FLAME UNIVERSITY

EVERLASTING *learning*

FUNDAMENTALS OF COMPUTER GRAPHICS (CSIT304)

SCANLINE FILLING ALGORITHM

CHIRANJOY CHATTOPADHYAY

Associate Professor,

FLAME School of Computation and Data Science

POLYGON FILLING

FILLING RECTANGLES

- The task of filling primitives can be broken into two parts:
  - The decision of which pixels to fill
    - shape of the primitive
  - With what value to fill them.

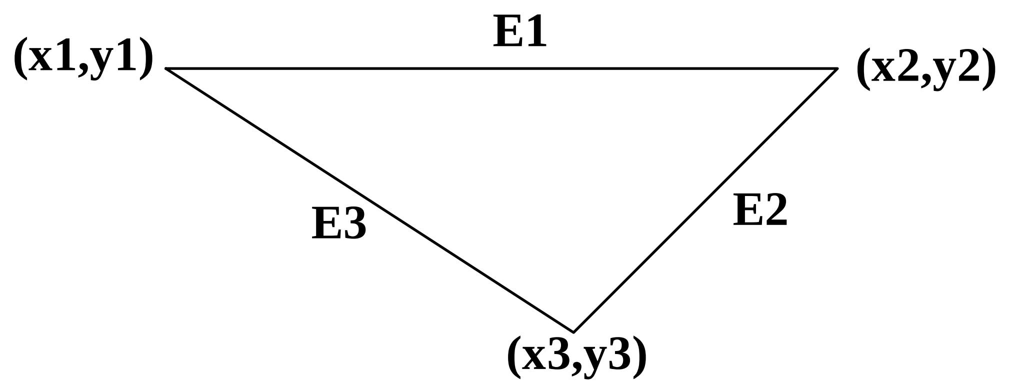
FILLING RECTANGLES

```
for (y=YMIN; y<YMAX; y++) // by scan line
{
 for (x=XMIN; x<XMAX; x++) // by pixel in span
 {
         WritePixel(x,y,color);
 }
}
```

POLYGONS

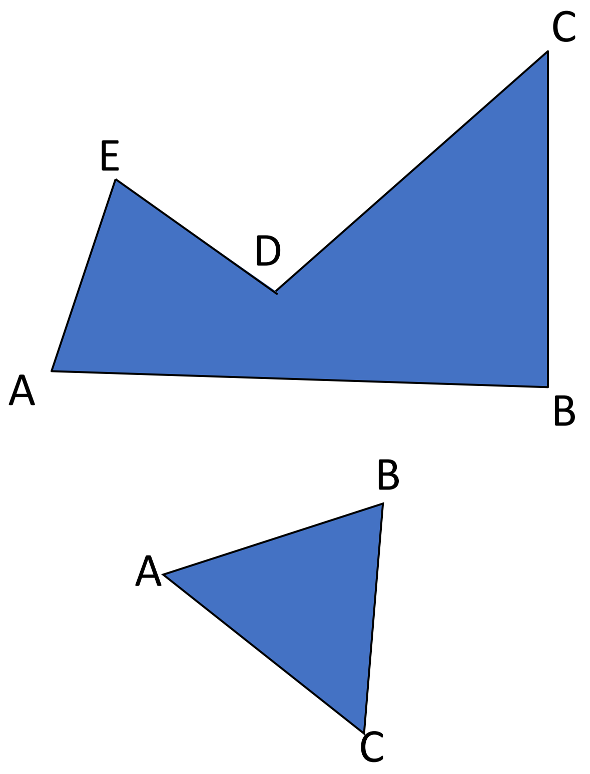
- A polygon is a many-sided planar figure composed of vertices and edges.
- A polygon is bounded (finite area) and closed (includes boundary).
- Vertices are represented by points (x,y).
- Edges are represented as line segments which connect two points, (x1,y1) and (x2,y2).

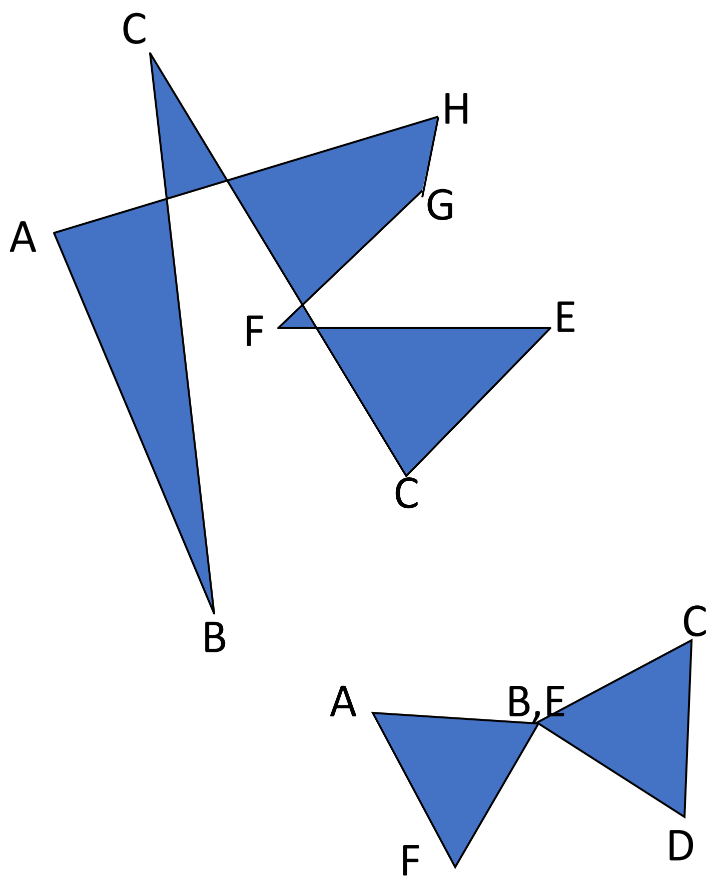
$P = \{ (x_i , y_i) \} \quad i=1,n$



POLYGONS: COMPLEX VS SIMPLE

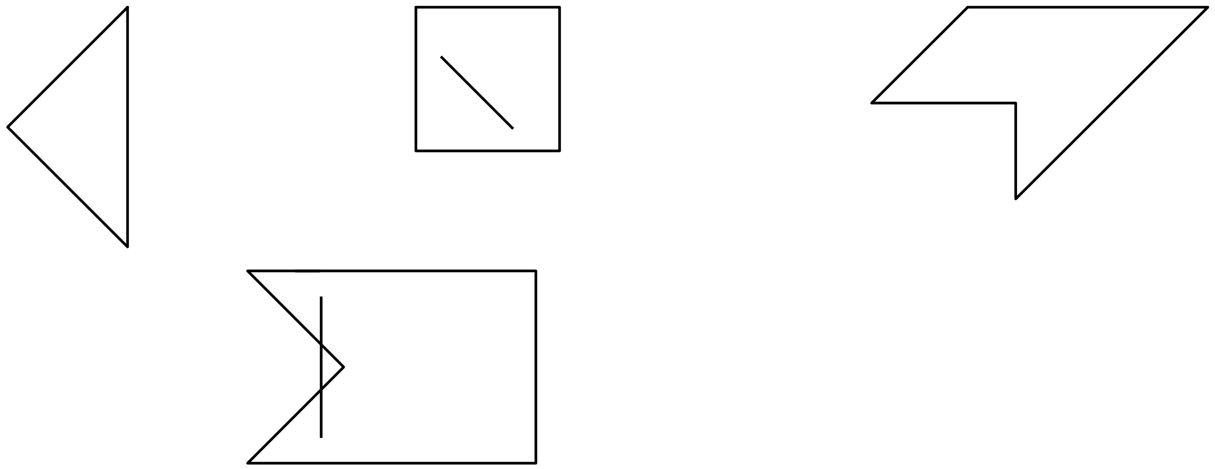
- A simple polygon – edges only intersect a vertices, no coincident vertices
- A complex polygon – edges intersect and/or coincident vertices





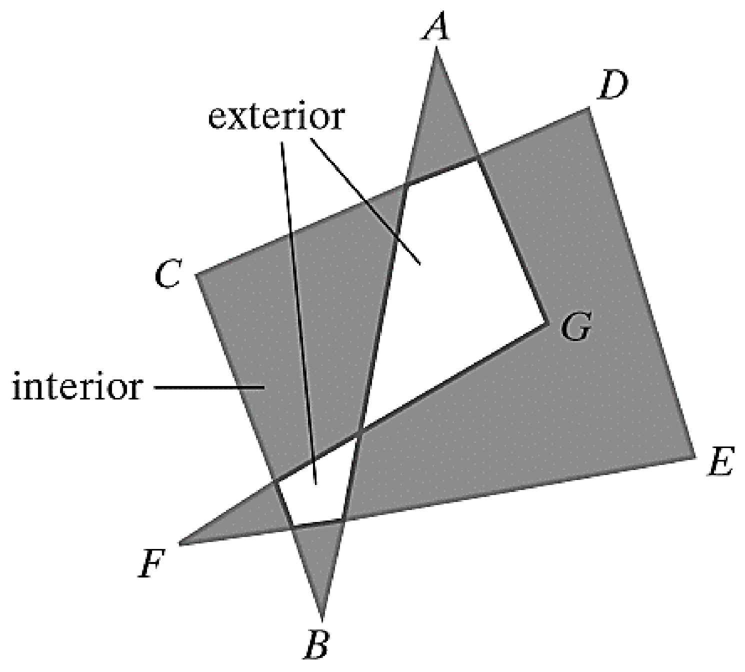
SIMPLE POLYGONS: CONVEX AND CONCAVE

- Convex Polygon - For any two points P1, P2 inside the polygon, all points on the line segment which connects P1 and P2 are inside the polygon.
- All points $P = uP1 + (1-u)P2$, u in $[0,1]$ are inside the polygon provided that P1 and P2 are inside the polygon.
- Concave Polygon - A polygon which is not convex.

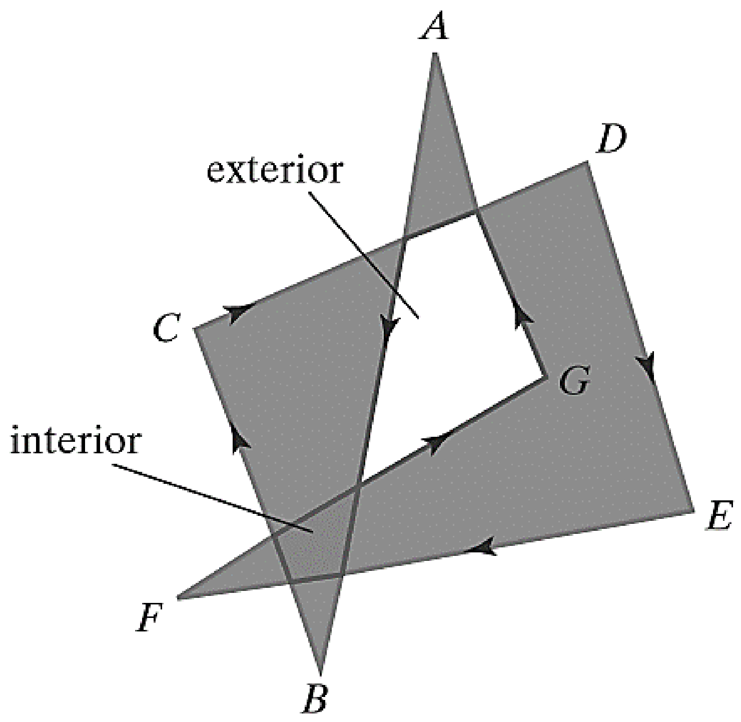


INSIDE-OUTSIDE TESTS

- Identifying the interior of a simple object



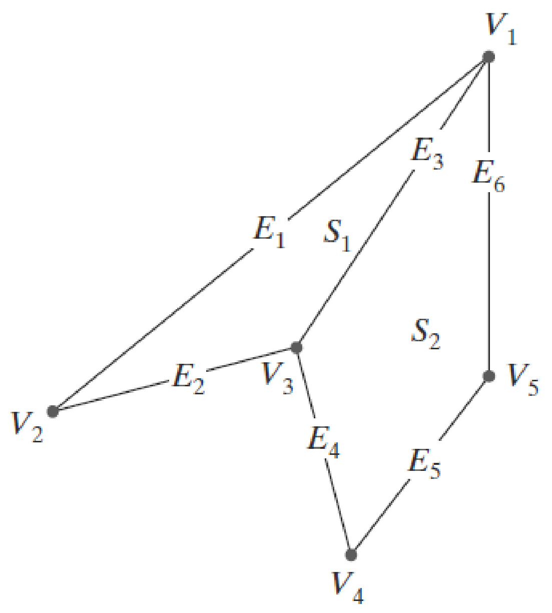
Odd-Even Rule



Non-zero winding number Rule

POLYGON TABLE

- The objects in a scene are described as sets of polygon surface facets.
- Define a surface shape as a **mesh** of polygon patches
- The data are placed into tables that are to be used in the subsequent processing
- Geometric data for the objects in a scene are arranged
- Several **tests** for consistency and completeness. [**EndPt**, **EdgePoly**, **Closed**, …]



VERTEX TABLE	
V_1:	x_1, y_1, z_1
V_2:	x_2, y_2, z_2
V_3:	x_3, y_3, z_3
V_4:	x_4, y_4, z_4
V_5:	x_5, y_5, z_5

EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1

SURFACE-FACET TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

FILLED- AREA PRIMITIVES

A standard output primitive in general graphics packages is a solid-color or patterned polygon area.

1. <u>The scan-line approach</u>

- Determine the overlap intervals for scan lines that cross the area.
- Typically used in general graphics packages to fill polygons, circles, ellipses

2. <u>Filling approaches</u>

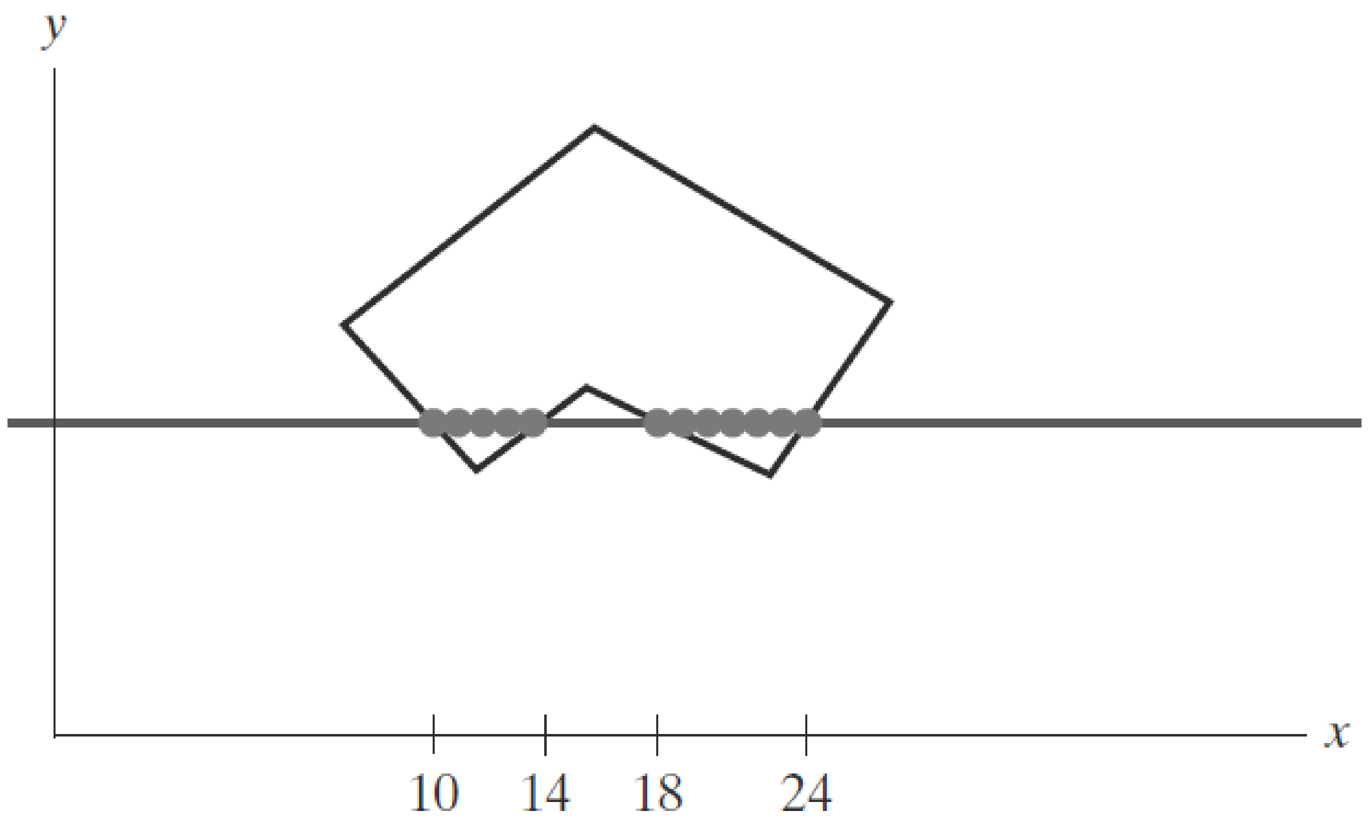
- Start from a given interior position and paint outward from this point until we encounter the specified boundary conditions.
- Useful with more complex boundaries and in interactive painting systems.

SCAN-LINE POLYGON-FILL

- For each scan line crossing a polygon
  - Locate the intersection points of the scan line with the polygon edges.

- These intersection points are then
  - Sorted from left to right,
  - The corresponding frame-buffer positions between each intersection pair are set to the specified fill color.

EXAMPLE



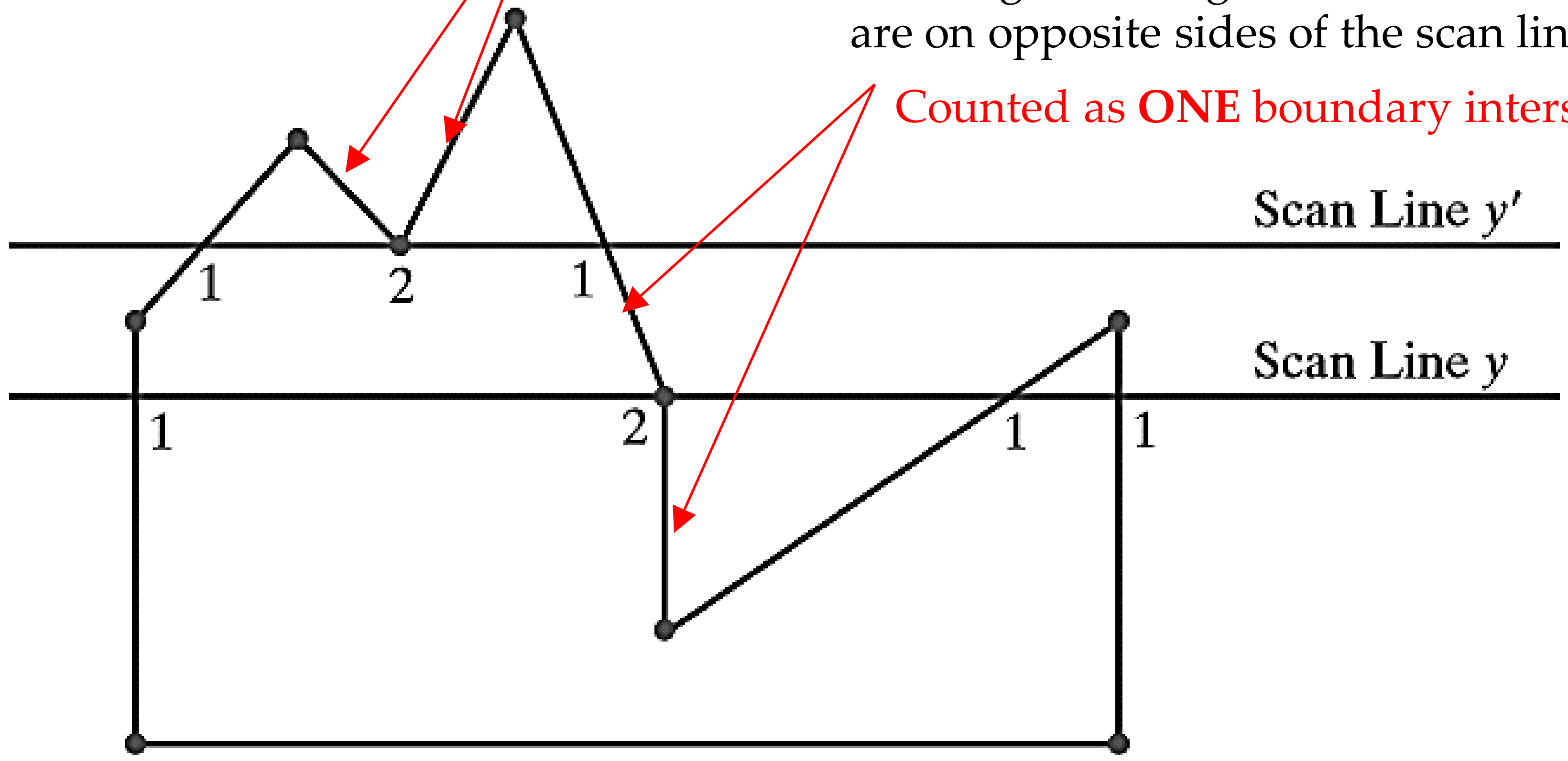
ISSUES

Two intersecting edges are both above the scan line

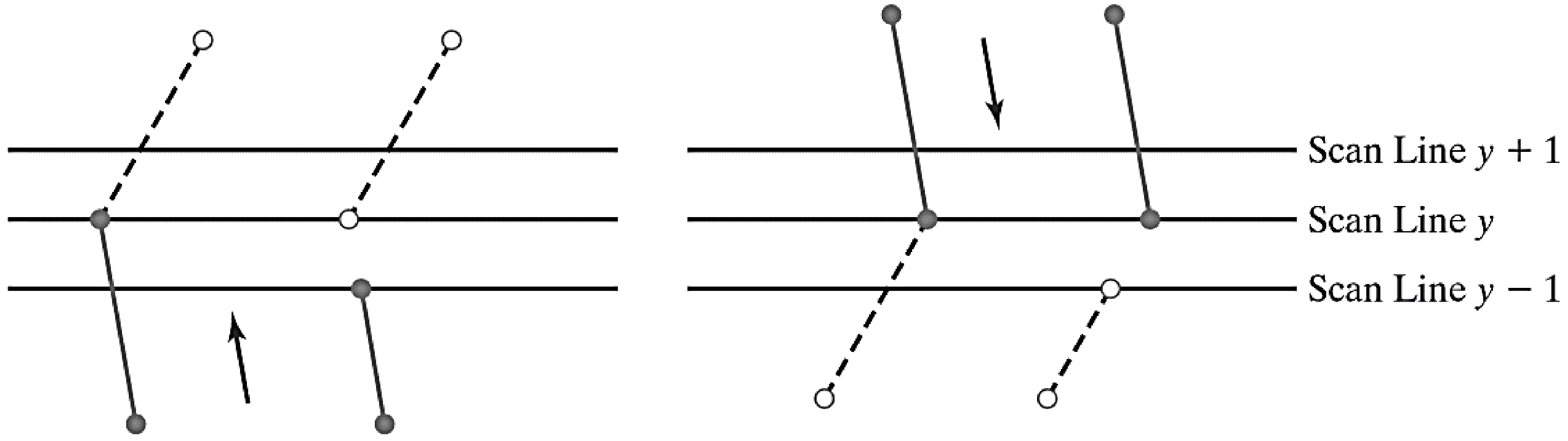
Counted as **TWO** boundary intersection point.

Two edges sharing an intersection vertex are on opposite sides of the scan line

Counted as **ONE** boundary intersection point.



ADJUST END POINT CALCULATION

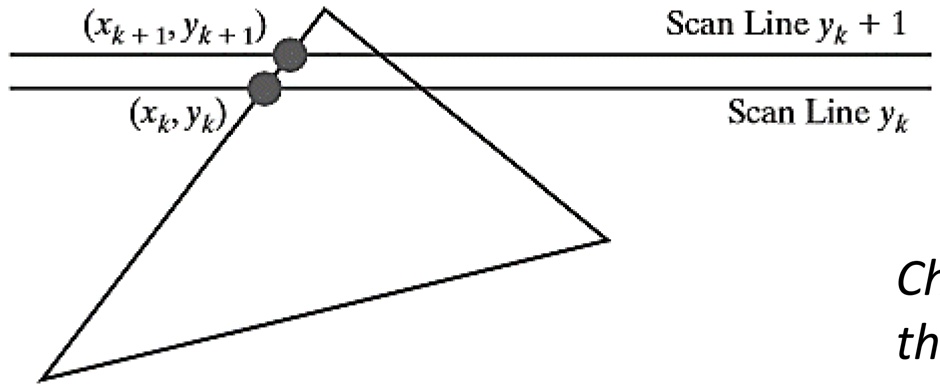


FILLED- AREA PRIMITIVES (CONT.)

- Scan-conversion algorithms typically take advantage of various coherence properties of a scene
- Coherence is simply that
  - the properties of one part of a scene are related in some way to other parts of the scene
  - so that the relationship can be used to reduce processing.
- Coherence methods often involve
  - incremental calculations applied along a single scan line
  - between successive scan lines.

COHERENCE EXAMPLE

Slope of the edge



$$\boxed{m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}}$$

Change in y coordinates between the two scan lines is ?

Given the current x-intercept, the next x-intercept coordinate =

$$\boxed{x_{k+1} = x_k + \frac{1}{m}}$$

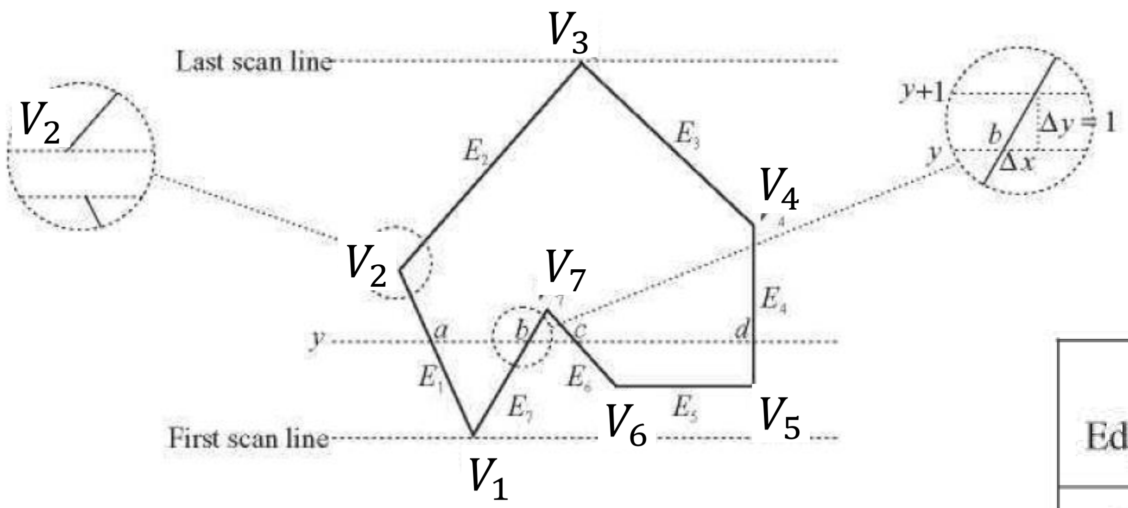
DIFFERENT SITUATIONS WHILE SCANNING

- Intersection point change by the amount of the slope of the line

- Edges may start / end

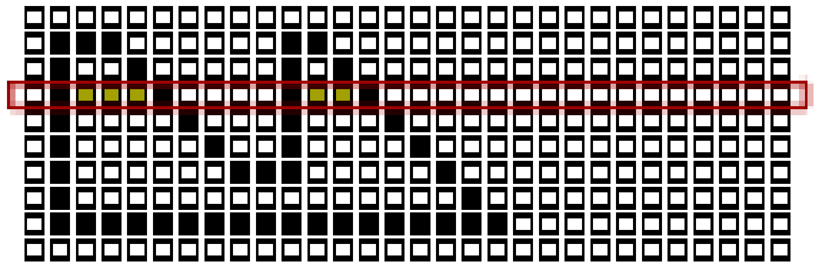
- **Tracking of intersection points is the key**
  - **Vertices**
  - **Edges**
- **Edge table (ET), Active Edge Table (AET)**

EXAMPLE



Edge	$y_{\min}$	$y_{\max}$	x coordinate of vertex with $y = y_{\min}$	$1/m$
E_1	y_1	$y_2 - 1$	x_1	$1/m_1$
E_7	y_1	y_7	x_1	$1/m_7$
E_4	y_5	$y_4 - 1$	x_5	$1/m_4$
E_6	y_6	y_7	x_6	$1/m_6$
E_2	y_2	y_3	x_2	$1/m_2$
E_3	y_4	y_3	x_4	$1/m_3$

SCAN LINE POLYGON FILLING ALGORITHM



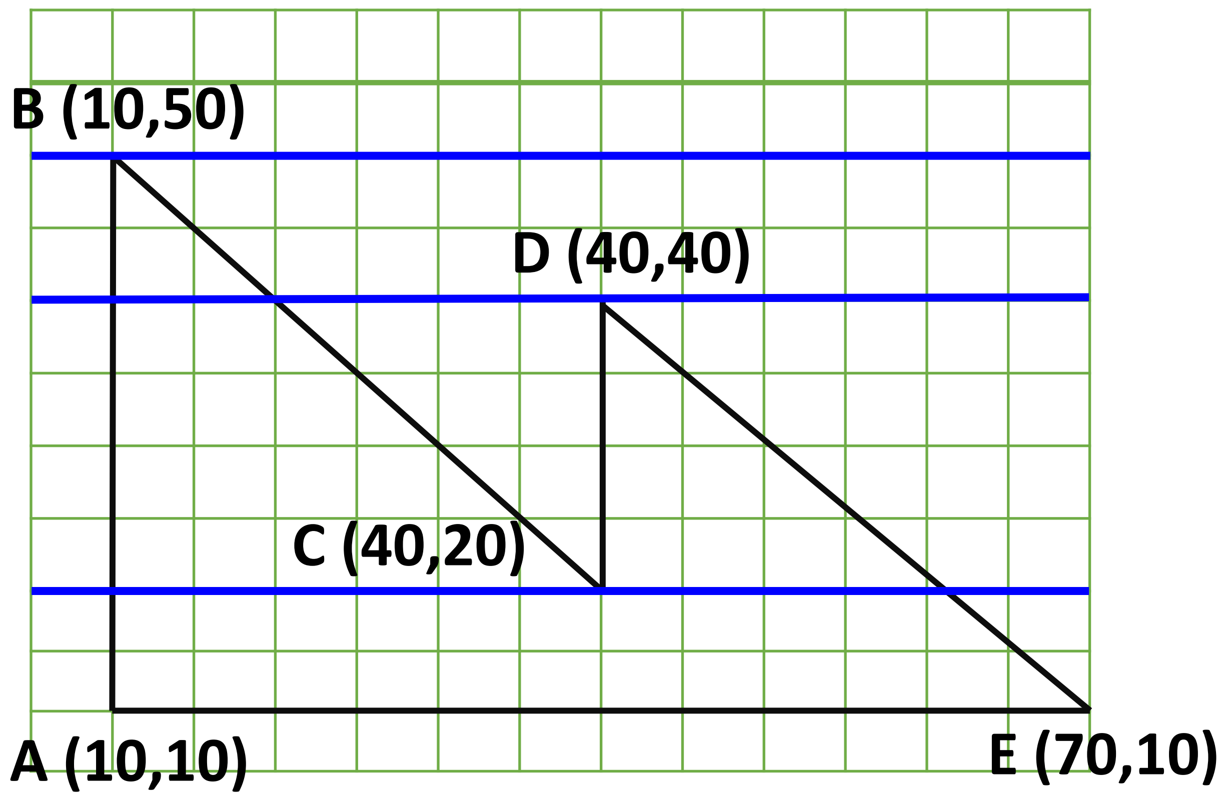
- Pixels within the boundary of a polygon belong to the polygon
- Moving from bottom to top up the polygon
- Starting at a left edge, fill pixels in spans until a right edge is reached

- Once we have an intersection
  - Incrementally compute the next intersection from the current one.

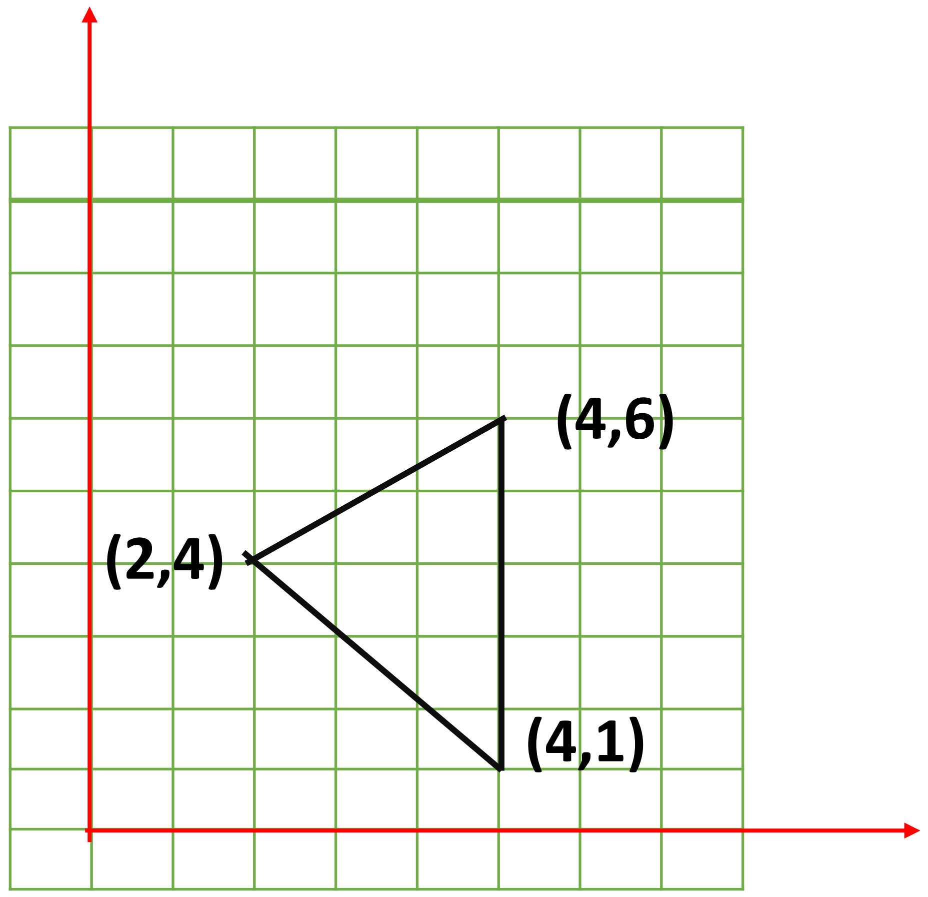
ALGORITHM

1. Set y to the smallest y coordinate that has an entry in the ET
2. Initialize the AET to be empty
3. Repeat until the AET and ET are empty:
    1. Move from ET bucket y to the AET those edges whose $y_{min} = y$ (entering edges).
    2. Remove from the AET those entries for which $y = y_{max}$ (edges not involved in the next scan line), then sort the AET on x
    3. Fill in desired pixel values on scan line y by using pairs of x coordinates from the AET
    4. Increment y by 1
    5. For each non-vertical edge remaining in the AET, update x for the new y

EXAMPLE



EXERCISE



Construct the

1. Global Edge Table (GET)
2. Active Edge Table (AET) for y=1 to 6

Identify the pair of points to be highlighted

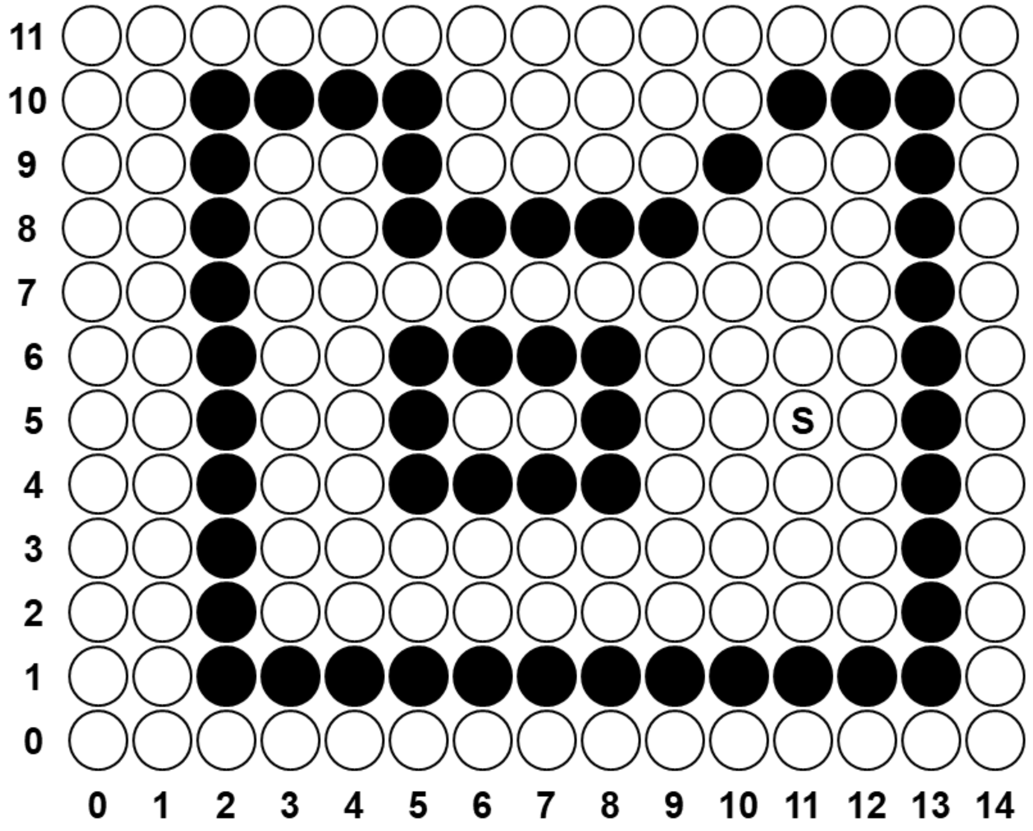
ALGORITHM

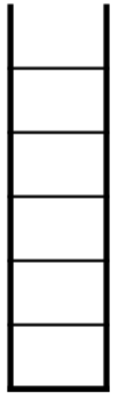
1. Starting from the **initial interior pixel**, then fill in the contiguous span of pixels on this starting scan line.

2. 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.

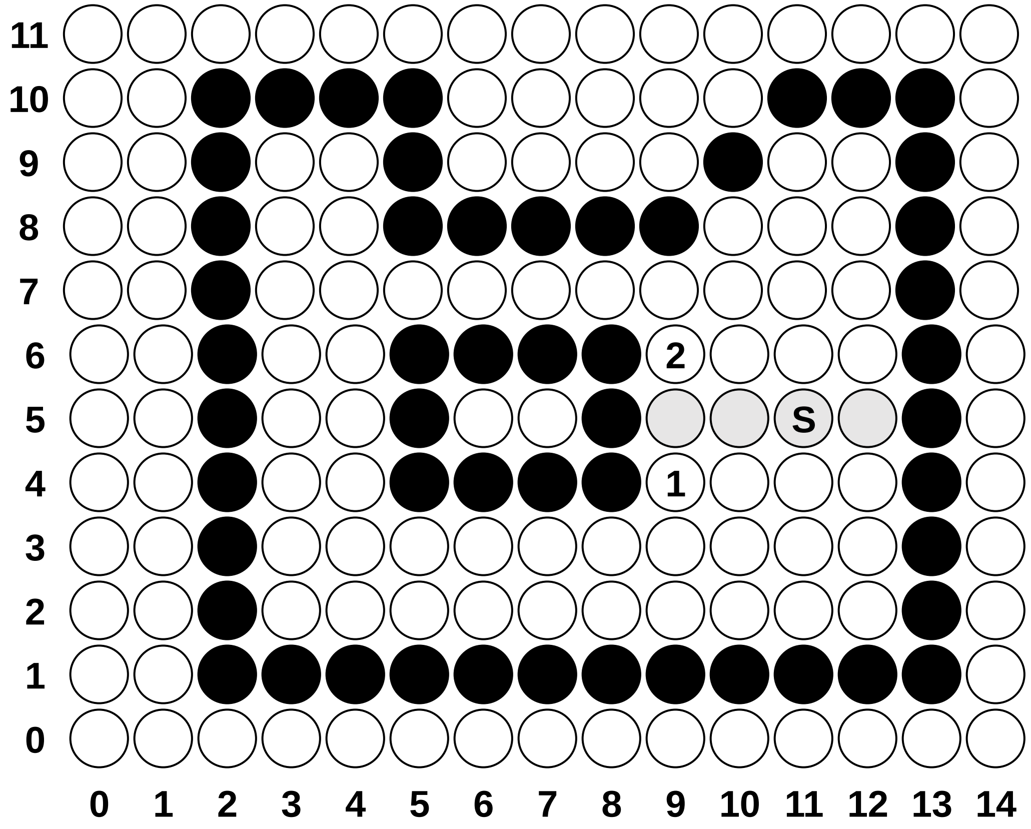
3. At each subsequent step, unstack the next start position and repeat the process.

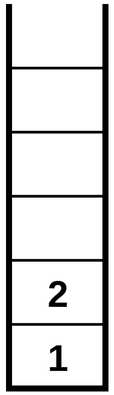
SPAN FLOOD-FILL ALGORITHM (EXAMPLE)



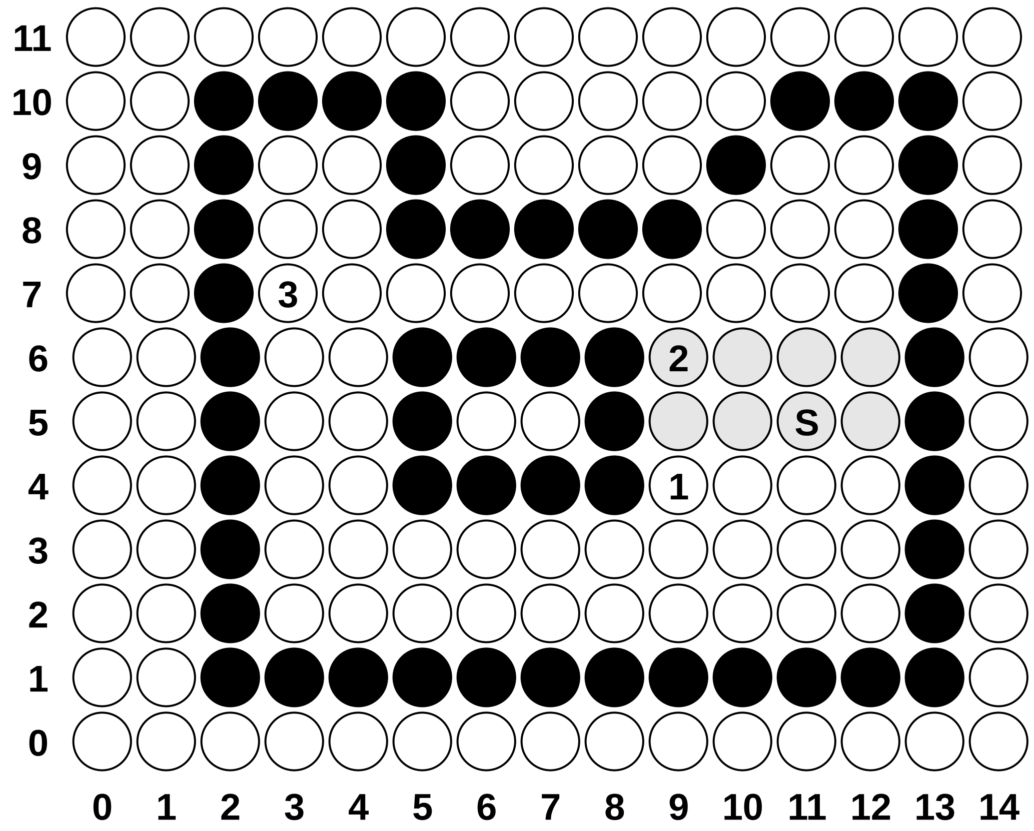


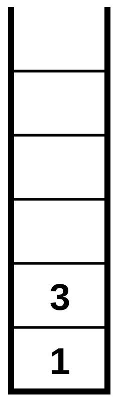
Span Flood-Fill Algorithm (example)



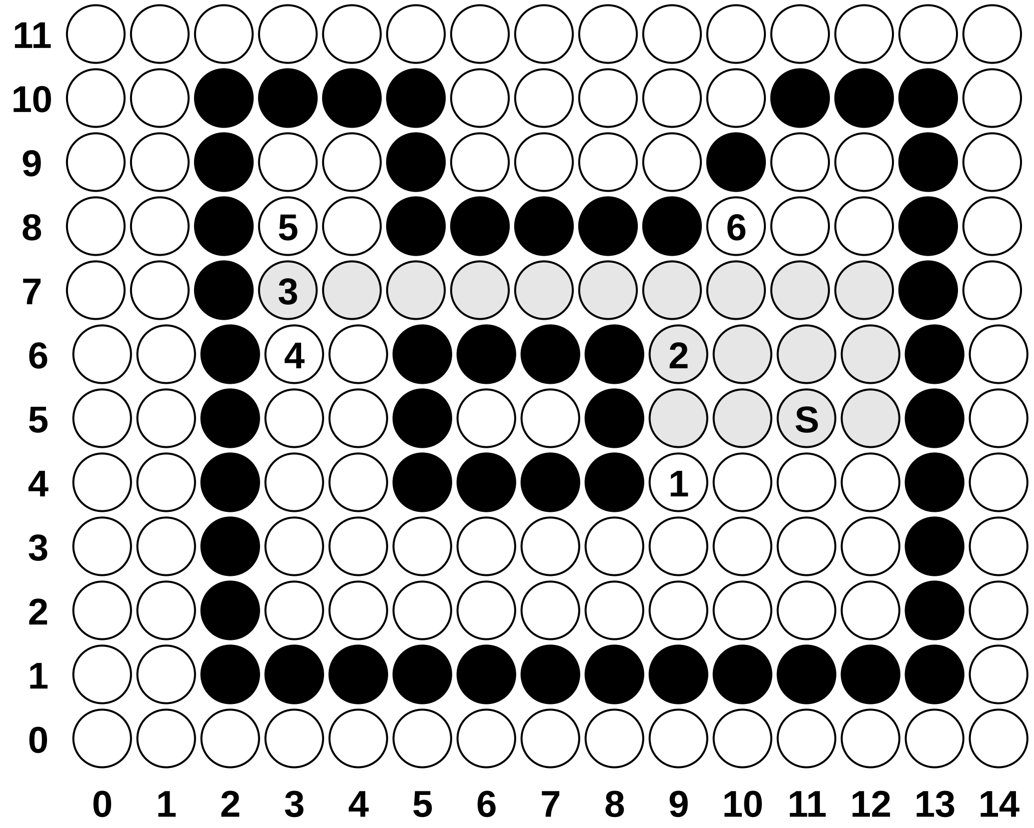


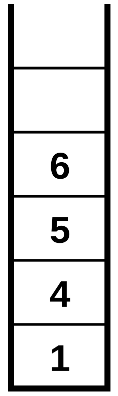
Span Flood-Fill Algorithm (example)



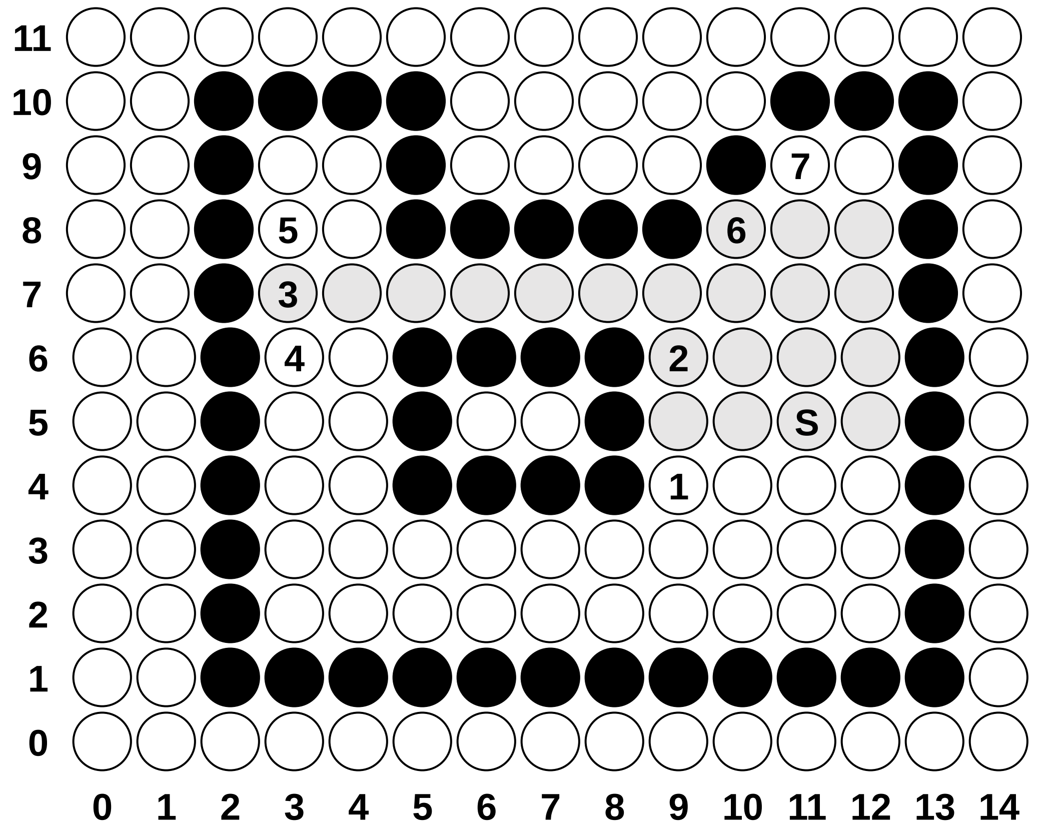


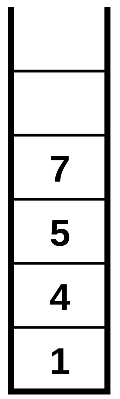
Span Flood-Fill Algorithm (example)



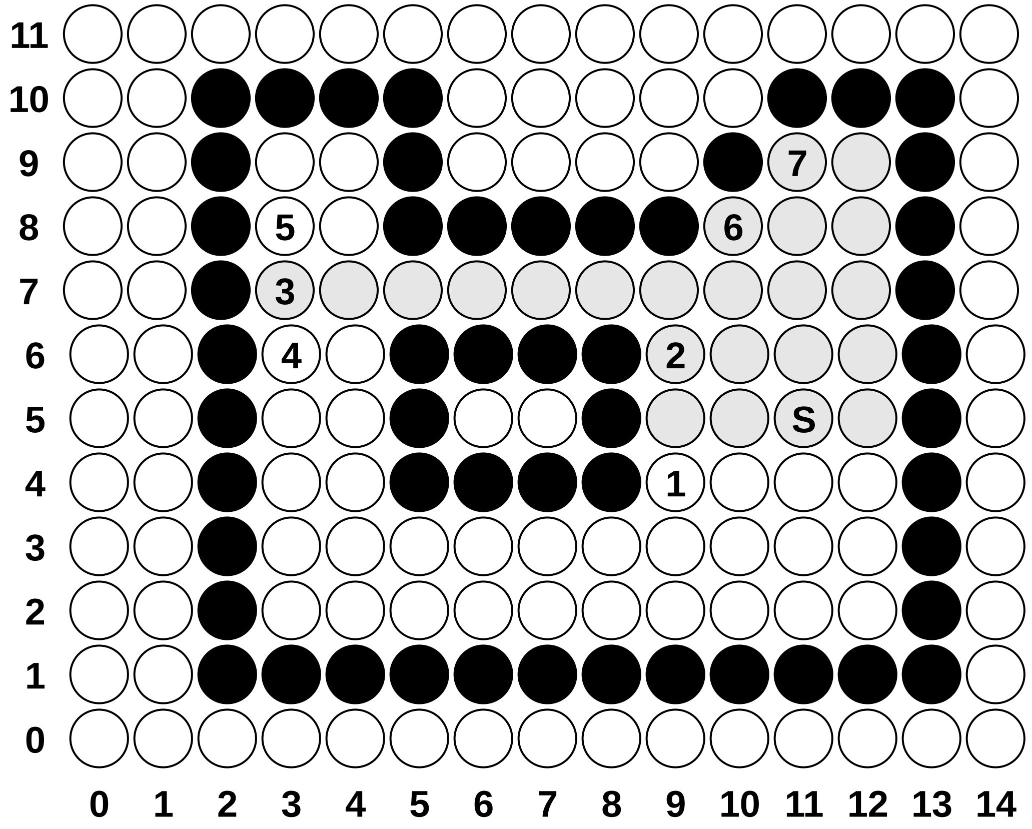


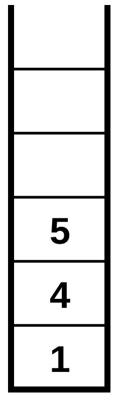
Span Flood-Fill Algorithm (example)



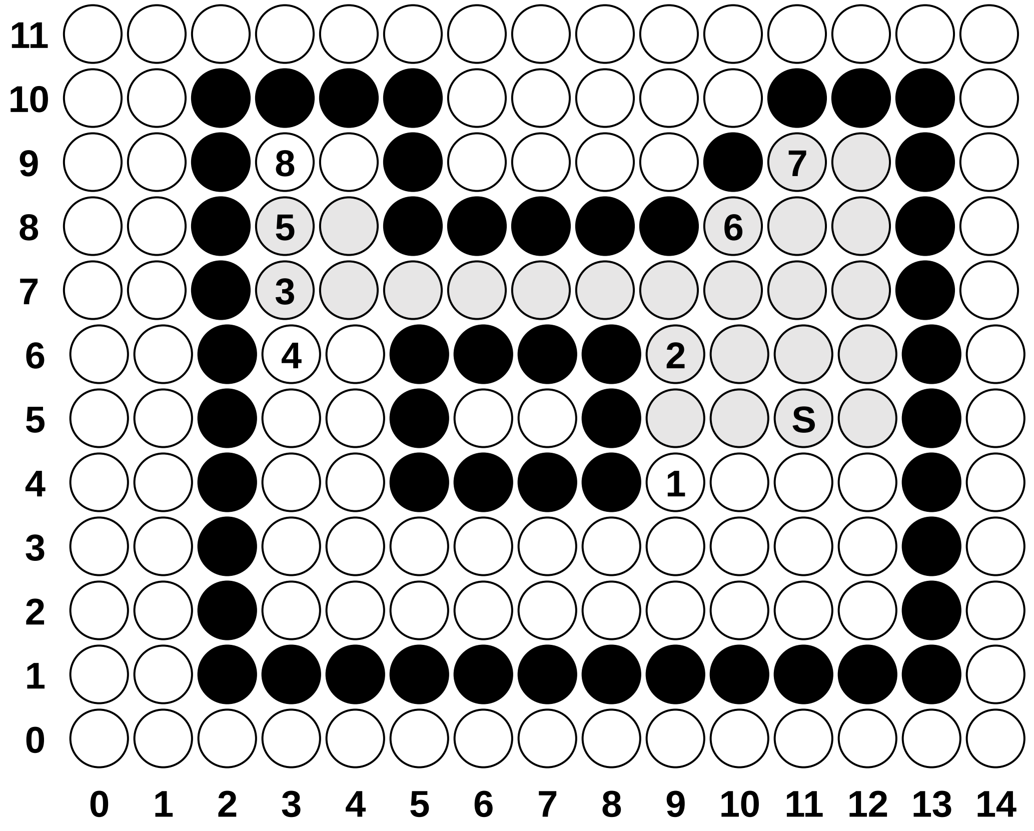


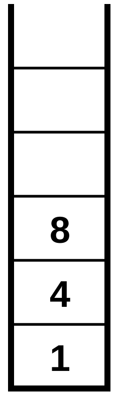
Span Flood-Fill Algorithm (example)



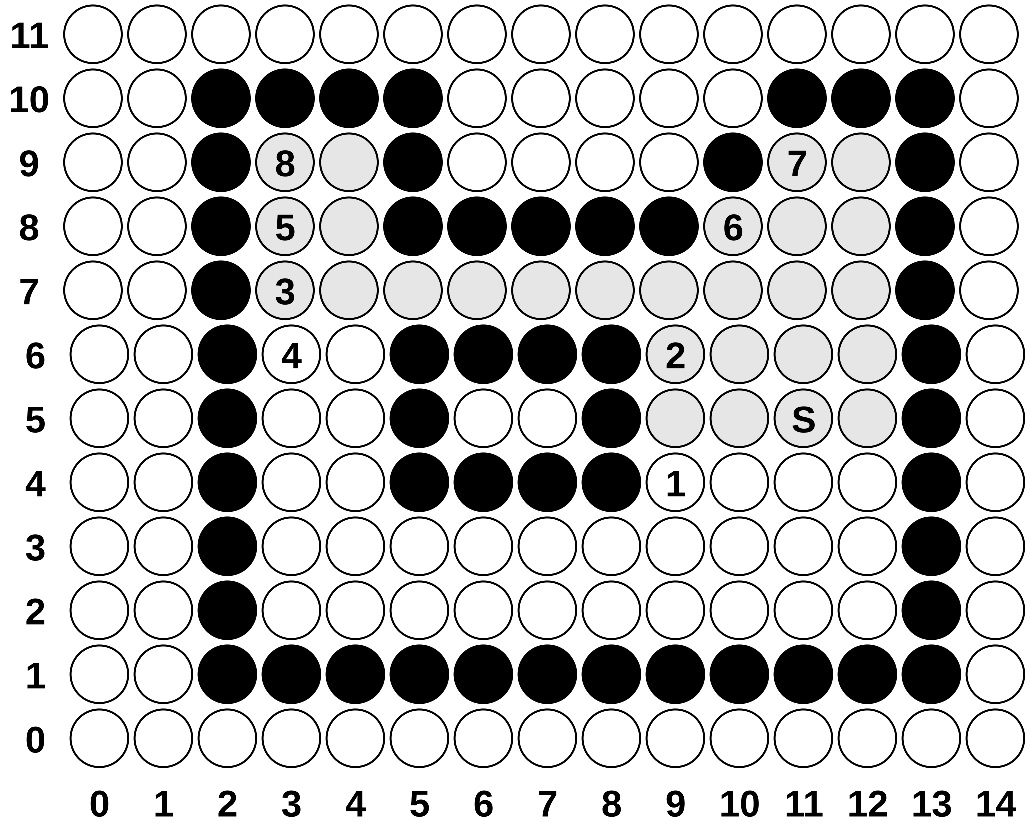


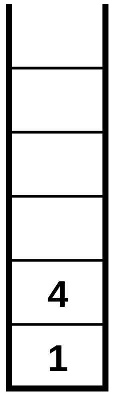
Span Flood-Fill Algorithm (example)



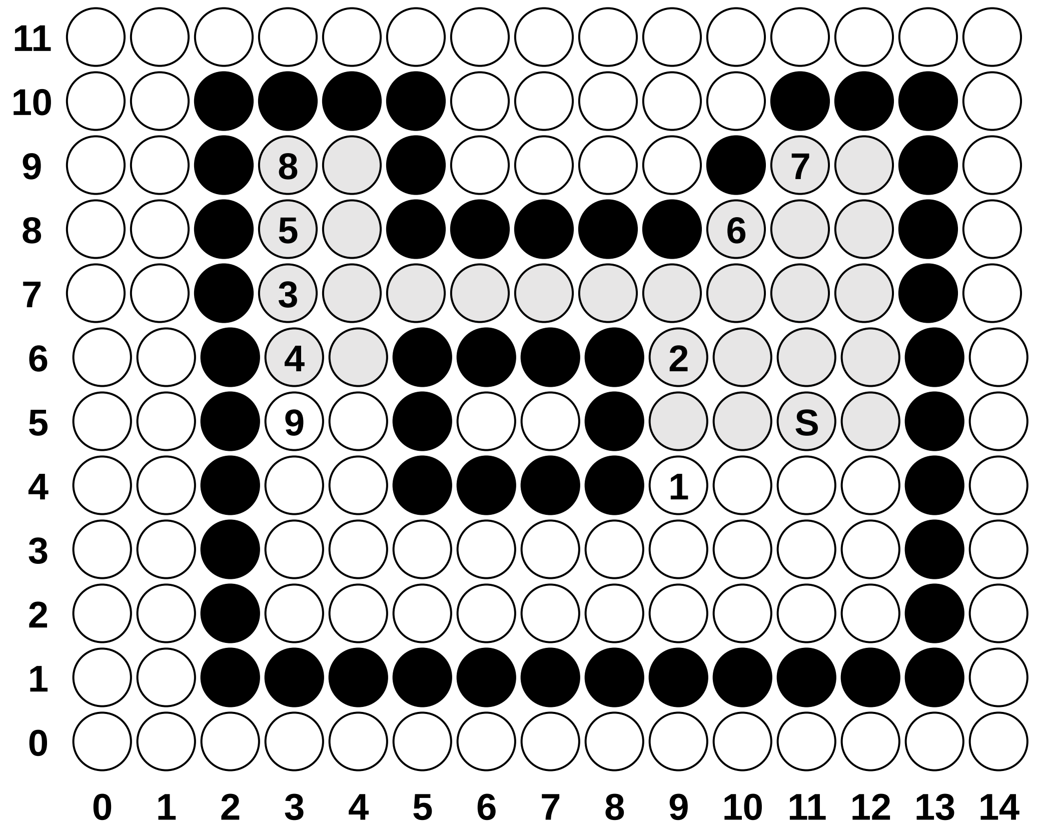


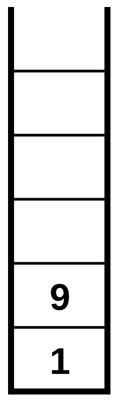
Span Flood-Fill Algorithm (example)



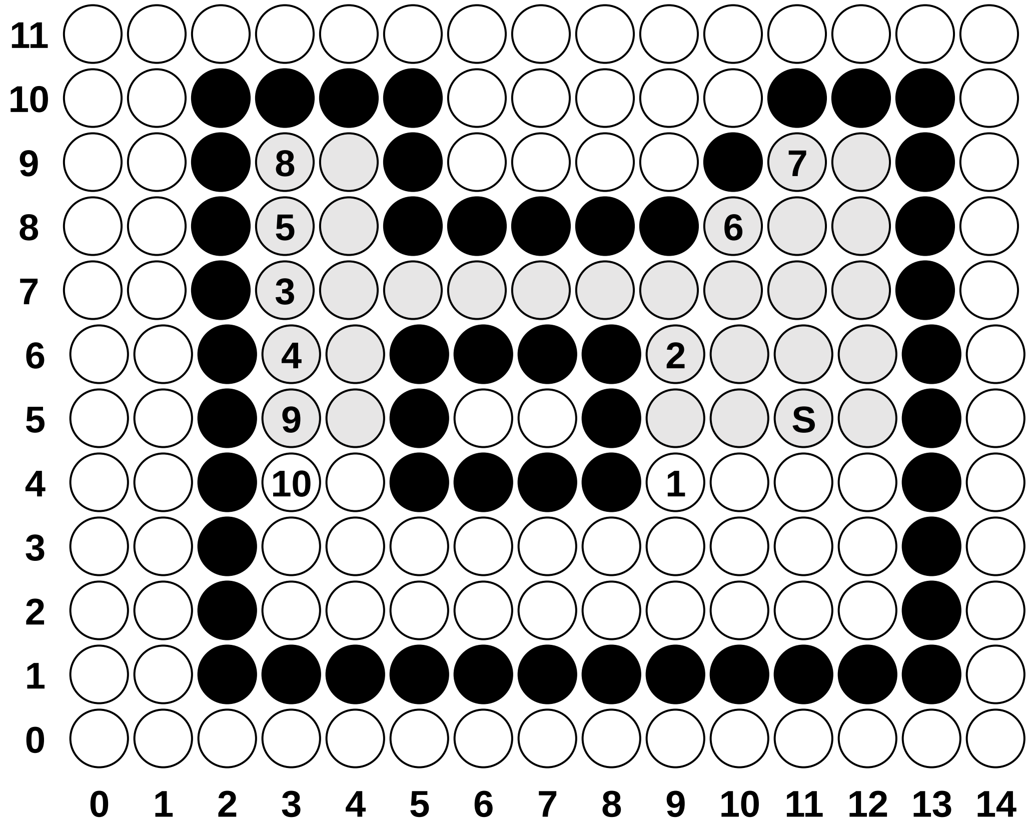


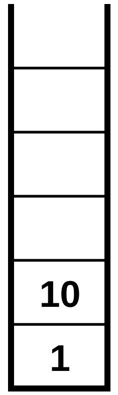
Span Flood-Fill Algorithm (example)



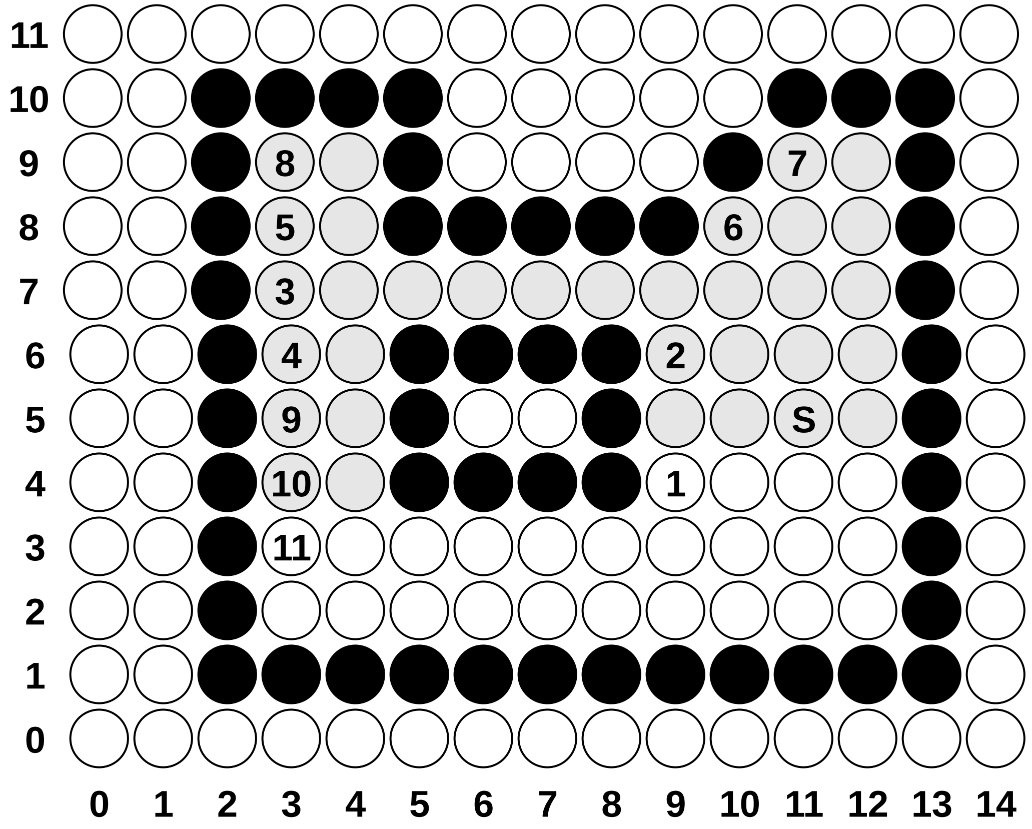


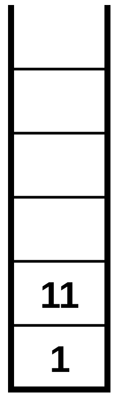
Span Flood-Fill Algorithm (example)



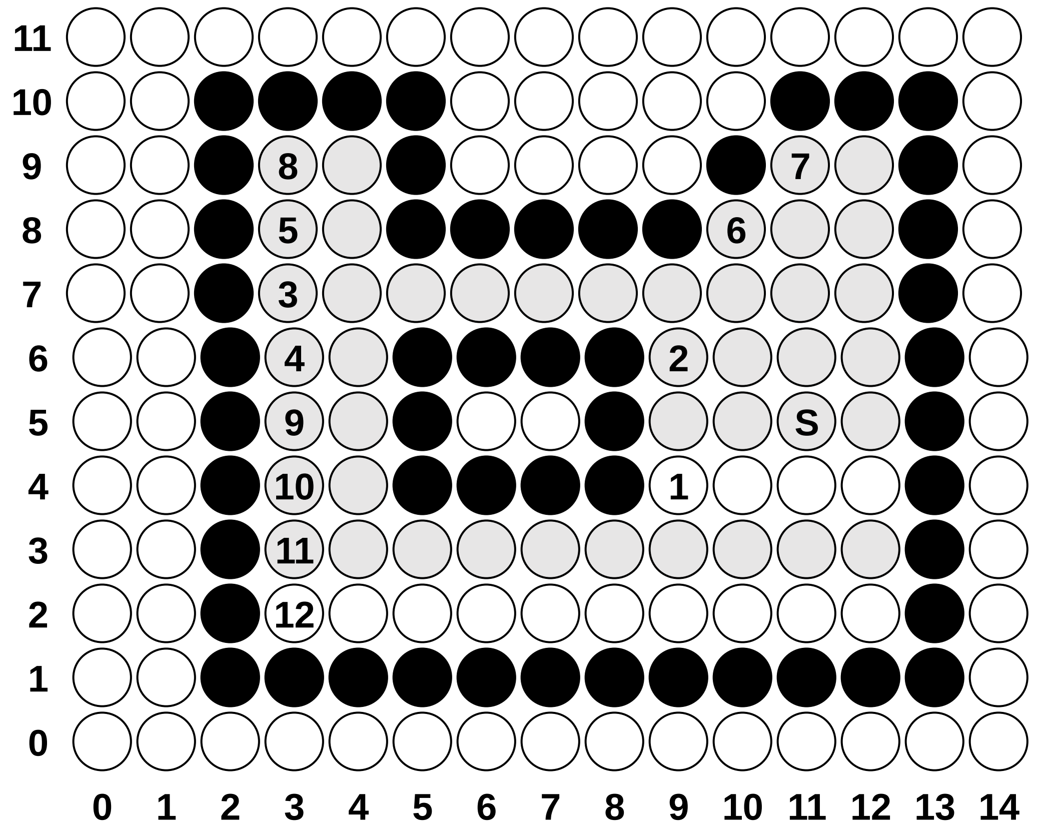


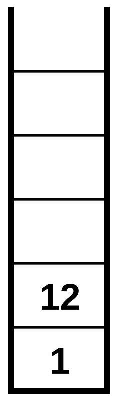
Span Flood-Fill Algorithm (example)



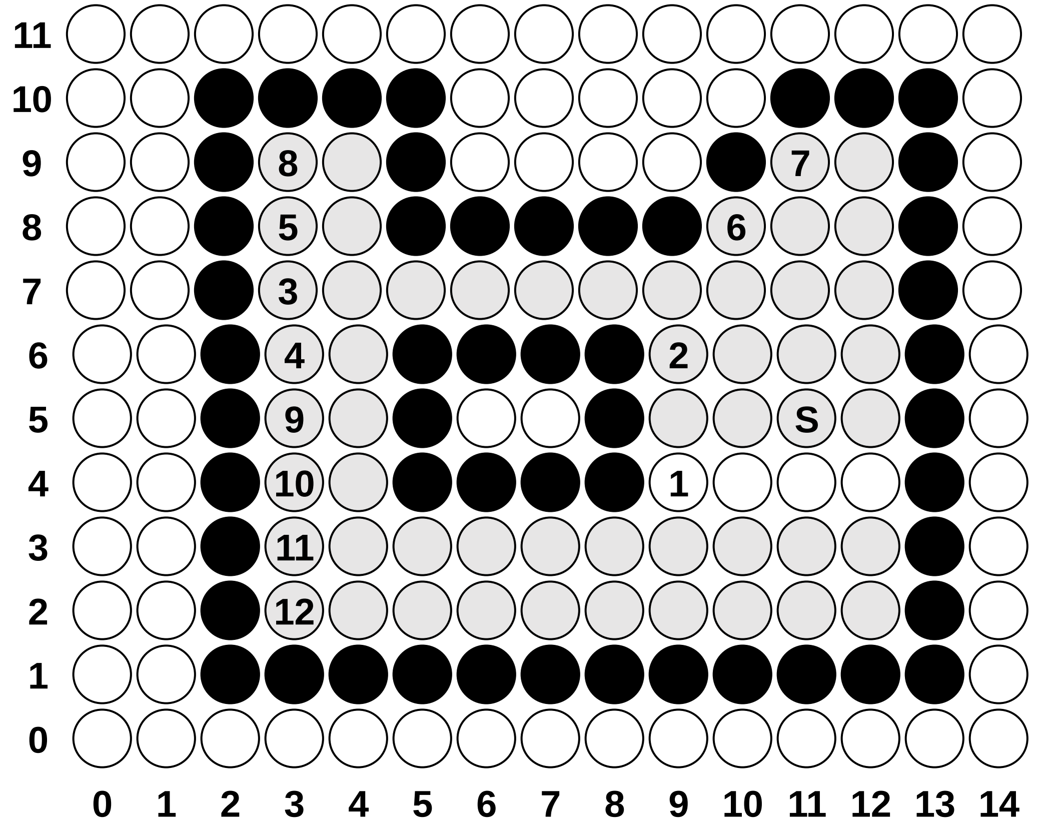


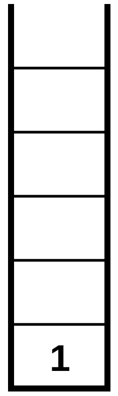
Span Flood-Fill Algorithm (example)



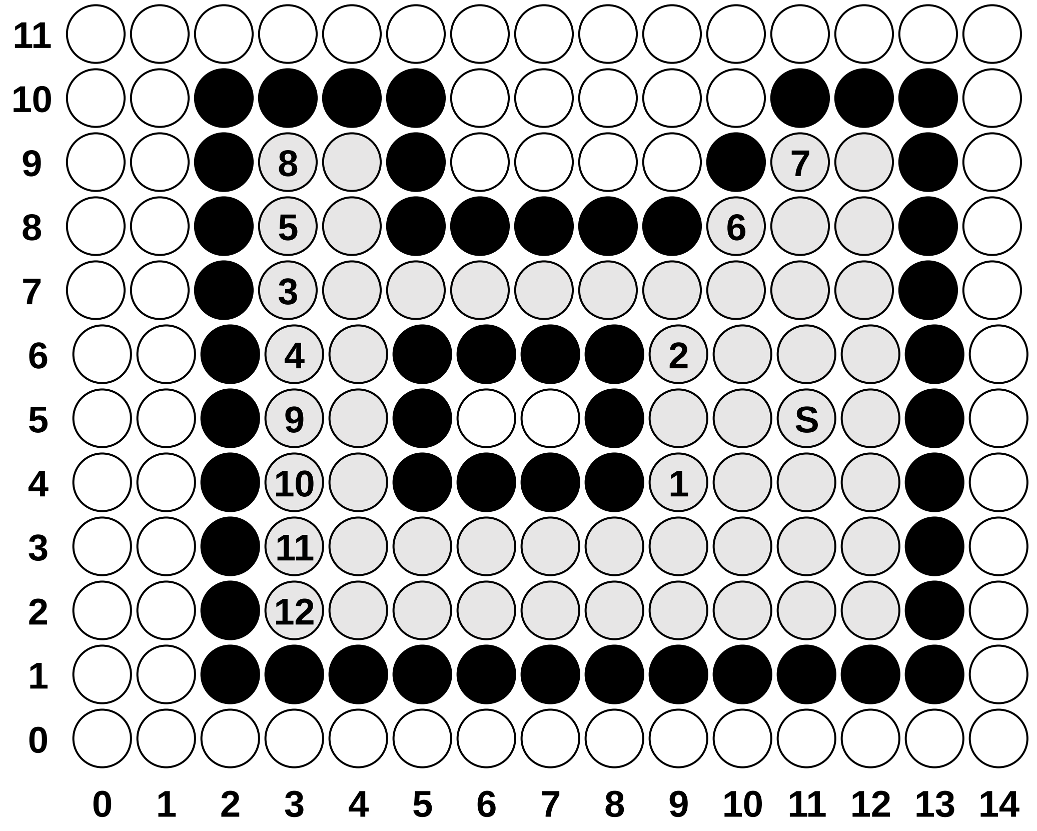


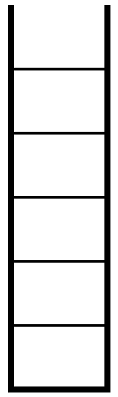
Span Flood-Fill Algorithm (example)





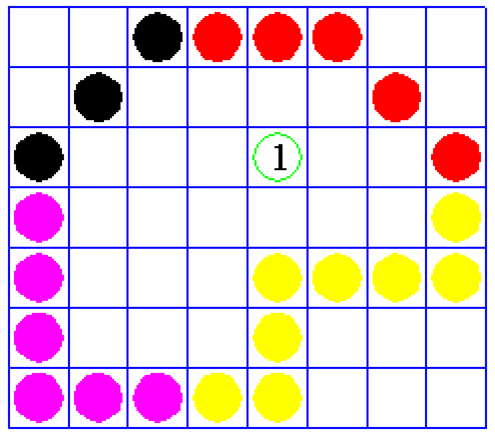
Span Flood-Fill Algorithm (example)





FLOOD FILL ALGORITHM

- Fill in (recolor) an area that is not defined within a single color boundary.
- Replace a specified interior color instead of searching for a boundary color value.
- This approach is called a **flood-fill algorithm**.



FLOOD FILL ALGORITHM

1. 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.

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

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

FILLED- AREA PRIMITIVES (CONT.)

- Scan-conversion algorithms typically take advantage of various coherence properties of a scene
- Coherence is simply that
  - the properties of one part of a scene are related in some way to other parts of the scene
  - so that the relationship can be used to reduce processing.
- Coherence methods often involve
  - incremental calculations applied along a single scan line
  - between successive scan lines.

FLAME UNIVERSITY

EVERLASTING *learning*

THANK YOU