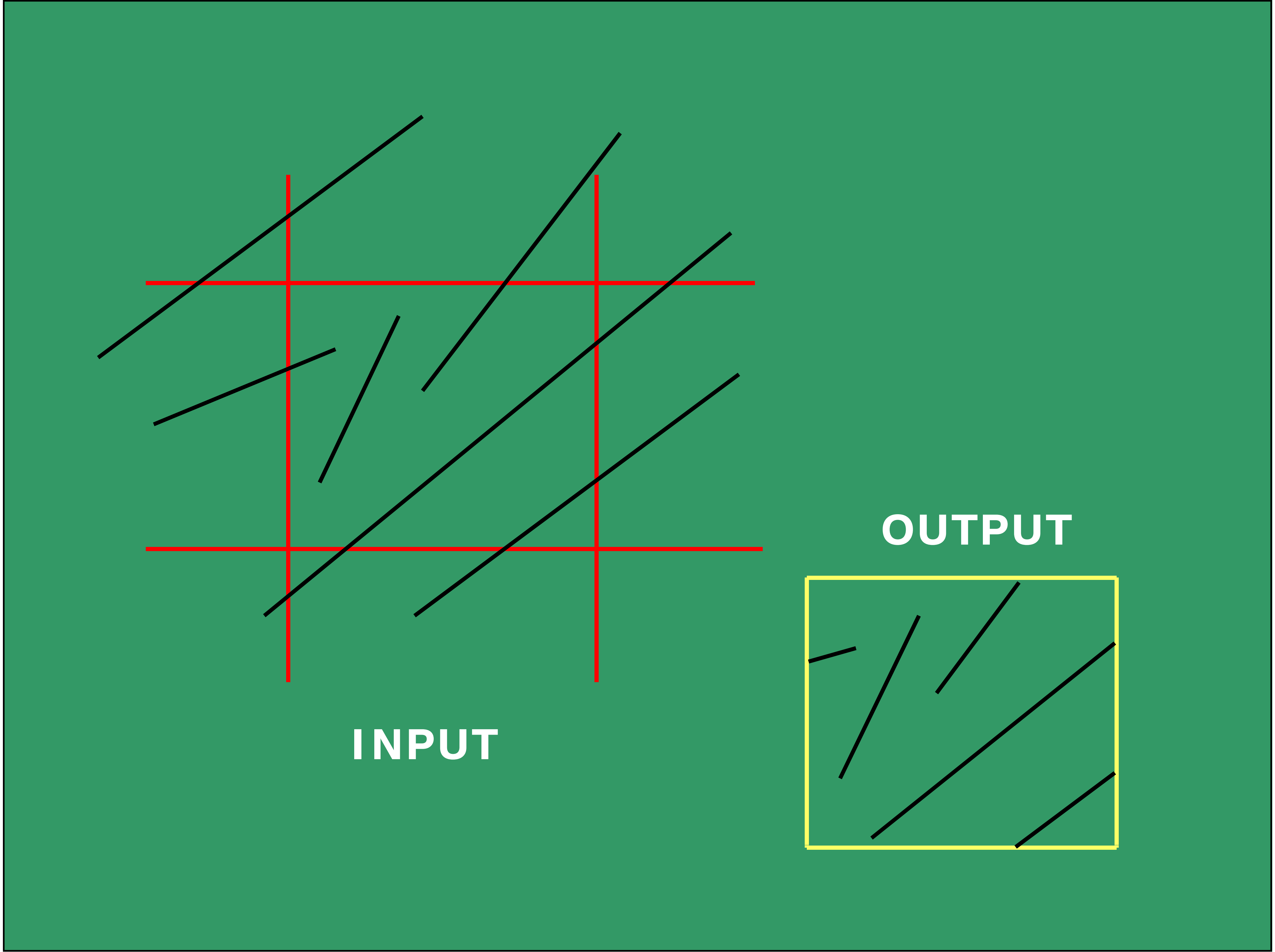
Clipping:

LINES

and

POLYGONS


OUTPUT
INPUT

Solving Simultaneous equations using parametric form of a line:

$$P(t) = (1 - t)P_0 + tP_1$$

$$where,\ P(0) = P_0;\ P(1) = P_1$$

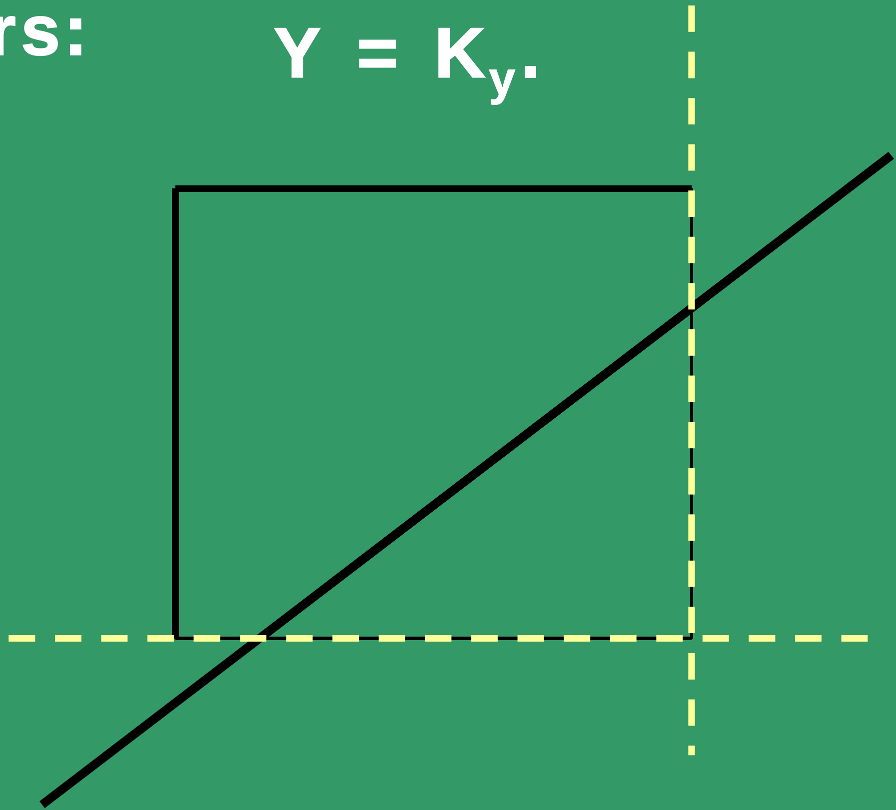
Vertical Line:
$X = K_x$;

Horizontal Line:
$Y = K_y$.

Solve with respective pairs:

$$t_{lx} = \frac{K_x - X_0}{X_1 - X_0}$$

$$t_{ly} = \frac{K_y - Y_0}{Y_1 - Y_0}$$



In general, solve for two sets of simultaneous equations for the parameters:

t_{edge} *and* t_{line}

Check if they fall within range [0 - 1].

i.e. Rewrite

$$P(t) = P_0 + t(P_1 - P_0)$$

and Solve:

$$t_1(P_1 - P_0) - t_2(P_1' - P_0') = P_0' - P_0$$

Cyrus-Beck

Line Clipping

CYRUS-BECK formulation

$$P(t) = P_0 + t(P_1 - P_0)$$

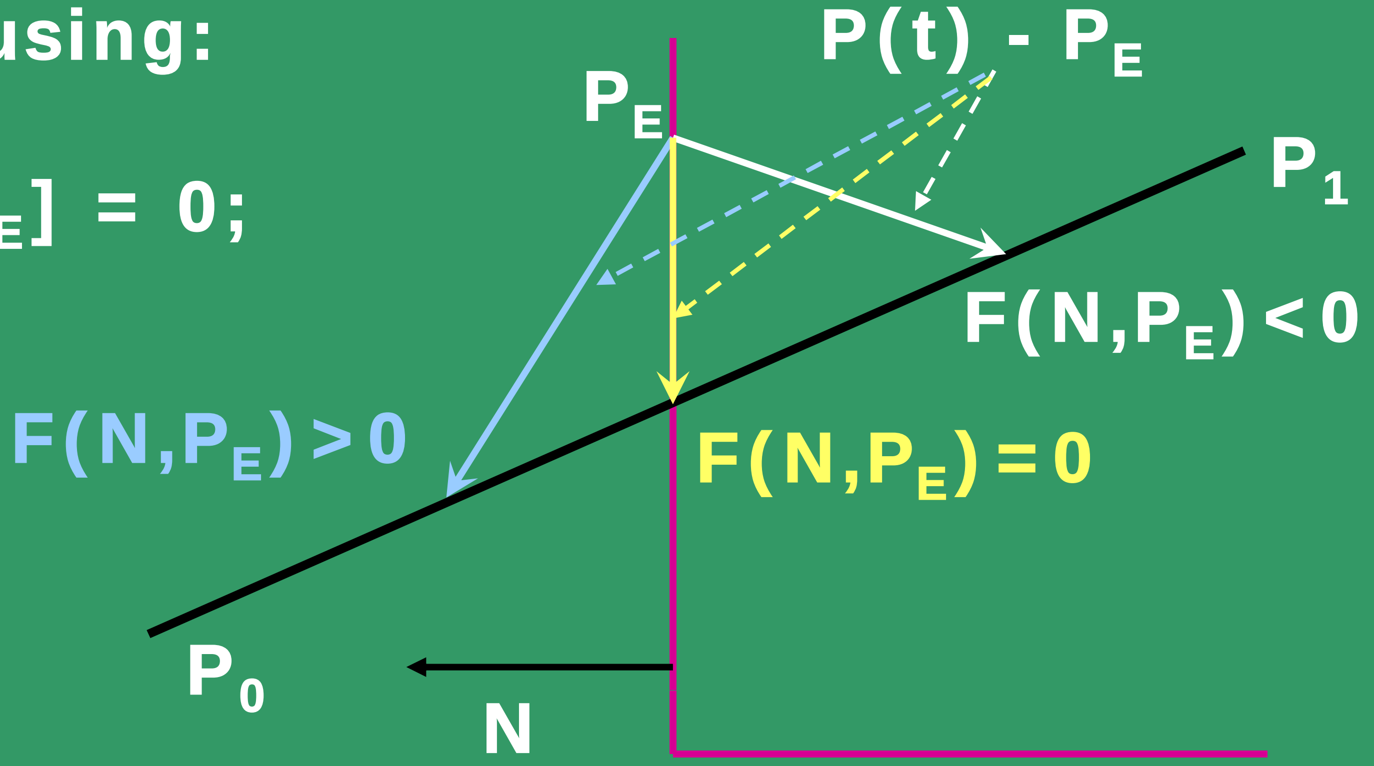
$$where,\ P(0) = P_0;\ P(1) = P_1$$

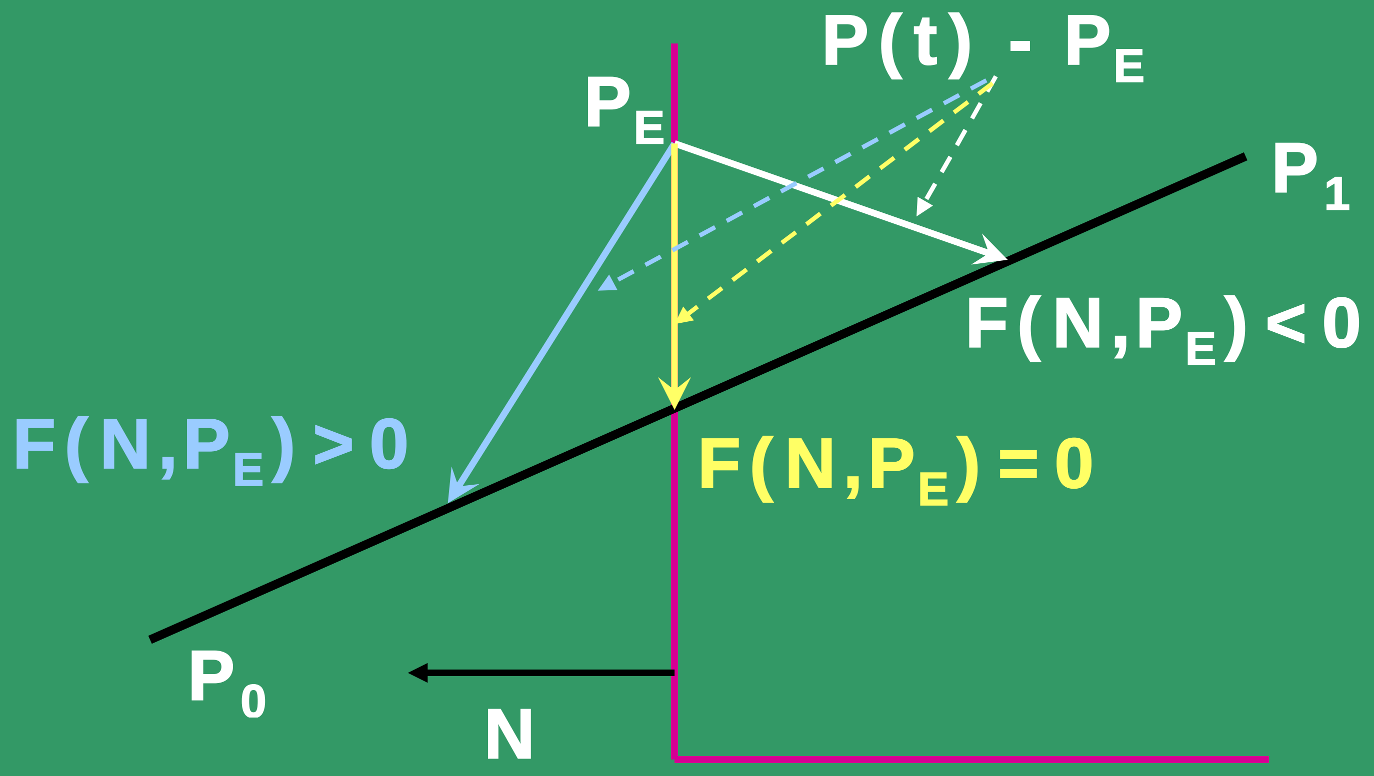
Define,

$F(N, P_E) = N.[P(t) - P_E]$

Solve for t using:

$N.[P(t) - P_E] = 0;$





$$N.[P_0 + (P_1 - P_0)t - P_E] = 0;$$

Substitute, $D = P_1 - P_0$;

To Obtain: $t = \frac{N.[P_0 - P_E]}{-N.D}$

Solve for t using:

$N.[P(t) - P_E] = 0;$

To ensure valid value of t, denominator must be non-zero.

Assuming, that D, N < > 0, check if:
N.D < > 0.   i.e. edge and line are not parallel.

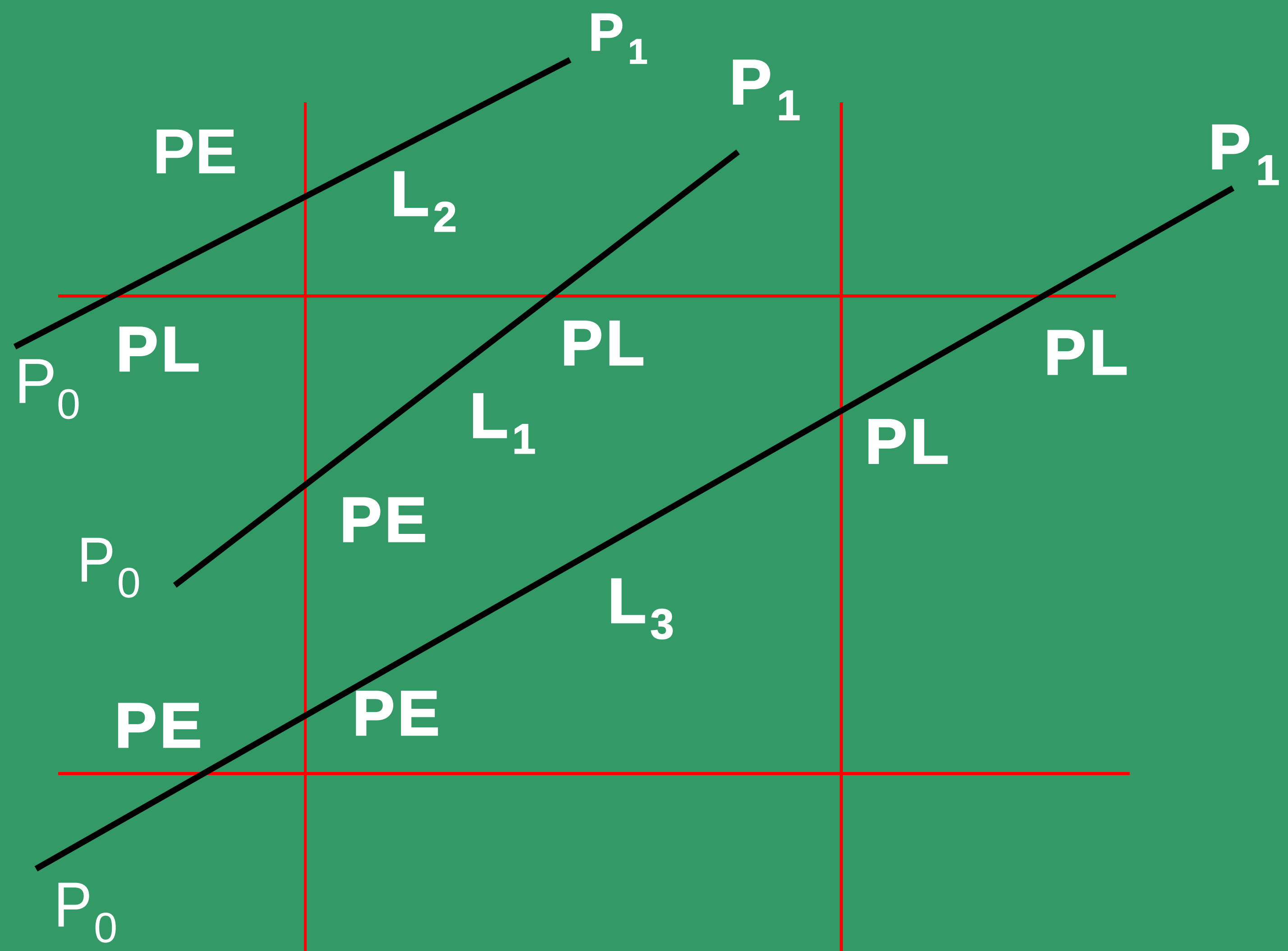
If they are parallel ?

Use the above expression of t to obtain all the four intersections:

- Select a point on each of the four edges of the clip rectangle.
- Obtain four values of t.
- Find valid intersections

How to implement the last step ?

Consider this example



Steps:

- If any value of t is outside the range [0 – 1] reject it.
- Else, sort with increasing values of t.

This solves L_1, but not lines L_2 and L_3.

Criteria to choose intersection points, PE or PL:

Move from point P_0 to P_1;

If you are entering edge's inside half-plane, then that intersection point is marked PE;

else, if you are leaving it is marked as PL.

Check the angle of D and N vectors, for each edge separately.

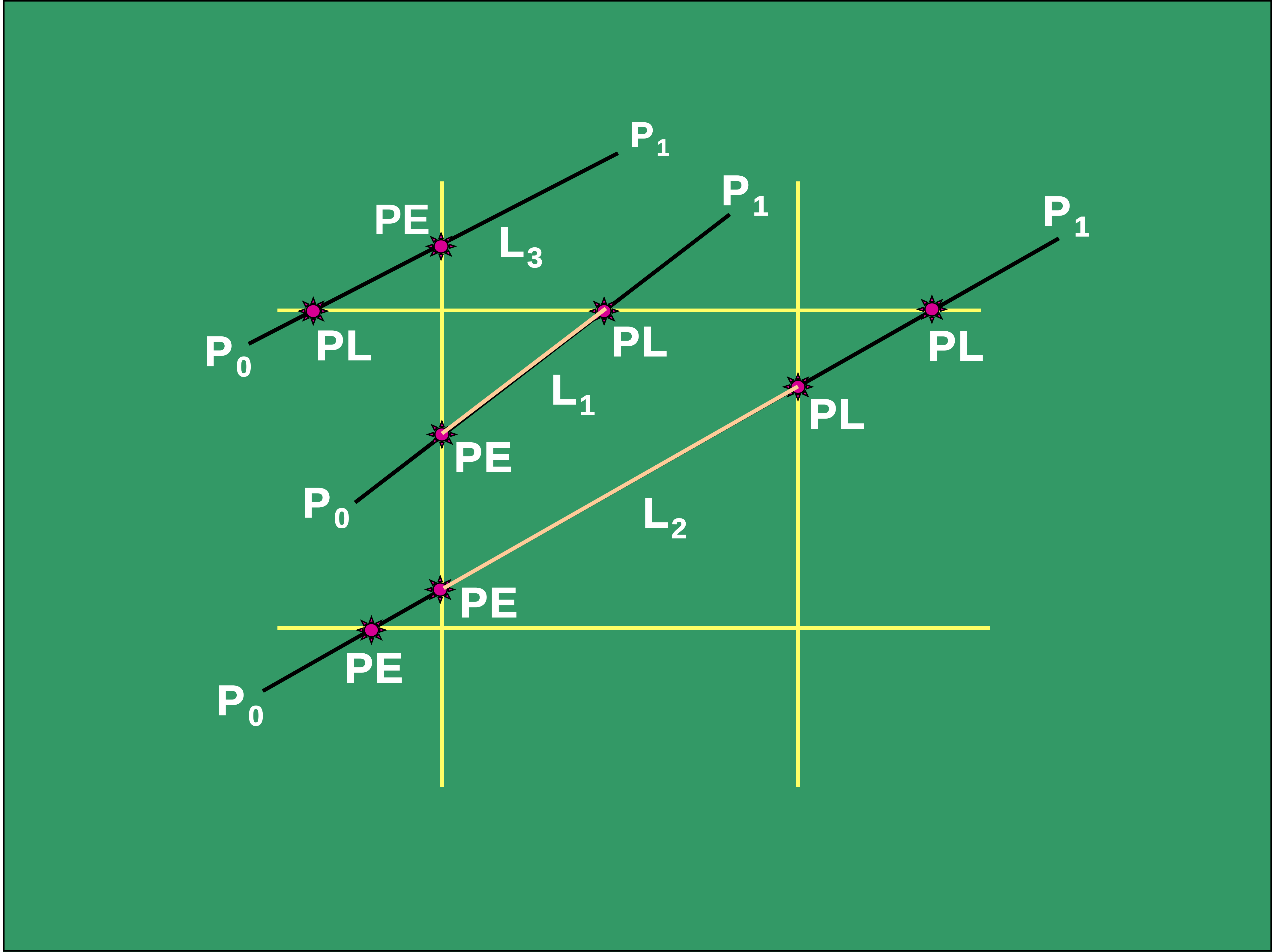
If angle between D and N is:

>90 deg., N.D < 0, mark the point as PE, store $t_E(i) = t$

<90 deg., N.D > 0, mark the point as PL, store $t_L(i) = t$

Find the maximum value of t_E, and minimum value of t_L for a line.

If $t_E < t_L$ choose pair of parameters as valid intersections on the line. Else NULL.



P1
P1
P1
PE
L3
PL
PL
PL
P0
L1
PL
PE
P0
L2
PE
PE
P0

Calculations for parametric line Clipping

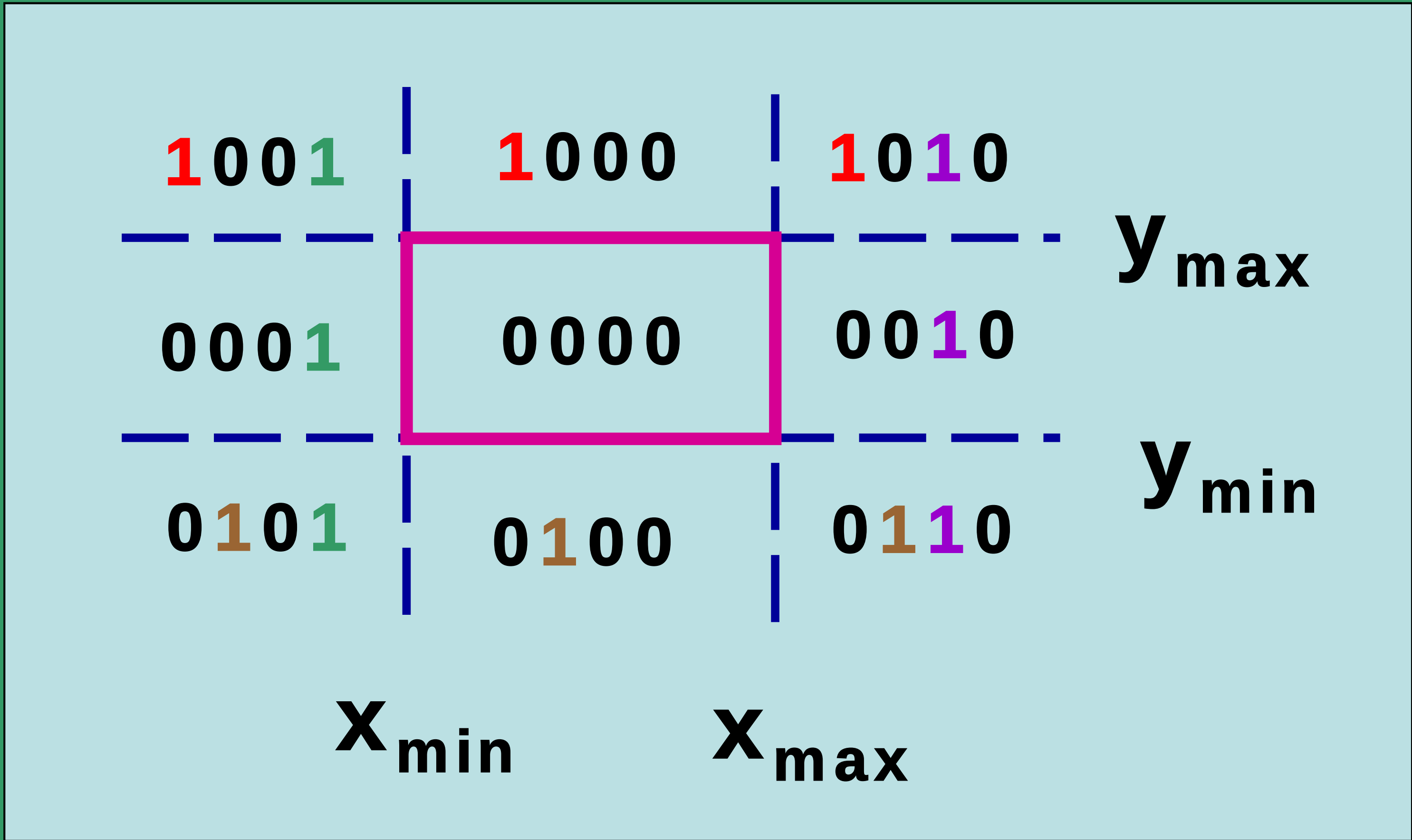
Clip Edge	Normal N	P_E§	$P_0 - P_E$	$t = \frac{N.[P_0 - P_E]}{-N.D}$
Left: $X = X_{min}$	(-1, 0)	(X_{min}, Y)	$(X_0 - X_{min}, Y_0 - Y)$	$\frac{-(X_0 - X_{min})}{(X_1 - X_0)}$
Right: $X = X_{max}$	(1, 0)	(X_{max}, Y)	$(X_0 - X_{max}, Y_0 - Y)$	$\frac{(X_0 - X_{max})}{-(X_1 - X_0)}$
Bottom: $Y = Y_{min}$	(0, -1)	(X, Y_{min})	$(X_0 - X, Y_0 - Y_{min})$	$\frac{-(Y_0 - Y_{min})}{(Y_1 - Y_0)}$
Top: $Y = Y_{max}$	(0, 1)	(X, Y_{max})	$(X_0 - X, Y_0 - Y_{max})$	$\frac{(Y_0 - Y_{max})}{-(Y_1 - Y_0)}$

§ - Exact coordinates for P_E is irrelevant.

Cohen-Sutherland

Line Clipping

Region Outcodes:



Bit Number	1	0
FIRST (MSB)	Above Top edge $Y > Y_{max}$	Below Top edge $Y < Y_{max}$
SECOND	Below Bottom edge $Y < Y_{min}$	Above Bottom edge $Y > Y_{min}$
THIRD	Right of Right edge $X > X_{max}$	Left of Right edge $X < X_{max}$
FOURTH (LSB)	Left of Left edge $X < X_{min}$	Right of Left edge $X > X_{min}$

First Step: Determine the bit values of the two end-points of the line to be clipped.

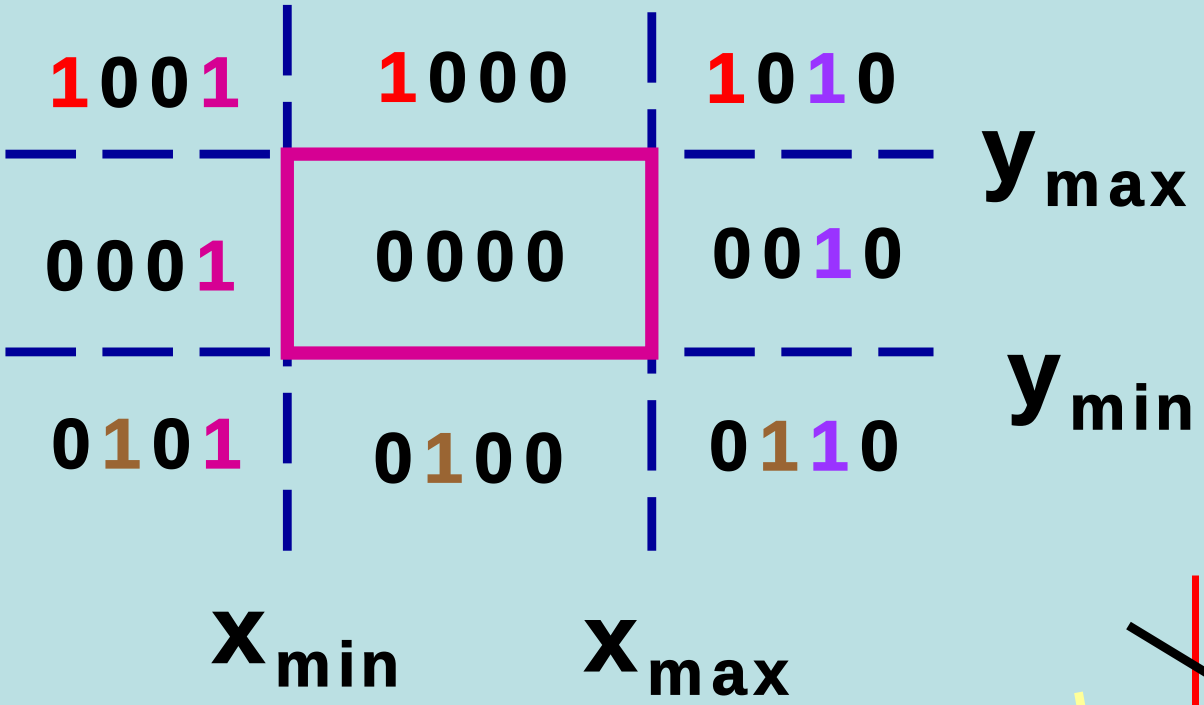
To determine the bit value of any point, use:

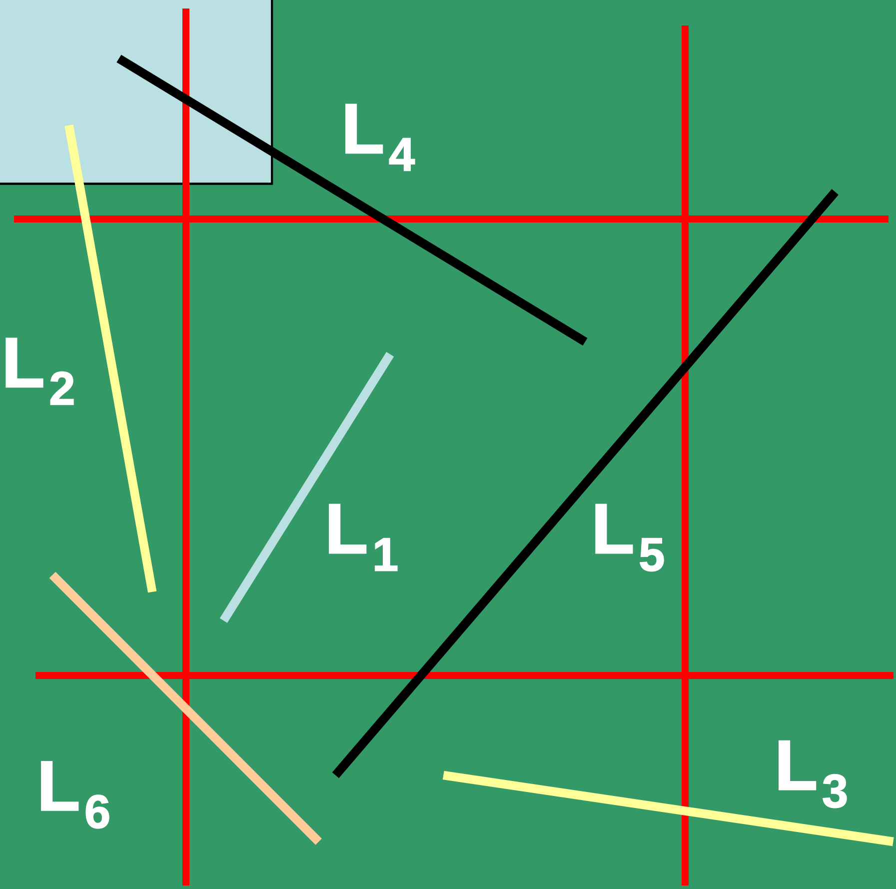
$$b_1 = sgn(Y_{max} - Y); \quad b_2 = sgn(Y - Y_{min});$$
$$b_3 = sgn(X_{max} - X); \quad b_4 = sgn(X - X_{min});$$

Use these end-point codes to locate the line. Various possibilities:

- **If both endpoint codes are [0000], the line lies completely inside the box, no need to clip. This is the simplest case (e.g. L_1).**

- **Any line has 1 in the same bit positions of both the endpoints, it is guaranteed to lie outside the box completely (e.g. L_2 and L_3).**





- **Neither completely reject nor inside the box:**

Lines: L_4 and L_5, - needs more processing.

- **What about Line L_6 ?**

Processing of lines, neither Completely IN or OUT; e.g. Lines: L_4, L_5 and L_6.

Basic idea:

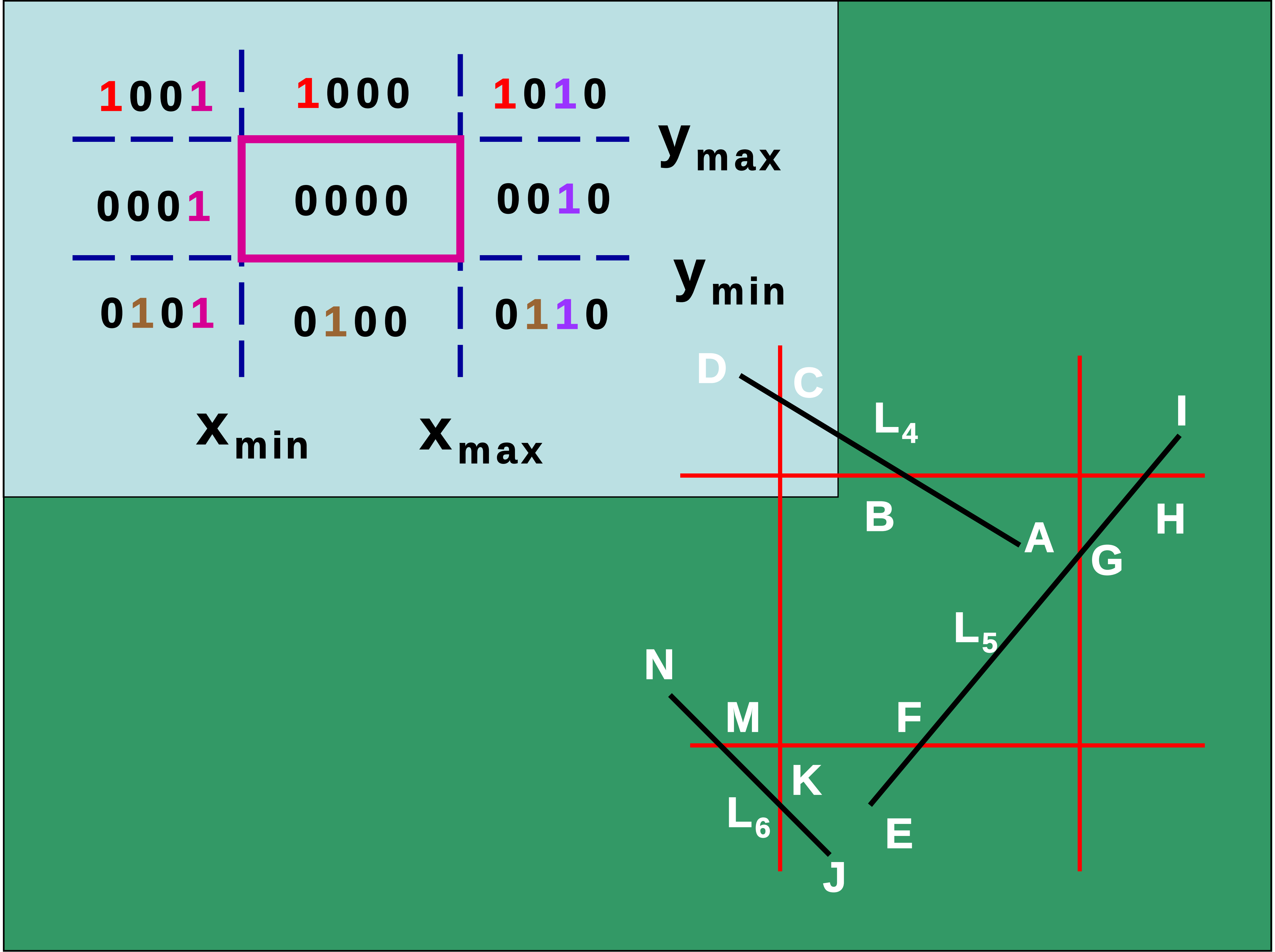
Clip parts of the line in any order (consider from top or bottom).

Algorithm Steps:

- Compute outcodes of both endpoints to check for trivial acceptance or rejection (AND logic).

- If not so, obtain an endpoint that lies outside the box (at least one will ?).

- Using the outcode, obtain the edge that is crossed first.


1001
1000
1010
0001
0000
0010
0101
0100
0110
y_{max}
y_{min}
x_{min}
x_{max}
D
C
L_4
I
B
A
H
G
L_5
N
M
F
K
L_6
E
J

Coordinates for intersection, for clipping w.r.t edge:

Inputs: **Endpoint coordinates:**
(X_0, Y_0) and (X_1, Y_1)

OUTPUT:
Edge for clipping (obtained using outcode of current endpoint).

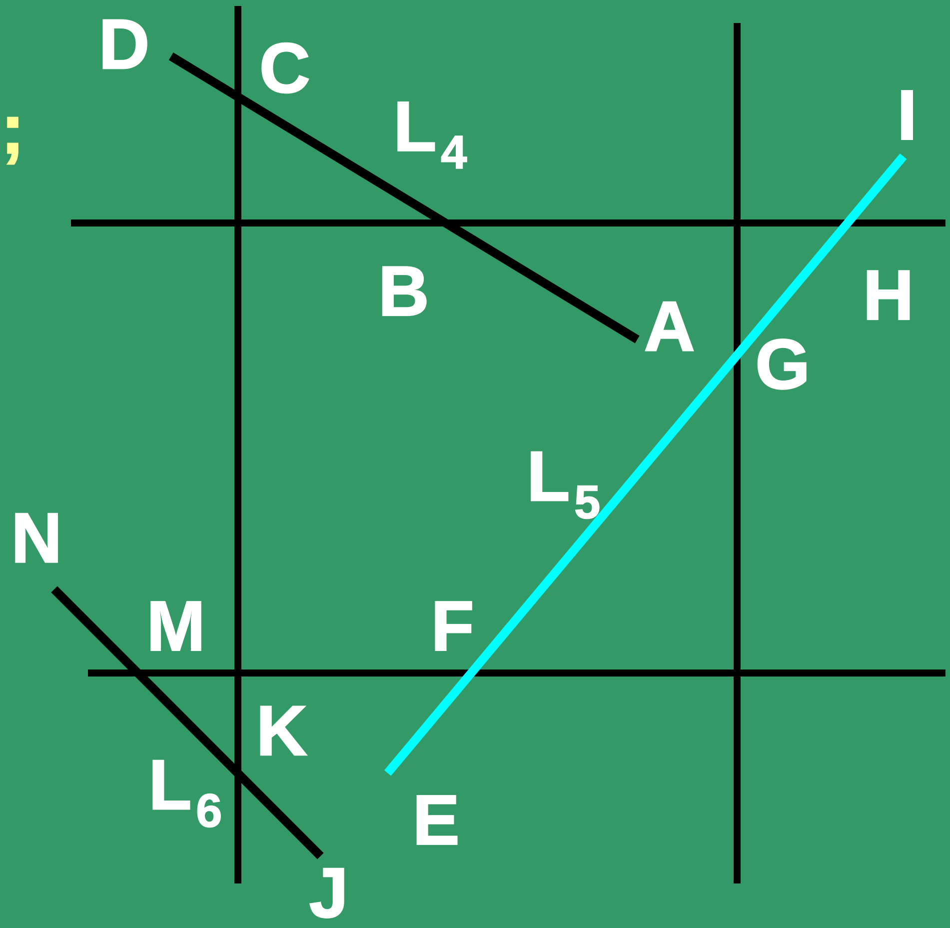
Obtain corresponding intersection points

- CLIP (replace the endpoint by the intersection point) w.r.t. the edge.
- Compute the outcode for the updated endpoint and repeat the iteration, till it is 0000.
- Repeat the above steps, if the other endpoint is also outside the area.

e.g. Take Line L_5 (endpoints - E and I):
E has outcode 0100 (to be clipped w.r.t. bottom edge);

So EI is clipped to FI;
Outcode of F is 0000;
But outcode of I is 1010;
Clip (w.r.t. top edge)
to get FH.
Outcode of H is 0010;
Clip (w.r.t. right edge)
to get FG;

Since outcode of G
is 0000, display the
final result as FG.



Formulas for clipping w.r.t. edge, in cases of:

Top Edge :

$$X = X_0 + (X_1 - X_0) * \frac{(Y_{max} - Y_0)}{(Y_1 - Y_0)}$$

Bottom Edge:

$$X = X_0 + (X_1 - X_0) * \frac{(Y_{min} - Y_0)}{(Y_1 - Y_0)}$$

Right Edge:

$$Y = Y_0 + (Y_1 - Y_0) * \frac{(X_{max} - X_0)}{(X_1 - X_0)}$$

Left edge:

$$Y = Y_0 + (Y_1 - Y_0) * \frac{(X_{min} - X_0)}{(X_1 - X_0)}$$

Let’s compare with Cyrus-Beck formulation →

Calculations for parametric line Clipping

Clip Edge	Normal N	P_E §	P_0 - P_E	$t = \frac{N.[P_0 - P_E]}{-N.D}$
Left: $X = X_{min}$	(-1, 0)	(X_{min}, Y)	$(X_0 - X_{min}, Y_0 - Y)$	$\frac{-(X_0 - X_{min})}{(X_1 - X_0)}$
Right: $X = X_{max}$	(1, 0)	(X_{max}, Y)	$(X_0 - X_{max}, Y_0 - Y)$	$\frac{(X_0 - X_{max})}{-(X_1 - X_0)}$
Bottom: $Y = Y_{min}$	(0, -1)	(X, Y_{min})	$(X_0 - X, Y_0 - Y_{min})$	$\frac{-(Y_0 - Y_{min})}{(Y_1 - Y_0)}$
Top: $Y = Y_{max}$	(0, 1)	(X, Y_{max})	$(X_0 - X, Y_0 - Y_{max})$	$\frac{(Y_0 - Y_{max})}{-(Y_1 - Y_0)}$

§ - Exact coordinates for P_E is irrelevant.

Liang-Barsky

Line Clipping

Consider parametric equation of a line segment:

$$X = X_1 + u\Delta X; Y = Y_1 + u\Delta Y, 0 \le u \le 1.$$

where,

$$\Delta X = X_2 - X_1; \ \Delta Y = Y_2 - Y_1$$

A point is considered to be within a rectangle, iff

$$XW_{\min} \le X_1 + u\Delta X \le XW_{\max};$$

$$YW_{\min} \le Y_1 + u\Delta Y \le YW_{\max}.$$

Each of these four inequalities, can be expressed as:

$$u.p_k = q_k ; k = 1,2,3,4$$

where, the parameters are defined as:

$$p_1 = -\Delta X, \quad q_1 = X_1 - XW_{\min}$$

$$p_2 = \Delta X, \quad q_2 = XW_{\max} - X_1$$

$$p_3 = -\Delta Y, \quad q_3 = Y_1 - YW_{\min}$$

$$p_4 = \Delta Y, \quad q_4 = YW_{\max} - Y_1$$

Based on these four inequalities, we can find the following conditions of line clipping:

- If $p_k = 0$, the line is parallel to the corresponding clipping boundary:

K = 1 → Left
K = 2 → Right
K = 3 → Bottom
K = 4 → Top

- If for any k, for which $p_k = 0$:

  - $q_k < 0$, the line is completely outside the boundary

  - $q_k \geq 0$, the line is inside the parallel clipping boundary.

- If $p_k < 0$, the line proceeds from the ***outside to the inside*** of the particular clipping boundary (visualize infinite extensions in both).

- If $p_k > 0$, the line proceeds from the ***inside to the outside*** of the particular clipping boundary (visualize infinite extensions in both).

In both these cases, the intersection parameter is calculated as:

$$u = q_k / p_k$$

The Algorithm:

- Initialize line intersection parameters to: $u_1 = 0$; $u_2 = 1$;

- Obtain p_i, q_i; for $i = 1, 2, 3, 4$.

- Using p_i, q_i - find if the line can be rejected or the intersection parameters must be adjusted.

- If $p_k < 0$, update u_1 as:

$$\max[0, (q_k / p_k)], k = 1 - 4$$

- If $p_k > 0$, update u_2 as:

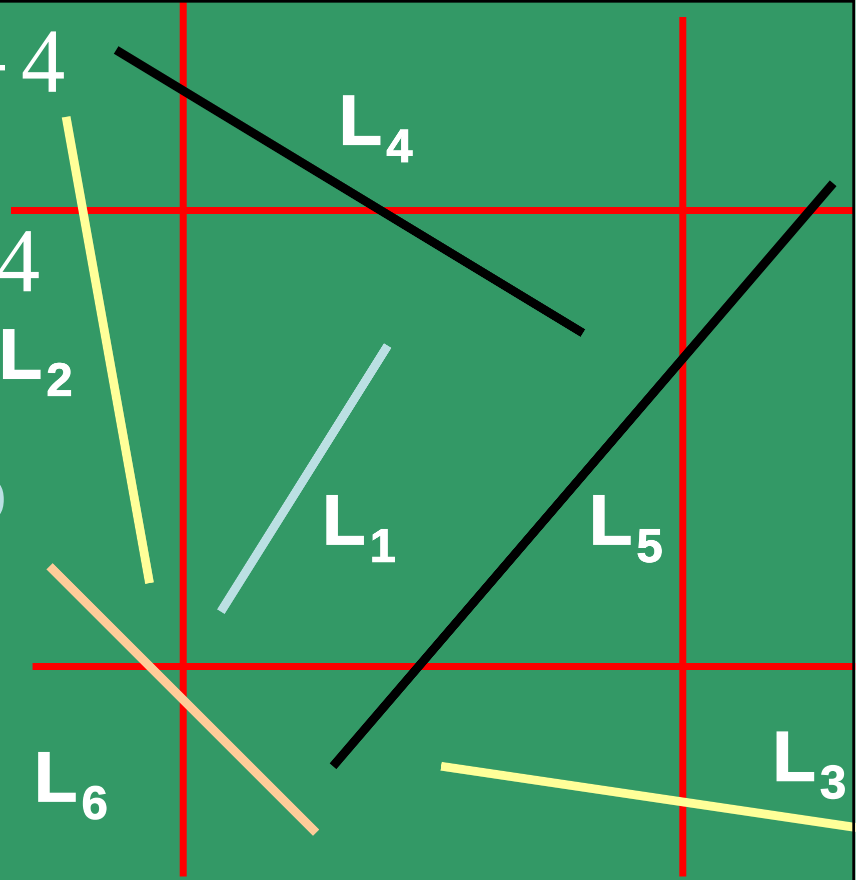
$$\min[1, (q_k / p_k)], k = 1 - 4$$

- After update, if $u_1 > u_2$: reject the line.

$$p_k < 0 : u_1 = \max[0, (q_k / p_k)], k = 1 - 4$$

$$p_k > 0 : u_2 = \min[1, (q_k / p_k)], k = 1 - 4$$

K = 1 → Left; K = 2 → Right
K = 3 → Bottom; K = 4 → Top



$$p_1 = -\Delta X, \quad q_1 = X_1 - XW_{\min}$$

$$p_2 = \Delta X, \quad q_2 = XW_{\max} - X_1$$

$$p_3 = -\Delta Y, \quad q_3 = Y_1 - YW_{\min}$$

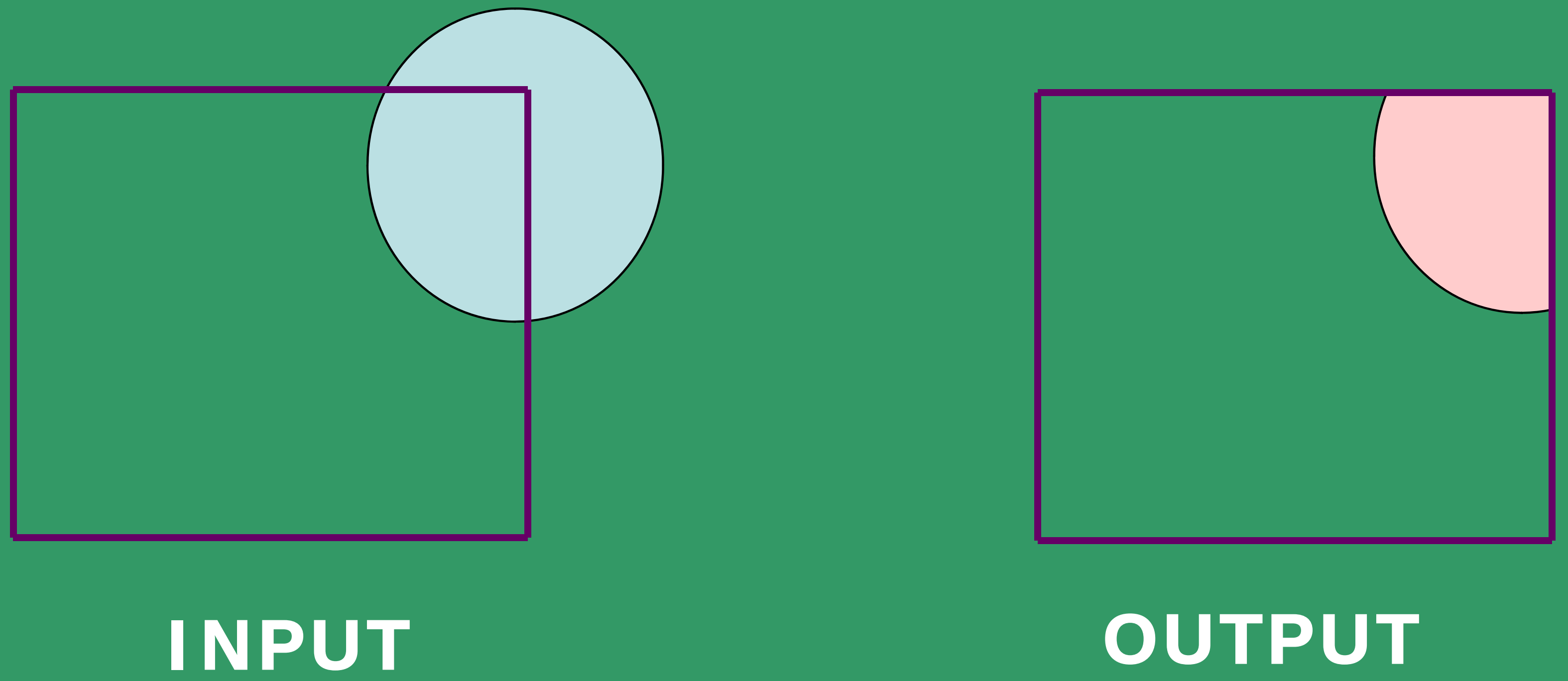
$$p_4 = \Delta Y, \quad q_4 = YW_{\max} - Y_1$$

L_1: (0, 1); */ *Analyze the line in both directions.*
L_2: [max(0, $-d_2$, $-d_3$) min(1, $-d_1$, d_4(<1))]
= (0, $-d_1$) – hence reject.
L_4: [max(0, $-d_2$, $-d_3$) min(1, d_1, d_4)]
= (0, d_4) (why ?) – so accept and clip

Do for L_3, L_5 & L_6.

What about Circle / Ellipse clipping

or for curves ??

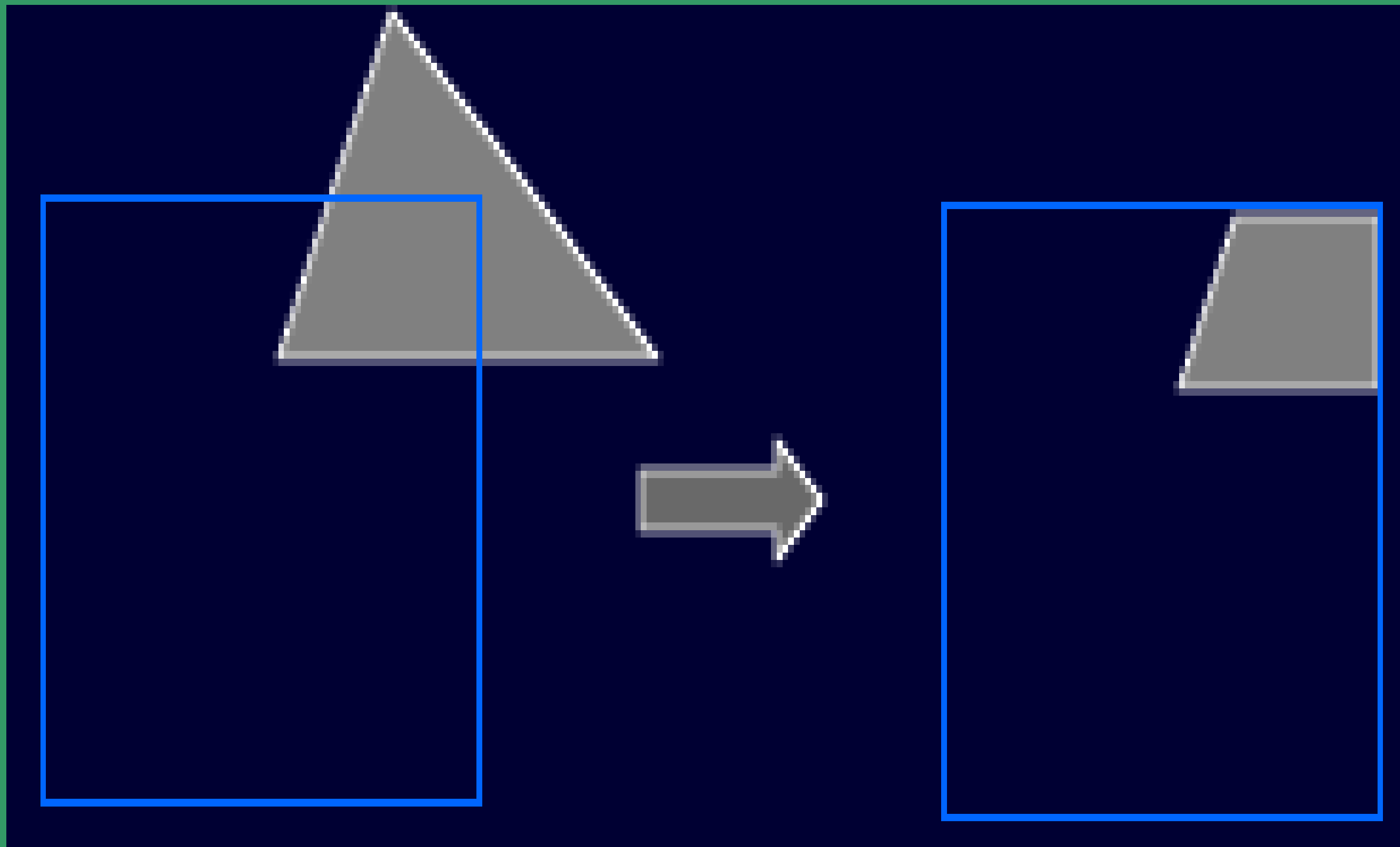


Can you do a inside-outside test,
for the object vs. rectangle ?

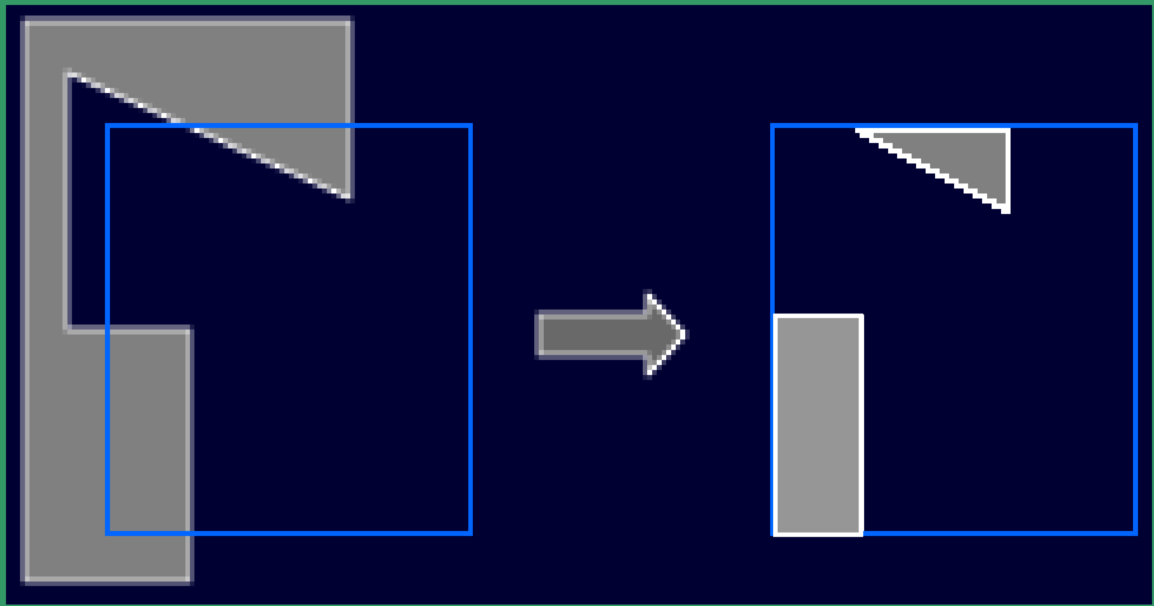
POLYGON

CLIPPING

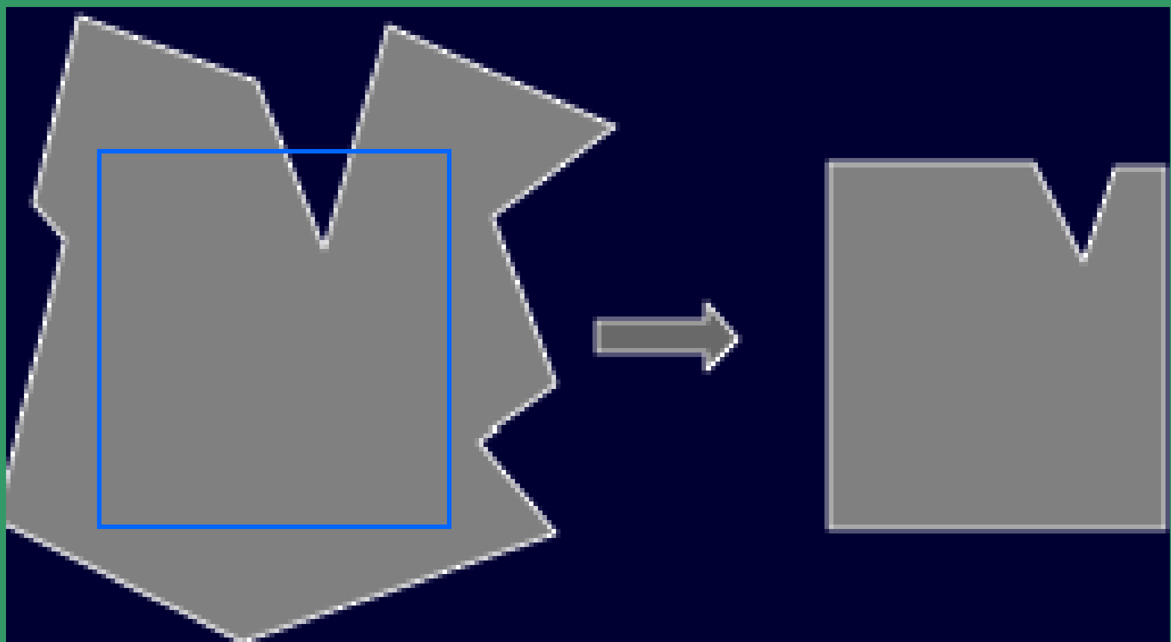
Examples of Polygon Clipping



CONVEX SHAPE



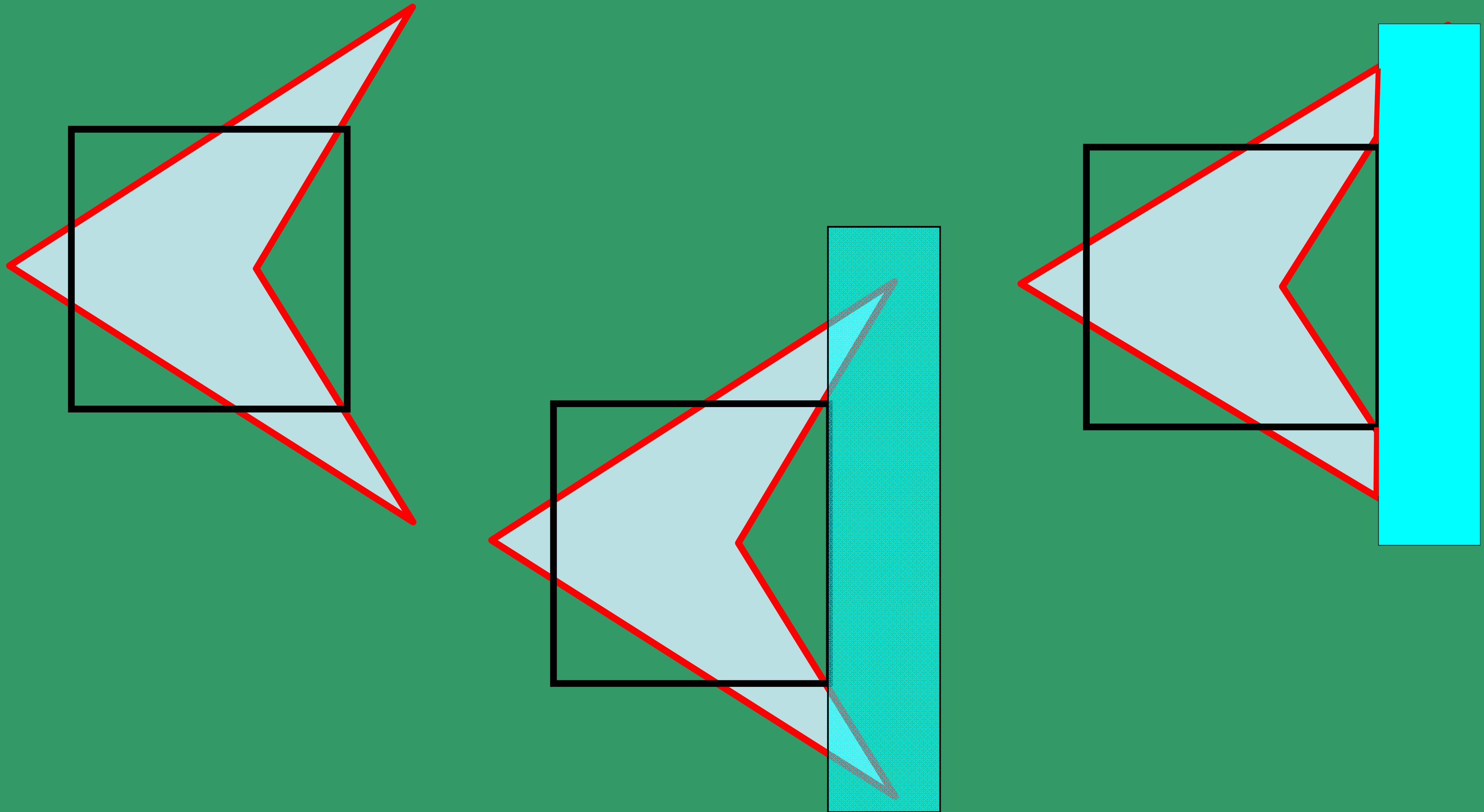
MULTIPLE COMPONENTS

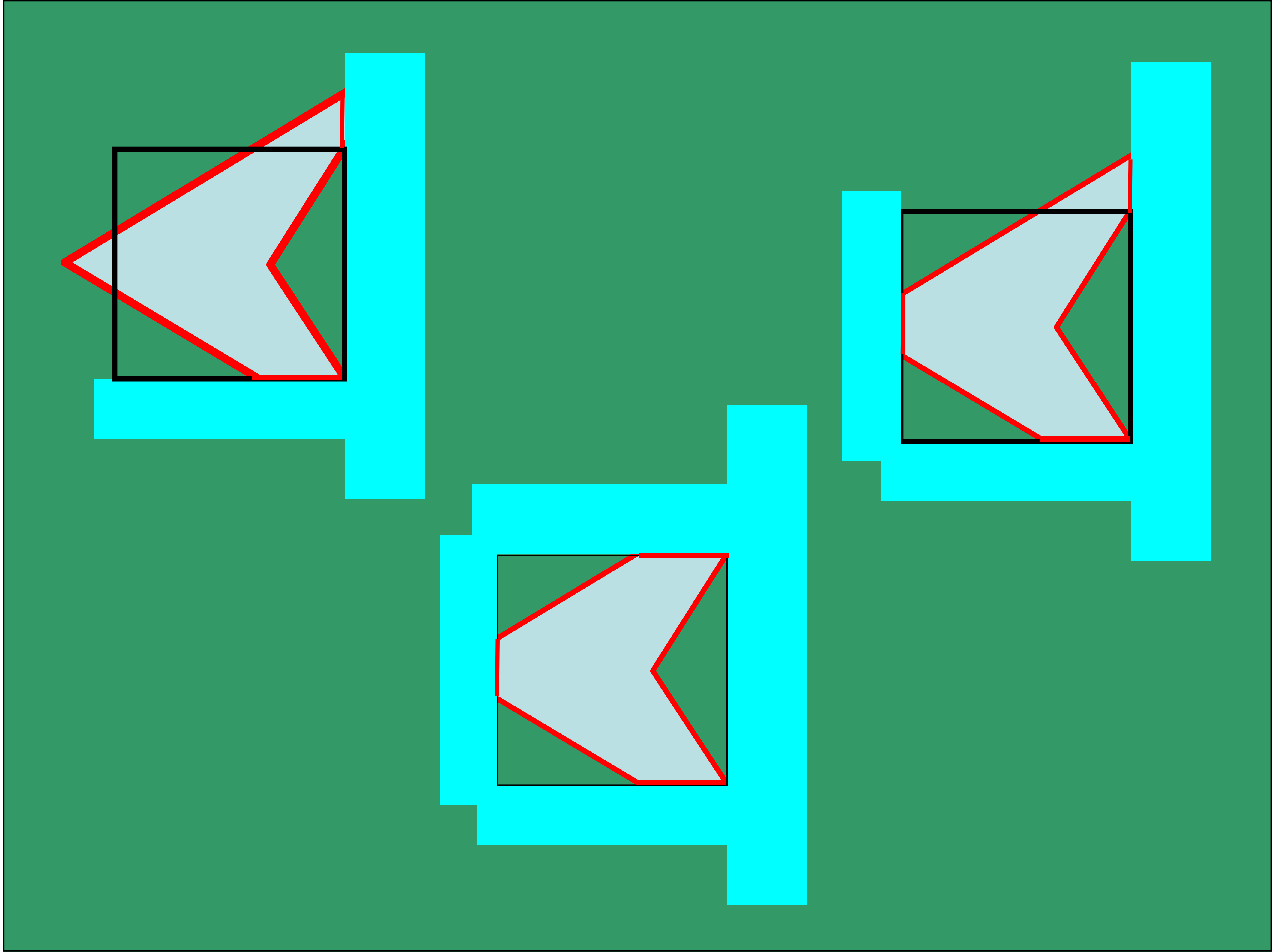


CONCAVE SHAPE

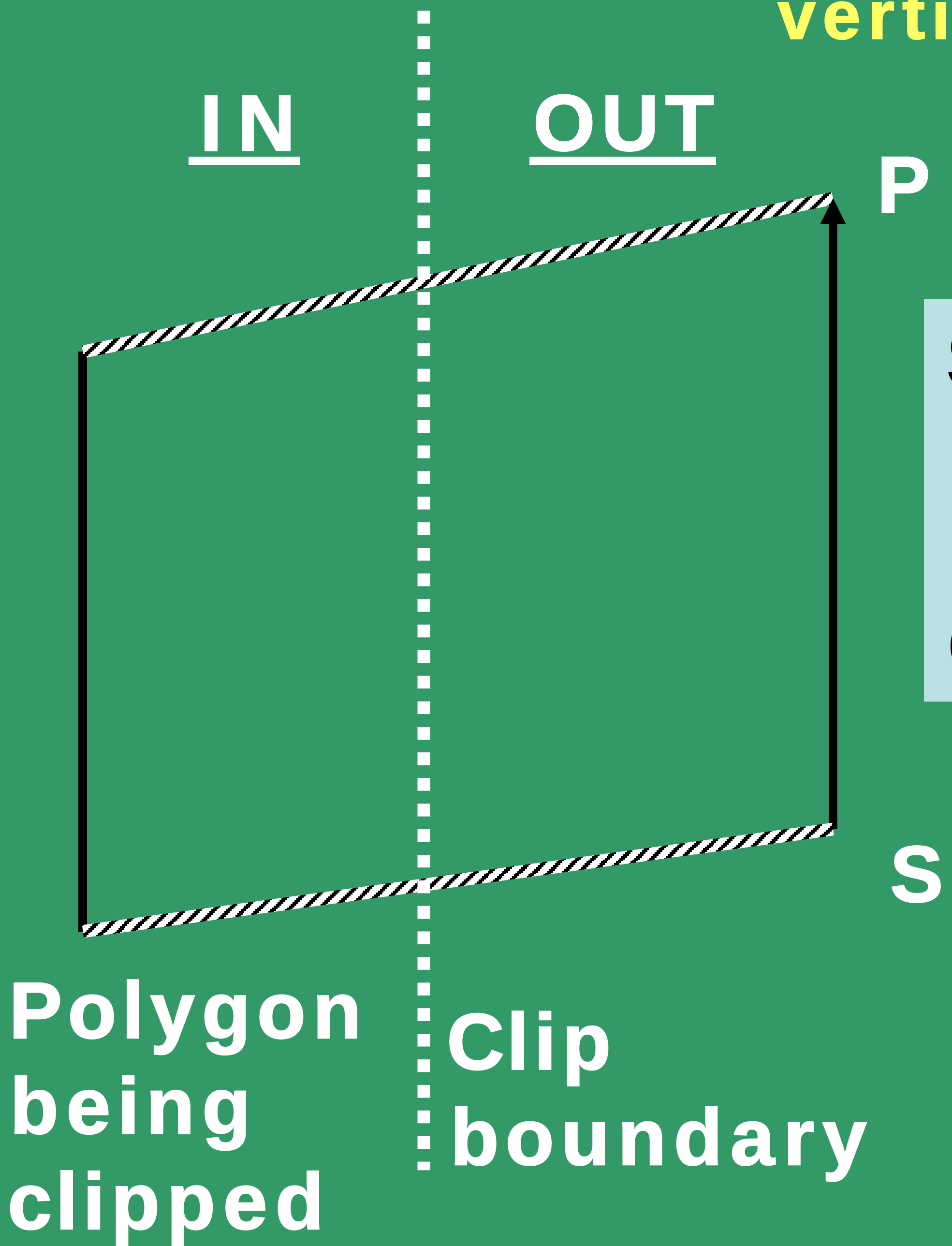
Methodology: CHANGE position of vertices for each edge by line clipping

May have to add new vertices to the list.





Processing of Polygon vertices against boundary

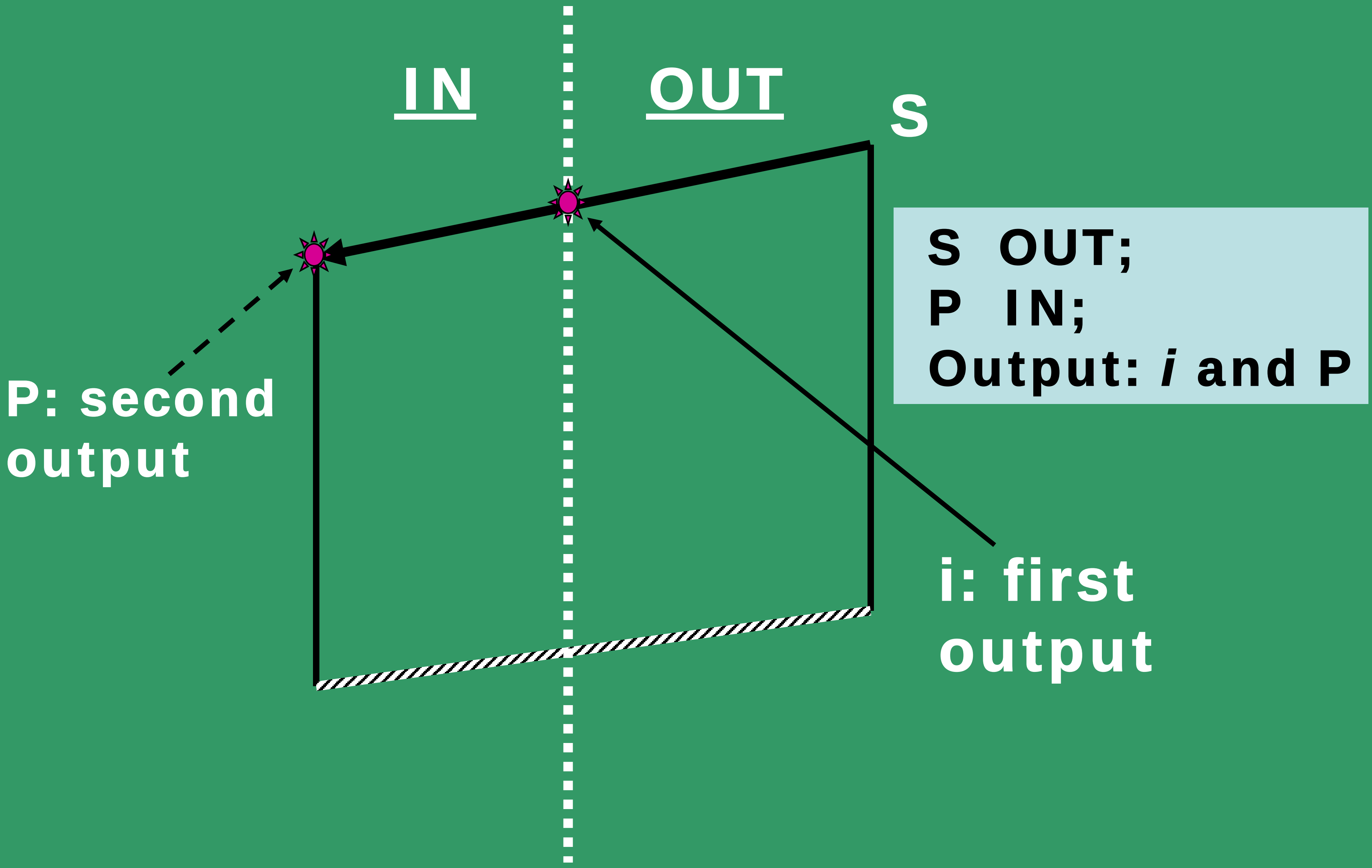


S and P both OUT

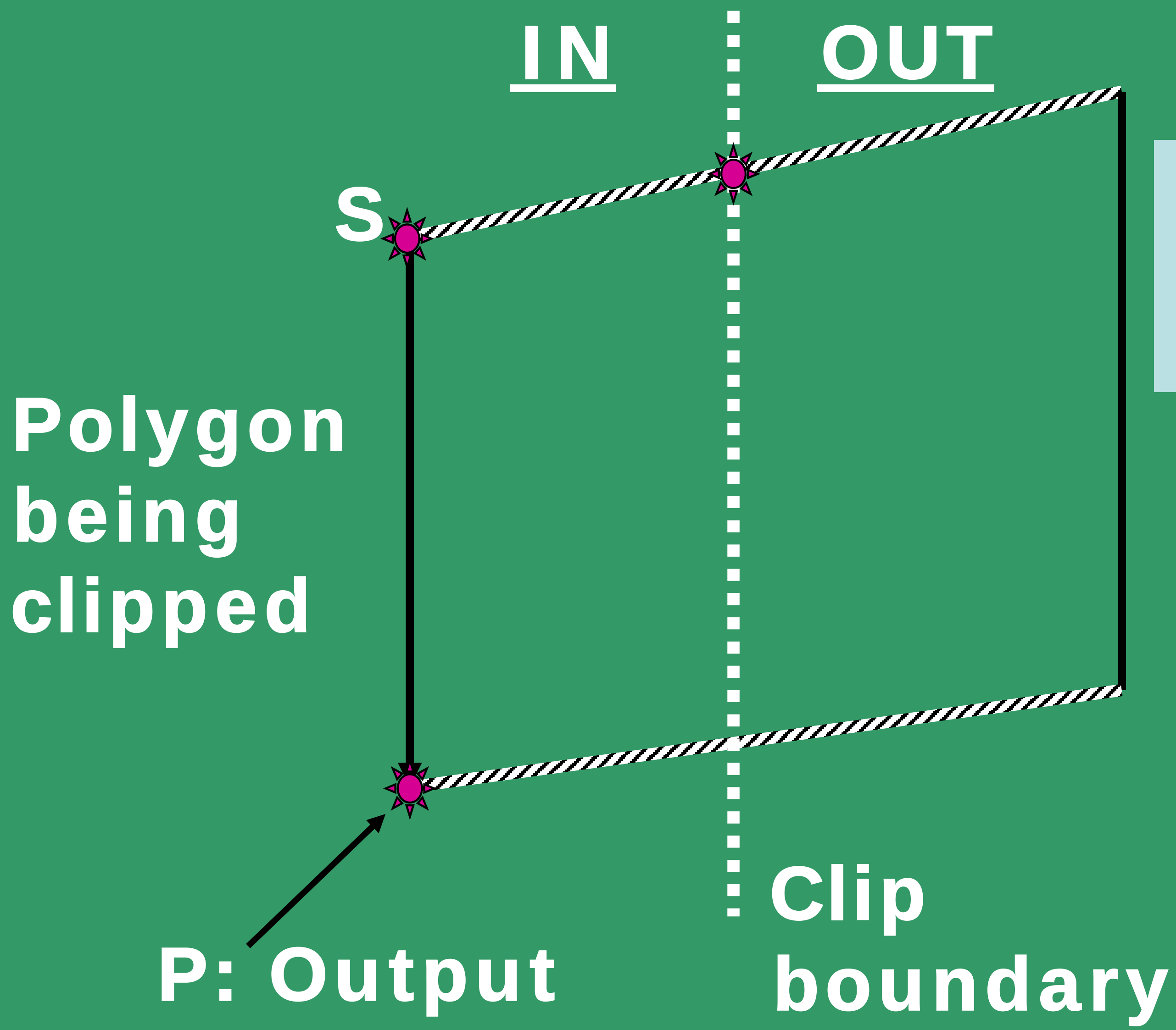
Output: Null.

(No output)

Processing of Polygon vertices against boundary



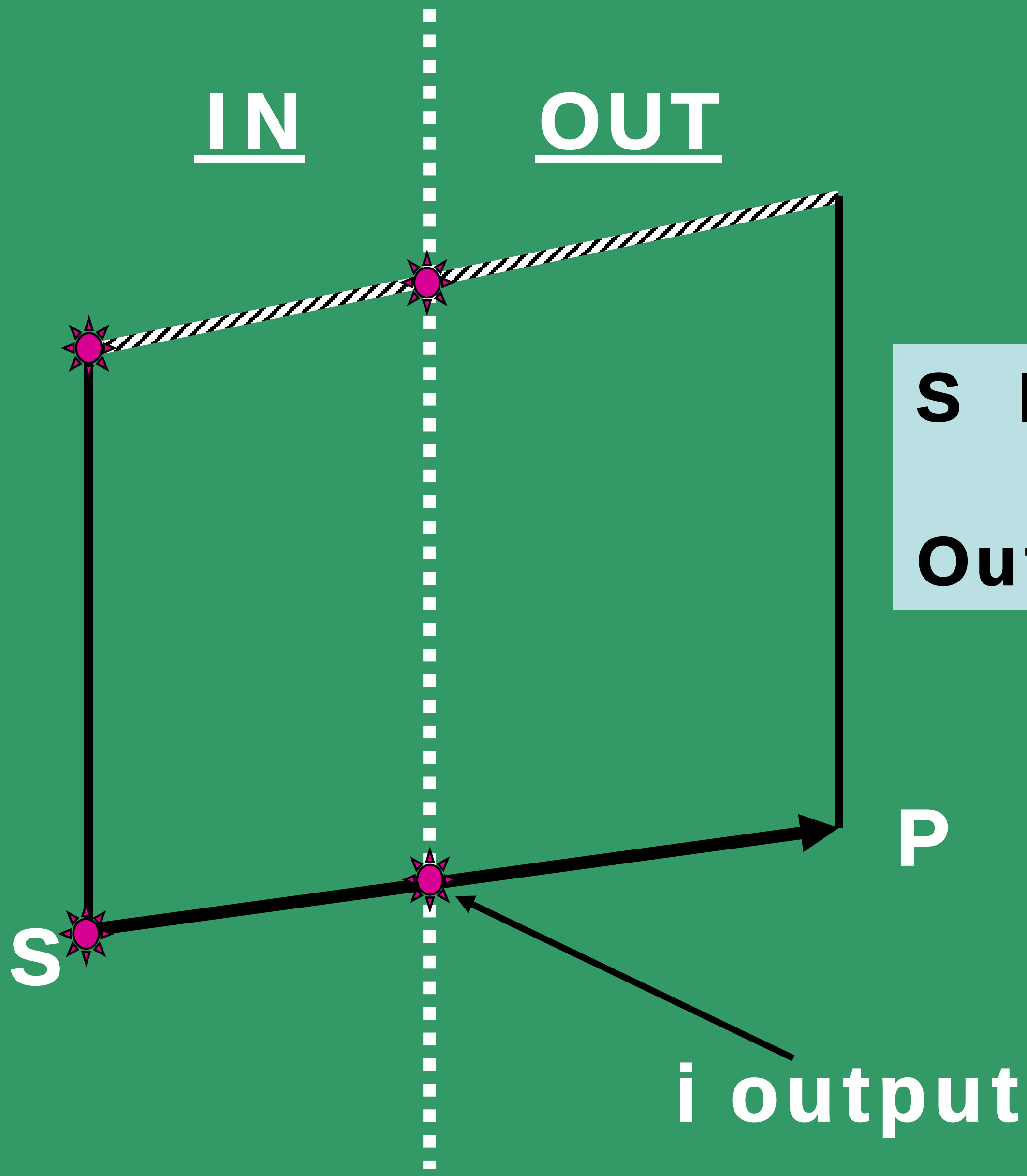
Processing of Polygon vertices against boundary



S and P both IN

Output: P.

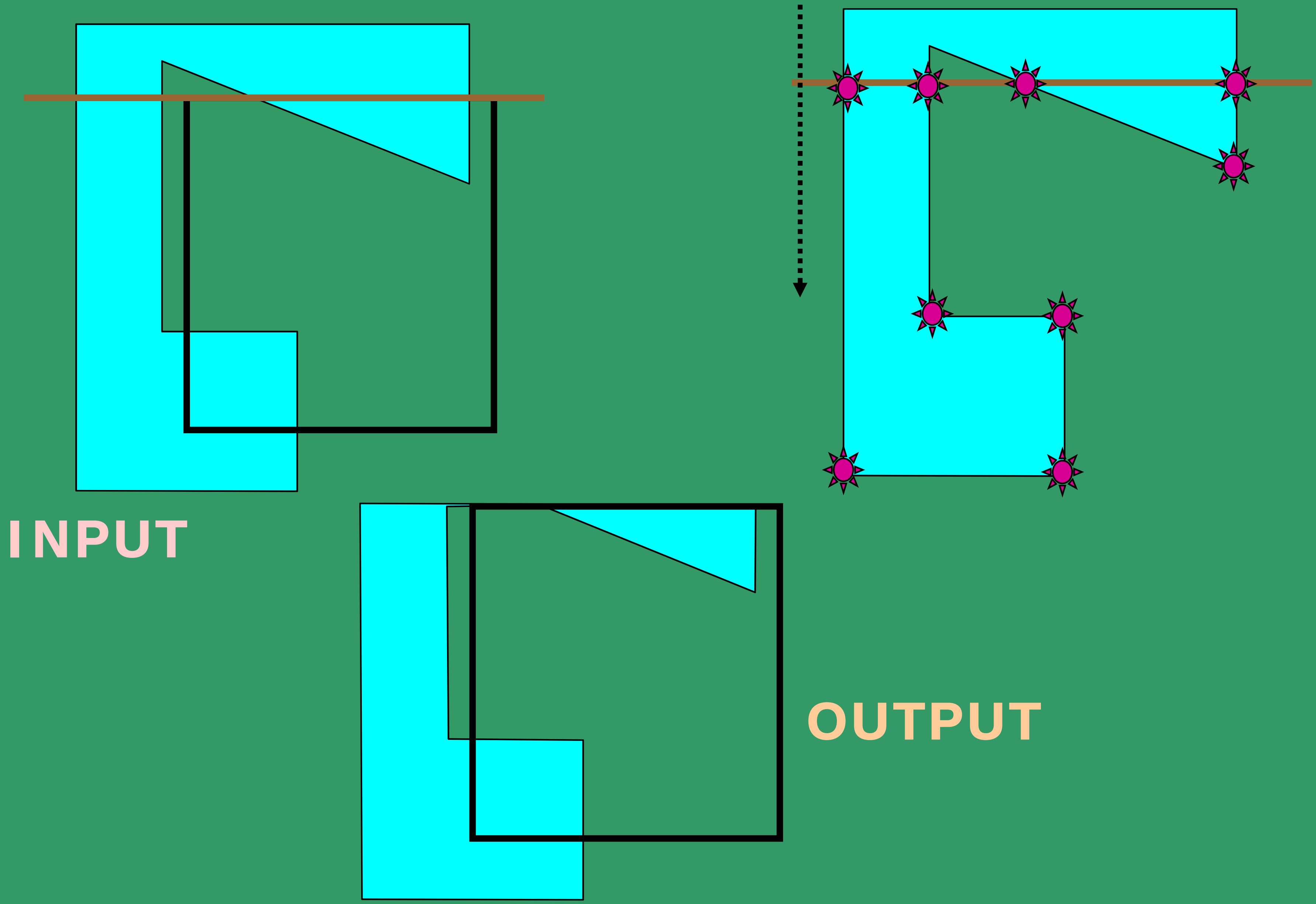
Processing of Polygon vertices against boundary



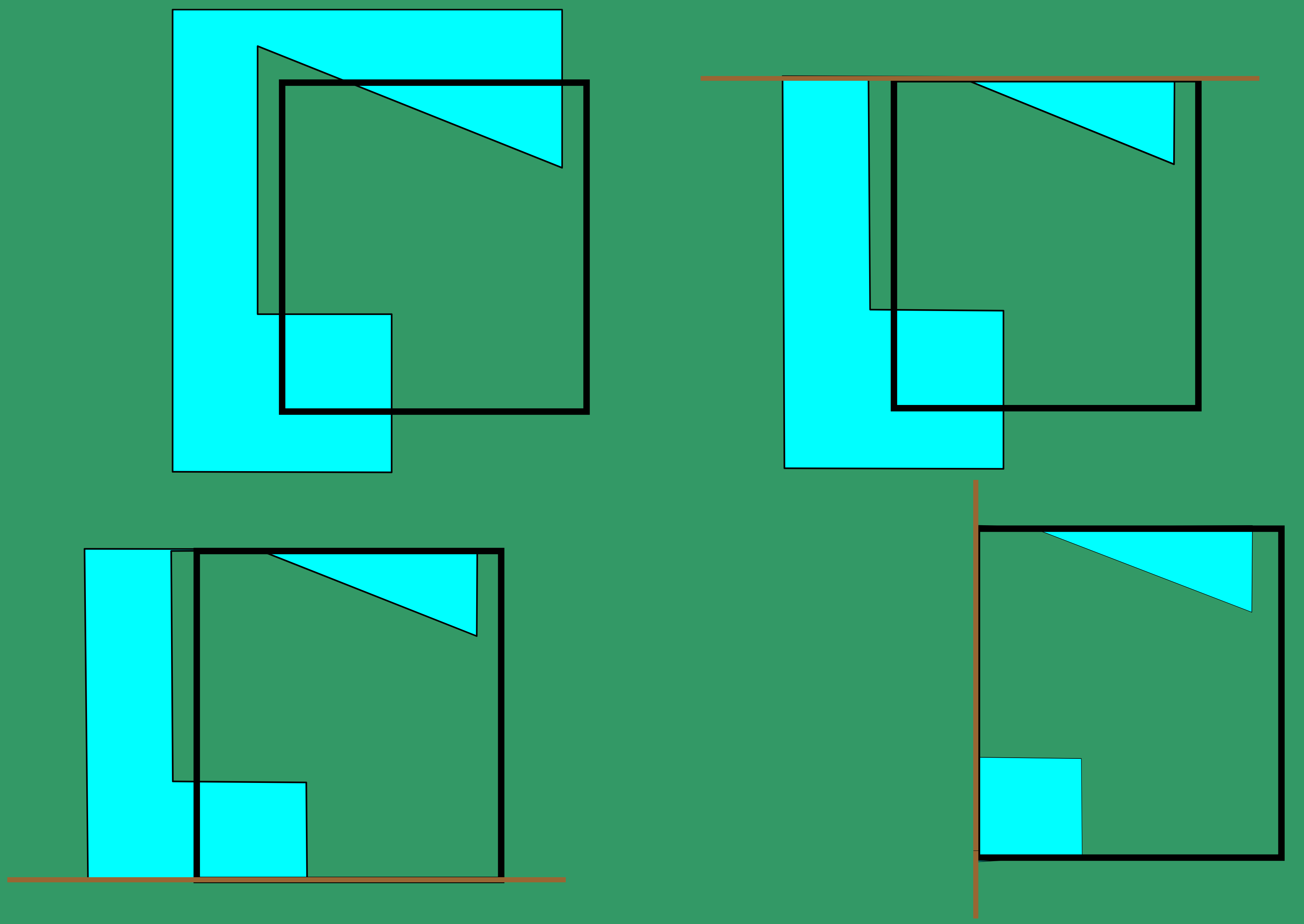
S IN; P OUT

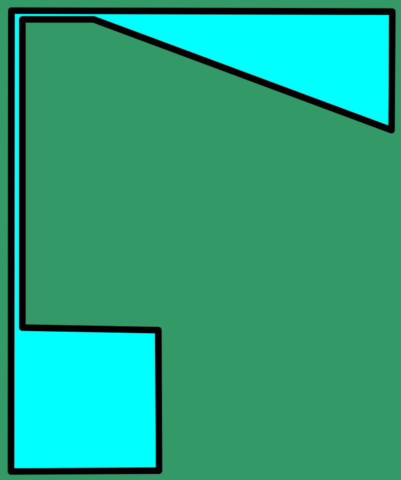
Output: *i*

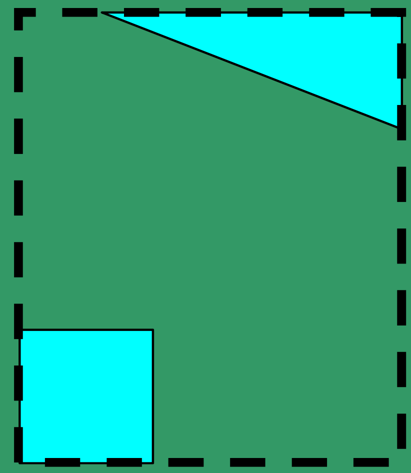
Problems with multiple components



Problems with multiple components



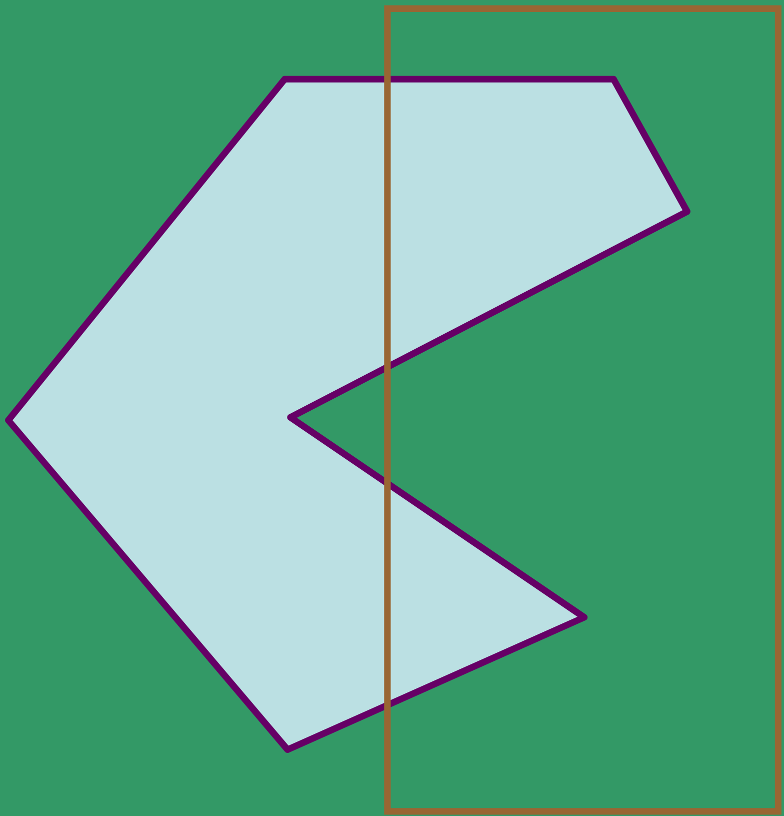




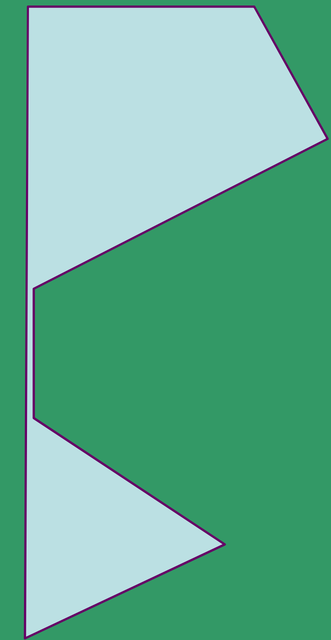
Now output is as above **Desired Output**

Any Idea ??

– the modified Weiler-Atherton algorithm



Solution for multiple components



For say, clockwise processing of polygons, follow:

- For OUT -> IN pair, follow the polygon boundary
- For IN -> OUT pair, follow Window boundary in clockwise direction

For say, clockwise processing of polygons, follow:

- For OUT -> IN pair, follow the polygon boundary
- For IN -> OUT pair, follow Window boundary in clockwise direction