Line Clipping in 2D

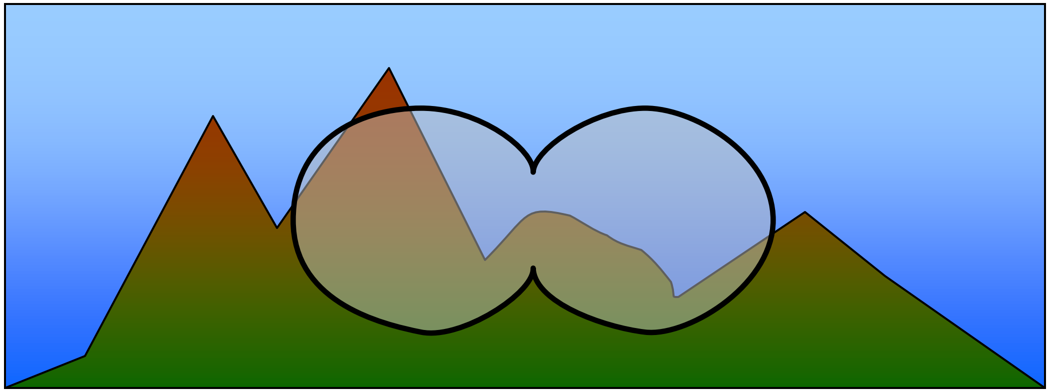
Why would we clip?

We clip objects to our view before rasterization. Why?

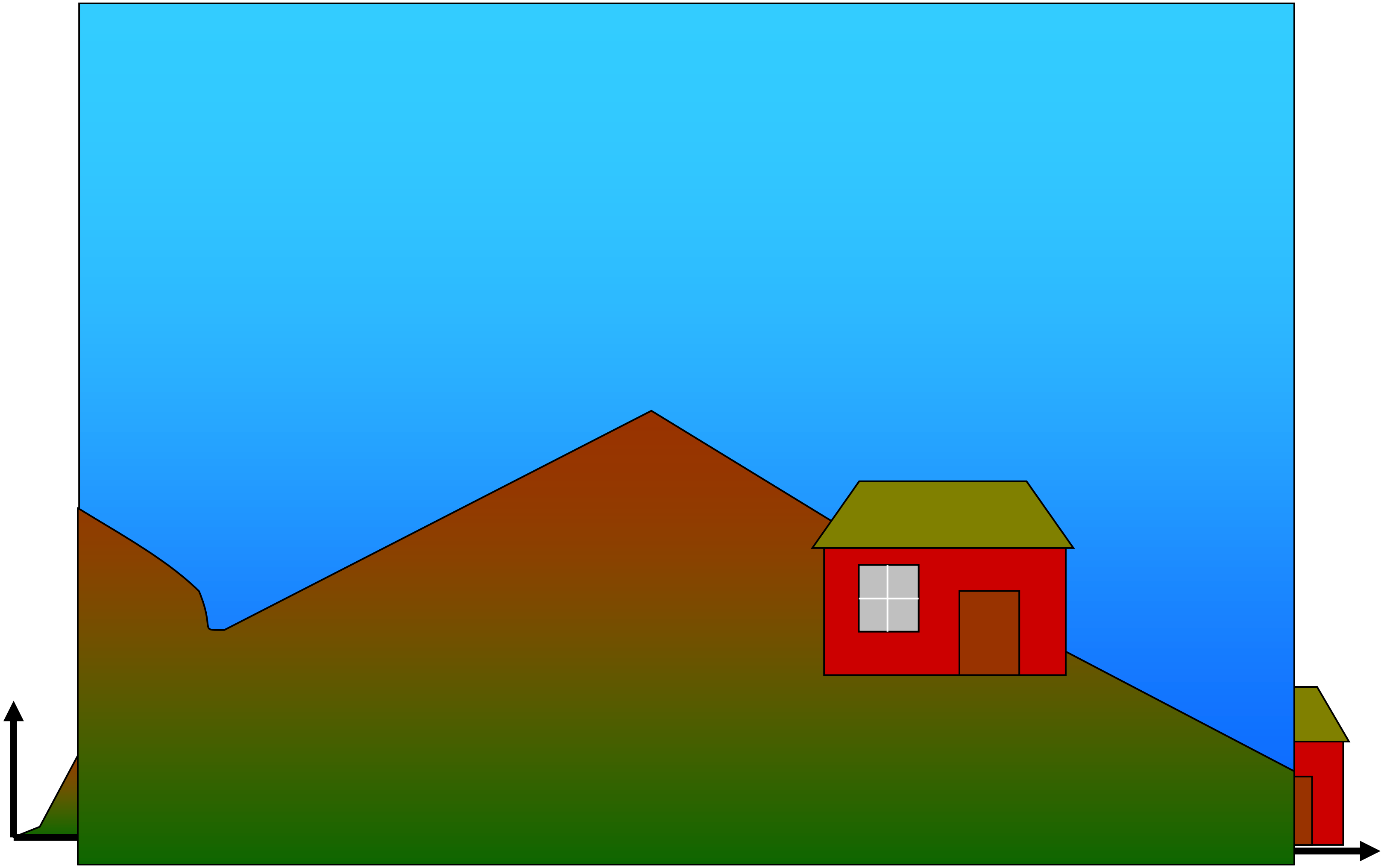
To avoid unnecessary work:

- pixel coordinate calculations
- parameter interpolation

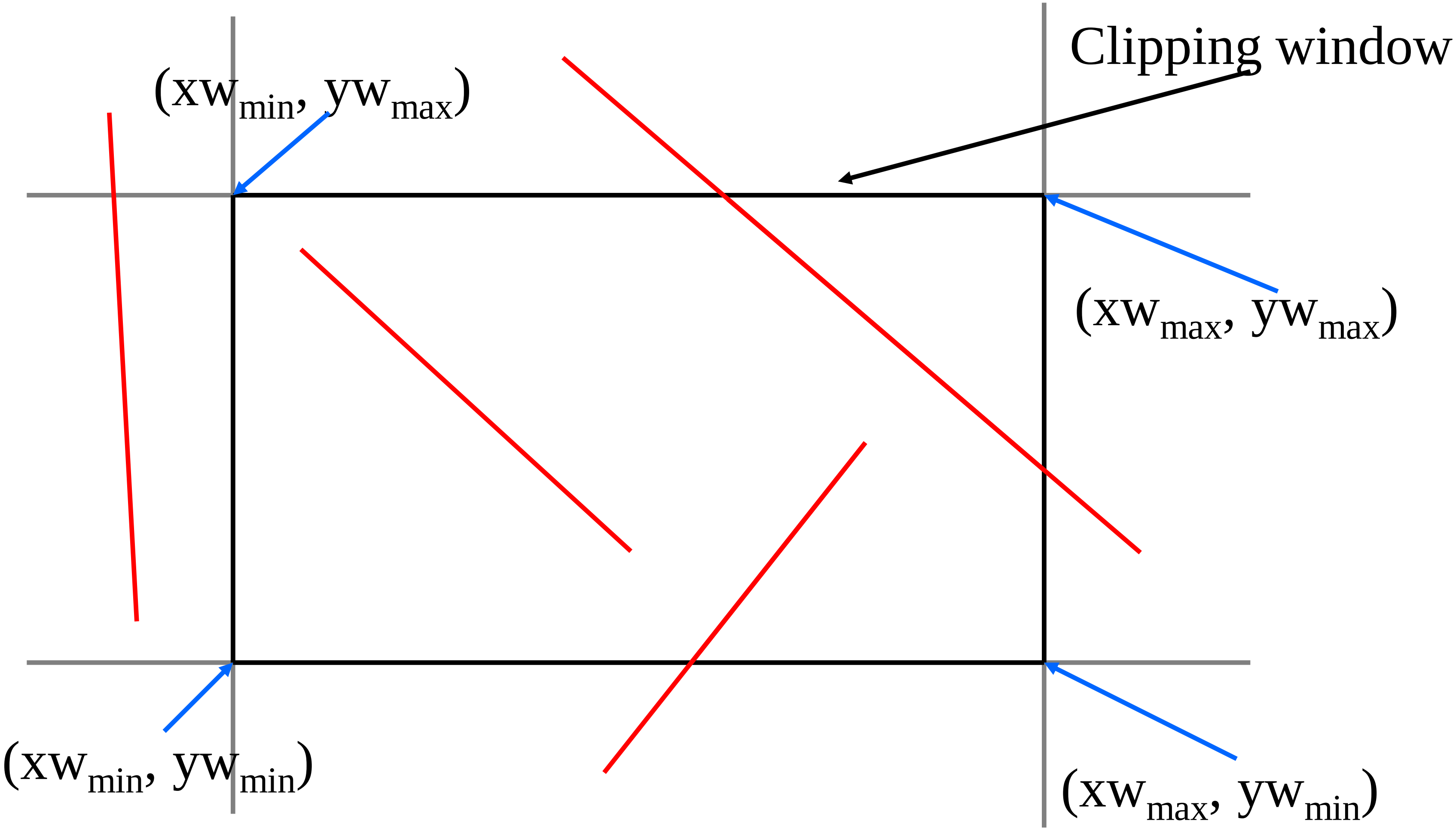
Any other reasons?



2D Viewing



What do we want out of clipping?



What are the basic steps?

1. **Determine if the line needs clipping**

   May be able to trivially accept or reject some lines

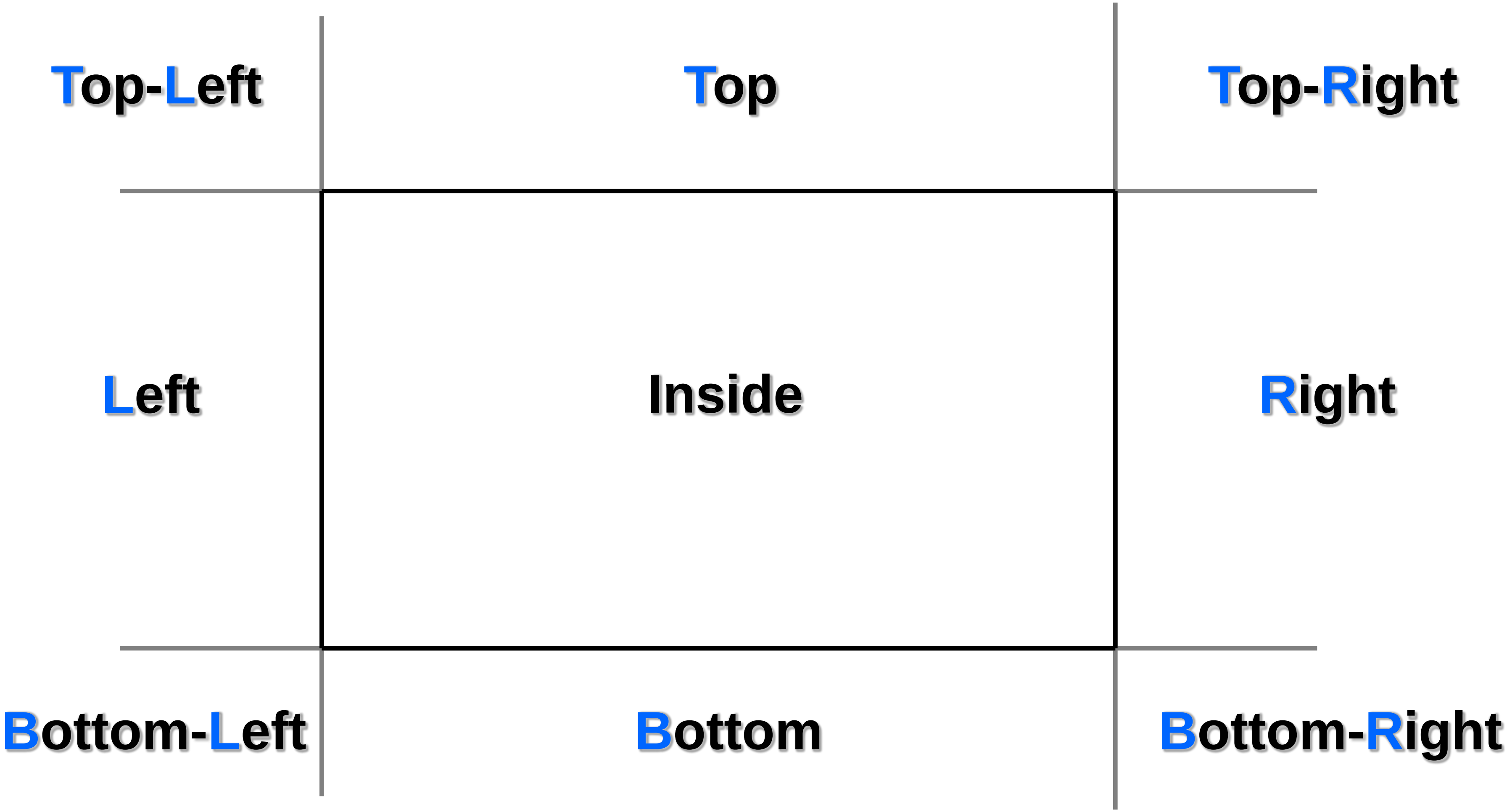
2. **Find intersections of line with viewport**

   We can use $y = mx + b$ to do this

We want to determine which edges of the viewport to test lines against and avoid unnecessary tests.

We'll start by categorizing the regions around the display.

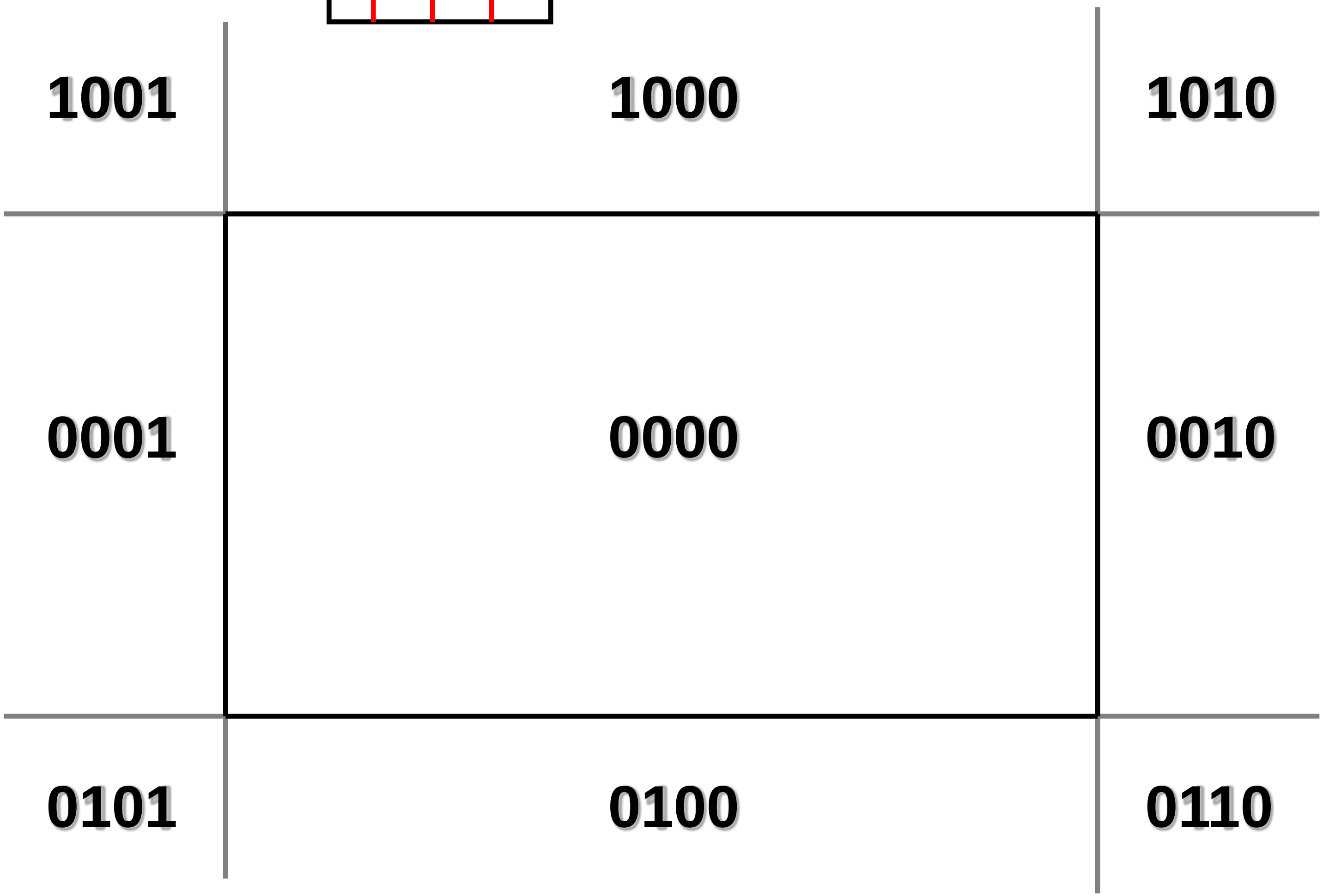
Cohen-Sutherland line clipping



TBRL

Cohen-Sutherland line clipping

Region codes	T	B	R	L
Bit	1	2	3	4



Region coding

How would you decide which region an endpoint is in?

e.g

1- IF $(x < xw_{min})$ && $(y > yw_{max})$

→ Point is at Top-Left

2- IF $(x < xw_{min})$ && $((y < yw_{max})$ && $(y > yw_{min}))$

→ the point is at the Left

3- IF $(x < xw_{min})$ && $(y < yw_{min})$

→ Point is at Bottom-Left

4- IF $((x > xw_{min})$ && $(x < xw_{max})$) && ($y > yw_{max})$

→ Point is at Top

5- IF $((x > xw_{min})$ && $(x < xw_{max}))$ && $((y > yw_{min})$ && $((y < yw_{max})$

→ Point is at Inside

6- IF $((x > xw_{min})$ && $(x < xw_{max}))$ && $(y < yw_{min})$

→ Point is at Bottom

Region coding

7- IF $(x > xw_{max})$ && $(y > yw_{max})$

→ Point is at **Top-Right**

8- IF $(x > xw_{max})$ && $((y > yw_{min})$ && $(y < yw_{max})$

→ Point is at **Right**

9- IF $(x > xw_{max})$ && $(y < yw_{min})$

→ Point is at **Bottom-Right**

Are there cases we can trivially accept or reject?

How would you test for those?

algorithm

1. Assign a region code for each endpoints.
2. If both endpoints have a region code 0000 → trivially accept this line.
3. Else, perform the logical AND operation for both region codes.
   - 3.1 if the result is not 0000 → trivially reject the line.
   - 3.2 else – (result = 0000, need clipping)
     - 3.2.1. Choose an endpoint of the line that is outside the window.
     - 3.2.2. Find the intersection point at the window boundary (base on region code).
     - 3.2.3. Replace endpoint with the intersection point and update the region code.
     - 3.2.4. Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

4. Repeat step 1 for other lines.

How to check for intersection?

if bit 1 = 1 → there is intersection on TOP boundary.

if bit 2 = 1 → .. .. BOTTOM ..

if bit 3 = 1 → .. .. RIGHT ..

If bit 4 = 1 → .. .. LEFT ..

How to find intersection point?

use line equation

intersection with LEFT or RIGHT boundary.

$x = xw_{min}$ (LEFT) $x = xw_{max}$ (RIGHT)

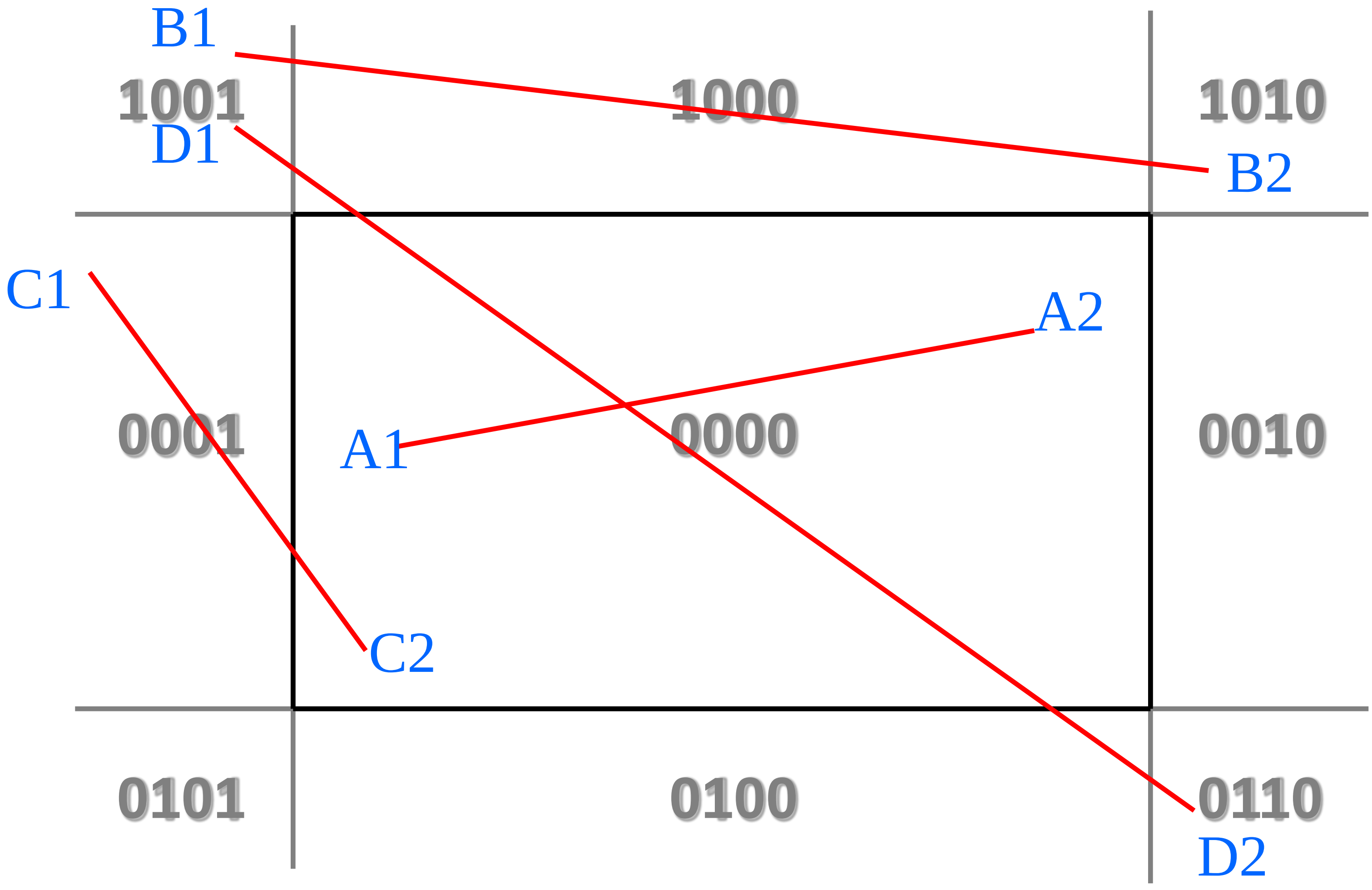
$y = y1 + m(x - x1)$

intersection with BOTTOM or TOP boundary.

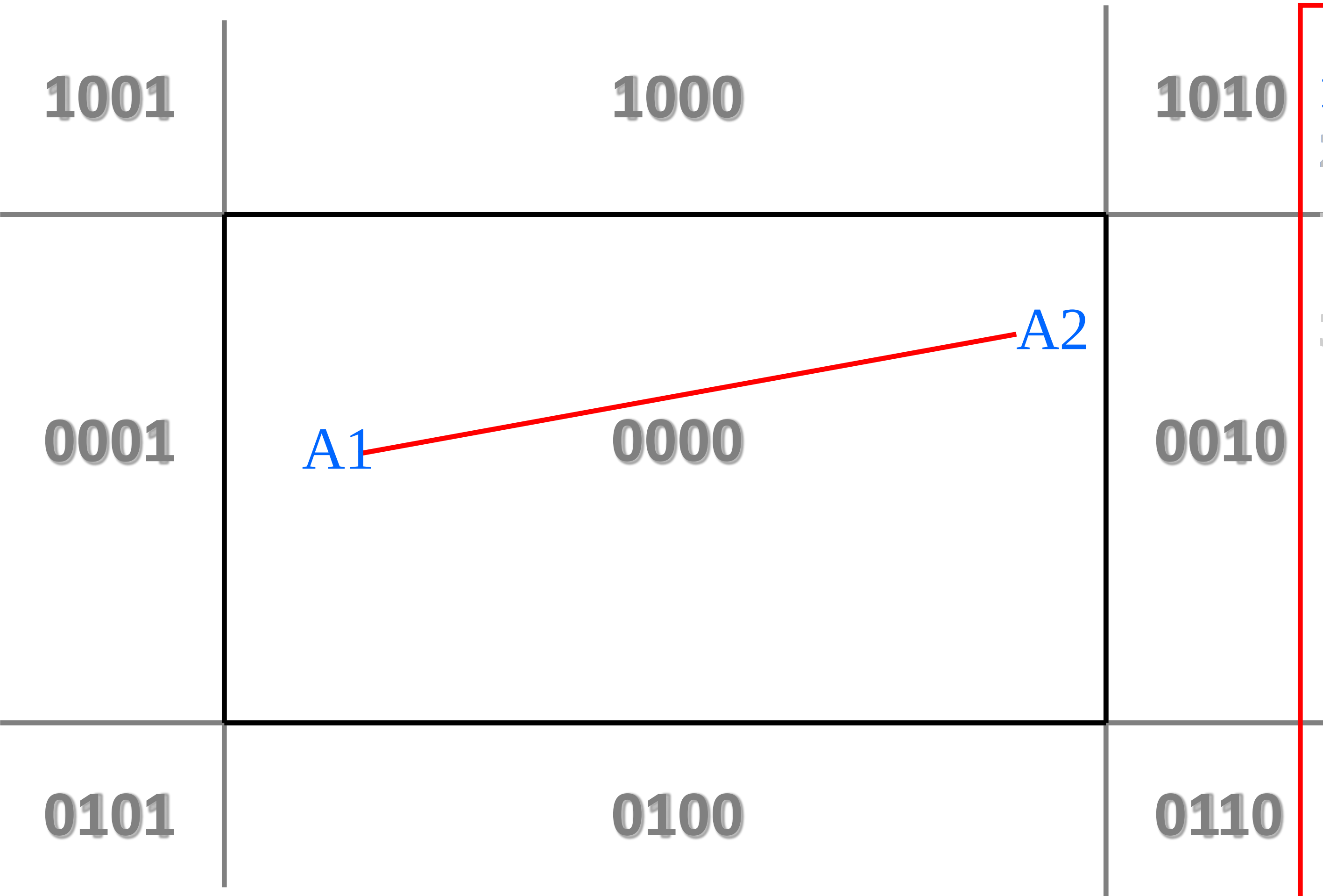
$y = yw_{min}$ (BOTTOM) $y = yw_{max}$ (TOP)

$x = x1 + (y - y1)/m$

Trivial accept & reject



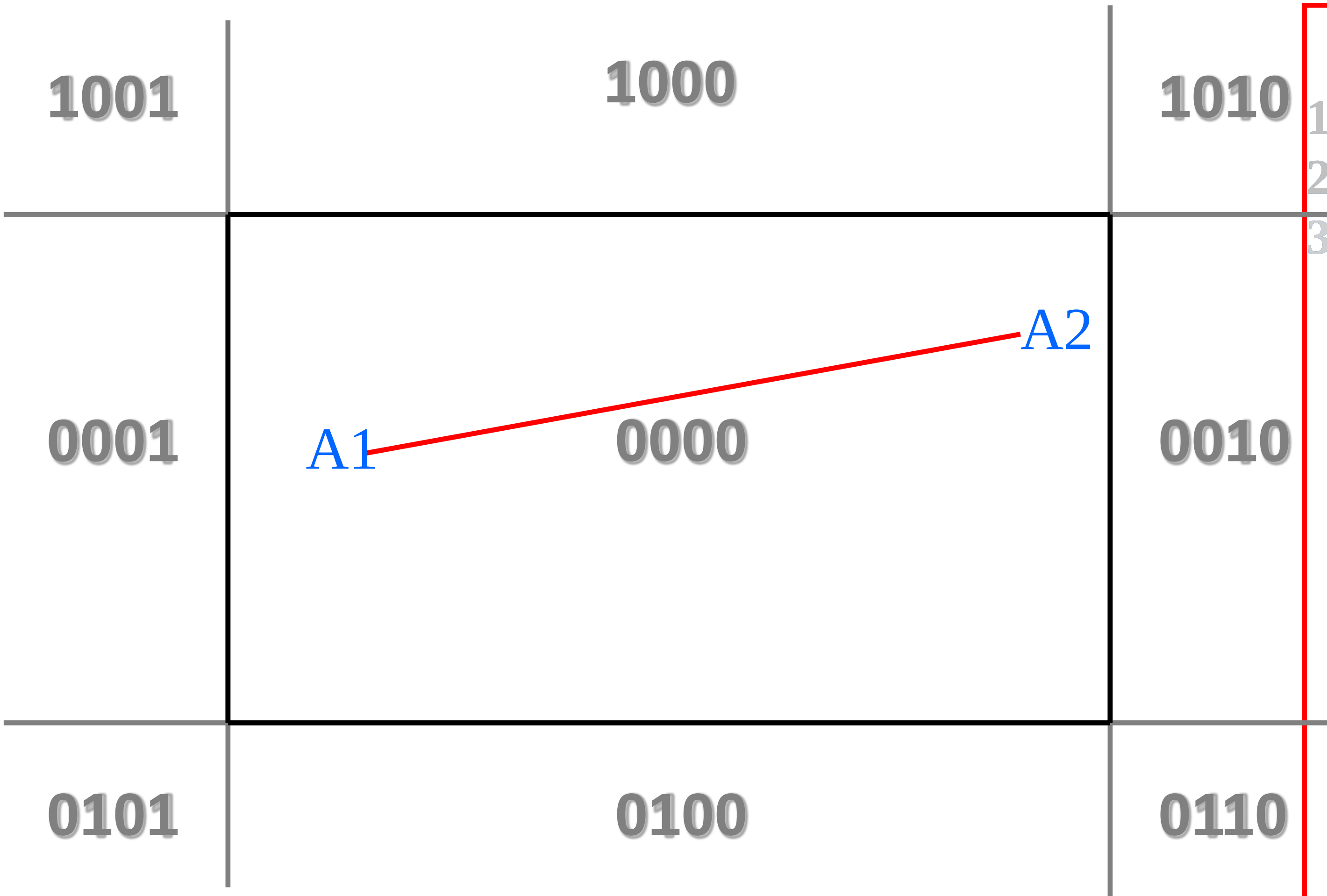
Example



algorithm

1. A1=0000,A2=0000
2. (both 0000) – Yes -> accept & draw
3.
   3.1
   3.2
      3.2.1
      3.2.2
      3.2.3
      3.2.4

Example



algorithm

1. B1=1001,B2=1010
2. (both 0000) – No
3. AND Operation

   B1 → 1001

   B2 → <u>1010</u>

   Result 1000

3.1 (not 0000) – Yes → reject

3.2

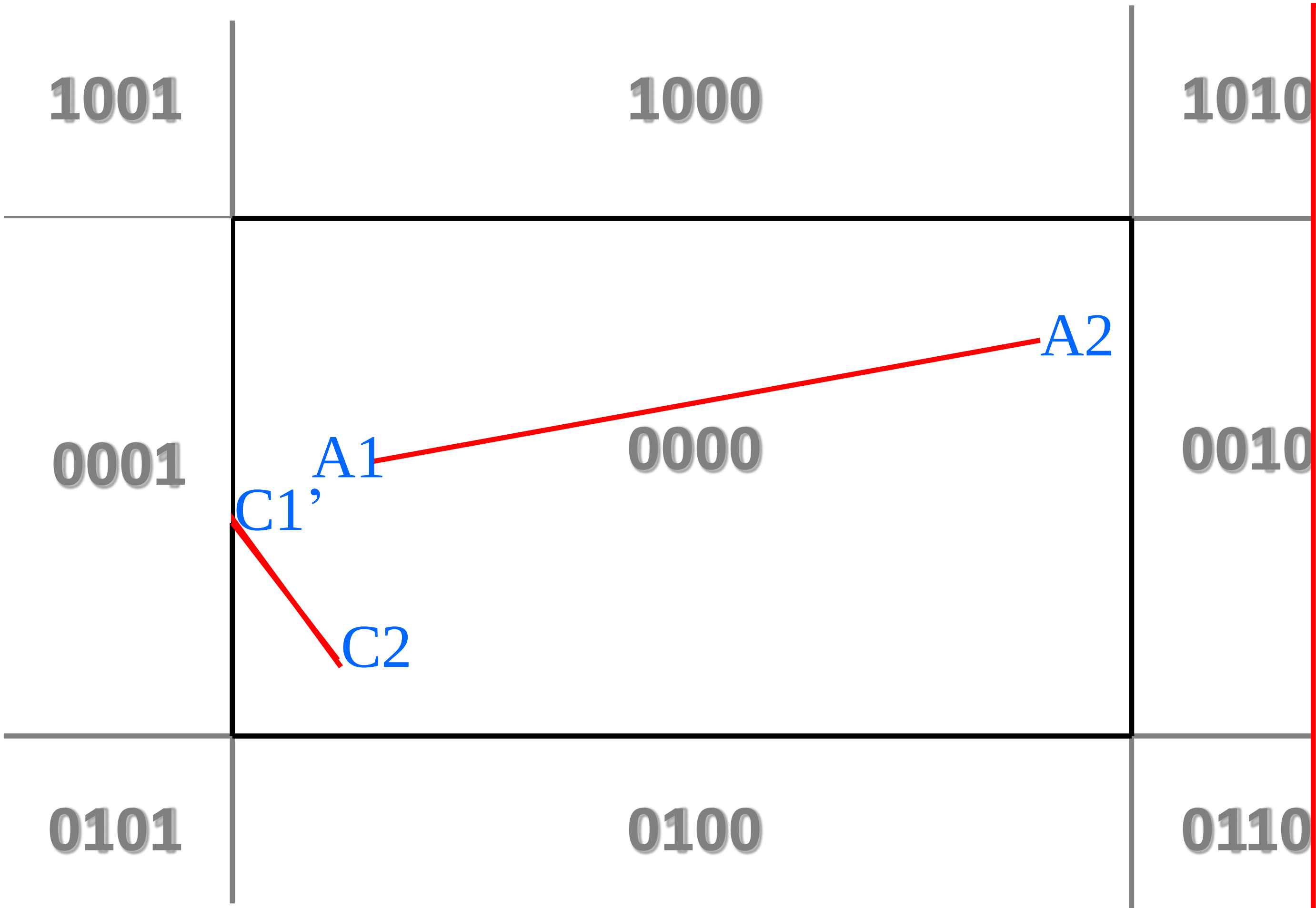
3.2.1

3.2.2

3.2.3

3.2.4

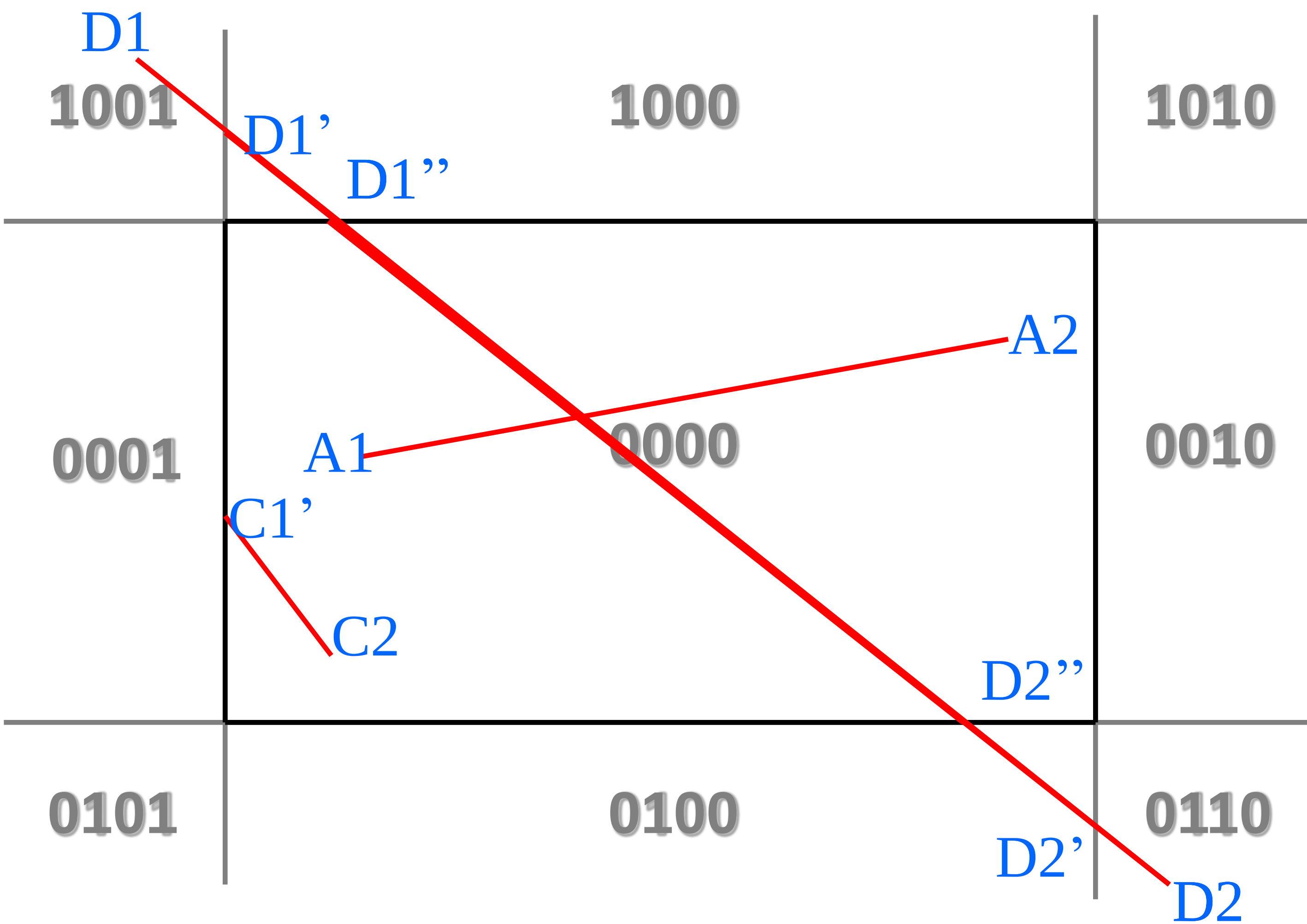
Example



algorithm

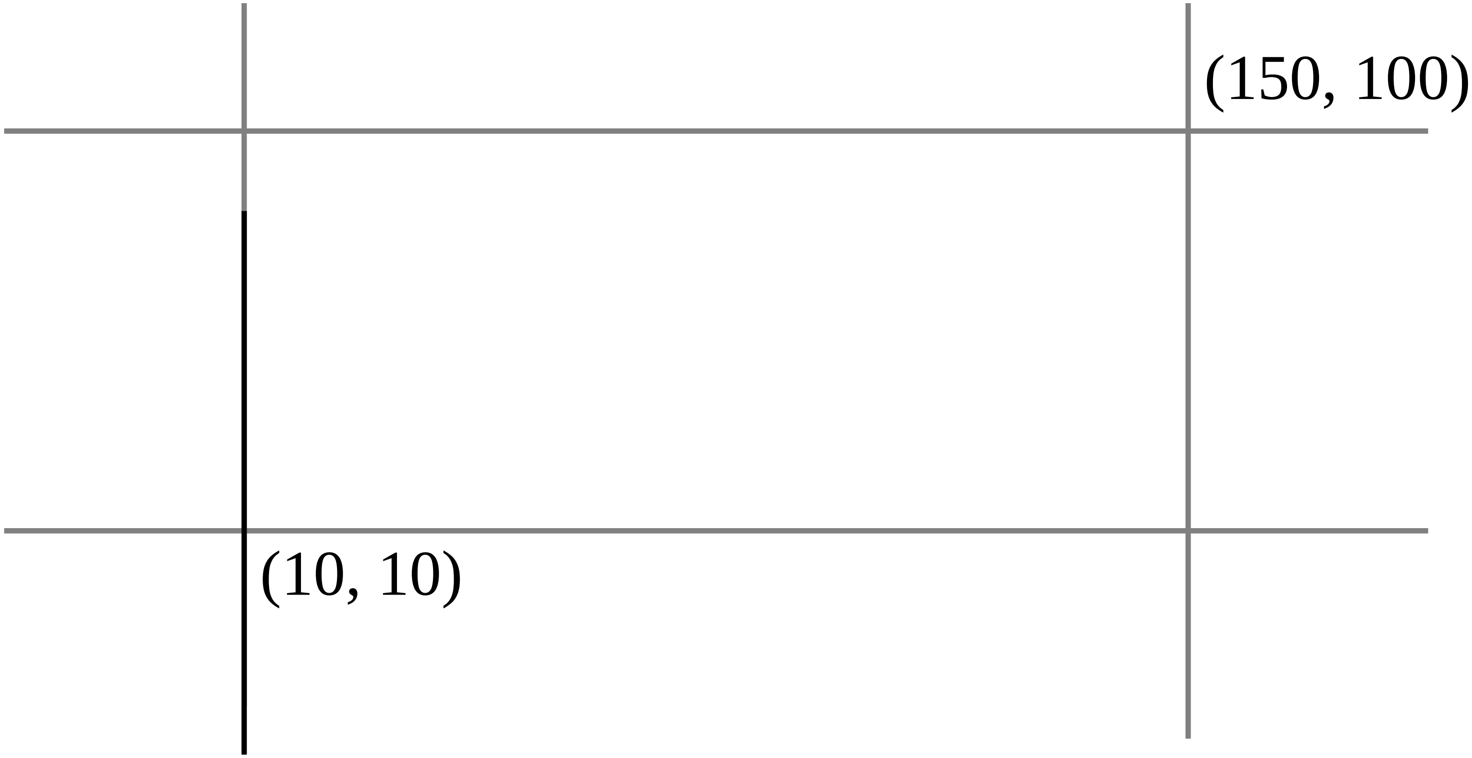
1. C1=0001,C2=0000
2. (both 0000) – Yes -> accept & draw
3.
   3.1
   3.2
      3.2.1
      3.2.2
      3.2.3
      3.2.4

Example



algorithm

Example



Diberi tetingkap ketipan seperti di atas. Sekiranya titik P1 ialah (0, 120) dan titik P2(130, 5) . Dapatkan titik-titik persilangan yang membentuk garisan selepas proses ketipan. Gunakan algoritma Cohen-Sutherland

answer

1. P1=1001, P2=0100

2. (both 0000) – No

2. (both 0000) – yes → ACCEPT & DRAW

3. AND Operation

P1 → 1001

P2 → 0100

Result 0000

Endpoints after clipping

P1" = (22, 100) P2' = 124, 10)

3.1 (not 0000) – no

3.2 (0000) yes

3.2.1choose P1'

3.2.1choose P2'

3.2.2 intersection with TOP boundary

3.2.2 intersection with BOTTOM boundary

m = (5-120)/(130-0) = -0.8846

- x = x1+(y-y1)/m where y = 100;
- x = 130+(100-5)/(-0.8846)=22
- P1"=(22,100)
- x = x1+(y-y1)/m where y = 10;
- x = 130+(10-5)/(-0.8846)=124
- P2"=(124,10)

3.2.3 update region code P1"=0000(TOP)

3.2.3 update region code P2"=0000

3.2.4 repeat step 2

The good and the bad

What's the maximum number of clips for an accepted line?

What's the maximum number of clips for a rejected line?

Good:

- **Easy to implement**
- **Early accept/reject tests**

Bad:

- **Slow for many clipped lines**

Liang-Barsky Line Clipping

- **Based on parametric equation of a line:**

$$x = x_1 + u.\triangle x \qquad 0 \le u \le 1$$
$$y = y_1 + u.\triangle y$$

- **Similarly, the clipping window is represented by:**

$$xw_{min} \le x_1 + u.\triangle x \le xw_{max}$$
$$yw_{min} \le y_1 + u.\triangle y \le yw_{max}$$

i.e.
$$- u.\triangle x \le x1 - xw_{min}$$
$$u.\triangle x \le xw_{max} - x1$$
$$- u.\triangle y \le y1 - yw_{min}$$
$$u.\triangle y \le yw_{max} - y1$$

… or, $u\, p_k \le q_k$ $\qquad k = 1, 2, 3, 4$

where,

$p_1 = - \triangle x$, $\qquad q_1 = x_1 - xw_{min}$
$p_2 = \triangle x$, $\qquad q_2 = xw_{max} - x_1$
$p_3 = - \triangle y$, $\qquad q_3 = y_1 - yw_{min}$
$p_4 = \triangle y$, $\qquad q_4 = yw_{max} - y_1$

Liang-Barsky (continued)

- Clipped line will be:

$$x_1' = x_1 + u_1 . \triangle x; \qquad u_1 \geq 0$$
$$y_1' = y_1 + u_1 . \triangle y;$$

$$x_2' = x_1 + u_2 . \triangle x; \qquad u_2 \leq 1$$
$$y_2' = y_1 + u_2 . \triangle y;$$

- Reject line with $p_k = 0$ and $q_k < 0$. (p_k=0 i.e. line is parallel to clip boundary & q_k<0 i.e. completely outside the clip window)
- Calculate u_k

$$\boxed{u_k = q_k / p_k}$$

Liang-Barsky (continued)

- u_1 : maximum value between 0 and u (for $p_k < 0$), where
  starting value for u_1 is 0 ($u_1 = 0$)
- u_2 : minimum value between u and 1 (for $p_k > 0$), where
  starting value for u_2 is 1 ($u_2 = 1$)

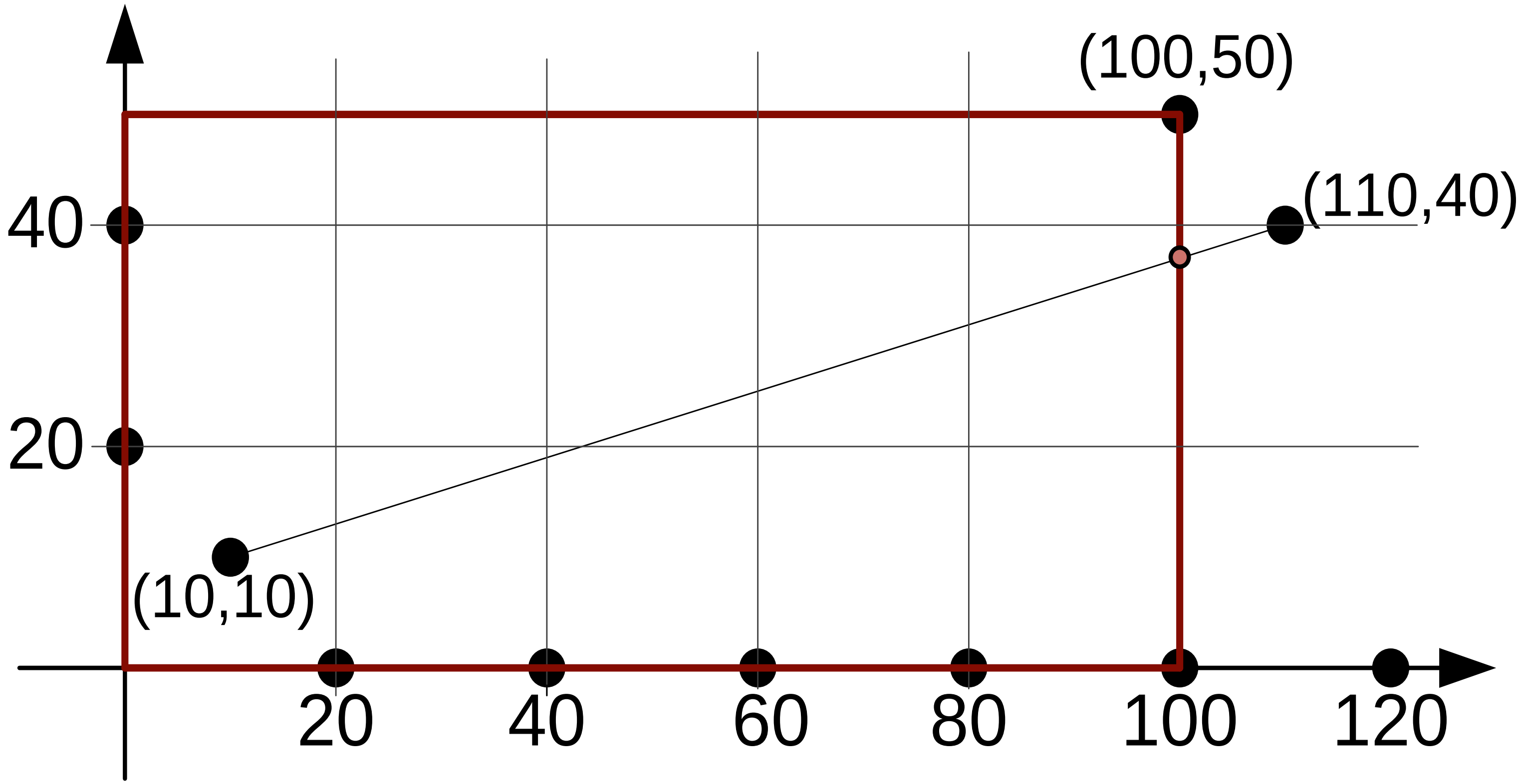
- Consider our previous example where:

  $xw_{min} = 0$, $xw_{max} = 100$

  $yw_{min} = 0$, $yw_{max} = 50$

And the line we want to clip connects $P_1(10, 10)$ and $P_2(110, 40)$

Example



Liang-Barsky (example)

- **Lets construct a table:**

k	p_k	q_k	u_k
1	$-\triangle x$ $= -(110-10)$ $= -100$	$x_1 - xw_{min}$ $= 10-0 = 10$	
2	$\triangle x$ $=110-10=100$	$xw_{max} - x_1$ $= 100 - 10 = 90$	
3	$-\triangle y$ $= -(40-10)$ $=-30$	$y_1 - yw_{min}$ $= 10-0 = 10$	
4	$\triangle y$ $= 40-10=30$	$yw_{max} - y_1$ $= 50 - 10 = 40$	

Liang-Barsky (example)

- **Lets construct a table:**

k	p_k	q_k	u_k
1	$-\Delta x$ $= -(110-10)$ $= -100$	$x_1 - xw_{min}$ $= 10-0 = 10$	
2	Δx $=110-10=100$	$xw_{max} - x_1$ $= 100 - 10 = 90$	
3	$-\Delta y$ $= -(40-10)$ $=-30$	$y_1 - yw_{min}$ $= 10-0 = 10$	
4	Δy $= 40-10=30$	$yw_{max} - y_1$ $= 50 - 10 = 40$	

u_1

u_1

Since $p_k < 0$

Liang-Barsky (example)

- u_1 : maximum value between 0 and u (for $p_k < 0$)!

u_1

u_1

k	p_k	q_k	u_k
1	$-\triangle x$ $= -(110-10)$ $= -100$	$x_1 - xw_{min}$ $= 10-0 = 10$	$u=10/(-100)$ $=-1/10$
2	$\triangle x$ $=110-10=100$	$xw_{max} - x_1$ $= 100 - 10 = 90$	
3	$-\triangle y$ $= -(40-10)$ $=-30$	$y_1 - yw_{min}$ $= 10-0 = 10$	$u=10/(-30)$ $=-1/3$
4	$\triangle y$ $= 40-10=30$	$yw_{max} - y_1$ $= 50 - 10 = 40$	

We opt $u_1 =0$,

Liang-Barsky (example)

- u_2 : minimum value between u (for $p_k > 0$) and 1

k	p_k	q_k	u_k
1	$-\triangle x$ $= -(110-10)$ $= -100$	$x_1 - xw_{min}$ $= 10-0 = 10$	$u=10/(-100)$ $=-1/10$
2	$\triangle x$ $=110-10=100$	$xw_{max} - x_1$ $= 100 - 10 = 90$	
3	$-\triangle y$ $= -(40-10)$ $=-30$	$y_1 - yw_{min}$ $= 10-0 = 10$	$u=10/(-30)$ $=-1/3$
4	$\triangle y$ $= 40-10=30$	$yw_{max} - y_1$ $= 50 - 10 = 40$	

u_2

u_2

We opt $u_1 = 0$,

Since $p_k > 0$

Liang-Barsky (example)

- u_2 : minimum value between u (for $p_k > 0$) and 1

k	p_k	q_k	u_k
1	$-\triangle x$ $= -(110-10)$ $= -100$	$x_1 - xw_{min}$ $= 10-0 = 10$	$u=10/(-100)$ $=-1/10$
2	$\triangle x$ $=110-10=100$	$xw_{max} - x_1$ $= 100 - 10 = 90$	$u=90/100$ $=9/10$
3	$-\triangle y$ $= -(40-10)$ $=-30$	$y_1 - yw_{min}$ $= 10-0 = 10$	$u=10/(-30)$ $=-1/3$
4	$\triangle y$ $= 40-10=30$	$yw_{max} - y_1$ $= 50 - 10 = 40$	$u=40/30)$ $=4/3$

u_2

u_2

We opt $u_1 = 0$,

We opt $u_2 = 0.9$

Liang-Barsky (example)

- If $u_1 > u_2$ then reject line (completely outside clipping window!)
- Clipped line will be:

$x_1' = x_1 + u_1 . \triangle x$ $\quad (u_1 = 0)$

$= 10 + 0.(100) = 10$

$y_1' = y_1 + u_1 . \triangle y$

$= 10 + 0.(30) = 10$

$x_2' = x_1 + u_2 . \triangle x$ $\quad (u_2 = 9/10)$

$= 10 + 0.9(100) = 100$

$y_2' = y_1 + u_2 . \triangle y$

$= 10 + 0.9(30) = 37$

**Homework: Use different values of xw_{min}, xw_{max}, yw_{min}, yw_{max}, P_1 and P_2 for exercise.

Algorithm

1. Initial value : u1 = 0, u2 = 1
2. For k = 1, 2, 3, 4;
   - 2.1 calculate P_k and q_k
   - 2.2 calculate $r_k = q_k / P_k$
   - 2.2 if ($P_k < 0$) →find u1 (if (r_k>u1), u1=r_k)
   - 2.3 if ($P_k > 0$) →find u2 (if (r_k<u2), u2=r_k)
   - 2.4 if ($P_k = 0$) and ($q_k < 0$) ;

     reject the line; goto step 5
3. If (u1> u2) ; reject the line; goto step 5
4. Find the clipped line

   $x1' = x1 + u1.\ \Delta x$

   $y1' = y1 + u1.\ \Delta y$

   $x2' = x1 + u2.\ \Delta x$

   $y2' = y1 + u2.\ \Delta y$
5. Repeat step 1 – 4 for other lines.