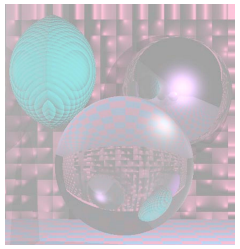
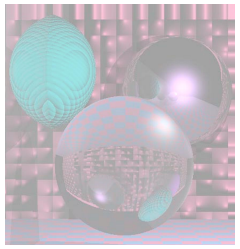


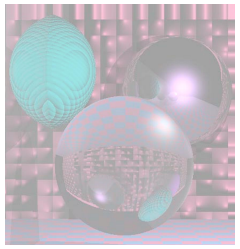
CS 4300
Computer Graphics

Prof. Harriet Fell
Fall 2012
Lecture 9 – September 24, 2012



Today's Topics

- Fill: Flood Boundary Fill vs. Polygon Fill
  - See Photoshop for Flood Fill
- 2D Polygon Fill

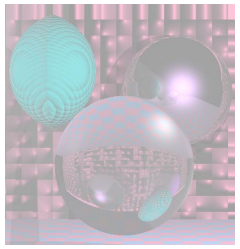


Simple 2D Polygon

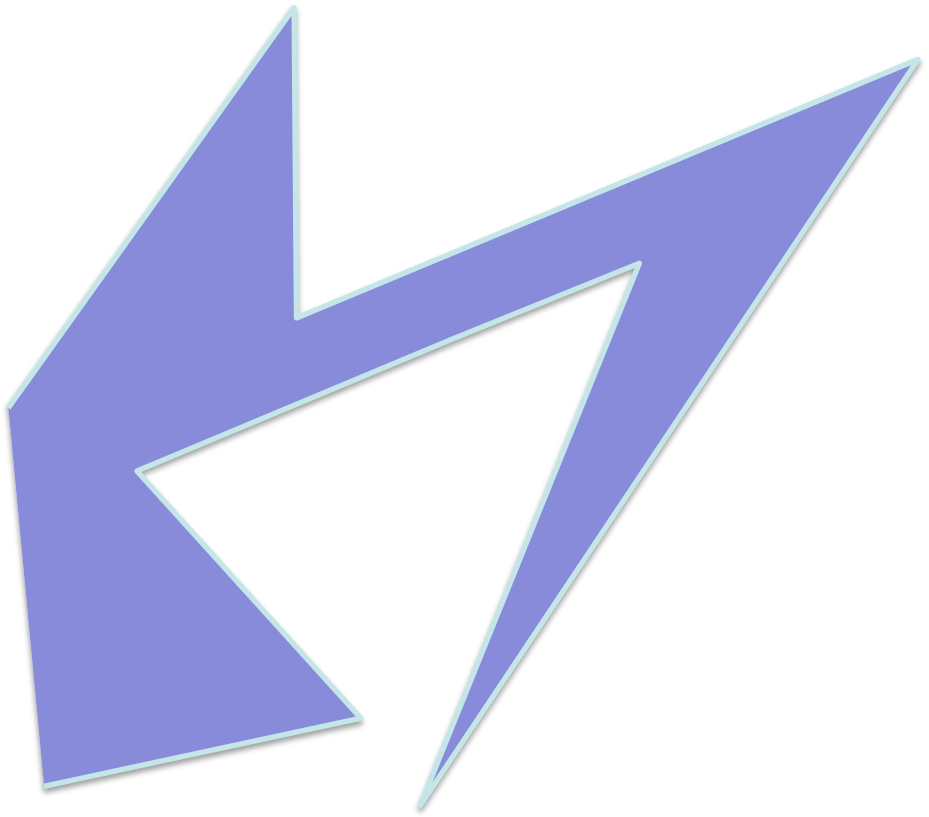
- an ordered sequence of line segments

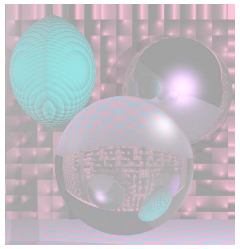
$$g_0, g_1, \ldots, g_{n-1} \quad n \geq 2$$

- such that
  - each edge $g_i = (s_i, e_i)$ is the segment from a start vertex s_i to an end vertex e_i
  - $e_{i-1} = s_i$ for $0 < i \leq n-1$ and $e_{n-1} = s_0$
  - non-adjacent edges do not intersect
  - the only intersection of adjacent edges is at their shared vertex



Scan Line Polygon Fill

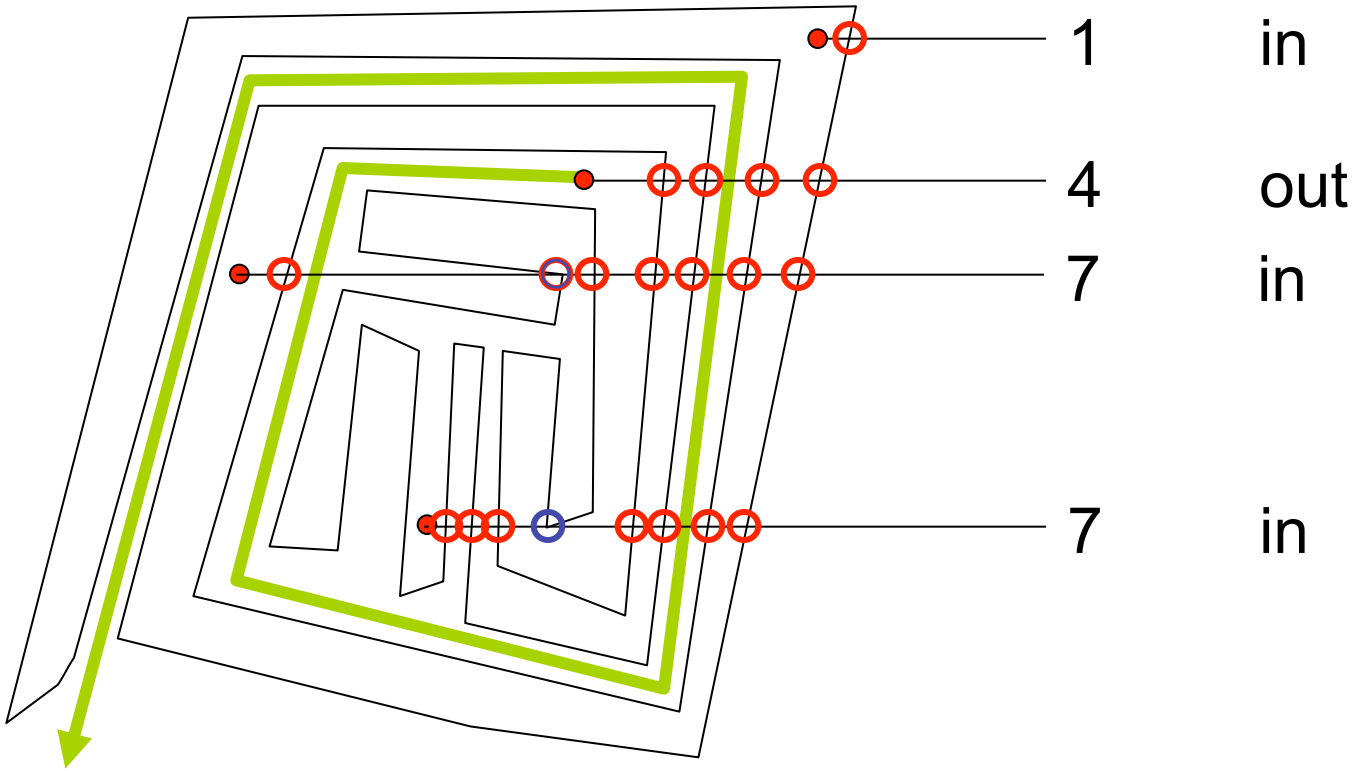


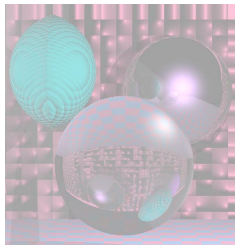


Parity Check

Draw a horizontal half-line from P to the right.

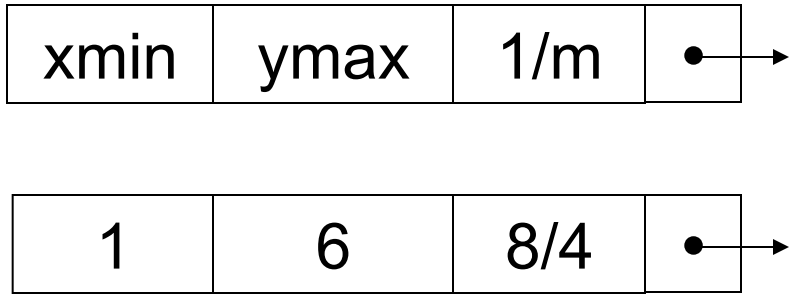
Count the number of times the half-line crosses an edge.

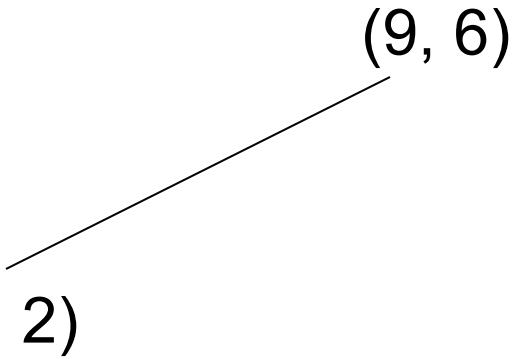




Polygon Data Structure

edges





xmin = x value at lowest y

ymax = highest y

Why 1/m?

If y = mx + b, x = (y-b)/m.

x at y+1 = (y+1-b)/m = (y-b)/m + 1/m.

Polygon Data Structure

13

12

11

10 ➔ e6

9

8

7 ➔ e4 ➔ e5

6 ➔ e3 ➔ e7 ➔ e8

5

4

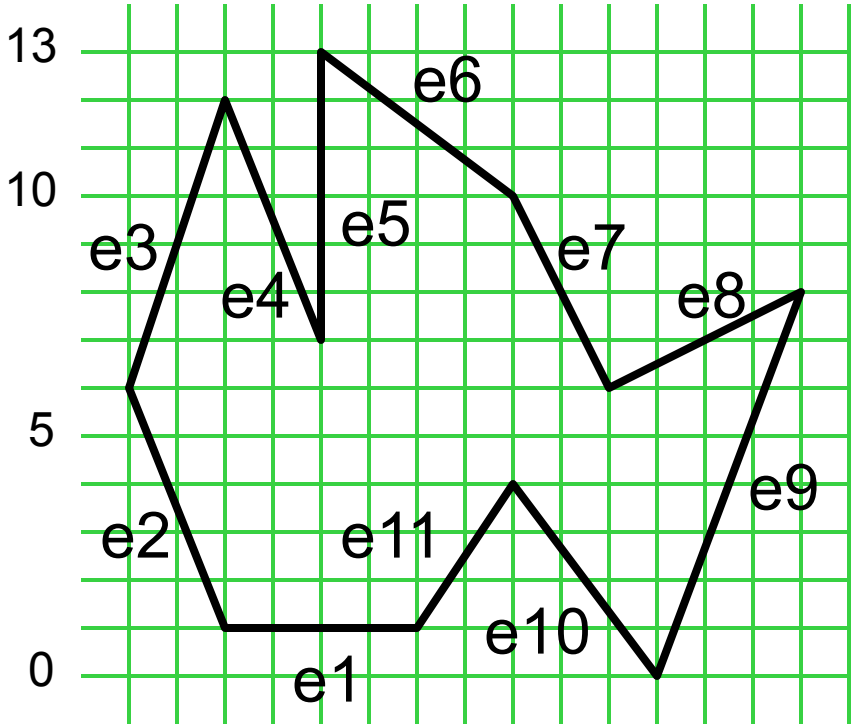
3

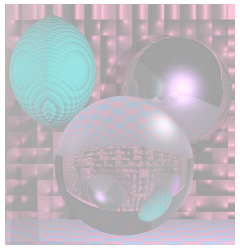
2

1 ➔ e2 ➔ e1 ➔ e11

0 ➔ e10 ➔ e9

Edge Table (ET) has a list of edges for each scan line.

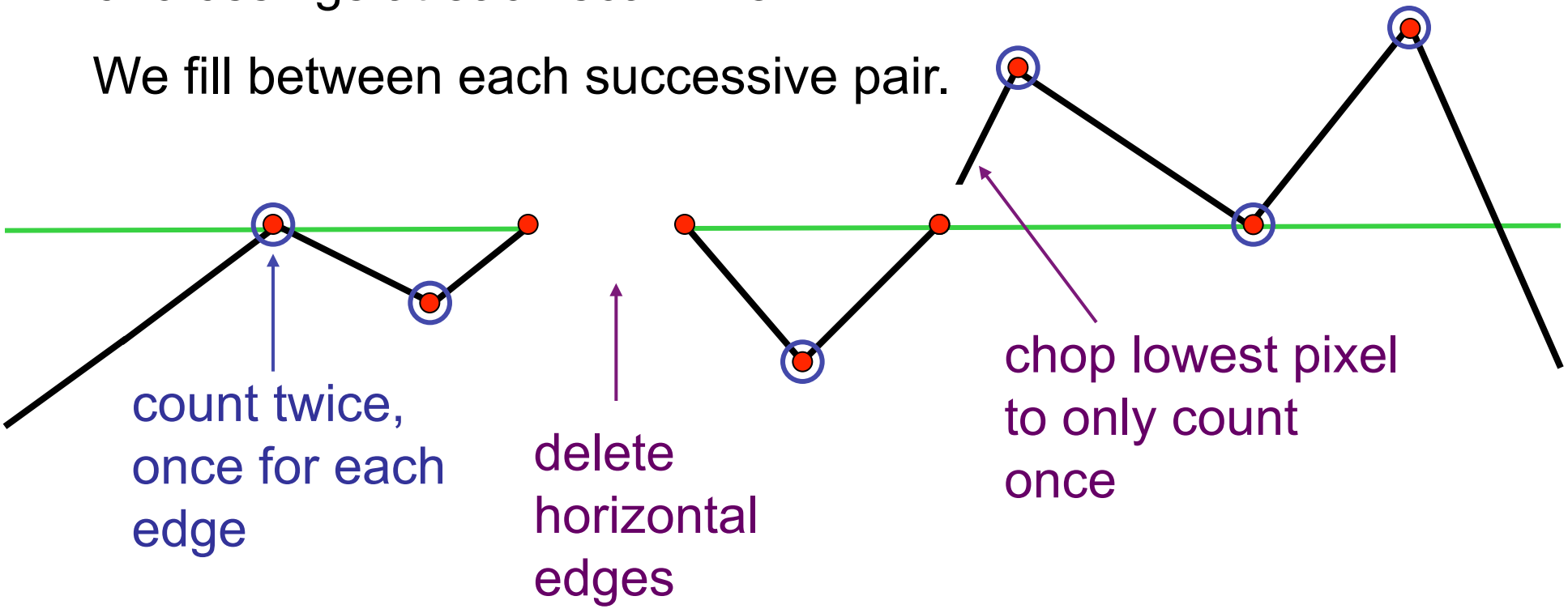




Preprocessing the edges

For a closed polygon, there should be an even number of crossings at each scan line.

We fill between each successive pair.



Polygon Data Structure

after preprocessing

Edge Table (ET) has a list of edges for each scan line.

13

12

11 → e6

10

9

8

7 → e3 → e4 → e5

6 → e7 → e8

5

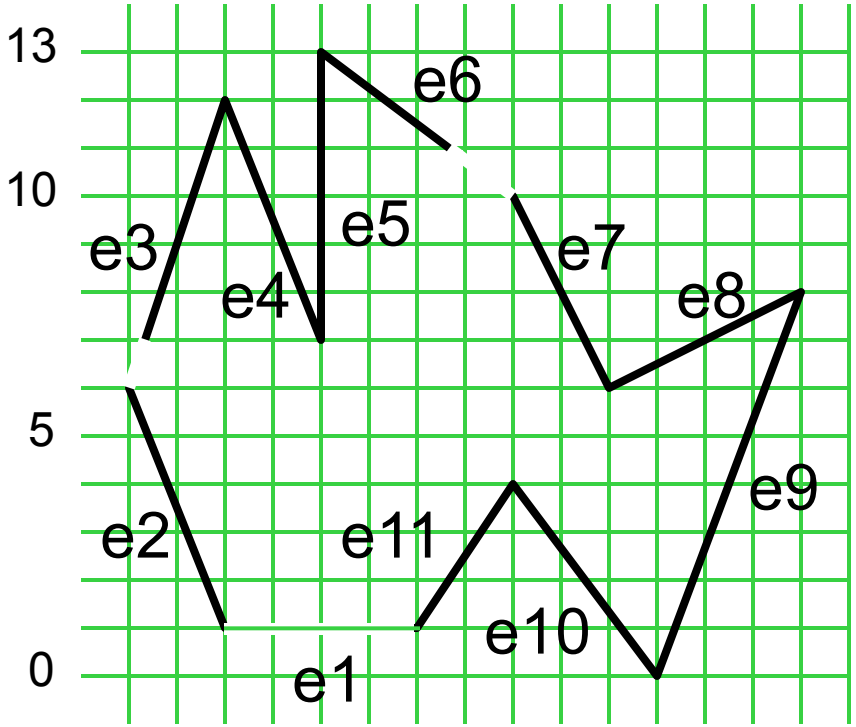
4

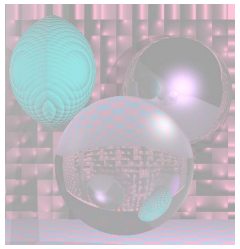
3

2

1 → e2 → e11

0 → e10 → e9





The Algorithm

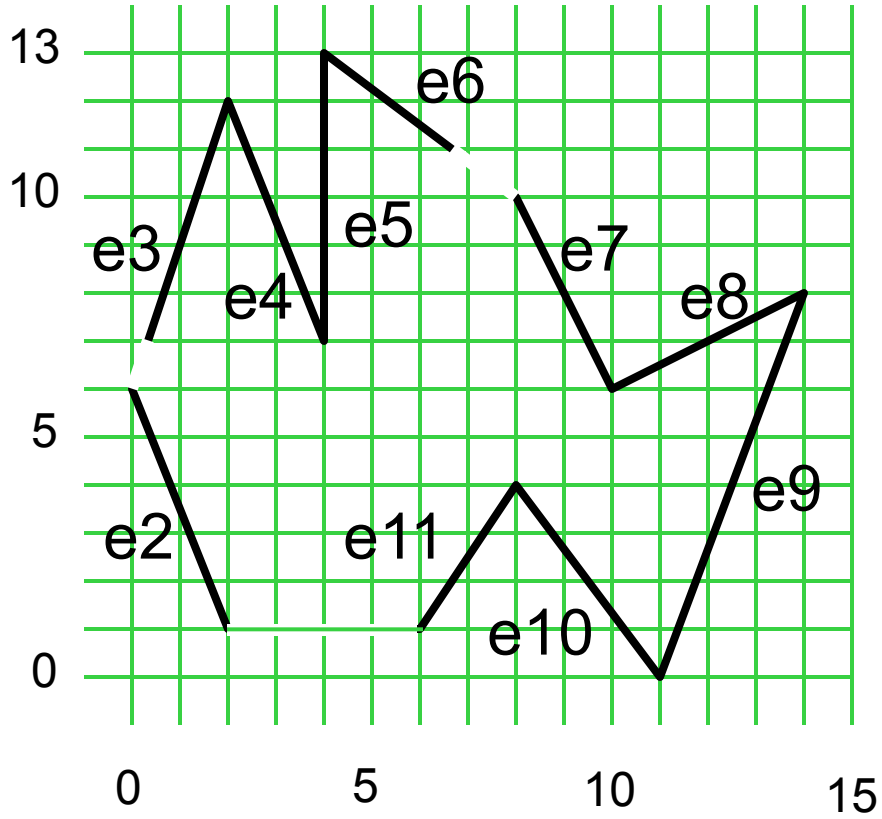
1. Start with smallest nonempty y value in ET.
2. Initialize SLB (Scan Line Bucket) to *nil*.
3. While current y ≤ top y value:
   a. Merge y bucket from ET into SLB; sort on xmin.
   b. Fill pixels between rounded pairs of x values in SLB.
   c. Remove edges from SLB whose ytop = current y.
   d. Increment xmin by 1/m for edges in SLB.
   e. Increment y by 1.

Running the Algorithm

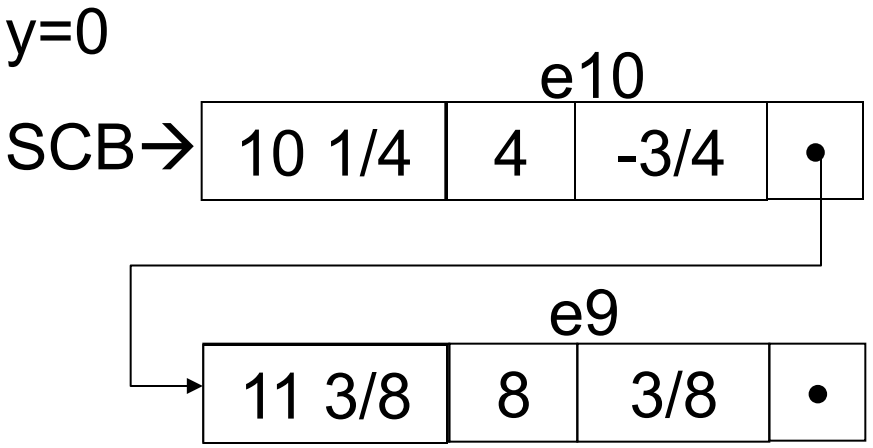
ET

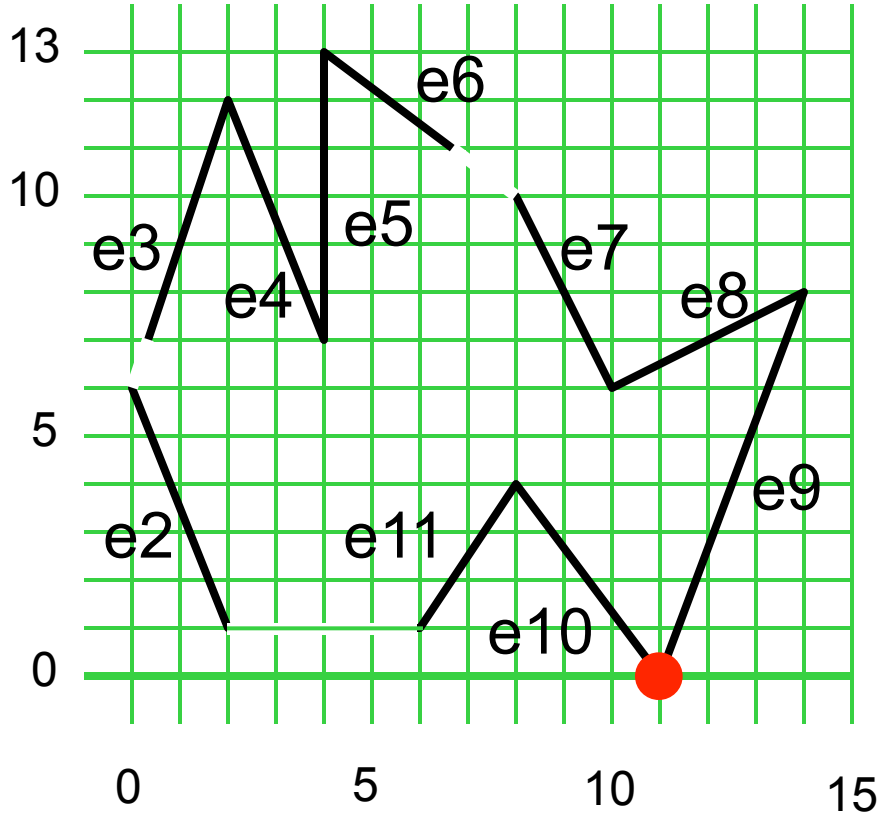
13
12
11 → e6
10
9
8
7 → e3 → e4 → e5
6 → e7 ve8
5
4
3
2
1 → e2 → e11
0 → e10 → e9

	xmin	ymax	1/m
e2	2	6	-2/5
e3	1/3	12	1/3
e4	4	12	-2/5
e5	4	13	0
e6	6 2/3	13	-4/3
e7	10	10	-1/2
e8	10	8	2
e9	11	8	3/8
e10	11	4	-3/4
e11	6	4	2/3

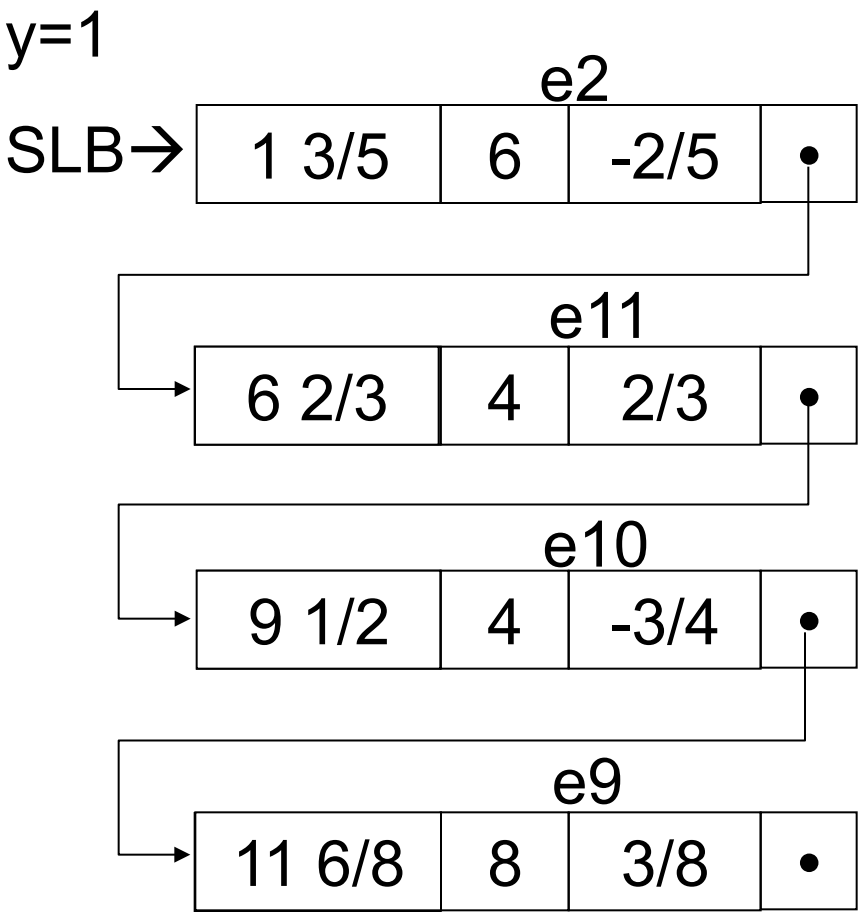


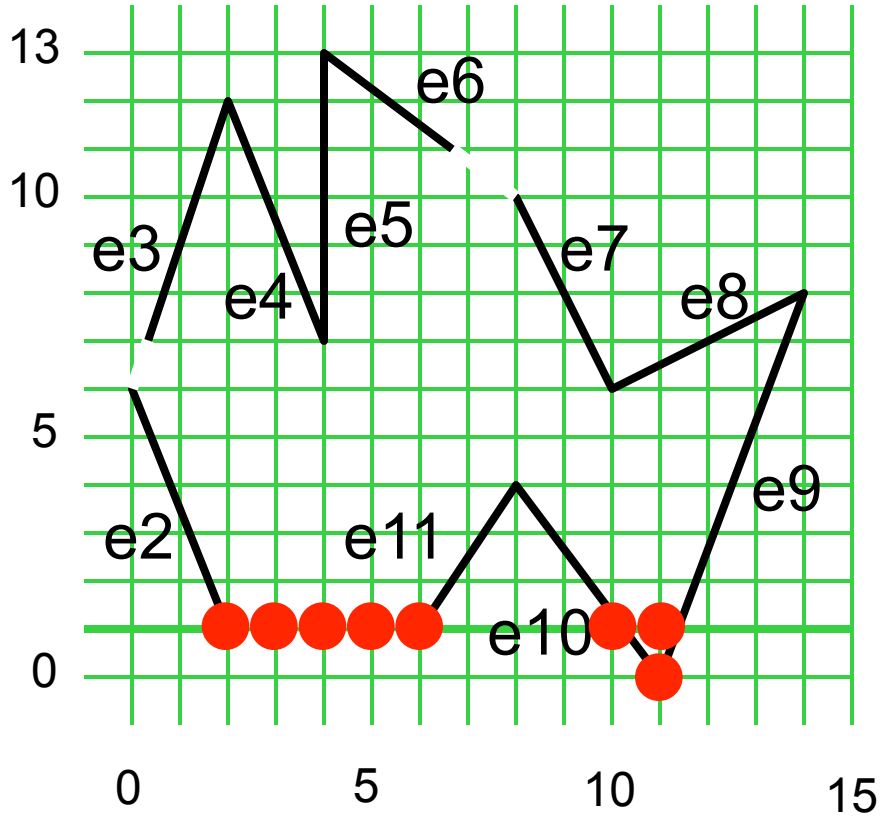
Running the Algorithm



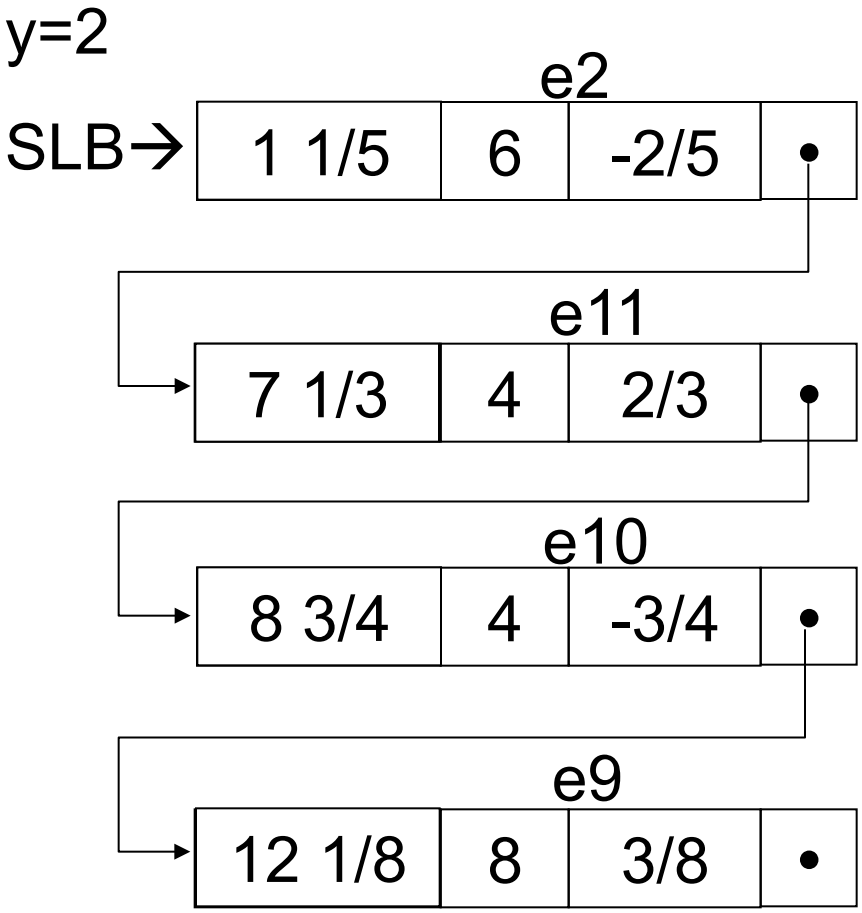


Running the Algorithm

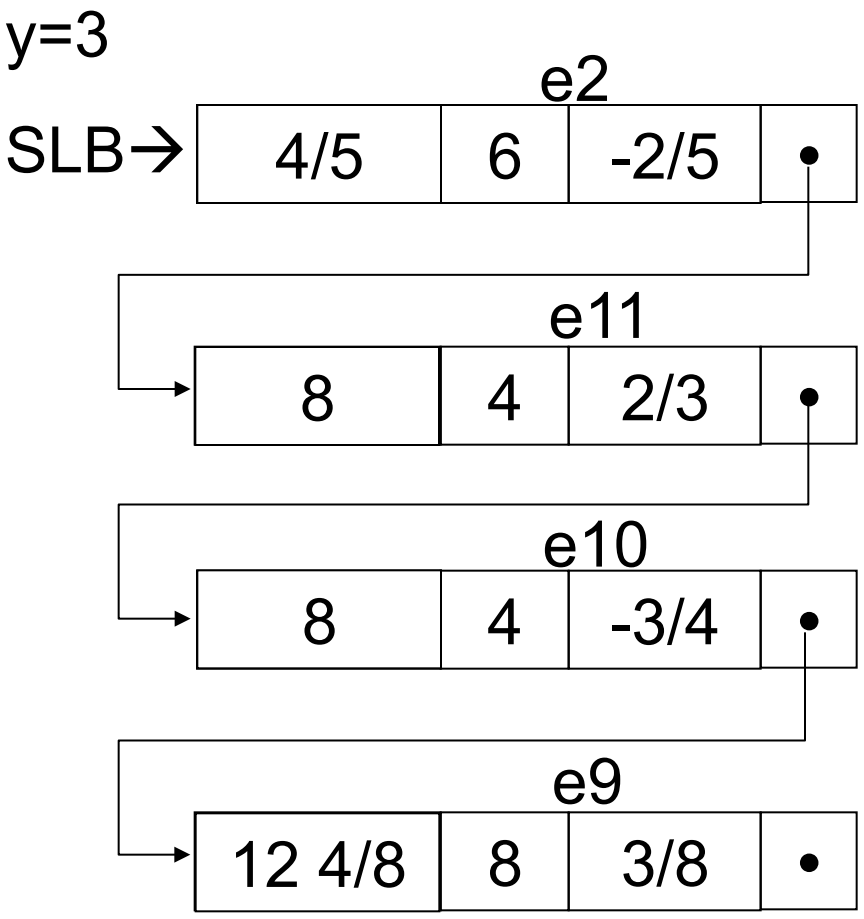


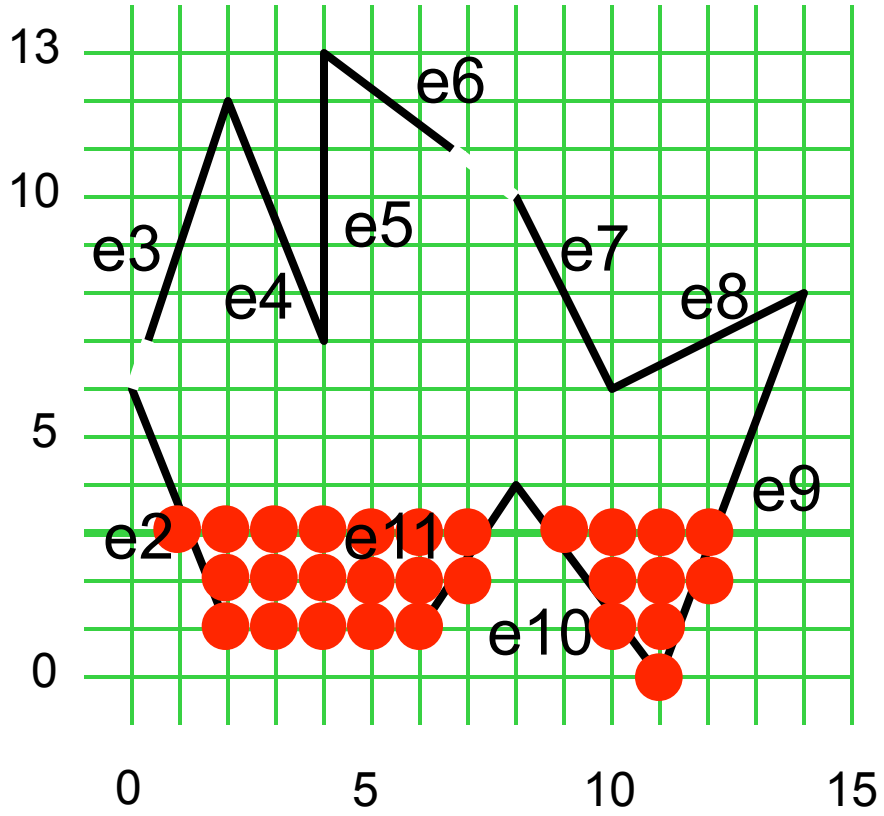


Running the Algorithm

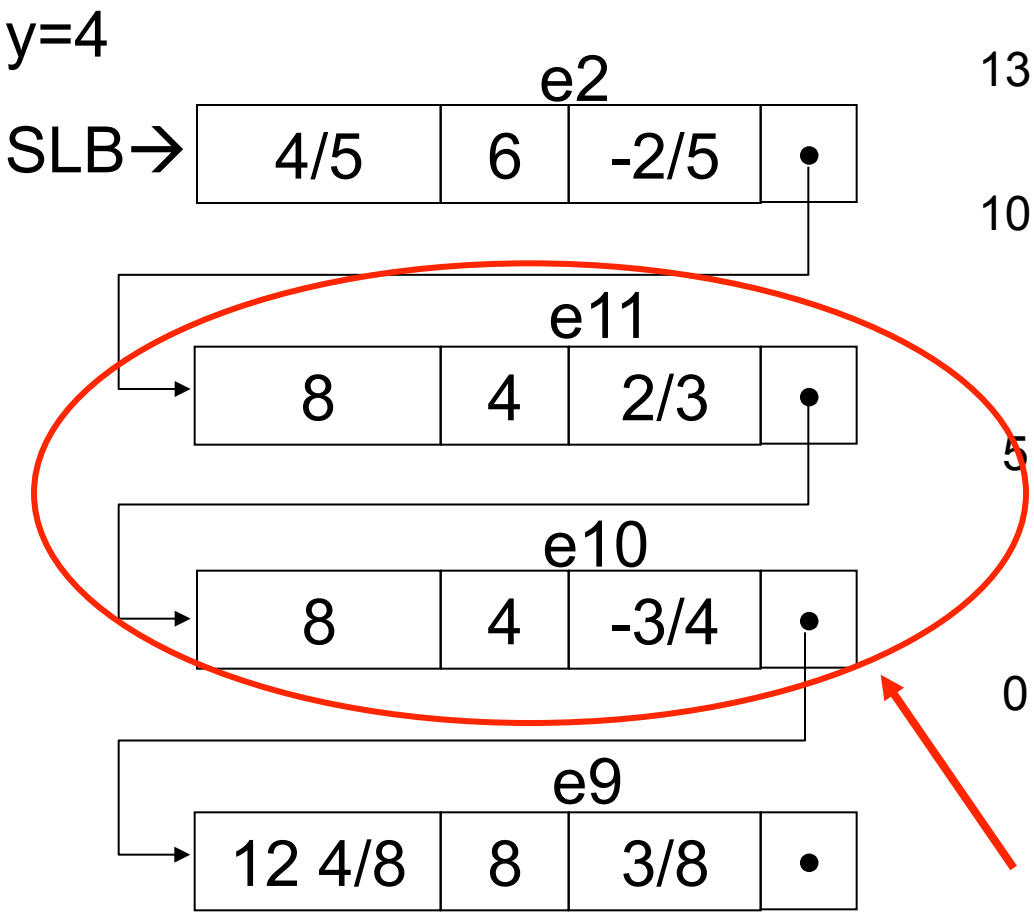


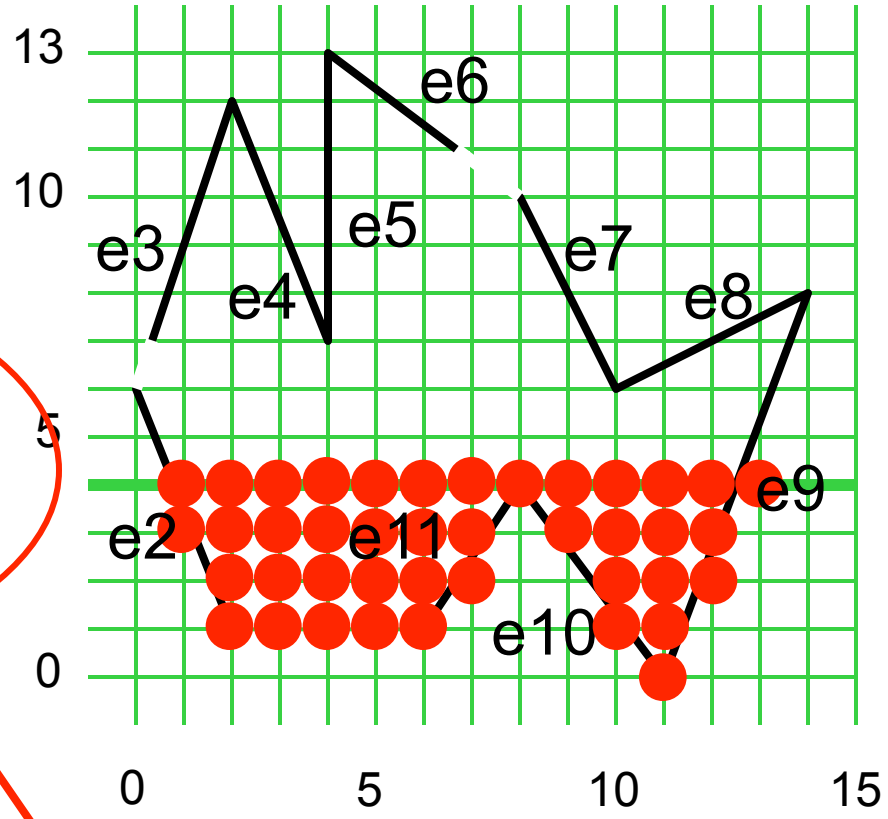
Running the Algorithm





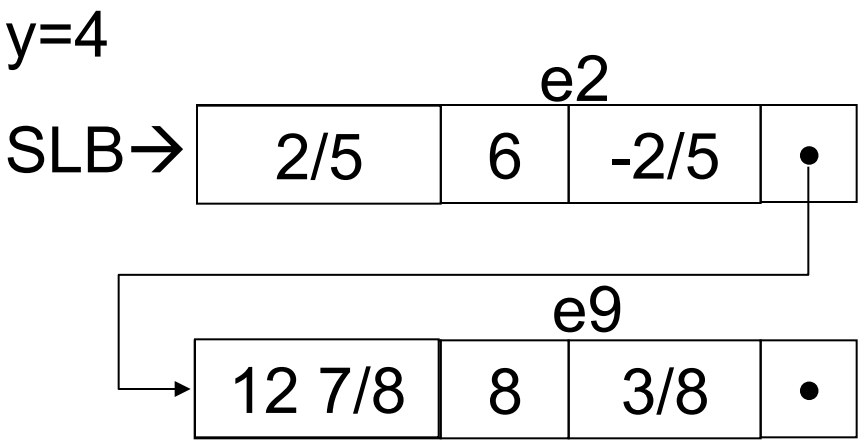
Running the Algorithm



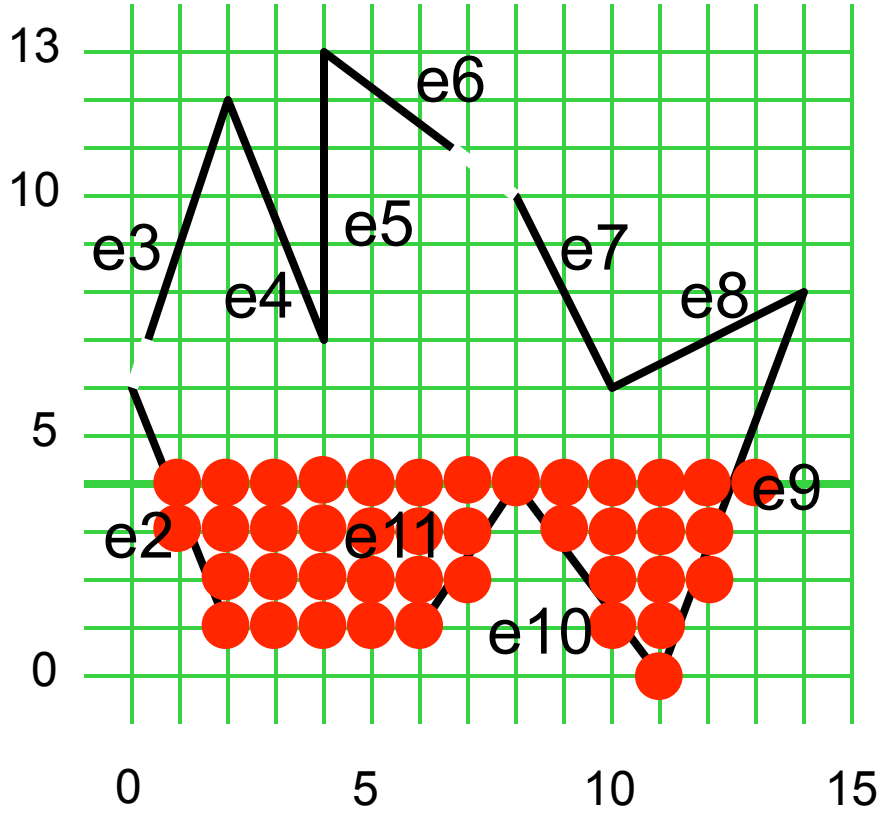


Remove these edges.

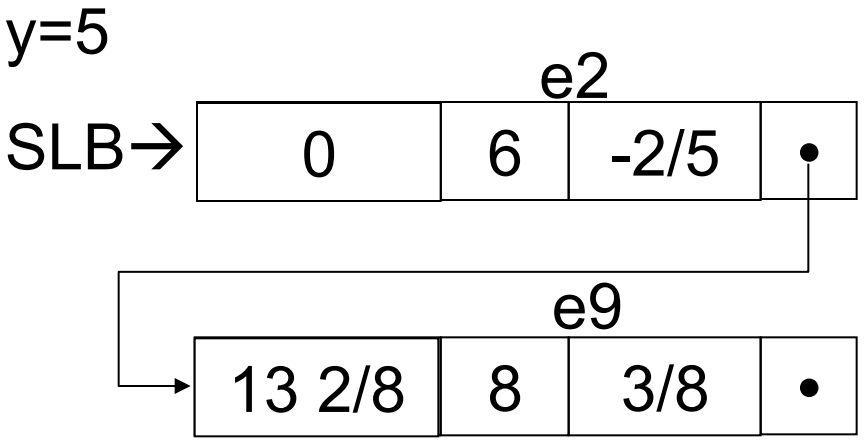
Running the Algorithm

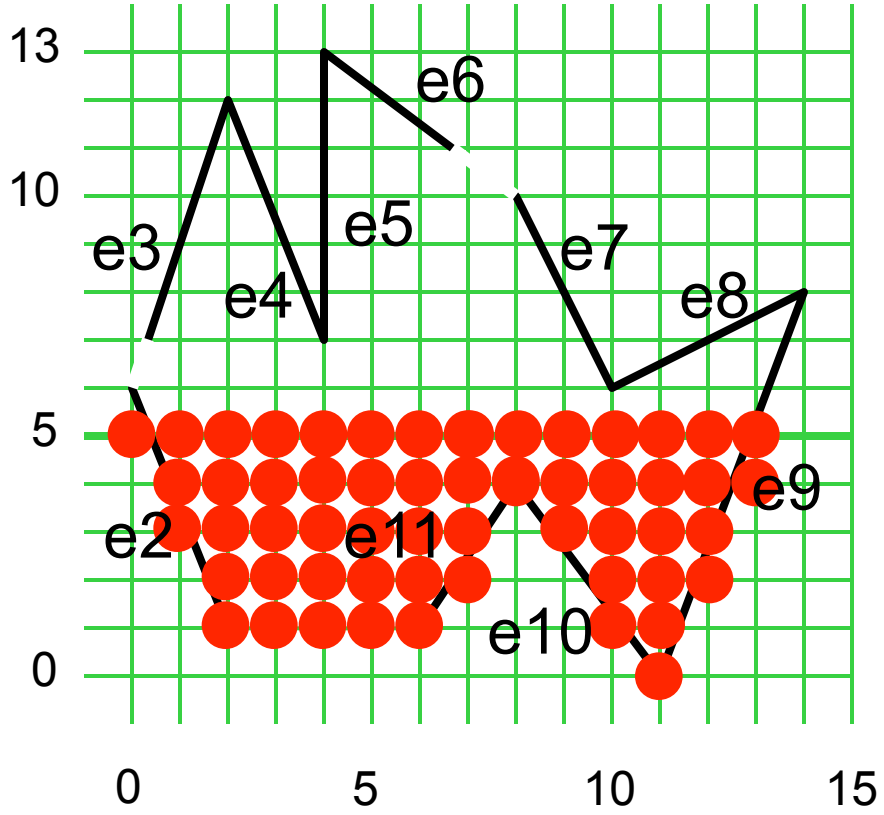


e11 and e10 are removed.

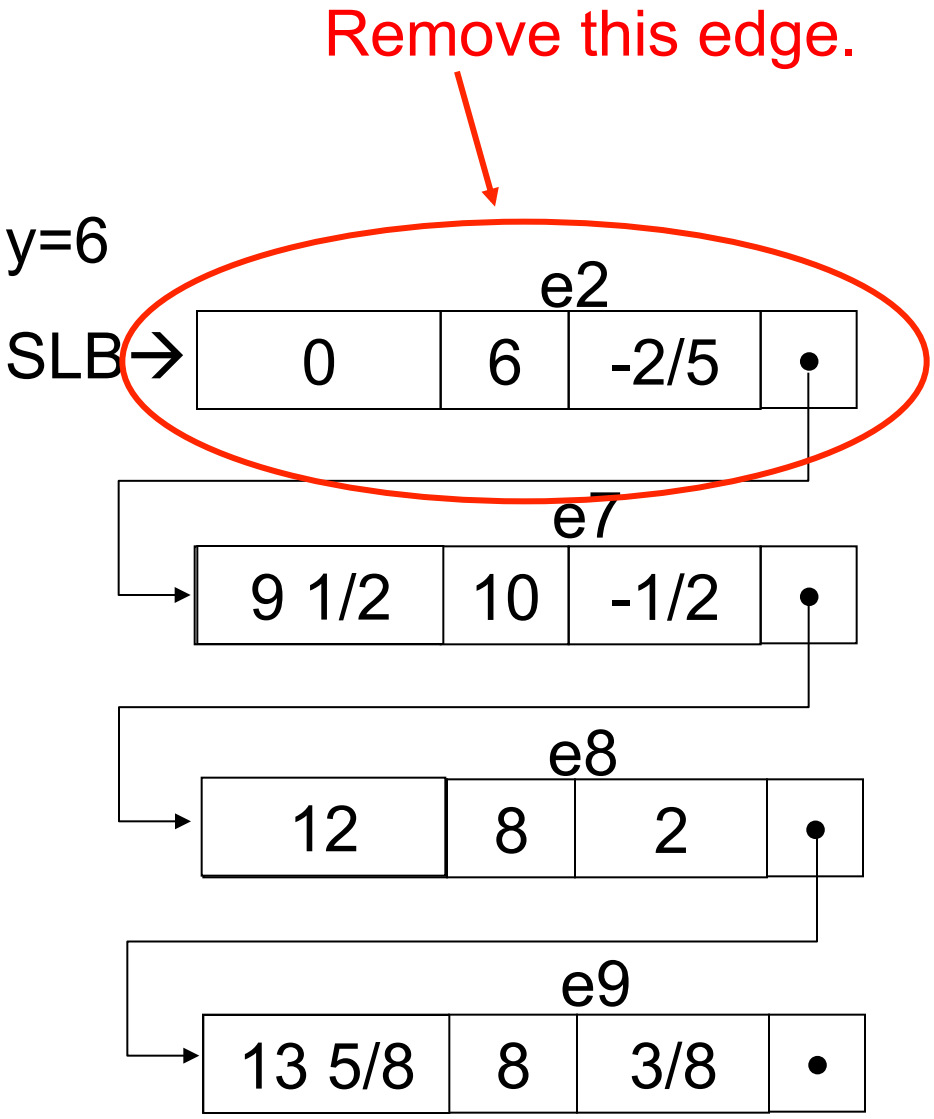


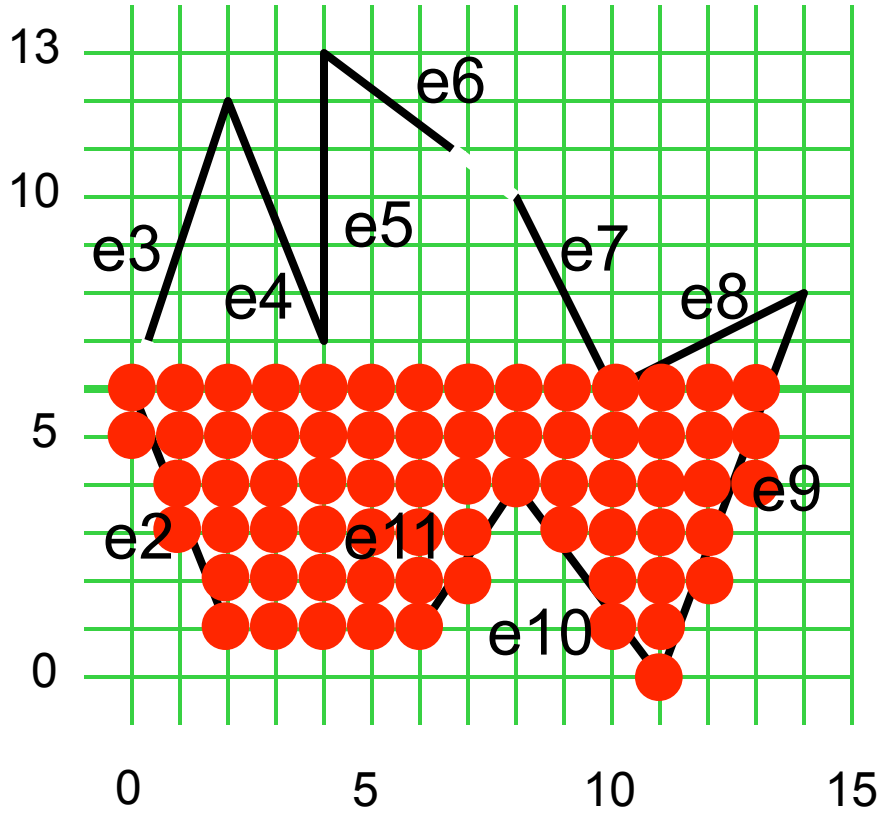
Running the Algorithm



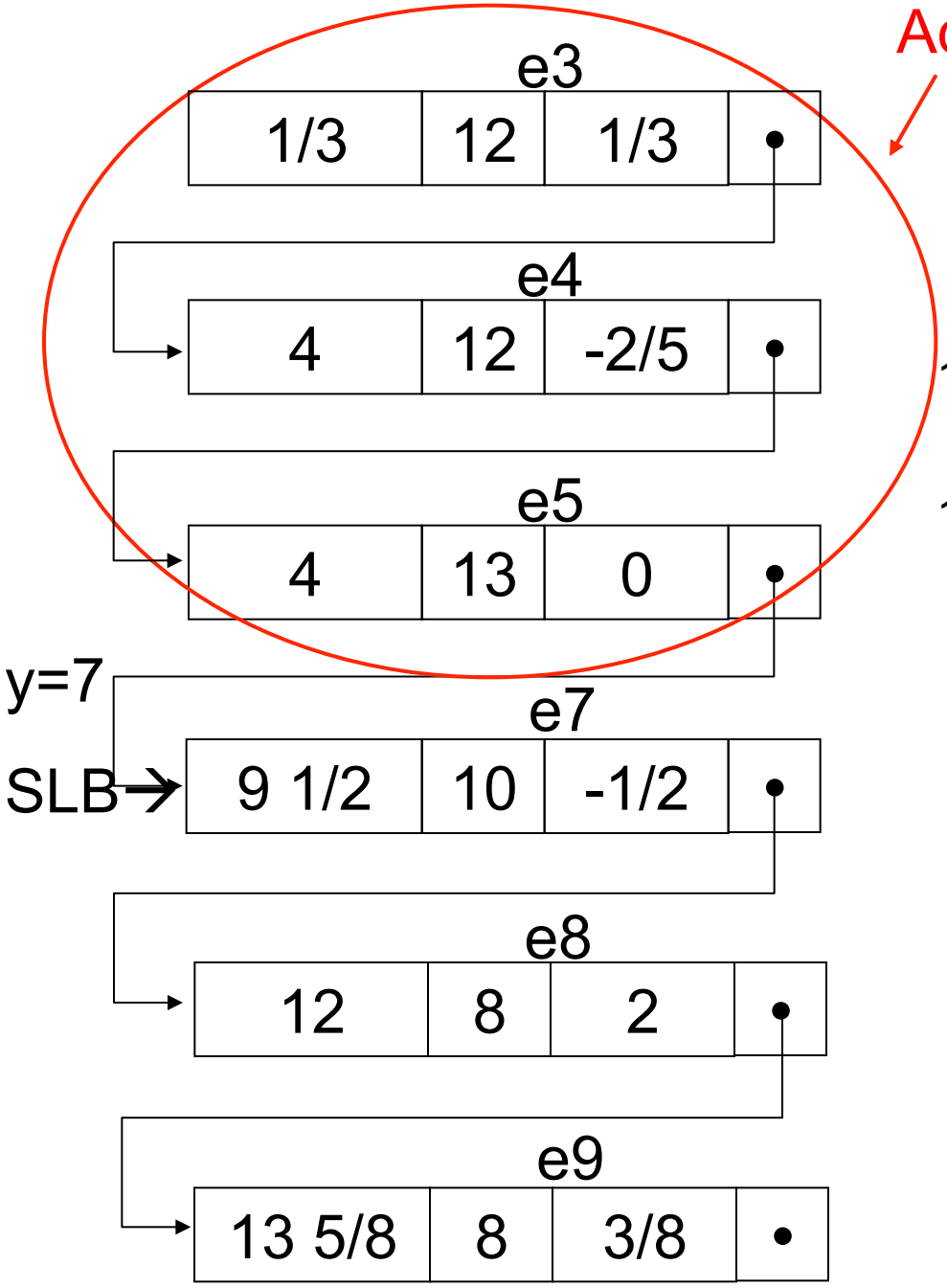


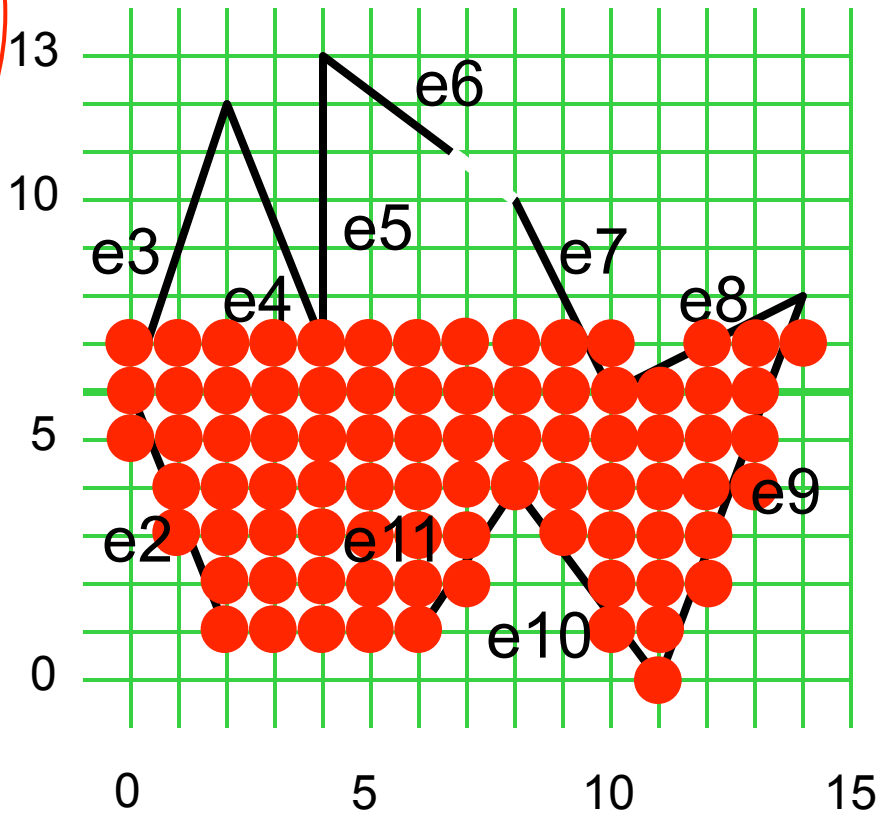
Running the Algorithm

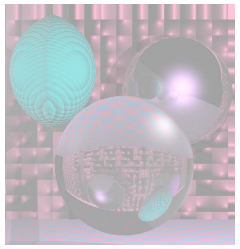




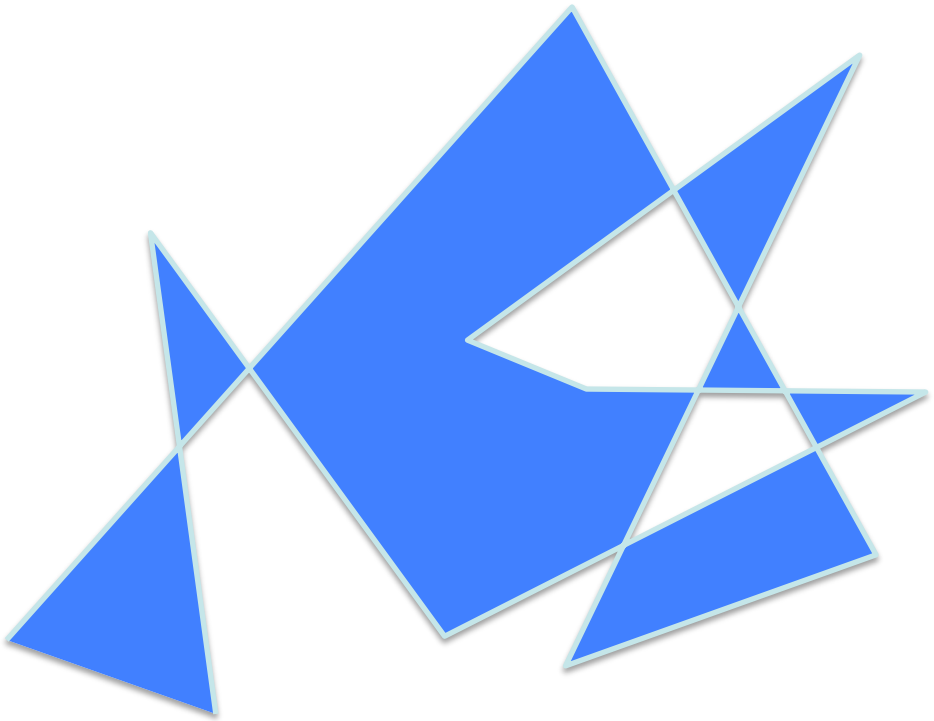
Running the Algorithm

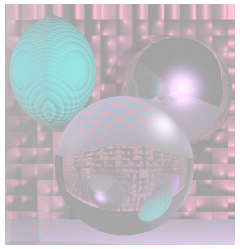






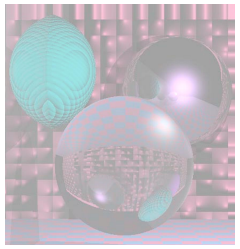
Non-Simple Polygons





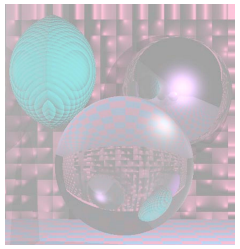
Even-odd Rule

- construct a ray r in an arbitrary direction from the test point p
- count number of intersections of r with the polygon. p is defined to be inside the poly if the intersection count is odd.



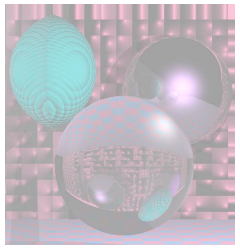
Reasoning

- each time an edge is crossed, either switch from inside to outside or outside to inside
- but since poly is closed, know that ray r must end up outside
- this is the method we just applied



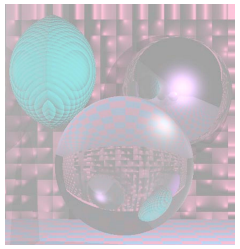
Nonzero Rule

- construct a ray r as before
- compute all intersections of r with the poly edges g_i, but this time keep track of whether the edge crossed from left to right (i.e. s_i on left side of r and e_i on right side of r) or right to left
- count +1 for left-to-right and –1 for right-to-left
- p is defined to be inside the poly if and only if (iff) final count is nonzero



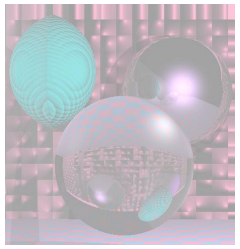
Reasoning

- consider sweeping a point q along the perimeter of the poly
- take the integral of the orientation angle of the line segment from p to q
- turns out that, for one complete sweep of the polygon, the integrated *winding angle* will be an integer multiple of *cover it)*



Intuition

- intuition: reasonable definition of inside-ness is to check if the poly "circled around" p in CCW direction different number times than in CW direction
- nonzero intersection test turns out to be a shortcut to compute (proof is a little subtle and we will not cover it)



The Results Can be Different