Area Filling

Instructor: Dinesh Maharjan

2022

Region Filing

- Fills specified region with specified color
- It involves two decision
- The decision of which pixel to fill
- And decision of which color to fill with

Type of Region Filling

- Scan line Polygon filling
- Seed filling (Boundary Fill and Flood Fill)

Seed Fill Algorithm

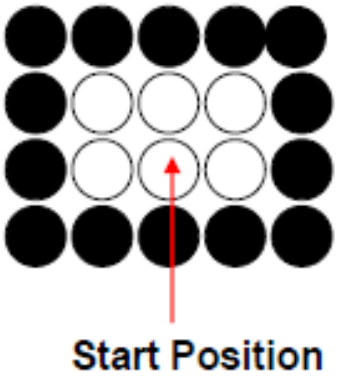
- Simple idea
- Start with one interior pixel and color the region progressively.
- Seed is one of the interior pixel.
- Progressively fills 4 connected or 8 connected pixels of the seed.
- 4 connected are top, down, left, right pixels
- 8 connected are top, top-left, top-right, left, right, bottom, bottom-left, bottom-right

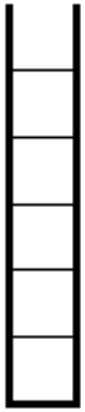
Boundary Fill Algorithm

- Boundary color, fill color, and the seed.
- Push seed to stack.
- repeat.
  - Current pixel=pop stack()
  - Apply specified color to current pixel
  - For each neighboring of seed
    - If neighbor color!= boundary color or neighbor color!=fill color,
      - Push neighboring pixel to stack.
- Until stack is empty

Boundary Fill Algorithm

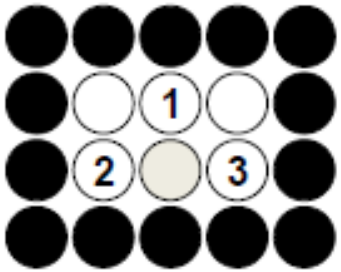
4-connected (Example)





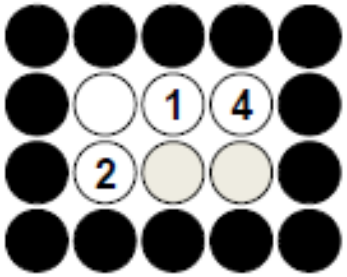
Boundary Fill Algorithm

4-connected (Example)



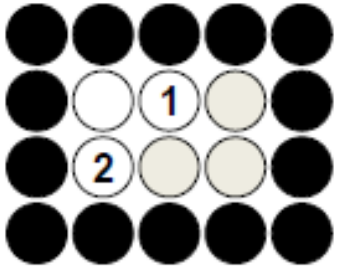
Boundary Fill Algorithm

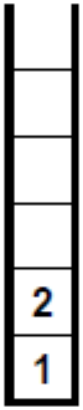
4-connected (Example)



Boundary Fill Algorithm

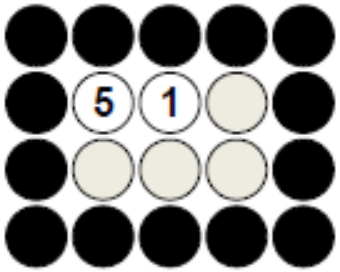
4-connected (Example)

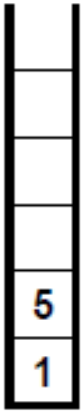




Boundary Fill Algorithm

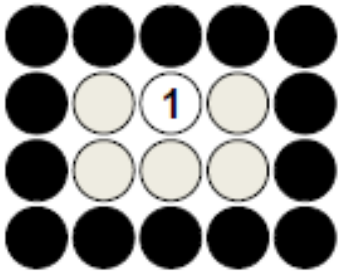
4-connected (Example)

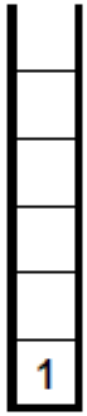




Boundary Fill Algorithm

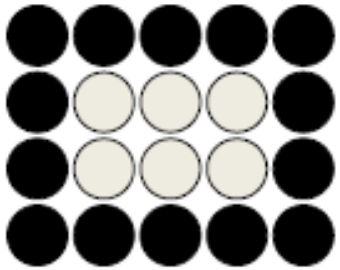
4-connected (Example)

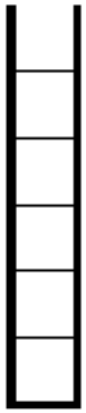




Boundary Fill Algorithm

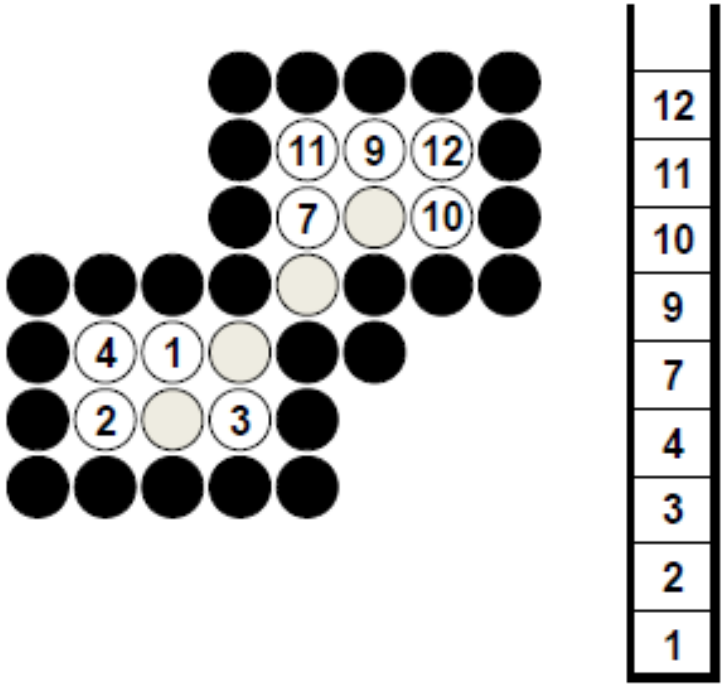
4-connected (Example)





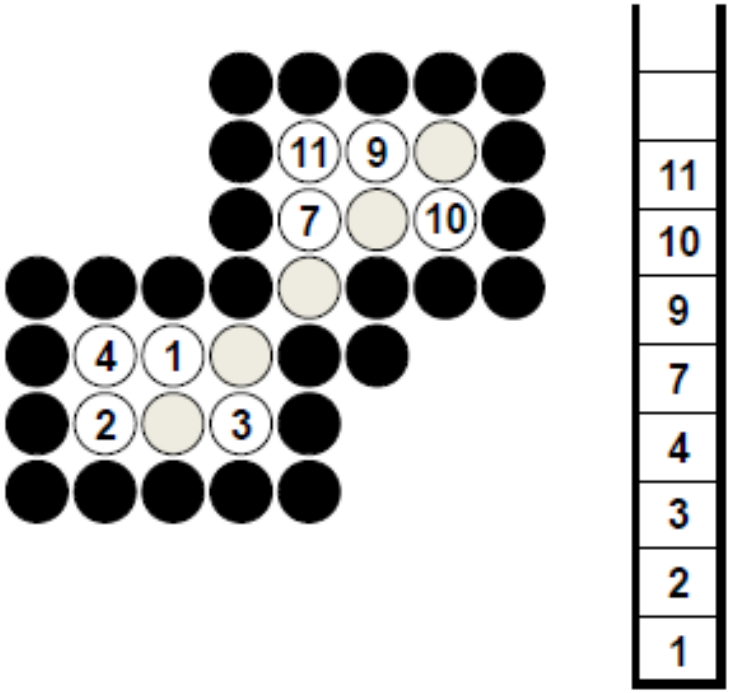
Boundary Fill Algorithm

8-connected (Example)



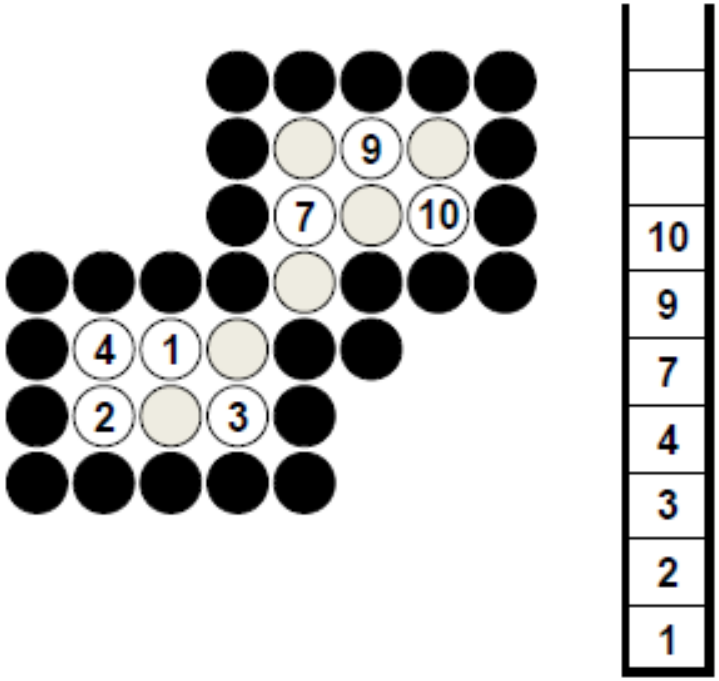
Boundary Fill Algorithm

8-connected (Example)



Boundary Fill Algorithm

8-connected (Example)



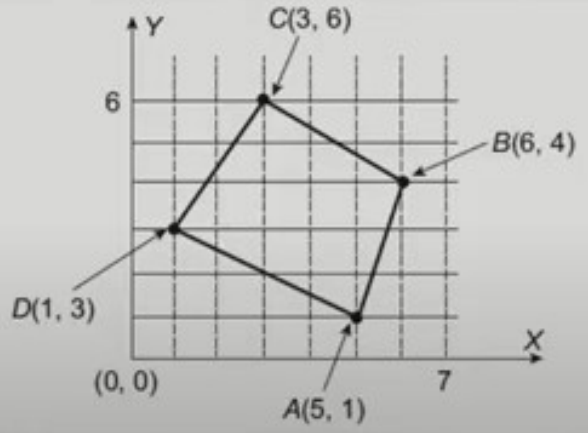
Flood Fill Algorithm

- interior color, fill color, boundary color and the seed.
- Push seed to stack.
- repeat.
  - Current pixel=pop stack()
  - Apply fill color to current pixel
  - For each neighboring pixel of seed
    - If neighbor color == interior pixel
    - Push neighboring pixel to stack.
- Until stack is empty

Scan Line Polygon Algorithm

Illustrative Example

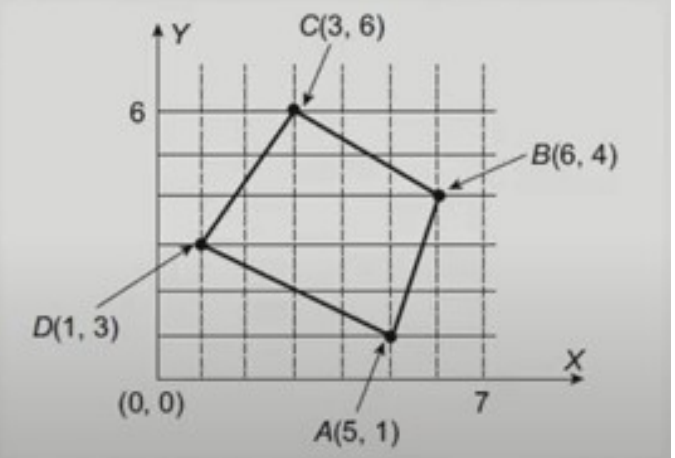
- Polygon specified with vertices *A*, *B*, *C*, and *D* (anti-clockwise vertex naming convention)
- Therefore, edges are *AB*, *BC*, *CD*, and *DA*



Scan Line Polygon Algorithm

Illustrative Example

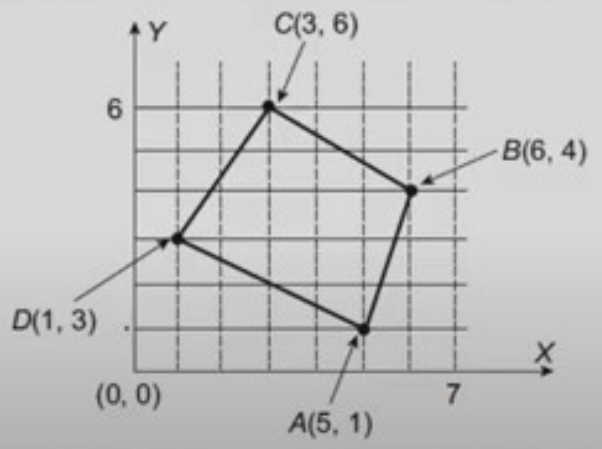
- First, determine max and min scanlines (line 3) from vertex coordinate
  - Max = max{1,4,6,3} = 6
  - Min = min{1,4,6,3} = 1



Scan Line Polygon Algorithm

Illustrative Example

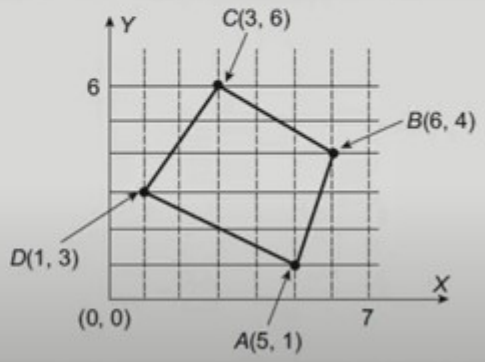
- First iteration of outer loop
  - First determine intersection points of the scan line $y = 1$ with all 4 edges in the inner loop (lines 6–10)



Scan Line Polygon Algorithm

Illustrative Example

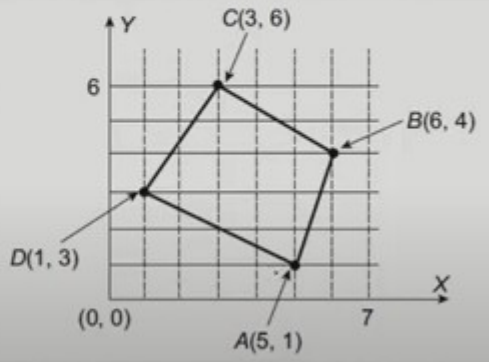
- For *AB*, IF condition satisfied - we determine intersection point as *A* (lines 7–8)
- For *BC* and *CD*, condition not satisfied
- For *DA*, again condition satisfied - we get *A* again



Scan Line Polygon Algorithm

Illustrative Example

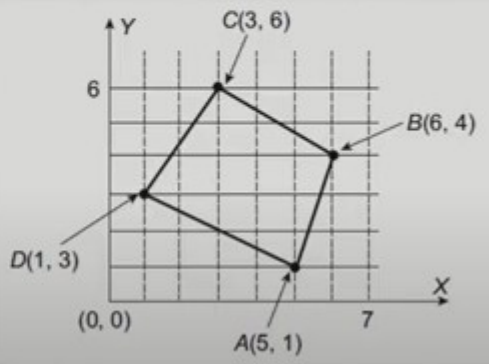
- Thus, 2 intersection points determined are the same vertex *A*
  - This is the only pixel between itself - apply specified color to it (lines 11–12)
- Set scanline = 2 (line 11)
- Since 2 ≠ *max* (= 6), we reenter outer loop



Scan Line Polygon Algorithm

Illustrative Example

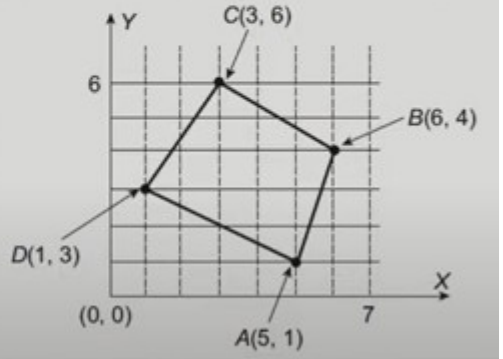
- 2nd iteration of outer loop
  - We check for intersection points between the edges and scanline $y = 2$



Scan Line Polygon Algorithm

Illustrative Example

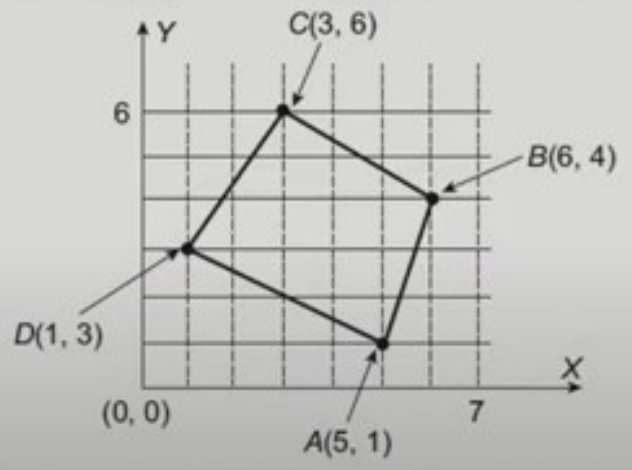
- For *AB*, IF condition satisfied - intersection point $(5\frac{1}{3}, 2)$
- *BC* and *CD* do not satisfy condition – no intersection
- Condition satisfied by *DA* - intersection point is (3,2)



Scan Line Polygon Algorithm

Illustrative Example

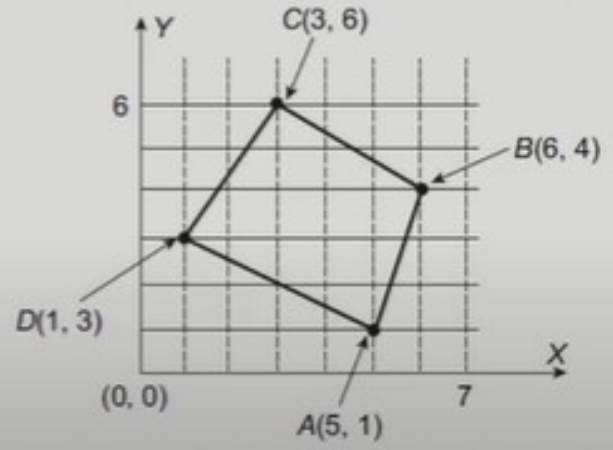
- After sorting (line 11), we get two intersection points (3,2), $(5\frac{1}{3}, 2)$
- Pixels in between - (3,2), (4,2), (5,2)
- Apply specified color to these pixels (line 12) and set scanline = 3
- Since $3 \neq max\ (= 6)$, we reenter the outer loop



Scan Line Polygon Algorithm

Illustrative Example

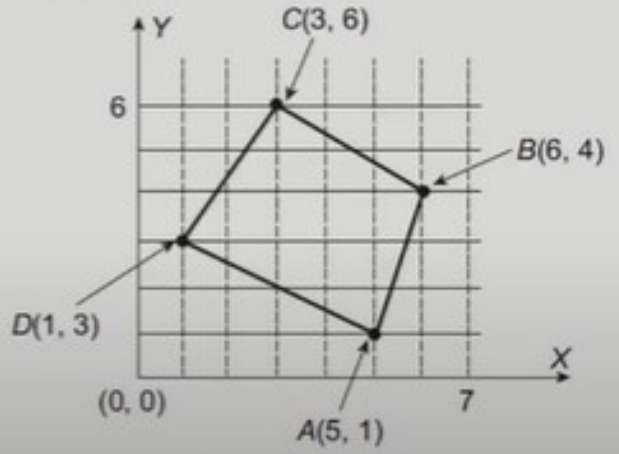
- Algorithm works in similar way for remaining scanlines $y = 3$, $y = 4$, $y = 5$, and $y = 6$



Scan Line Polygon Algorithm

Illustrative Example

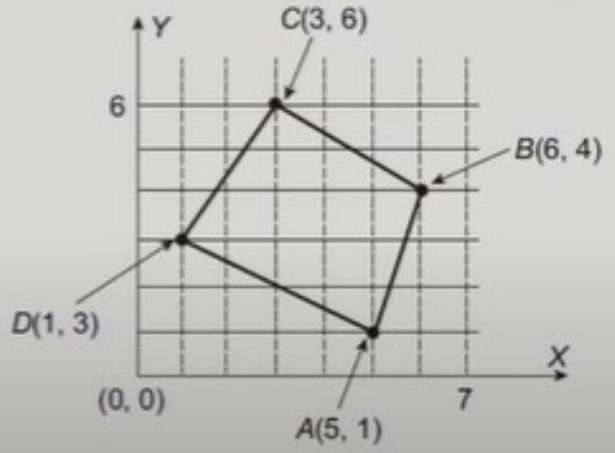
- TWO things in the algorithm require elaboration



Scan Line Polygon Algorithm

Illustrative Example

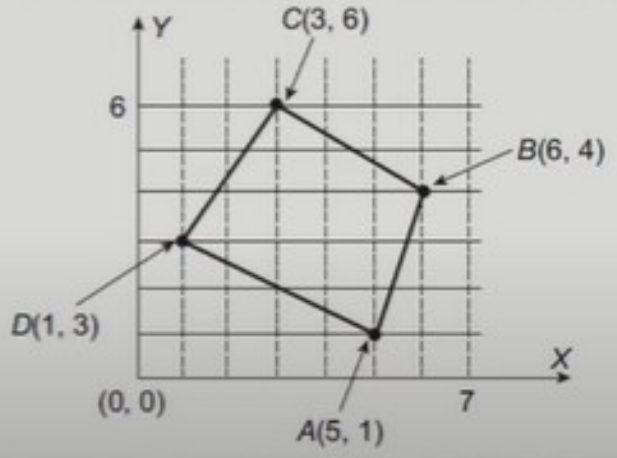
- First, how do we determine the edge–scanline intersection point?
  - Determine line eq from vertices
  - Evaluate with scan line value



Scan Line Polygon Algorithm

Illustrative Example

- Second, how do we determine pixels within two intersection points?
  - Start from left most pixel (x>intersection point)
  - Continue along scan line till x<right intersection



Scan Line Polygon Algorithm

- Is used to fill the polygon region
- Polygon region is the space enclosed by three or more edges.
- So, polygon is defined in terms of vertices and edges.

- Input: Set of vertices, v , specified color, c
- mn=minimum of among y-coordinate from v
- mx=maximum among y-coordinate from v
- s=mn
- Repeat
  - For each edge ((x1,x2), (x2,y2))
    - If (y1<=s<=y2) or (y2<=s<=y1)
      - Find edge and scanline intersection
  - Sort intersection points in increasing order of x-coordinate
  - Apply specified color to the points between intersection points.
- S=s+1
- Until s==mx

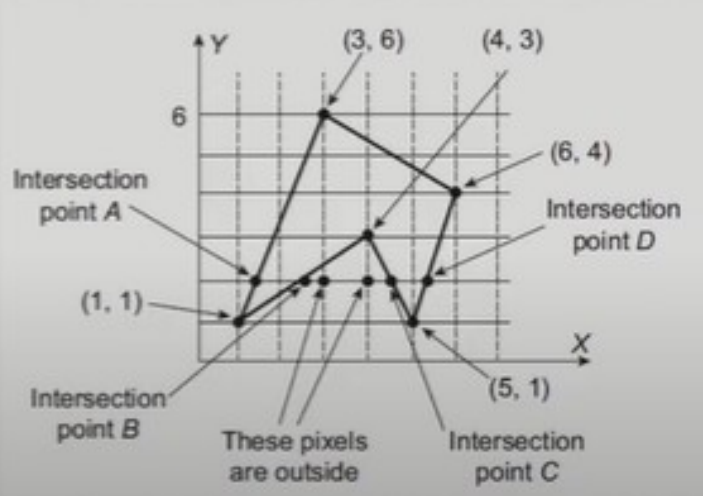
Scan Line Fill for Concave Polygon

- The points between edge and scanline may lie outside if the polygon is concave.
- For such polygon, we need to determine the points that lie inside polygon.
- For testing if a point, p1 lies outside of polygon,
- We need to determine max x and max y from all the vertices of polygon.
- Then choose an arbitrary outside point p2 which is easy as we have max x and max y.

Scan Line Polygon Algorithm

Scan Line Fill for Concave Polygons

- Earlier, we determine pixels between pair of edge–scanline intersection points
  - All these pixels may not be **inside** in case of concave polygon
- We also need to determine **inside** pixels



Scan Line Fill for Concave Polygon

- Join the points p1 and p2.
- If the line cuts the polygon at even number of points then it is outside of polygon.
- Otherwise line is inside of polygon.