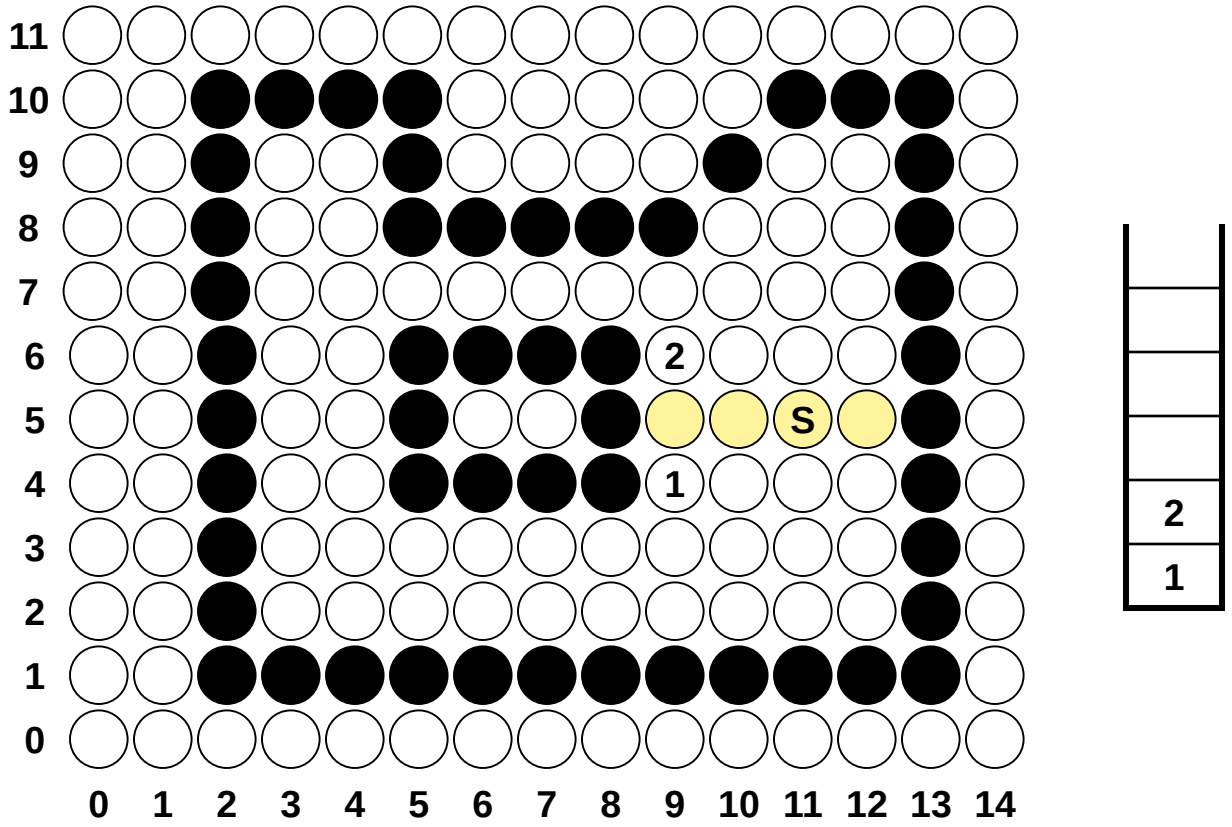
- **Then locate** and **stack** starting positions for spans on the adjacent scan lines, where spans are defined as the contiguous horizontal string of positions bounded by pixels displayed in the area border color.

- **At each subsequent step**, **unstack** the next start position and repeat the process.

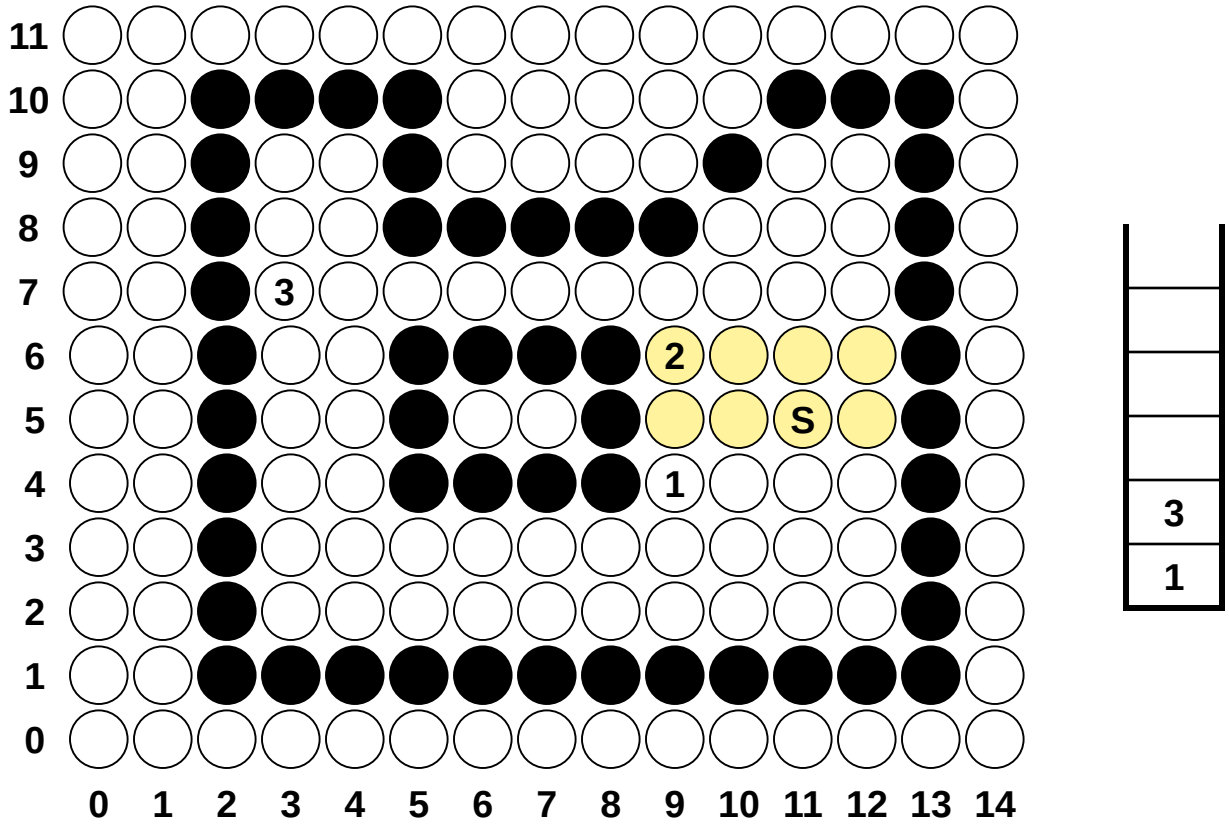
Span Flood-Fill Algorithm (example)



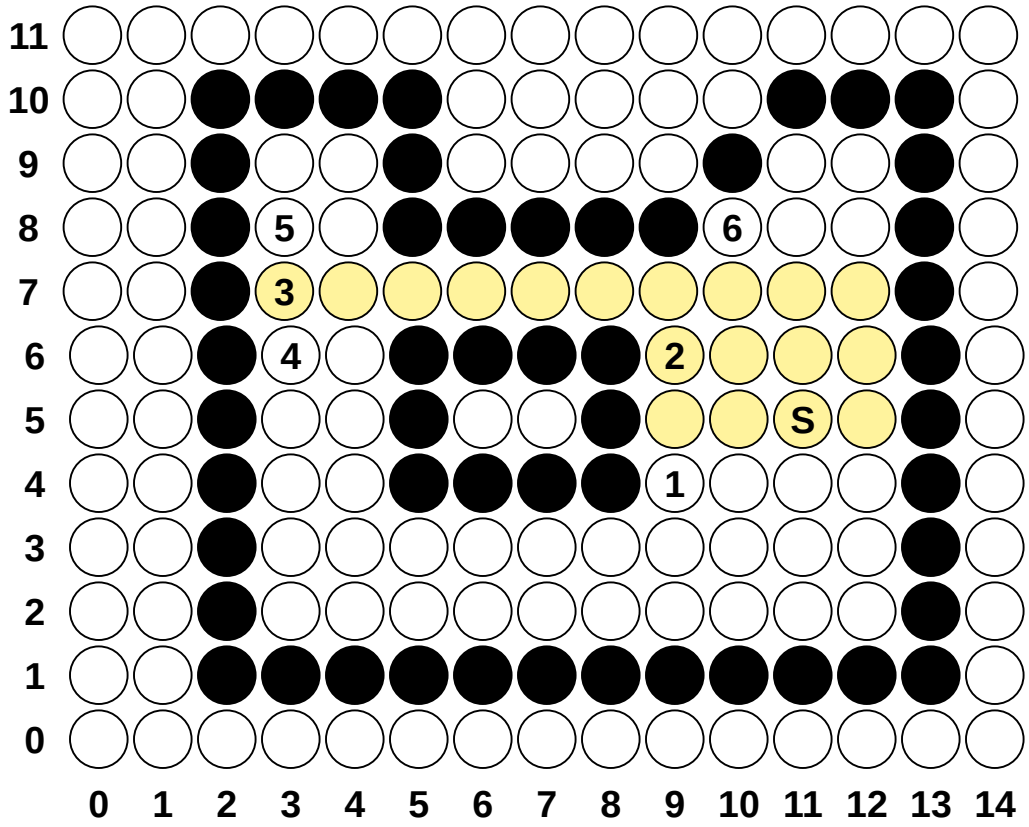
Span Flood-Fill Algorithm (example)

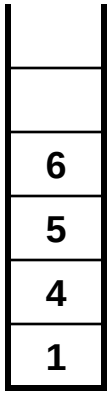


Span Flood-Fill Algorithm (example)

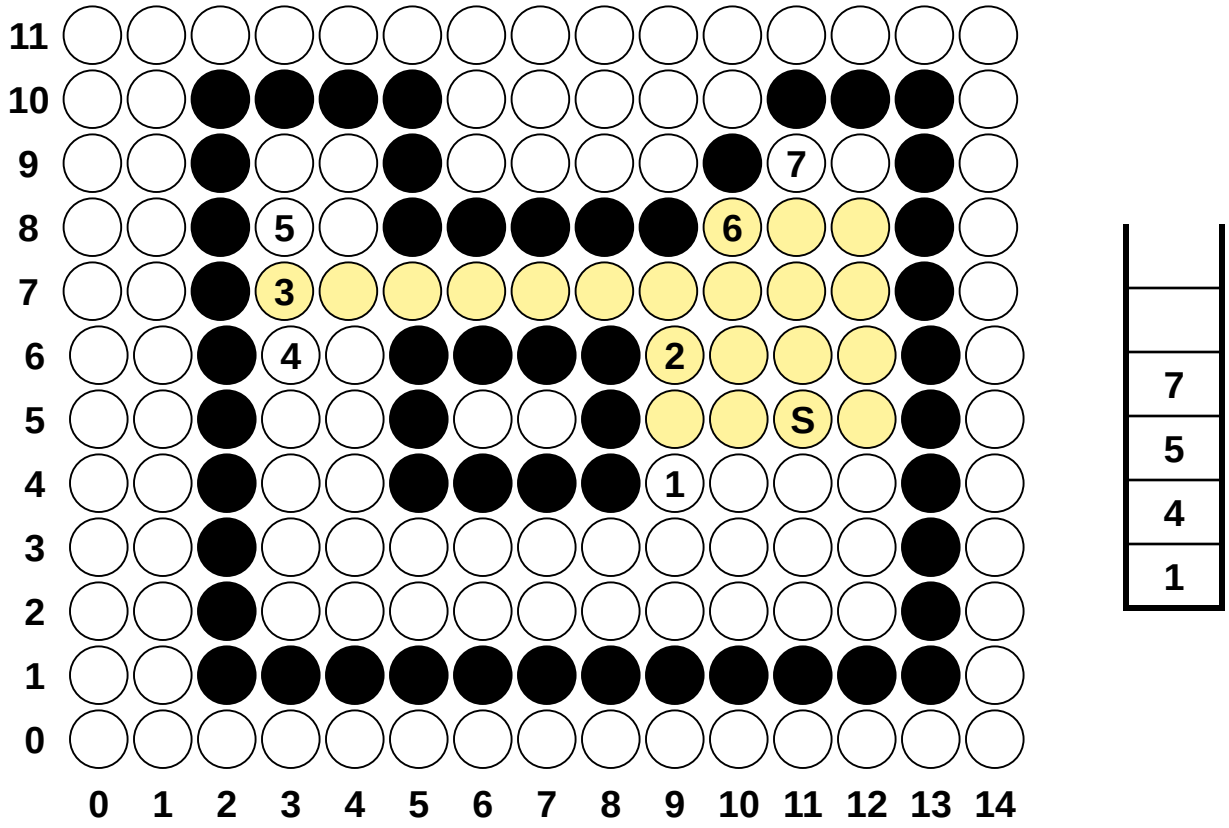


Span Flood-Fill Algorithm (example)

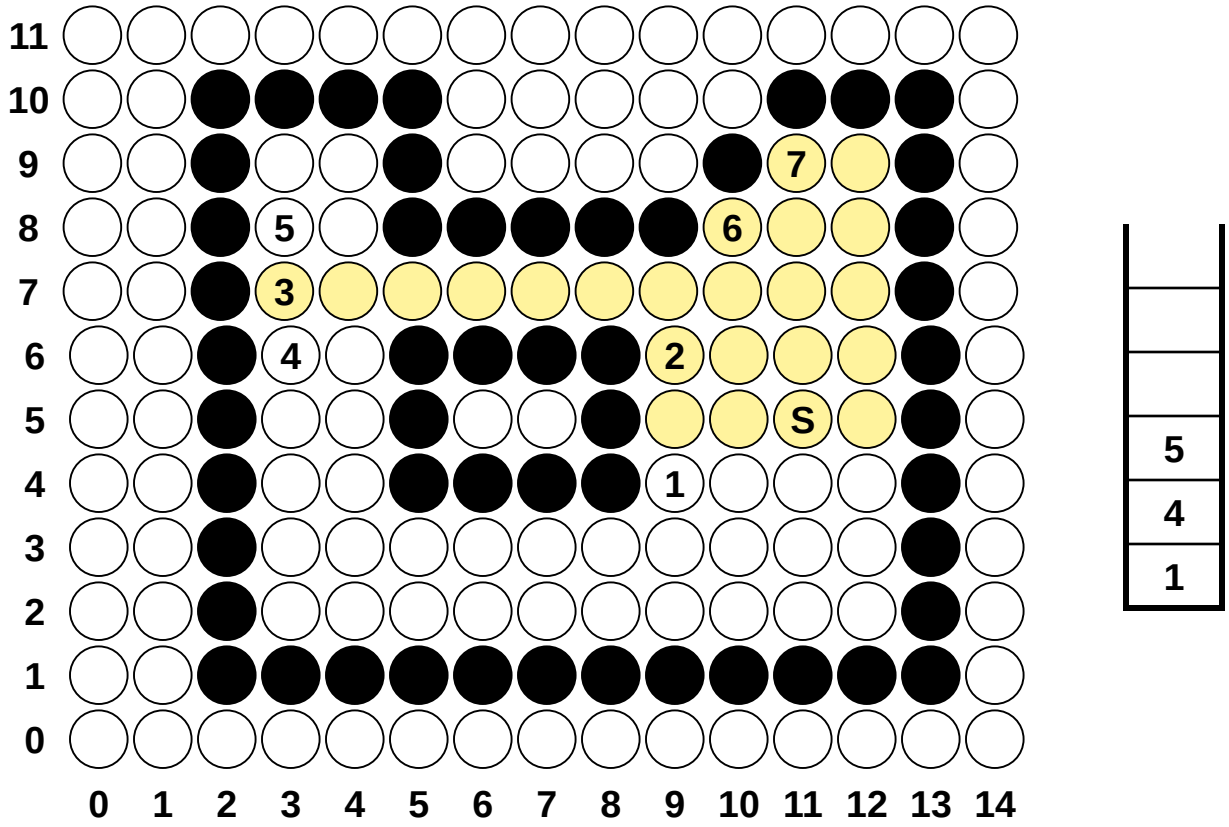




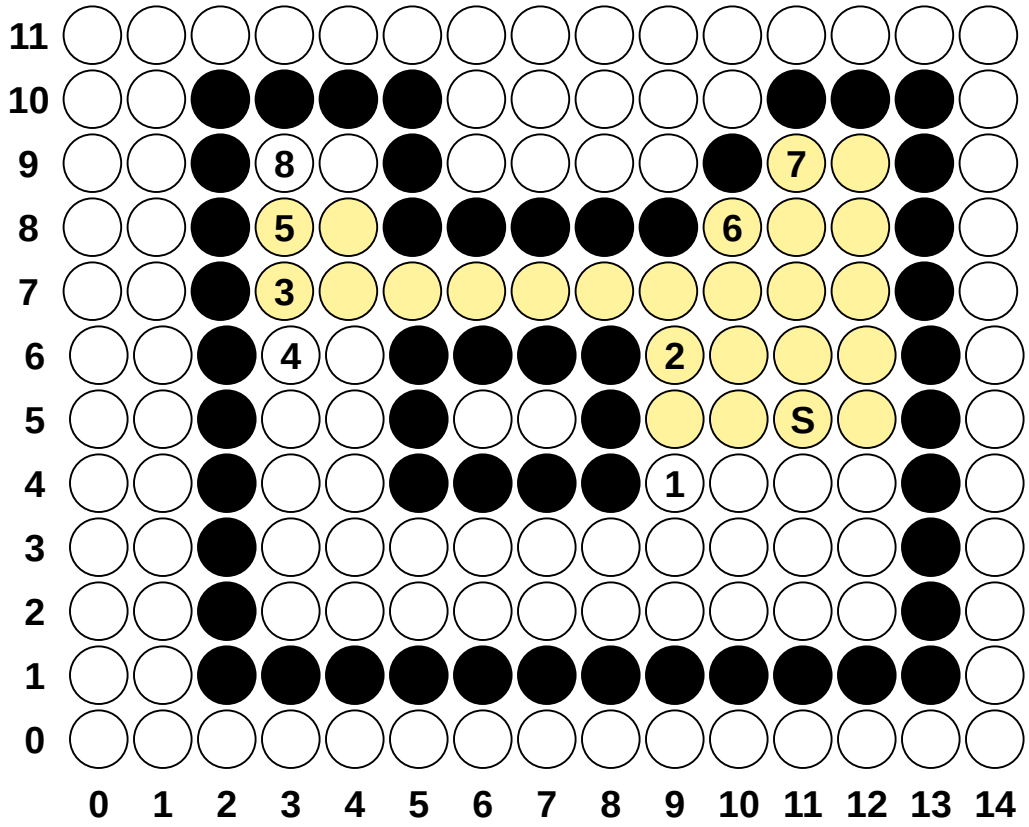
Span Flood-Fill Algorithm (example)



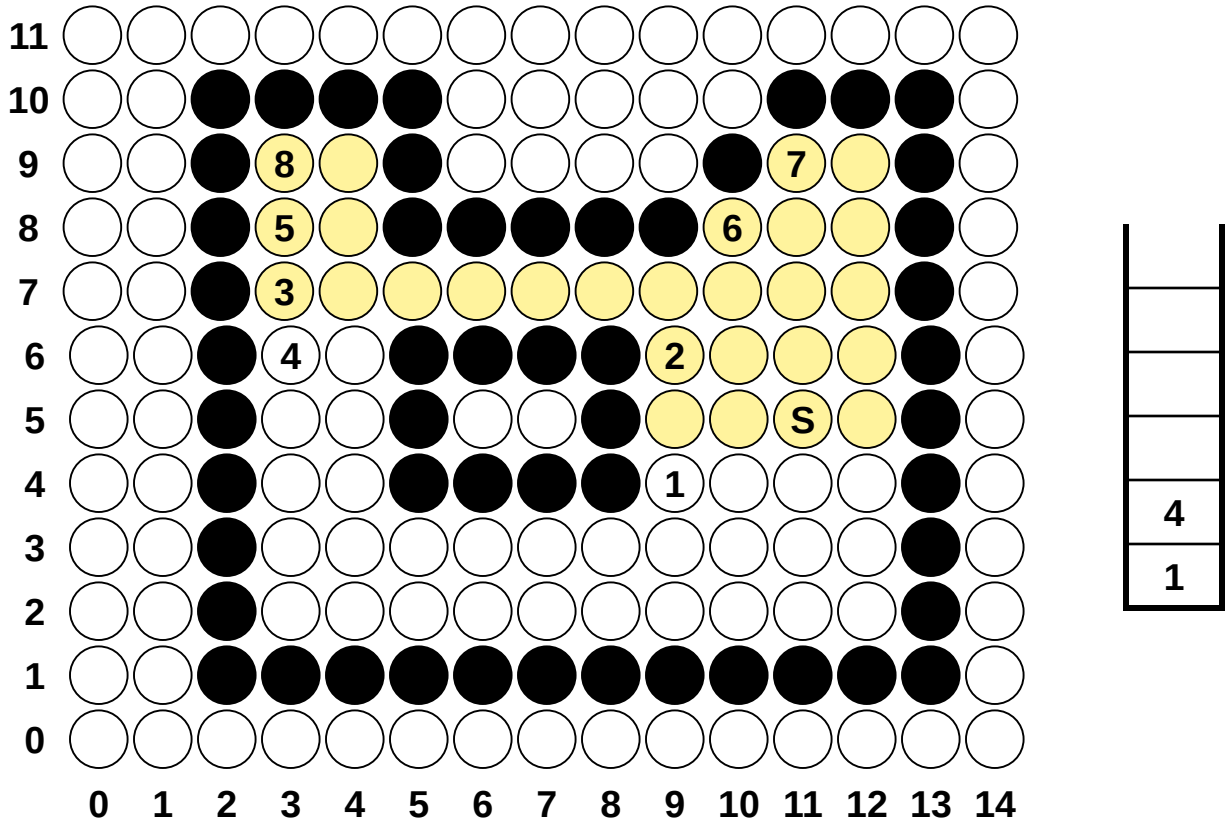
Span Flood-Fill Algorithm (example)



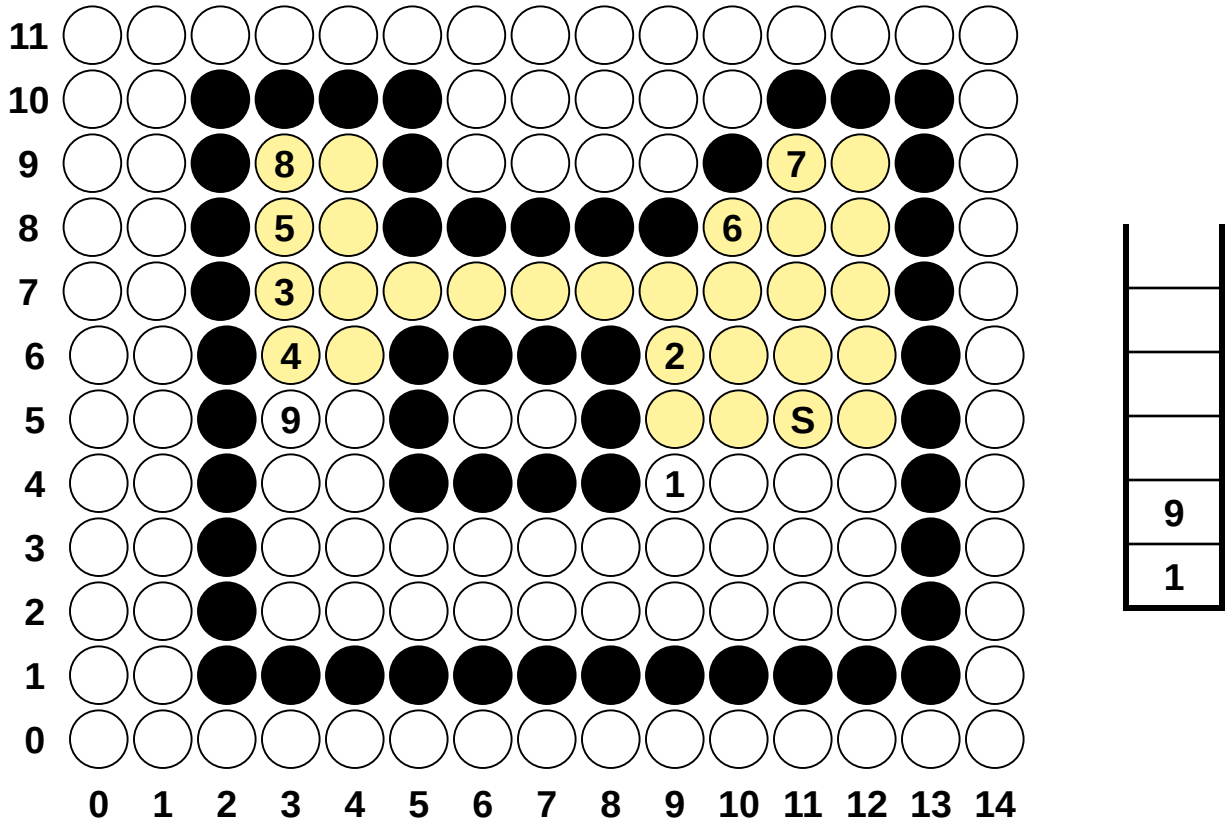
Span Flood-Fill Algorithm (example)



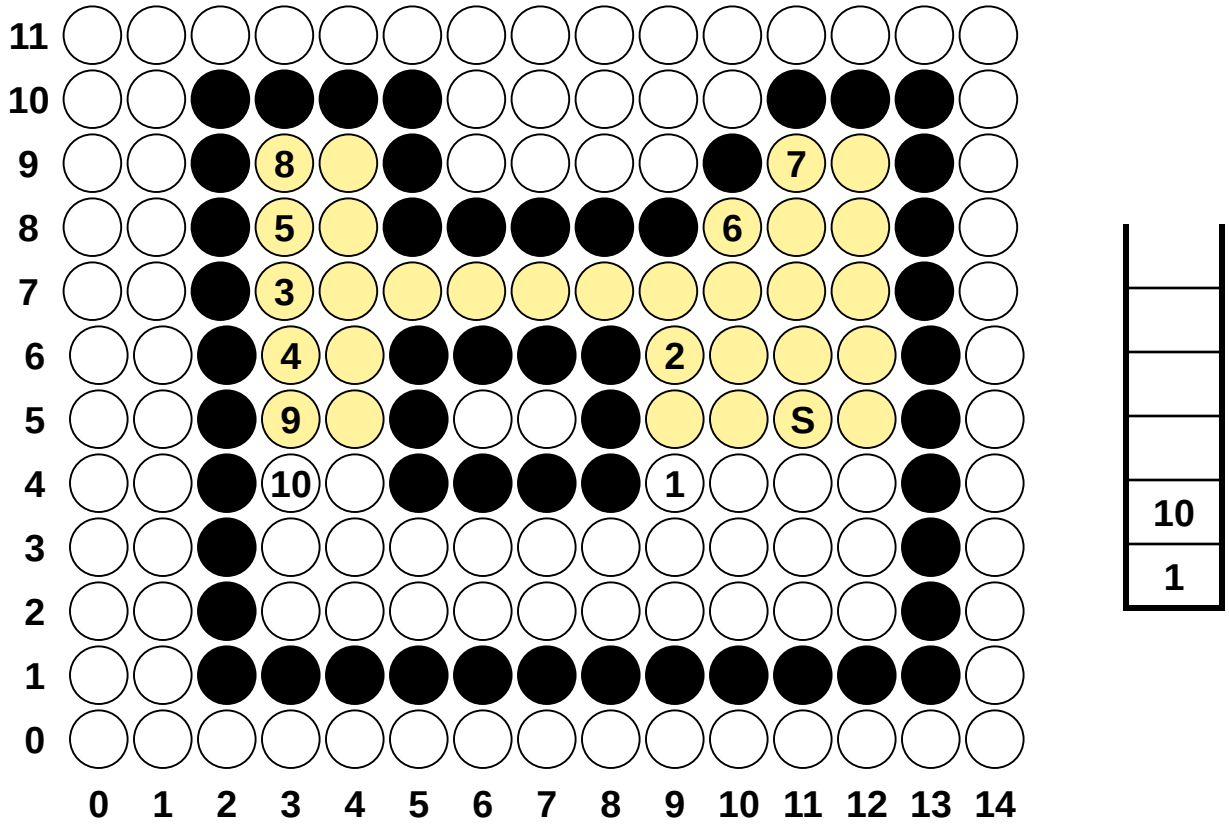
Span Flood-Fill Algorithm (example)



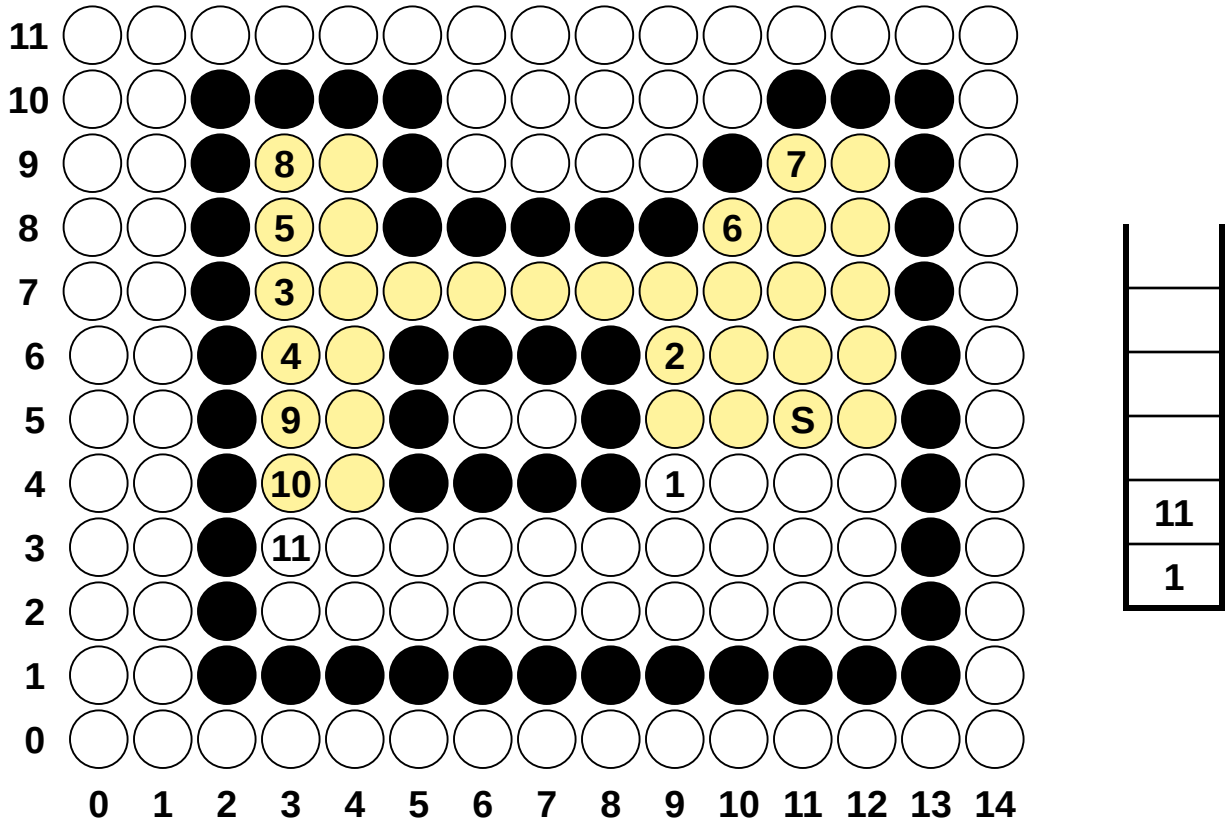
Span Flood-Fill Algorithm (example)



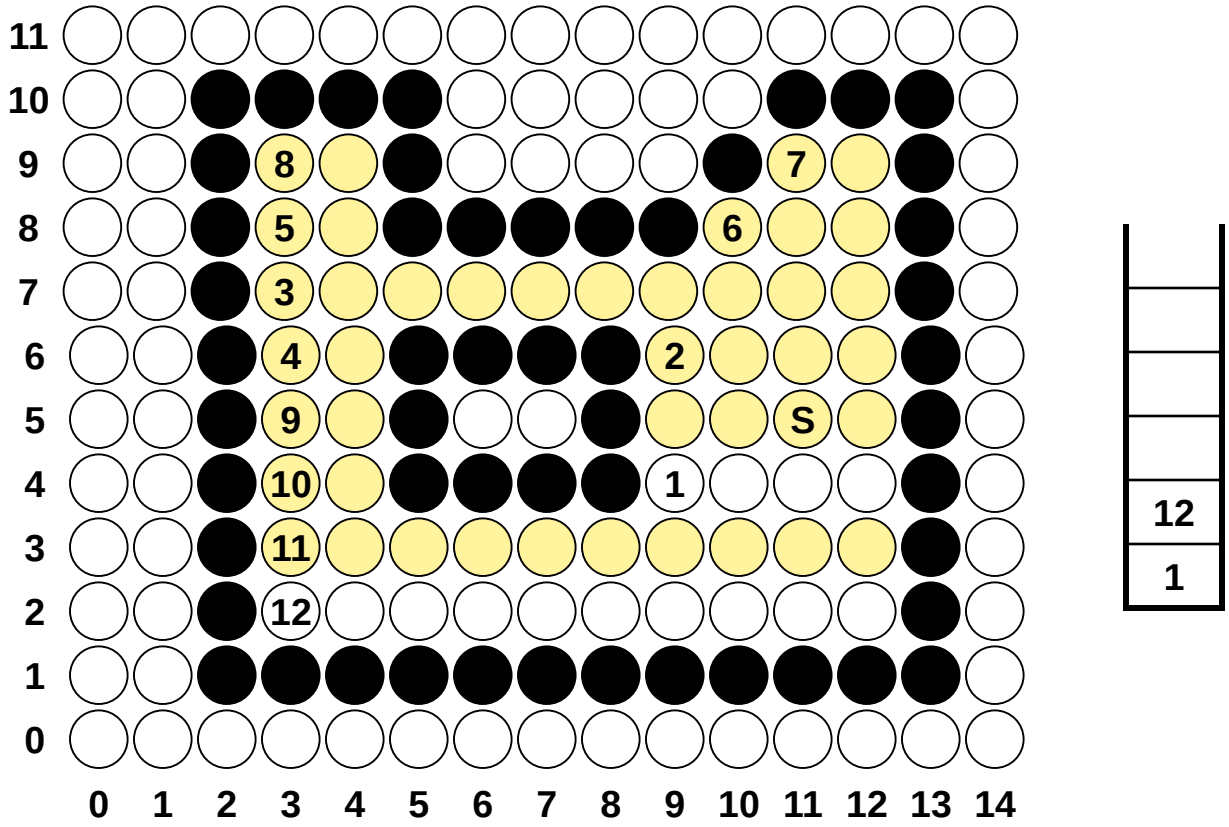
Span Flood-Fill Algorithm (example)



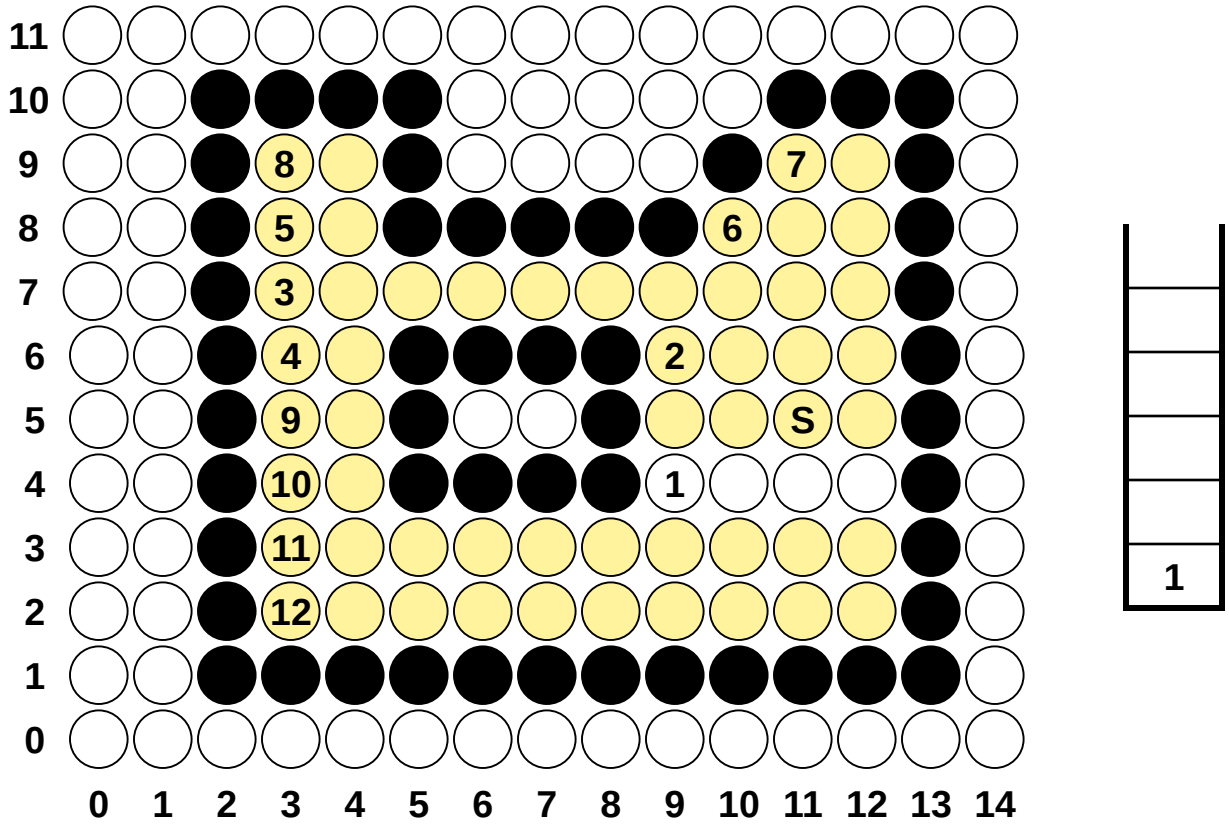
Span Flood-Fill Algorithm (example)



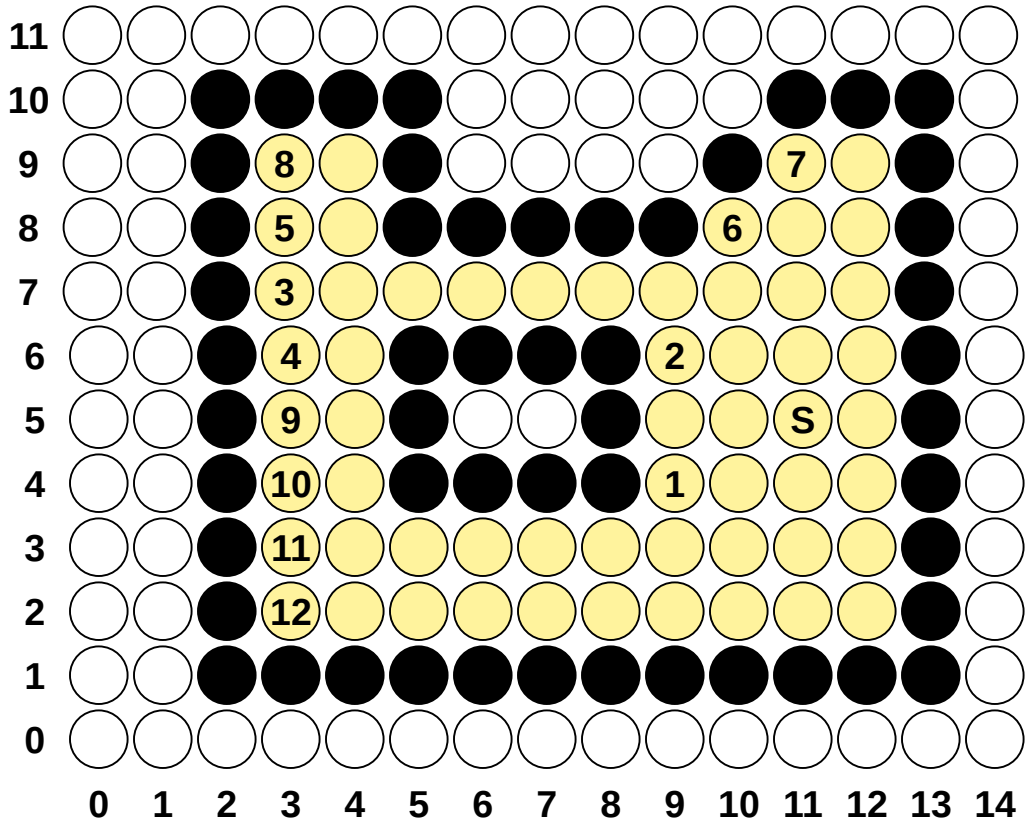
Span Flood-Fill Algorithm (example)

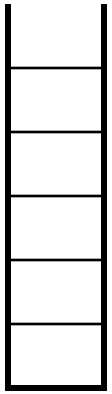


Span Flood-Fill Algorithm (example)



Span Flood-Fill Algorithm (example)



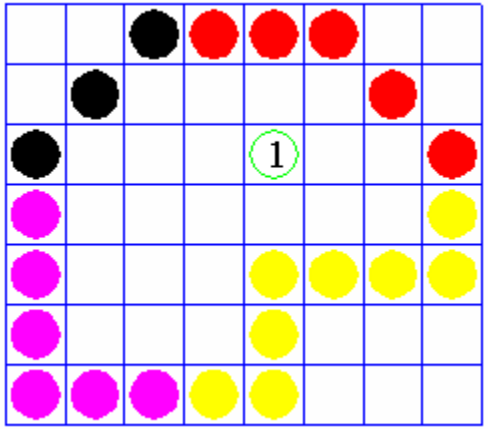


Flood Fill Algorithm

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.

Thank You