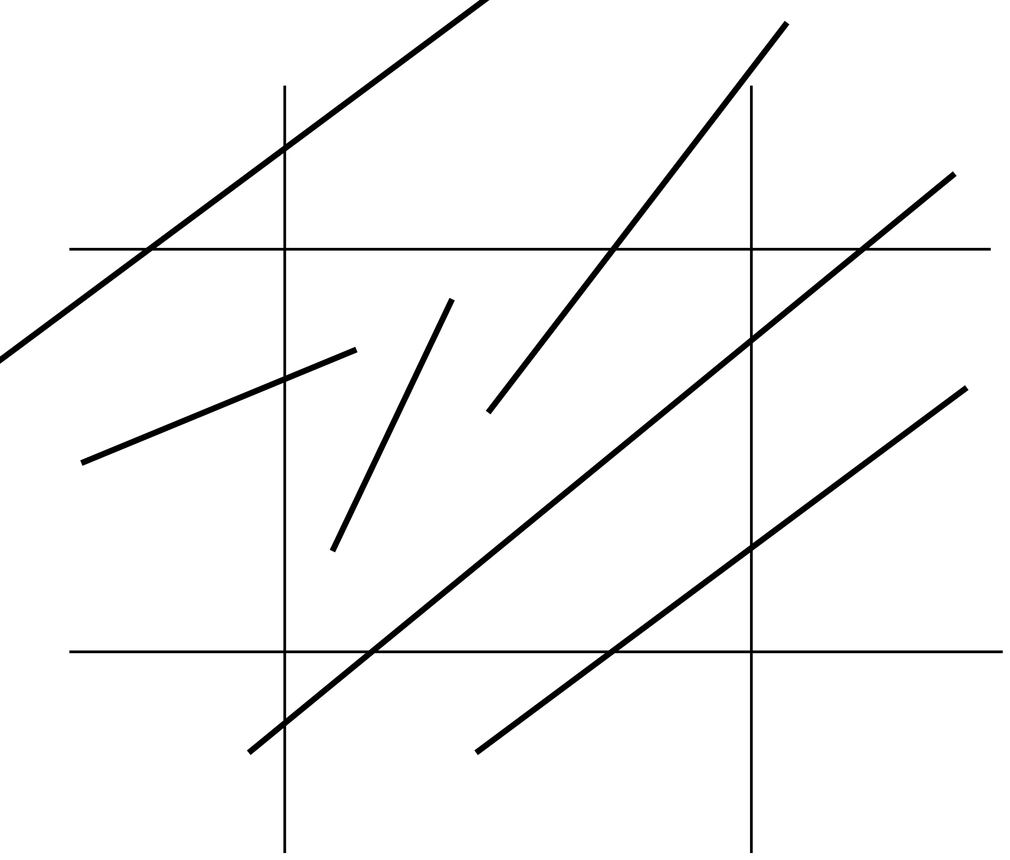
Clipping:

LINES

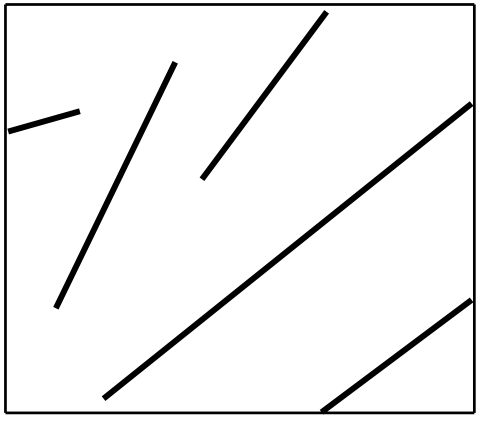
and

POLYGONS



INPUT

OUTPUT



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: $Y = K_y$.

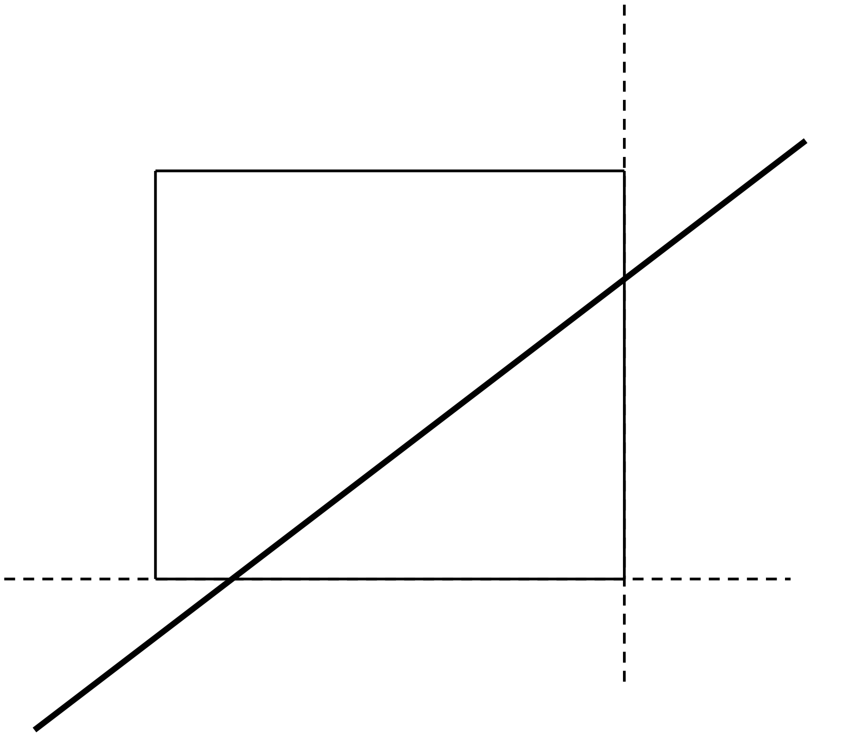
Solve with respective pairs:

$$t_{lx} = \frac{K_x - X_0}{X_1 - X_0} \qquad t_{ly} = \frac{K_y - Y_0}{Y_1 - Y_0}$$

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

t_{edge} and t_{line}.

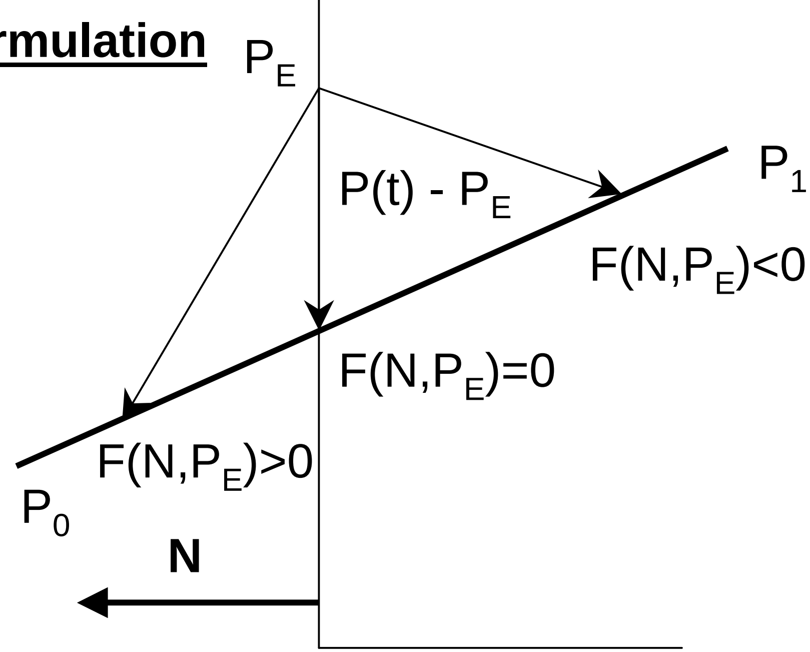
Check if they fall within range [0 - 1].



i.e. Solve:

$$t_1(P_1 - P_0) - t_2(P_1^{'} - P_0^{'}) = P_0^{'} - P_0$$

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;**

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

$$\text{Substitute, } D = P_1 - P_0;$$

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

To ensure valid value of t, denominator must be non-zero.
Assuming D, N <> 0, check if:
N.D <> 0. i.e. edge and line are not parallel. **If they are parallel ?**

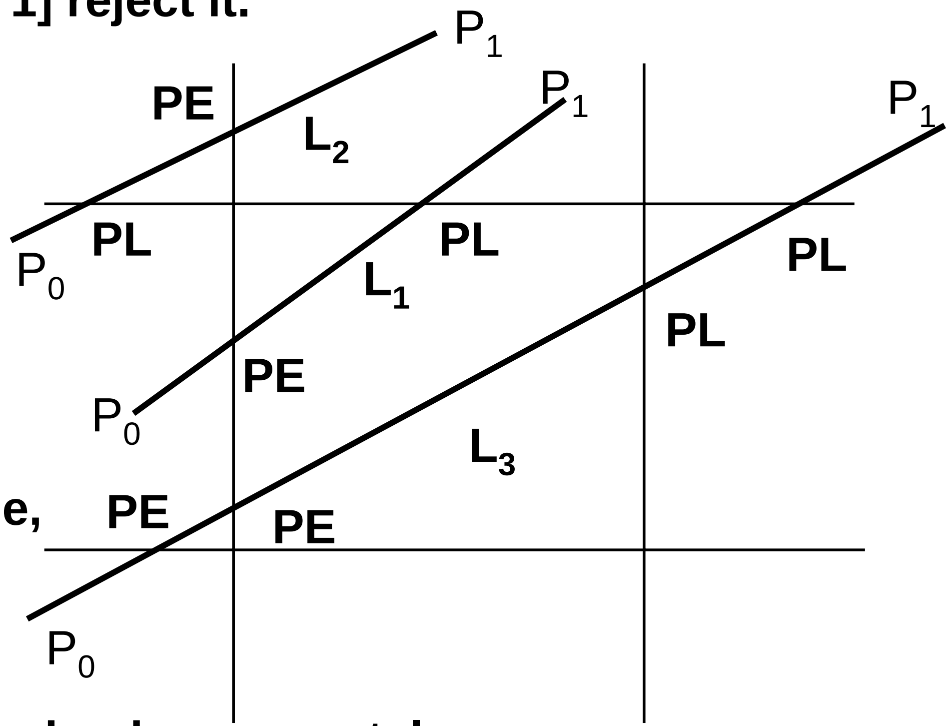
Use the above expression of t to obtain all the four intersection:

- **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 ?

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 L_2 and L_3 lines.

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

Simplified Calculations for parametric line Clipping, applicable for any other algorithm too.

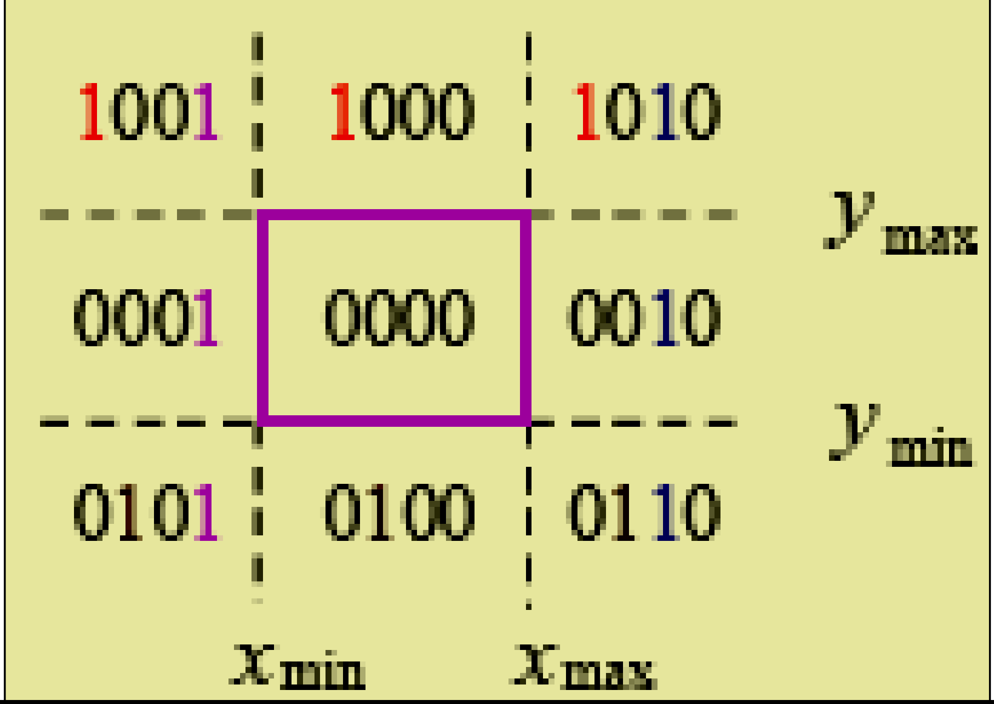
Clip Edge	Normal N	$P_E^{\$}$	$P_0 - P_E$	$t = \frac{N.[P_0 - P_E]}{-N.D}$
Left: $X = X_{min}$	(-1, 0)	(X_{min}, Y)	$(X_0 - X_{min}, Y_0 - Y)$	$\frac{-(X_0 - X_{min})}{(X_1 - X_0)}$
Right: $X = X_{max}$	(1, 0)	(X_{max}, Y)	$(X_0 - X_{max}, Y_0 - Y)$	$\frac{(X_0 - X_{max})}{-(X_1 - X_0)}$
Bottom: $Y = Y_{min}$	(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. Put any value, as shown in table.

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:

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

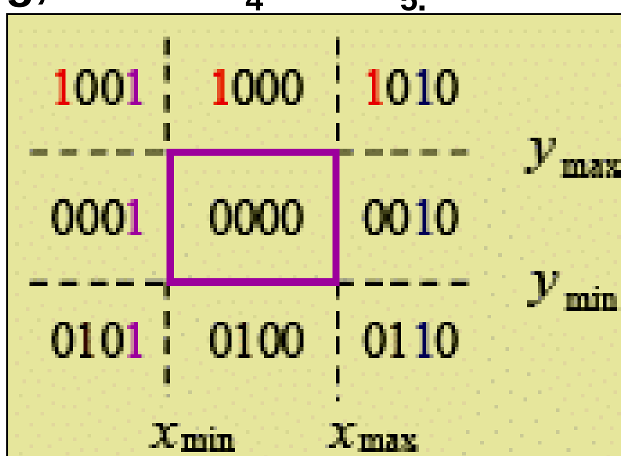
Use these end-point codes to locate the line. Various possibilities:

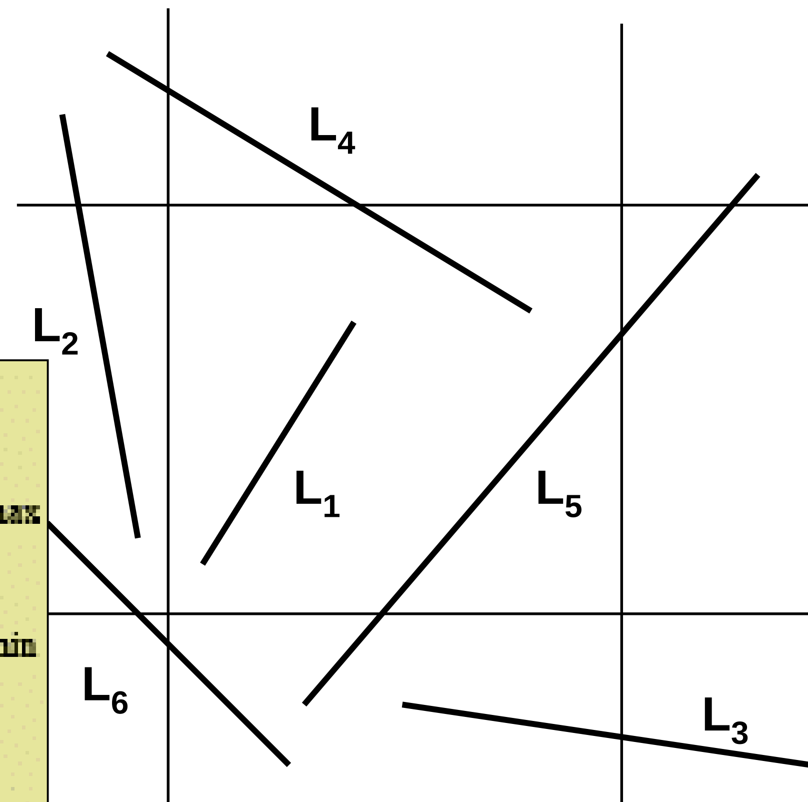
- **If both endpoint codes are [0000], the line lies completely inside the box, no need to clip. This is the simplest case (e.g. L_1).**

- **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).**
Reject it.
Test is performed by bit-wise AND operation.
If the results in not [0000], reject the line.

- **Neither completely reject nor inside the box:**
Needs more processing, Lines: L_4 and L_5.

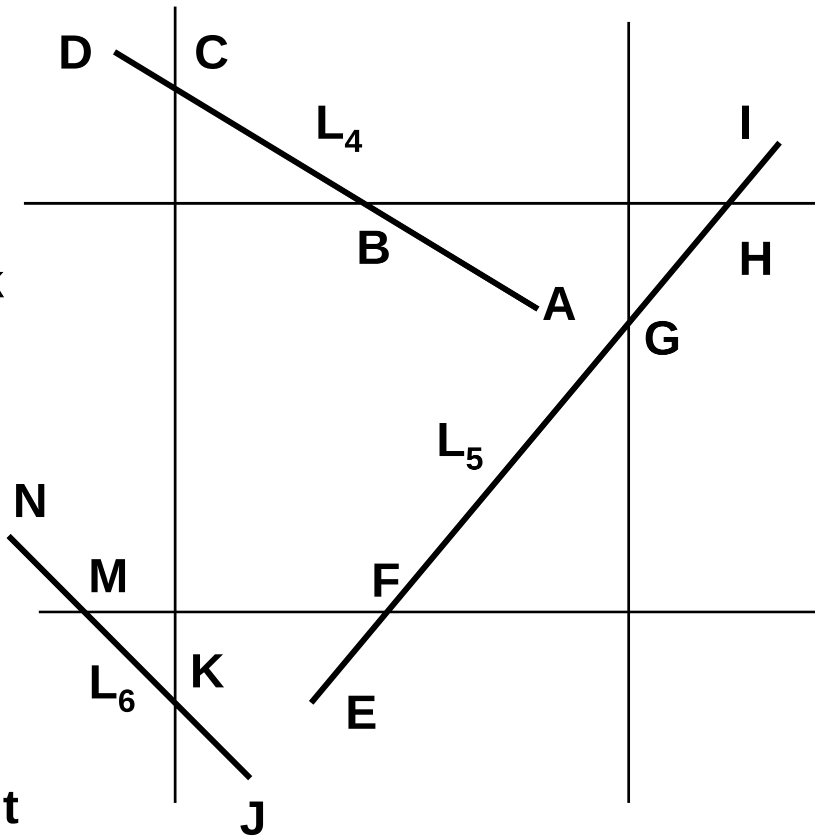
- **What about Line L_6 ?**





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).



Algm. 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.
- ***Obtain corresponding intersection point.***
- 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.

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

Inputs:

- **Endpoint coordinates: (X_0, Y_0) and (X_1, Y_1)**
- **Edge for clipping (obtained using outcode of current endpoint).**

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

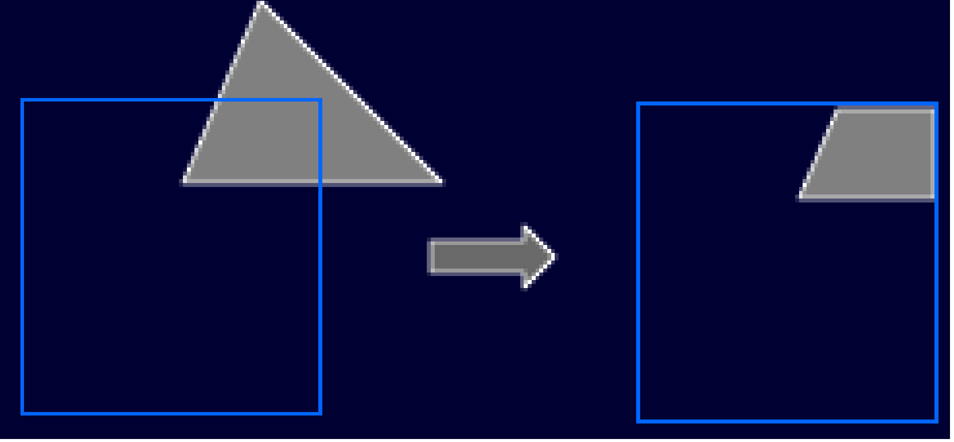
Right Edge: $$Y = Y_0 + (Y_1 - Y_0) * \frac{(X_{max} - X_0)}{X_1 - X_0)}$$

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

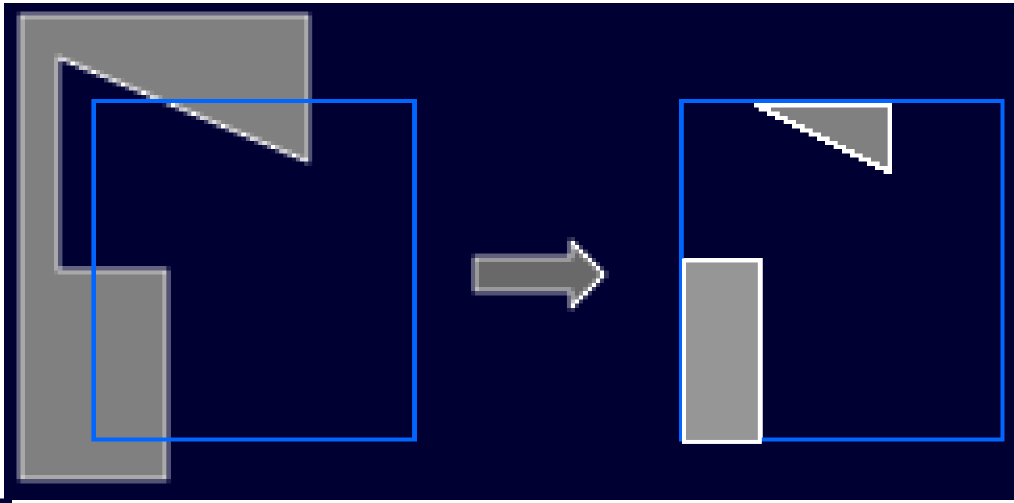
POLYGON

CLIPPING

Examples of Polygon Clipping

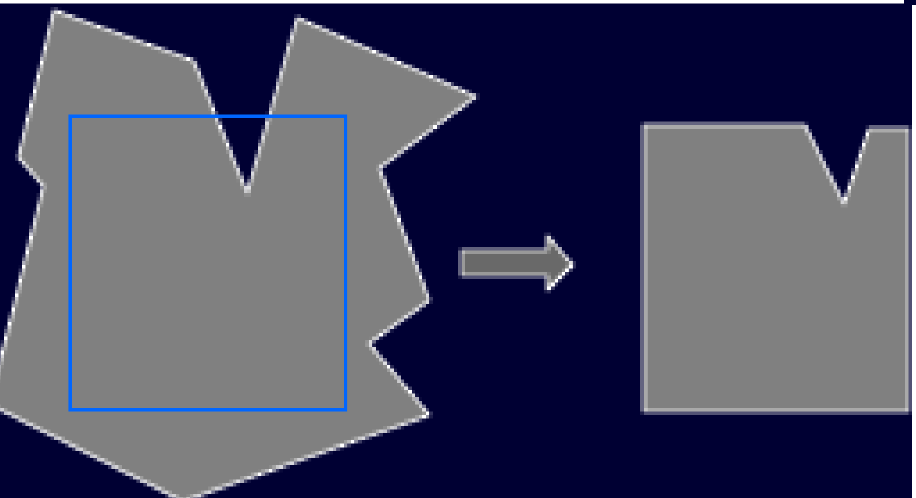


CONVEX SHAPE



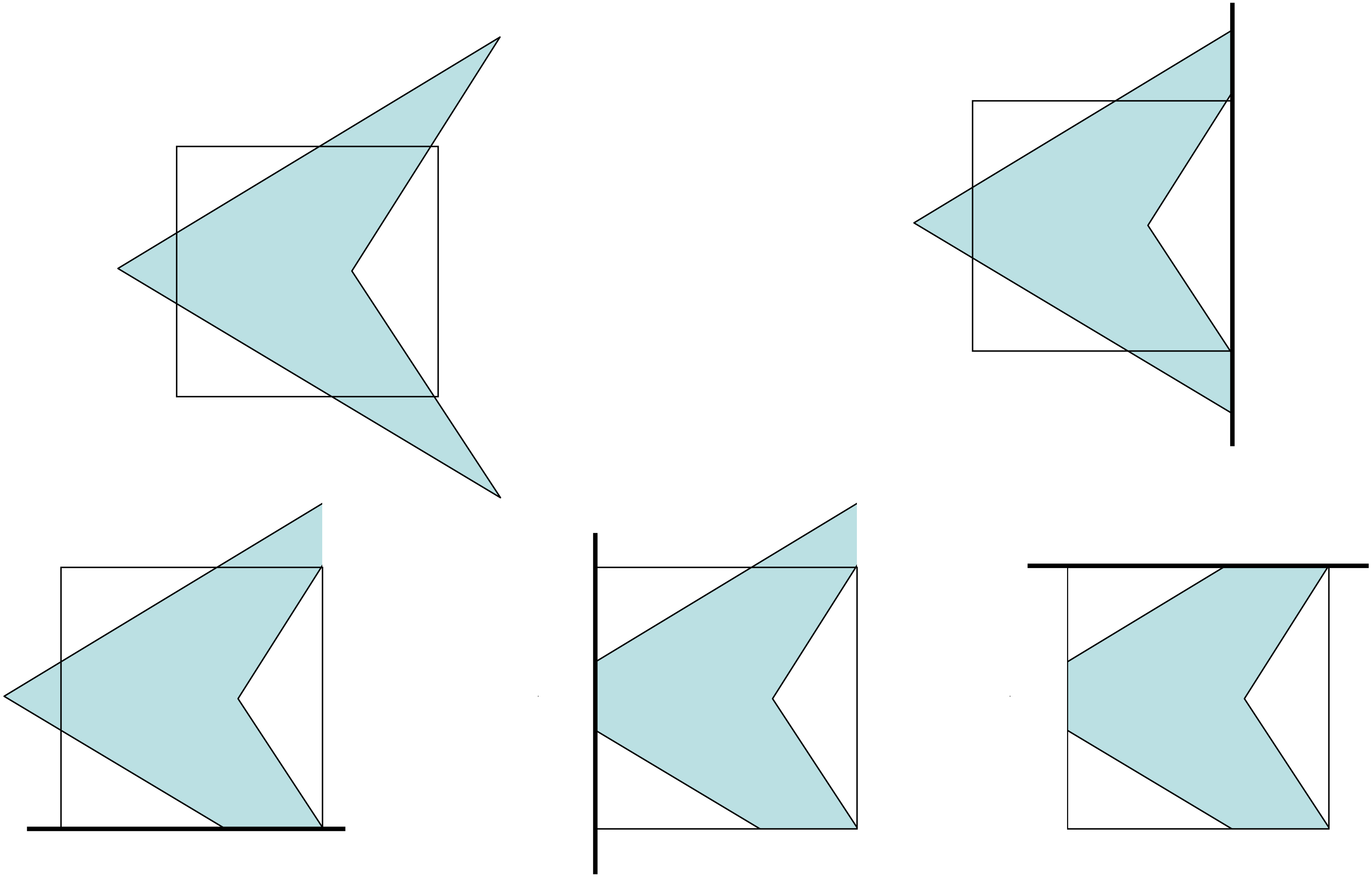
MULTIPLE COMPONENTS

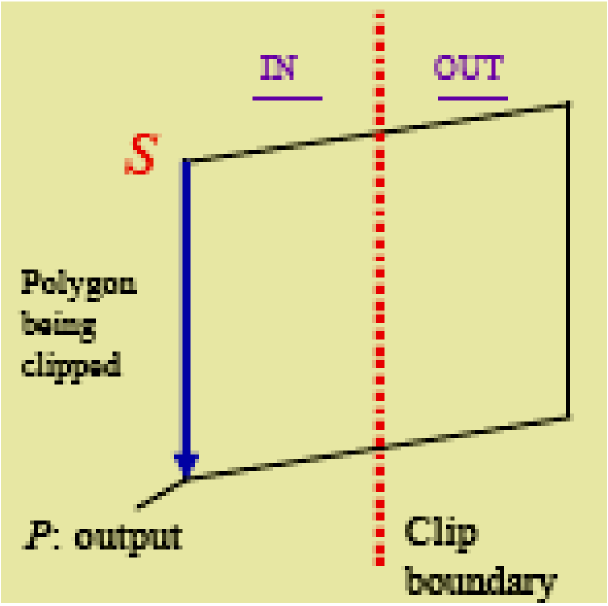
CONCAVE SHAPE



<u>Methodology: CHANGE position of vertices for each edge by line clipping</u>

May have to add new vertices to the list.



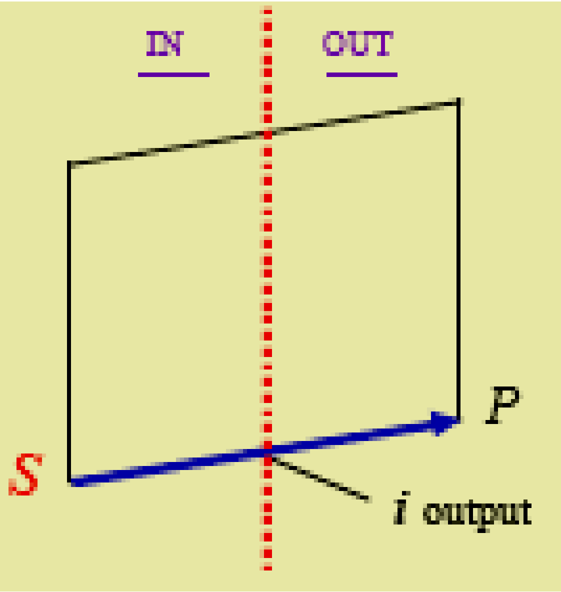


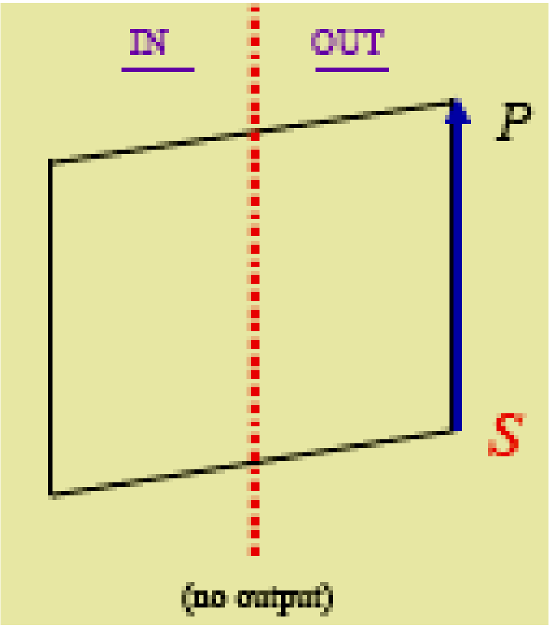
S and P both IN

Output: P.

S IN; P OUT

Output: *i*

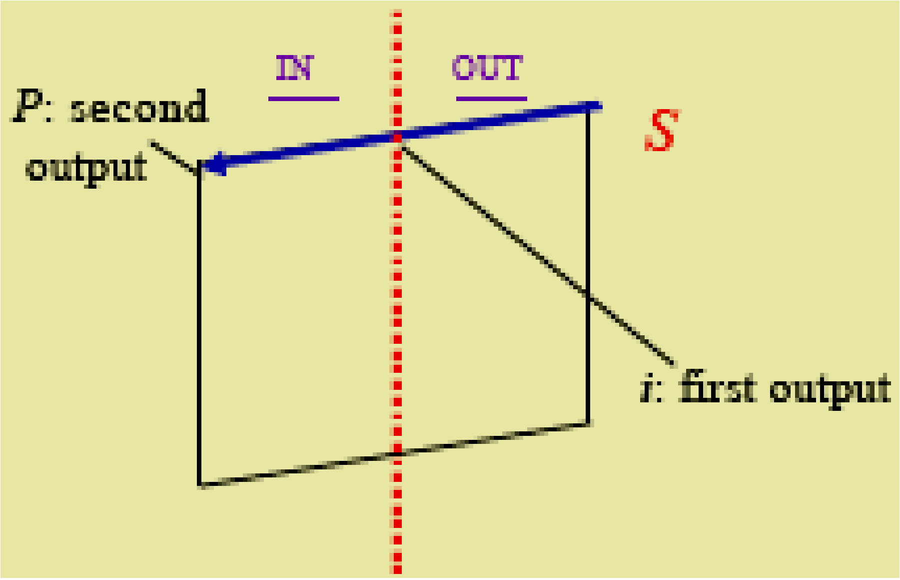


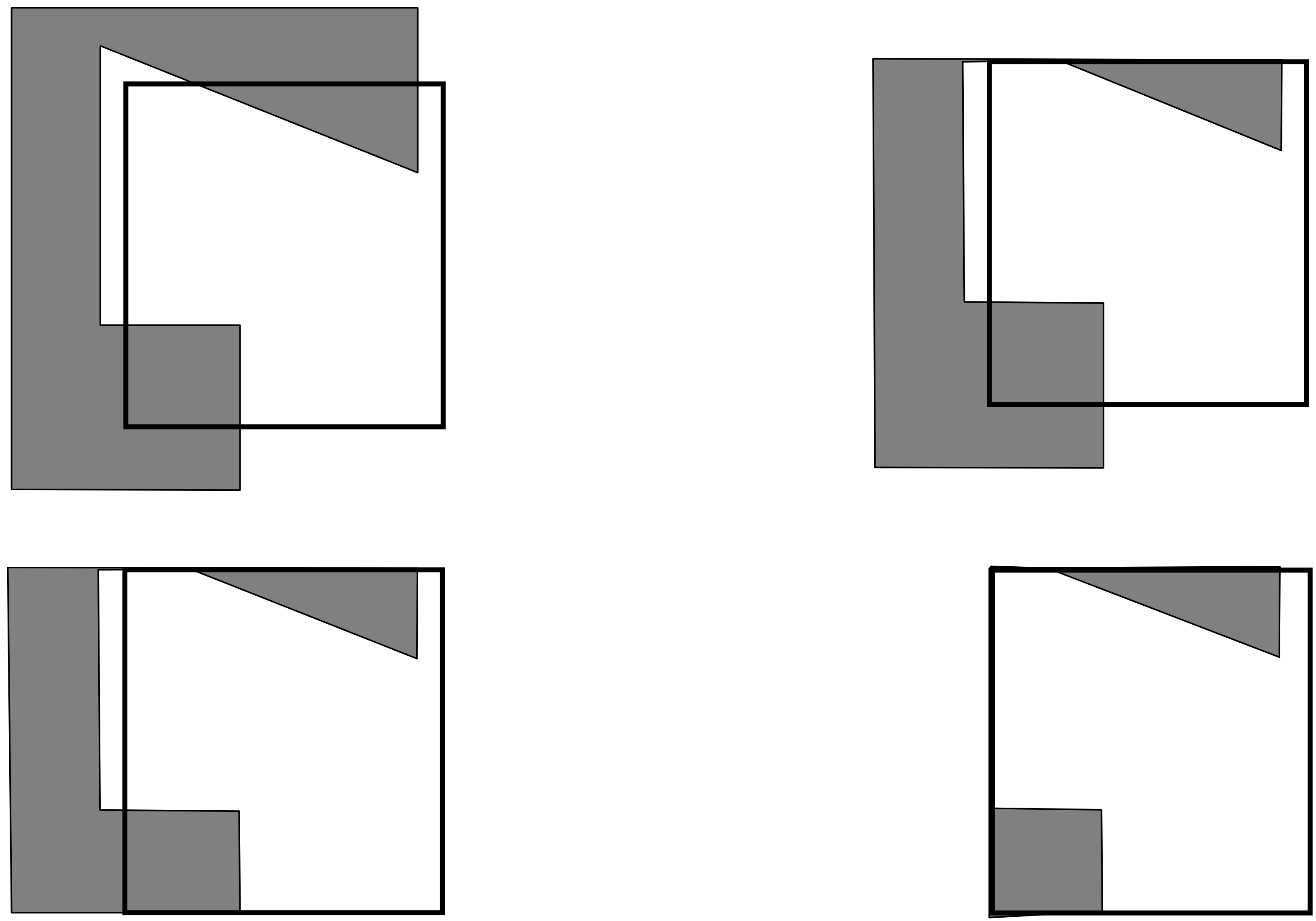


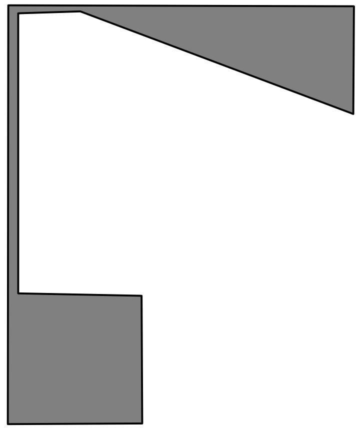
S and P both OUT

Output: Null.

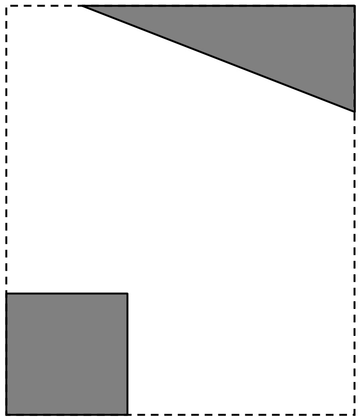
S OUT; P IN; Output: *i* and P







Now output is as above



Desired Output

Any Idea ?? – the modified algorithm