Clipping

Grafika Komputer

Murinto, M.Kom

Clipping

- Prosedur yang mendefinisikan bagian gambar, baik di dalam maupun di luar suatu bidang tertentu di sebut dengan algoritma clipping/clipping
- Pada transformasi viewing, perlu ditampilkan bagian gambar yang terdapat dalam window. Semua yang berada di luar window akan dibuang.
- Clipping dapat diterapkan pada world coordinate, sehingga hanya isi yang berada dalam window dipetakan ke device coordinat.

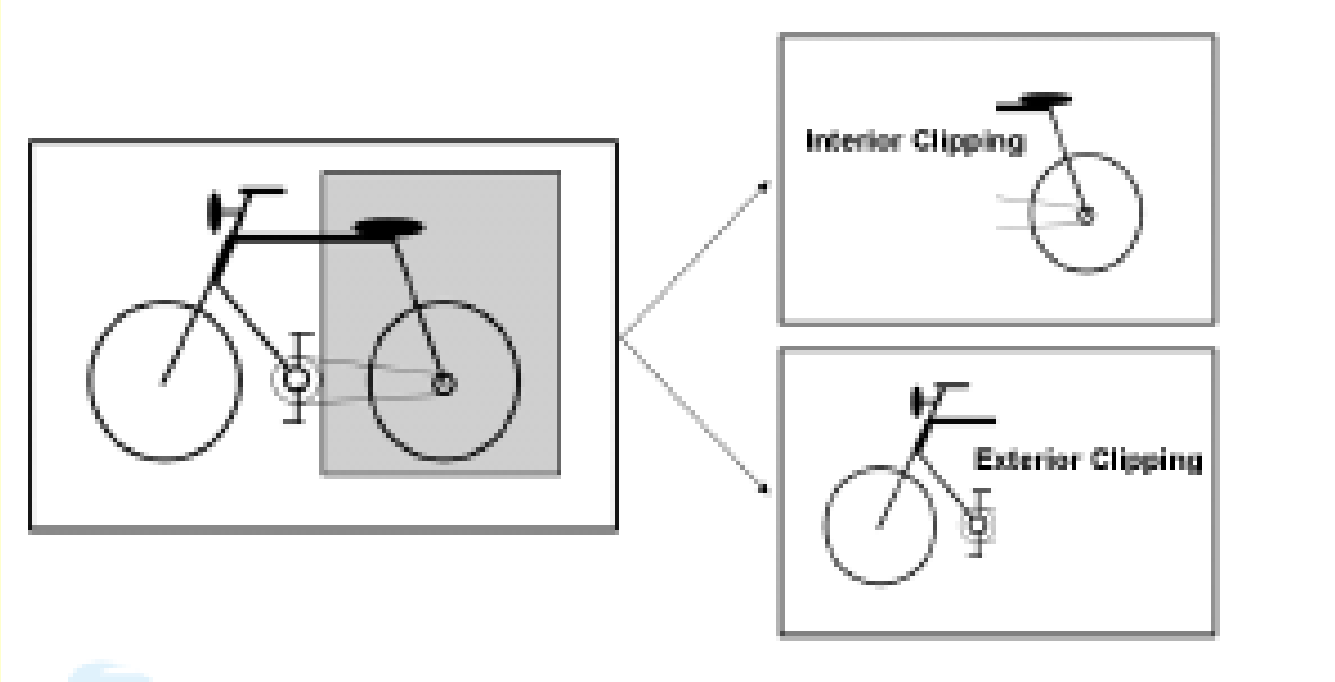
Istilah Cliping

- Istilah Kliping (*Clipping)* = kumpulan guntingan koran
- *Clipping* = memotong objek dengan bentuk tertentu.
- Sarana pemotong objek ➔ *clipping window*
- Dalam konteks grafika komputer, untuk melakukan *clipping*, kita lebih dulu harus menentukan bentuk *window* dan baru kemudian menentukan hanya objek yang terdapat di dalam *window* tersebut yang akan ditampilkan.

Algoritma Clipping

- Algoritma clipping digunakatn untuk berbagai macam primitif, yaitu :
  - Clipping titik
  - Clipping garis
  - Clipping area (poligon)
  - Clipping kurva
  - Clipping teks

Contoh Cliping



CLIPPING TITIK

- Pada Clipp window yang mempunyai bentuk persegi empat dengan posisi standar, titik P(x,y) disimpan untuk ditampilkan bila :

$$xw_{\min} \le x \le xw_{\max} \qquad yw_{\min} \le y \le yw_{\max}$$

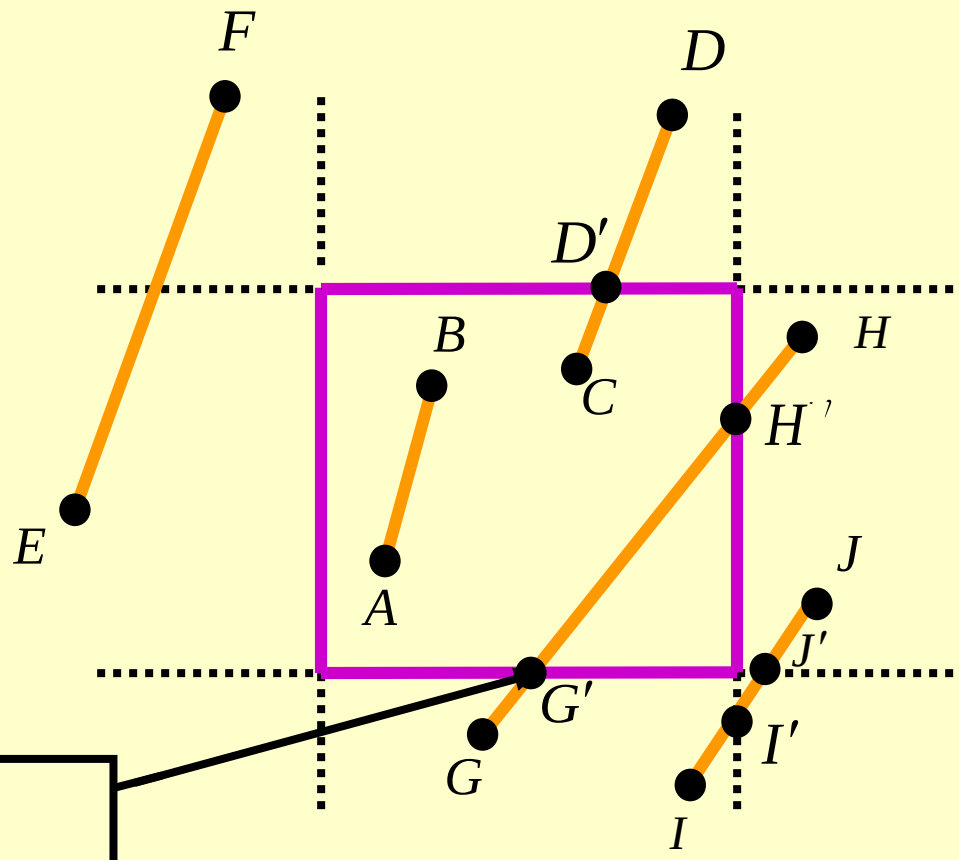
- dimana batas clip window dapat berada di dalam batas world coordinate atau viewport coordinate.

Clipping Garis

- Prosedur clipping untuk garis dapat dijelaskan sebagai berikut :

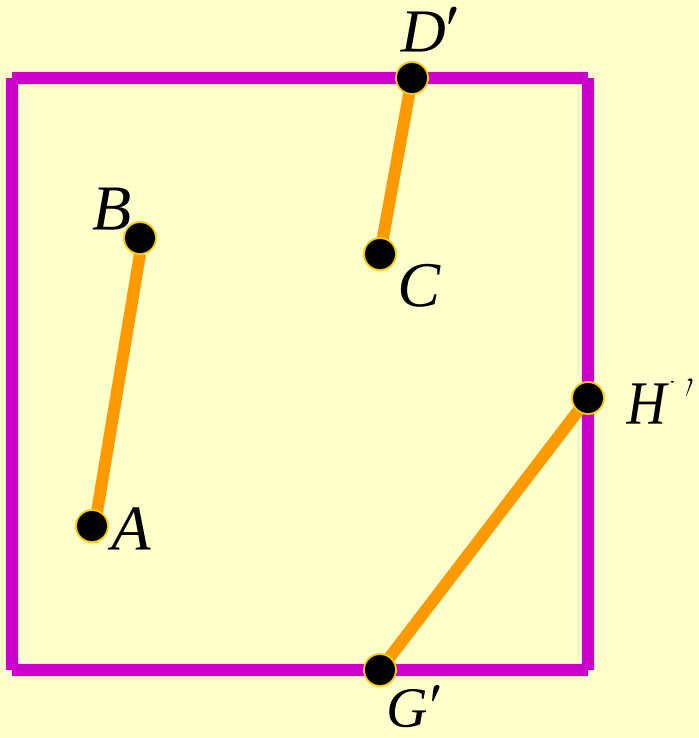
  Clipping garis diproses dengan inside-outside tes dengan memeriksa endpoint dari garis. Garis yang mempunyai kedua endpoint di dalam batas clipping, maka garis tersebut disimpan. Sedang bila kedua endpoint tidak berada di dalam, maka garis tersebut berada di luar window. Semua garis lain yang memotong satu atau lebih batas clipping memerlukan algoritma clipping yang dapat mengidentifkasi dengan efisien bahwa garis di luar batas clipping.

Clipping Garis



clip
Persegi empat

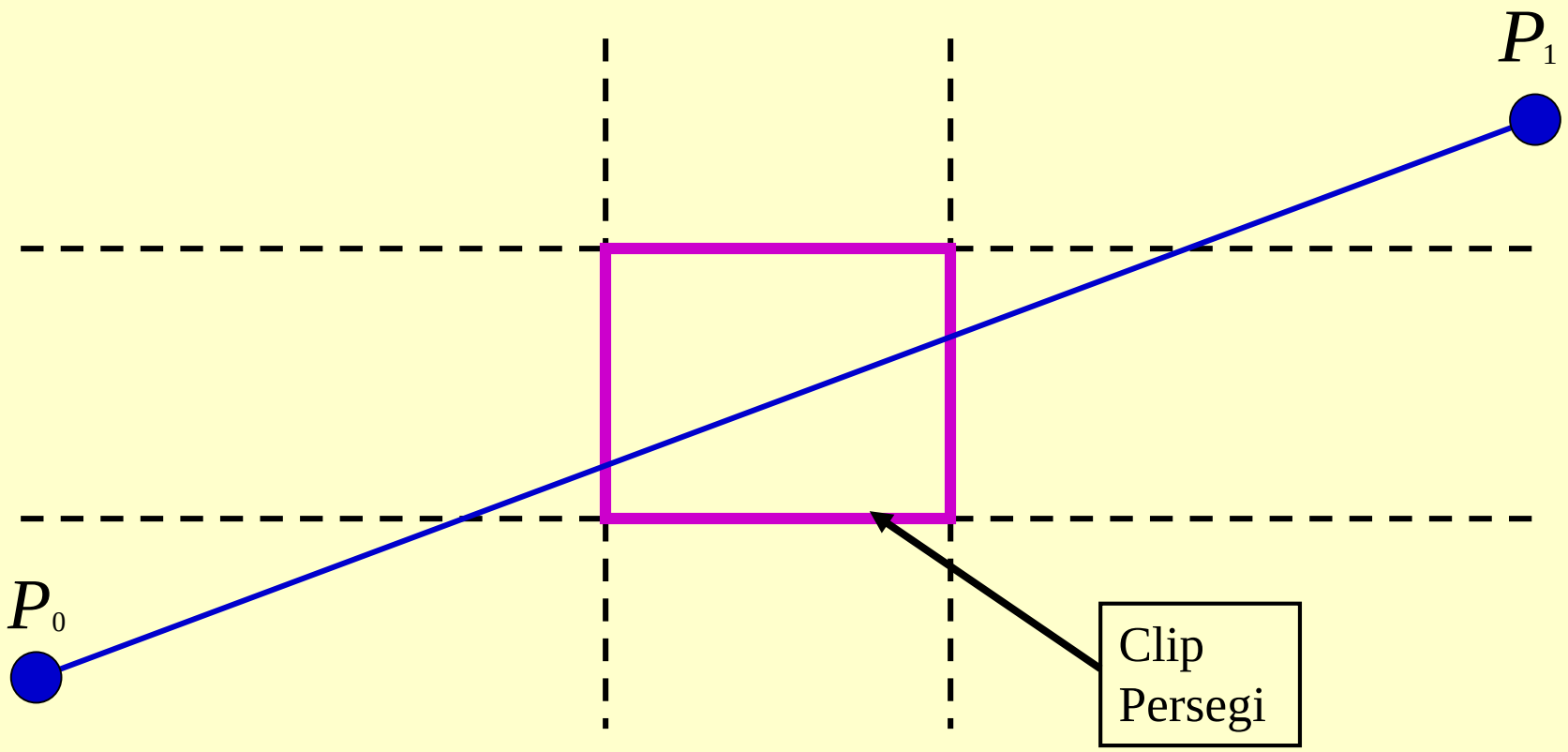
Clipping Garis



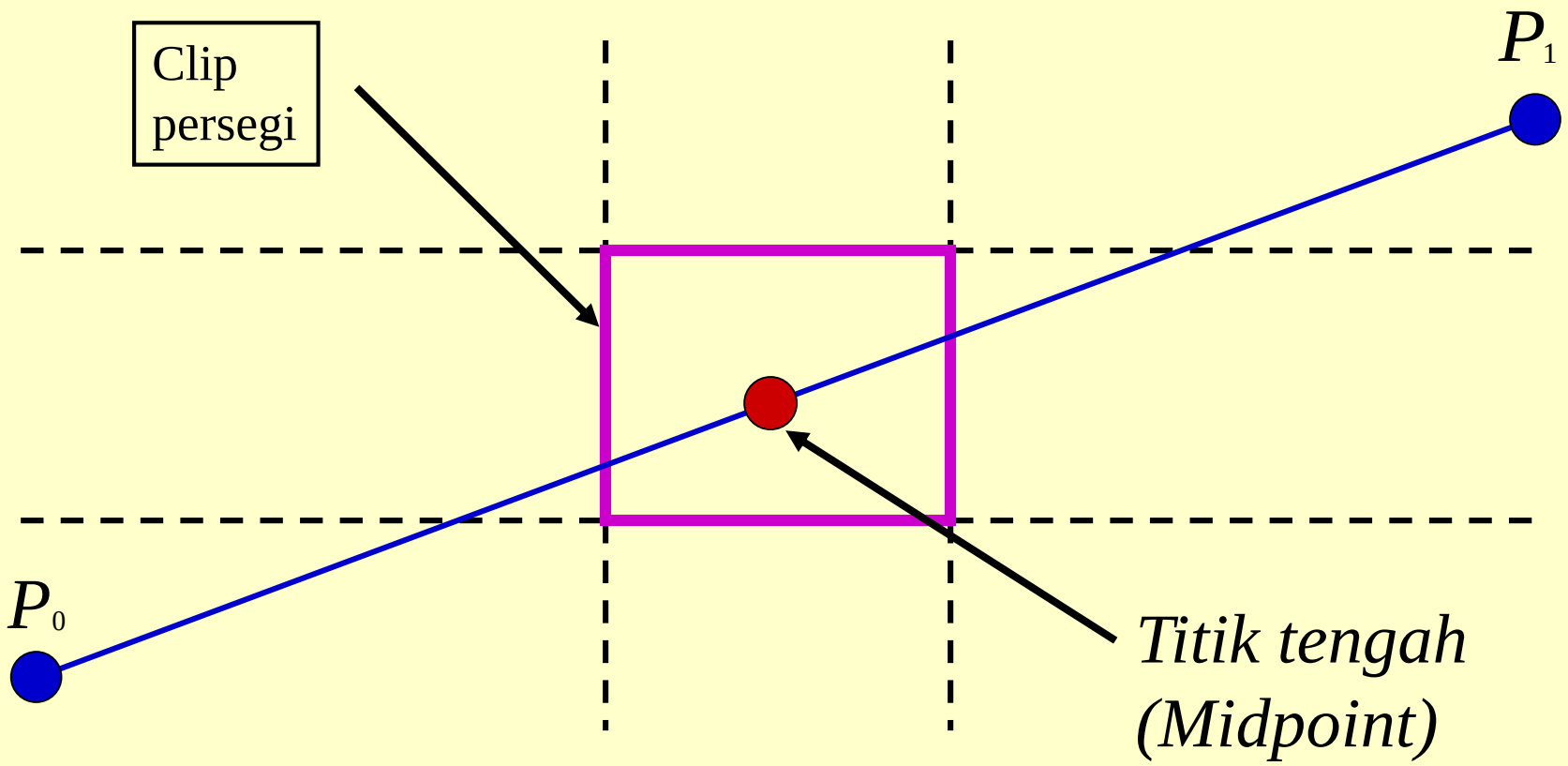
Algoritma

- Recursive Subdivision (membagi garis pada titik tengah)
- Bagus untuk binary processing
- Bounded number (10 atau 12) dari step (melalui ukuran pixel)

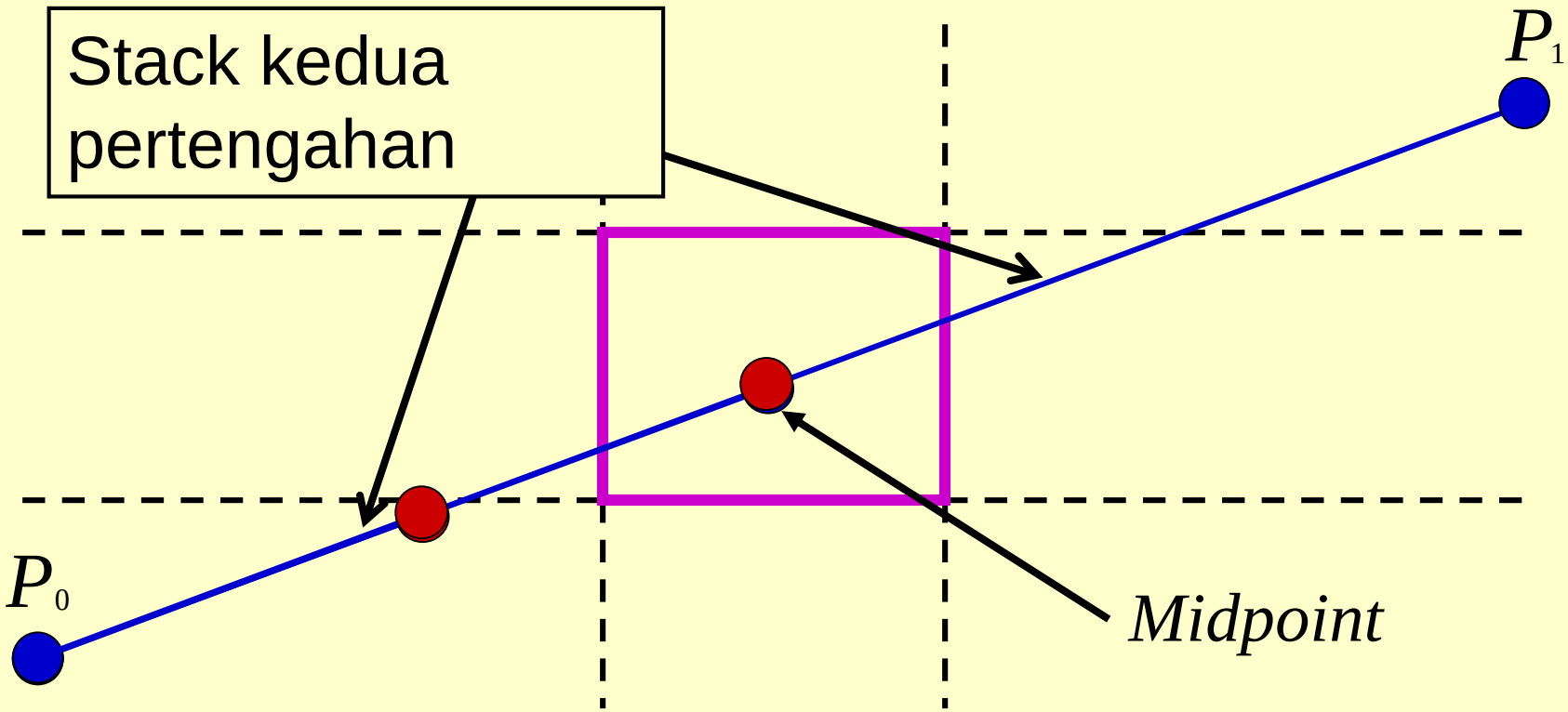
Recursive Subdivision Clipping



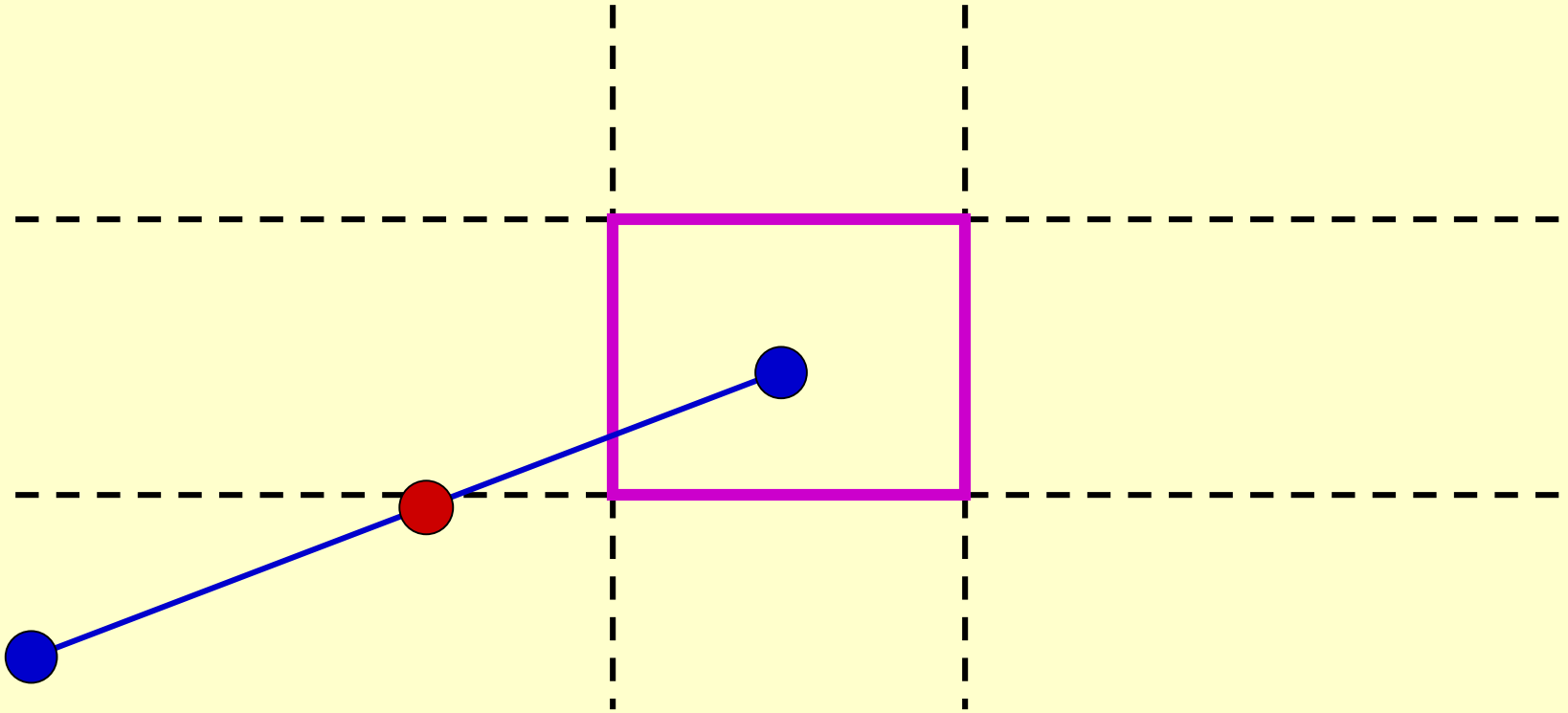
Recursive Subdivision Clipping



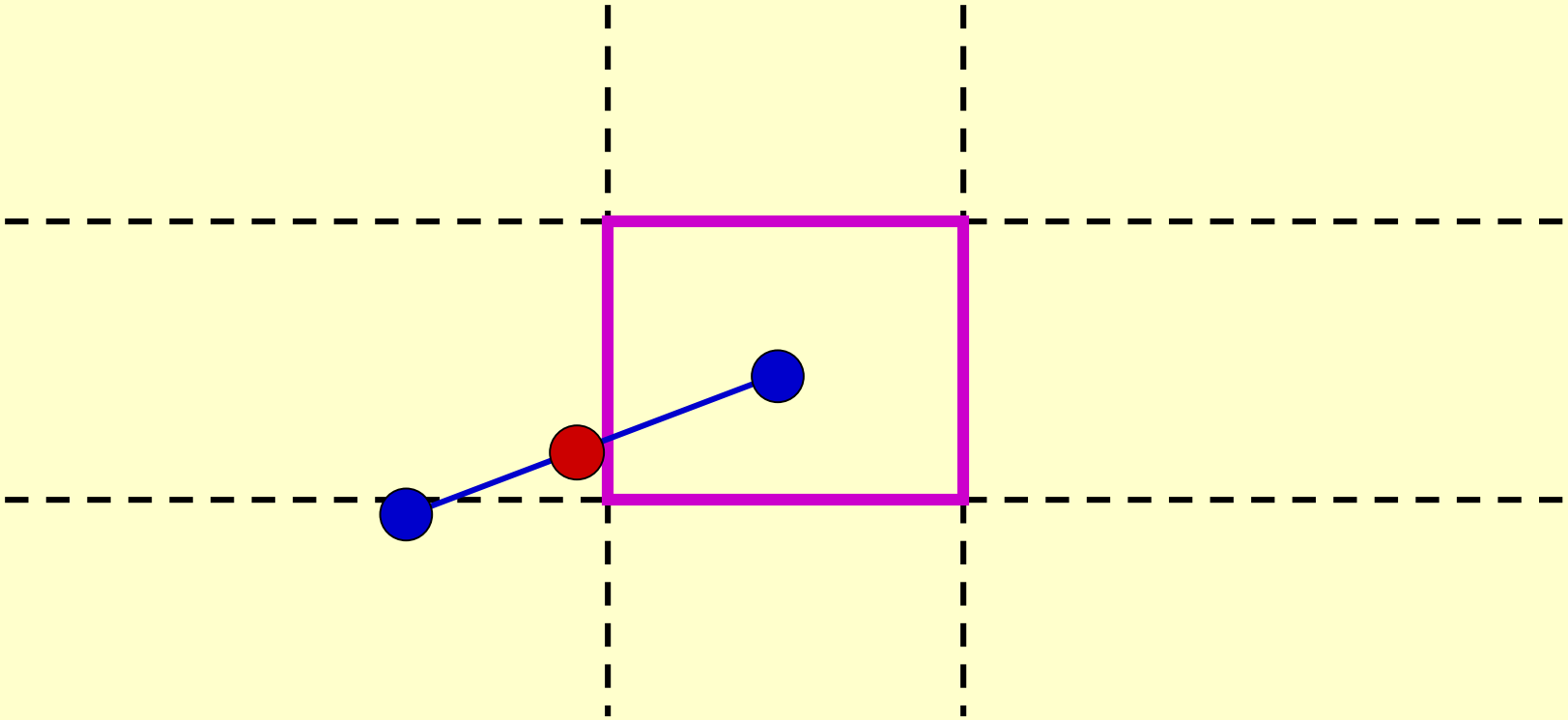
Recursive Subdivision Level 1



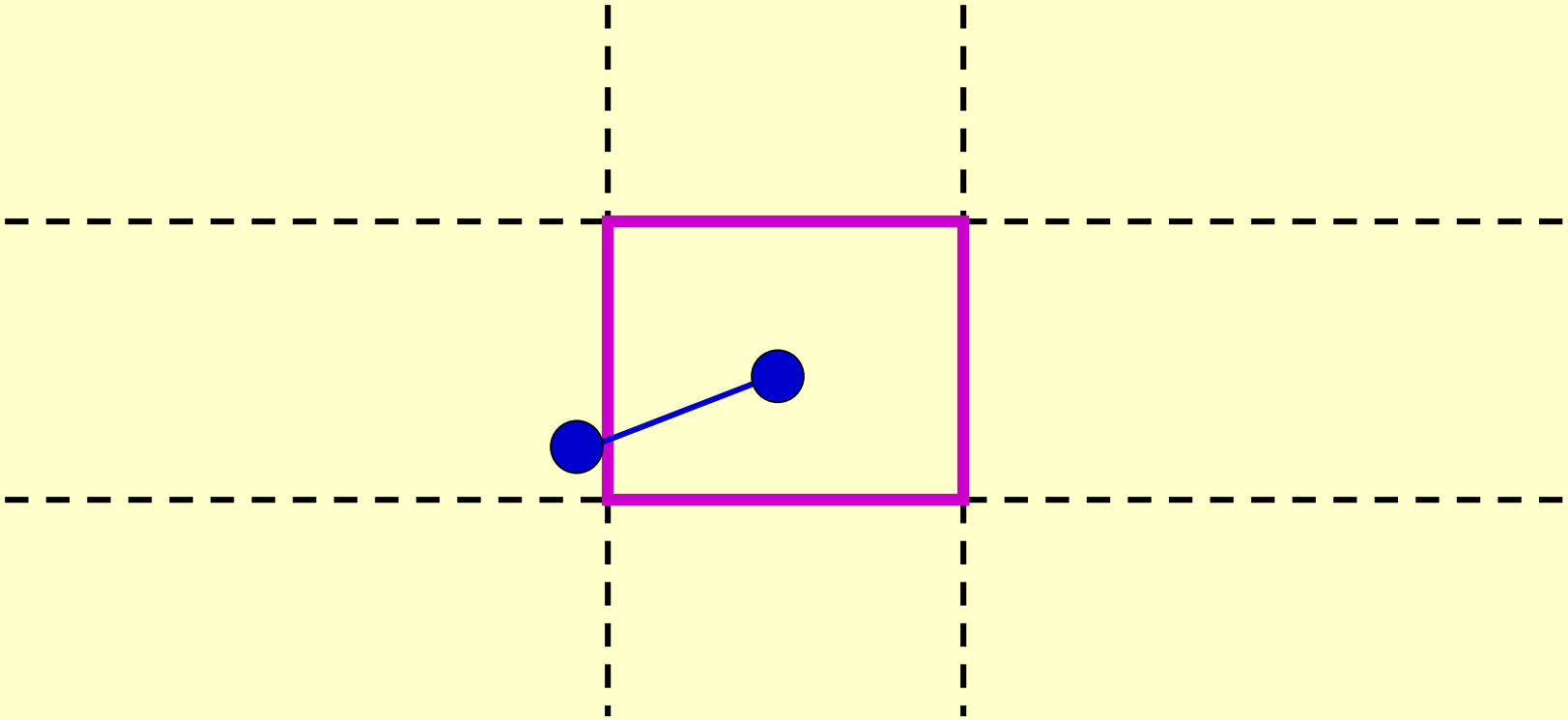
Recursive Subdivision Level 2



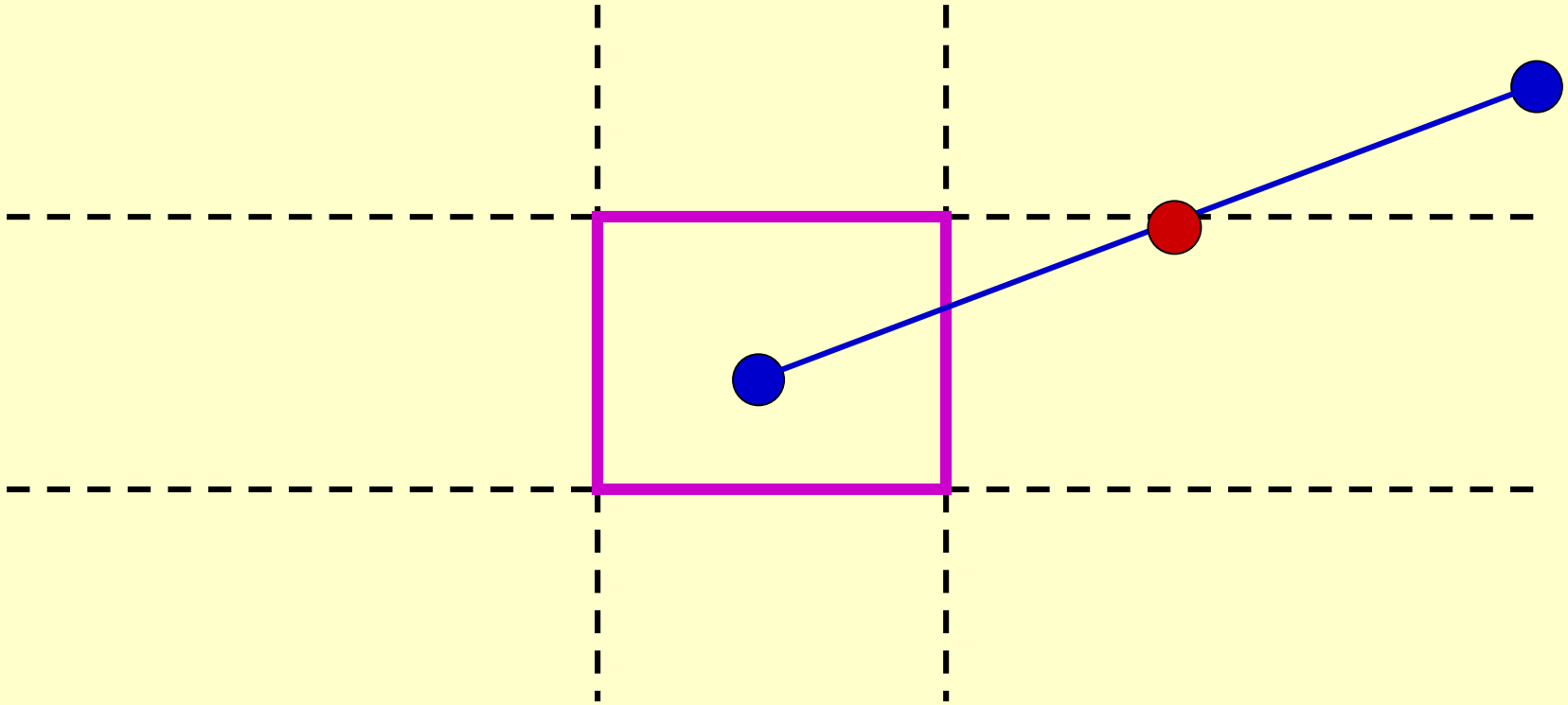
Recursive Subdivision Level 3



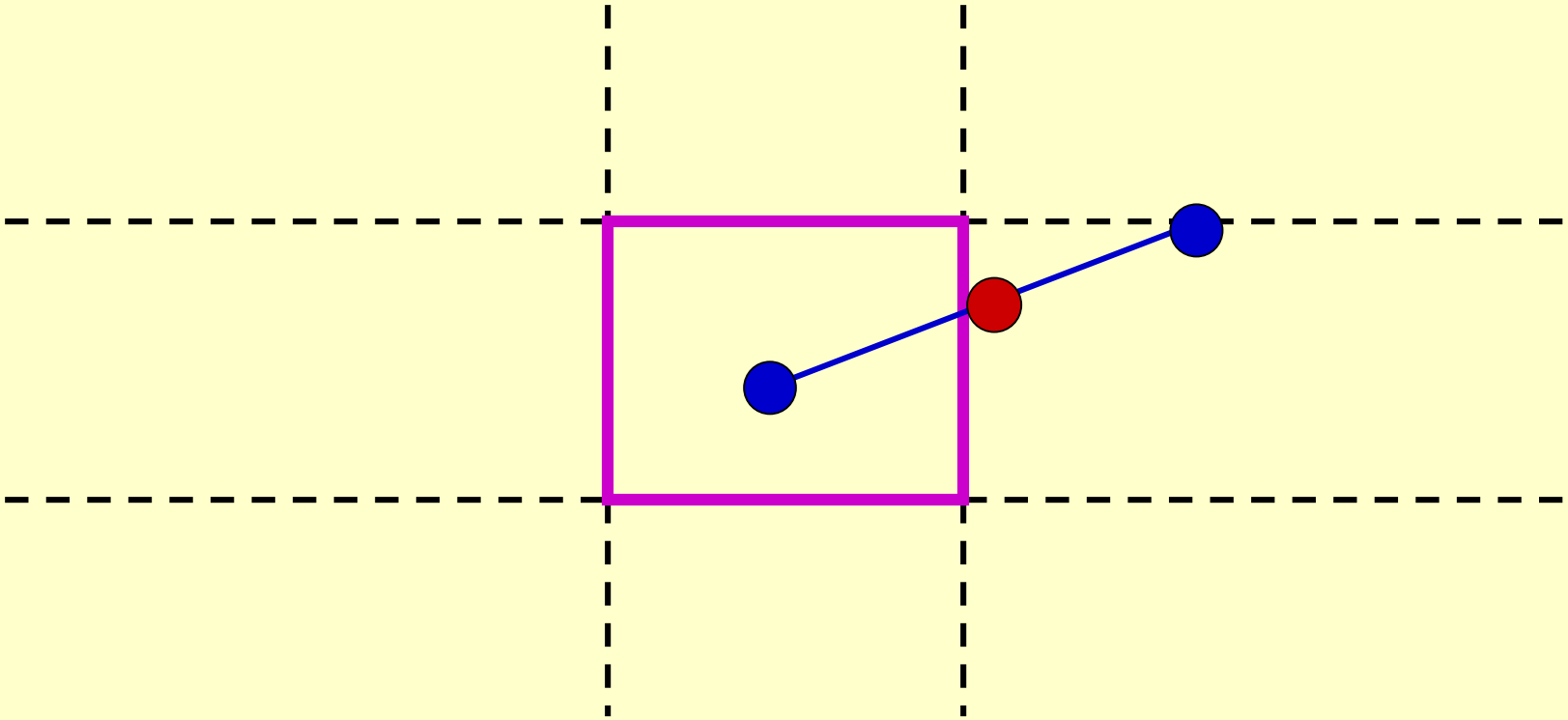
Recursive Subdivision Level 3



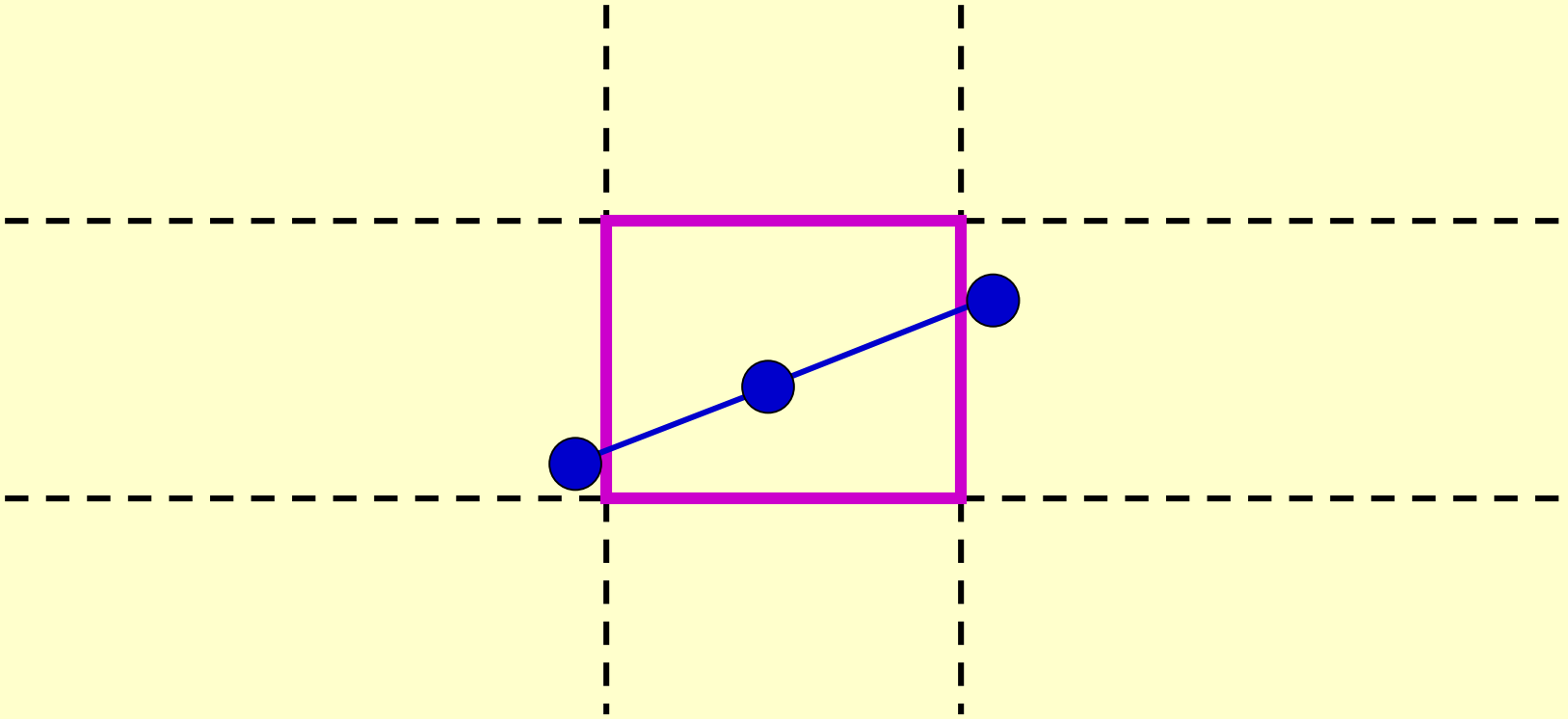
Recursive Subdivision Level 2



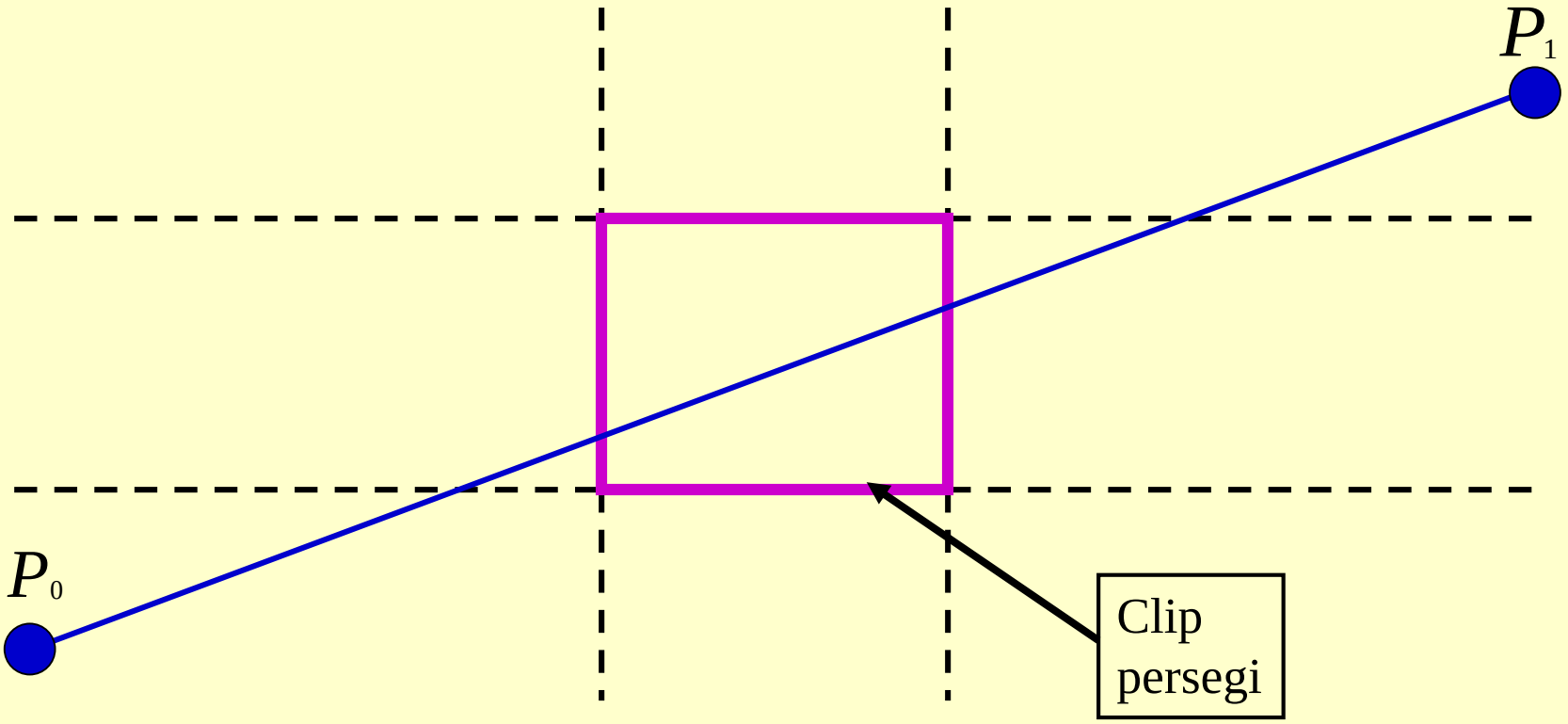
Recursive Subdivision Level 3



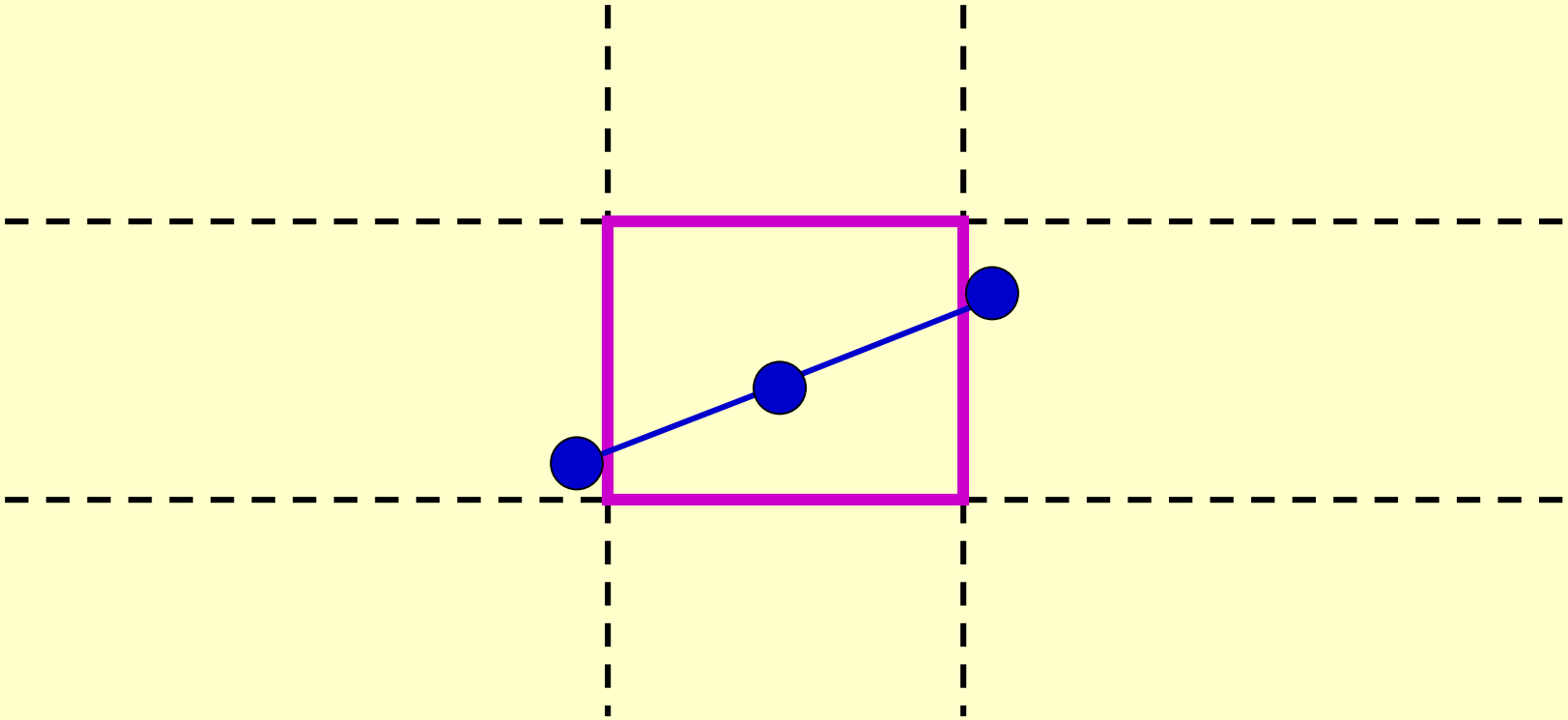
Recursive Subdivision Level 3



Recursive Subdivision Clipping



Akurasi sampai 3 Binary Digits



Recursive Subdivision

- Algoritma Kovergensinya Linear
- Menghitung 1 binary digit tiap satu loop
- Secara alami bekerja dengan *shift* register
- stabil

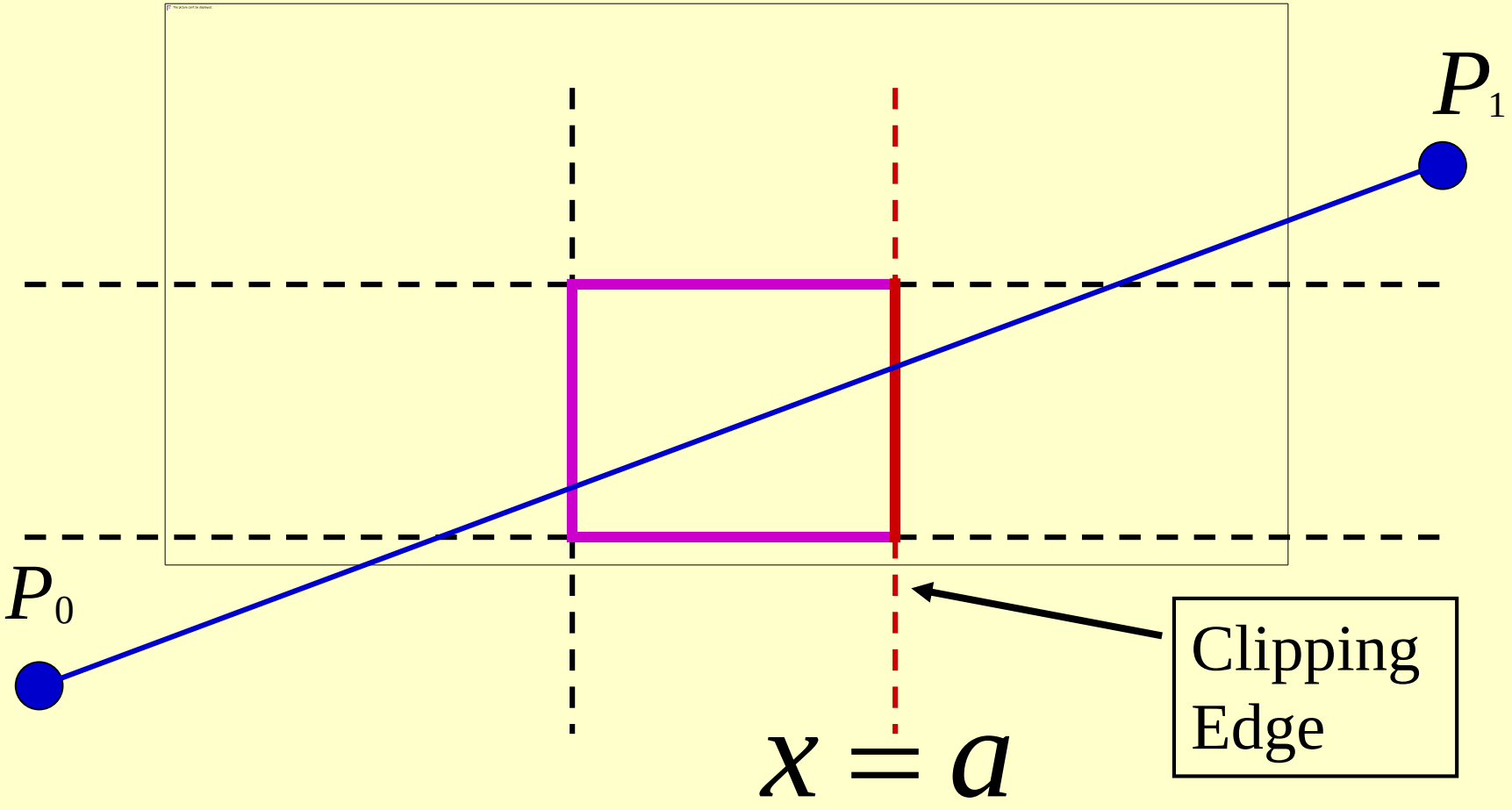
Parameter persamaan Garis

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

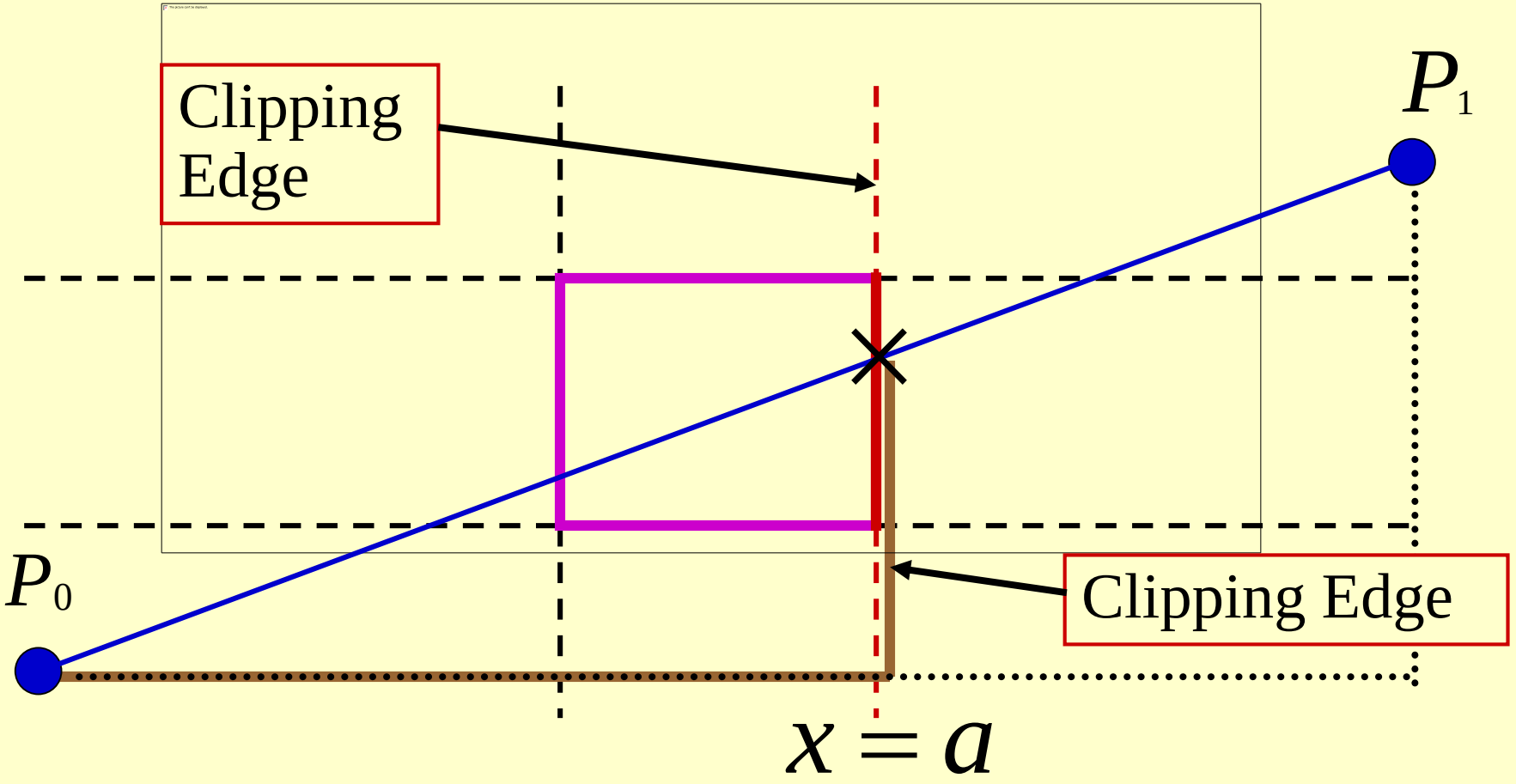
where,

$$P(0) = P_0 \;\; ; \quad P(1) = P_1$$

Clip batas-batas $x = a$



Gunakan Kesamaan Segitiga



Gunakan Kesamaan Segitiga

Gunakan rasio garis ini

{ .....

Yaitu, $$t' = \frac{a - x_0}{x_1 - x_0}$$

Dan , sama juga untuk garis eksplisit

Clipping garis Cohen-Sutherland

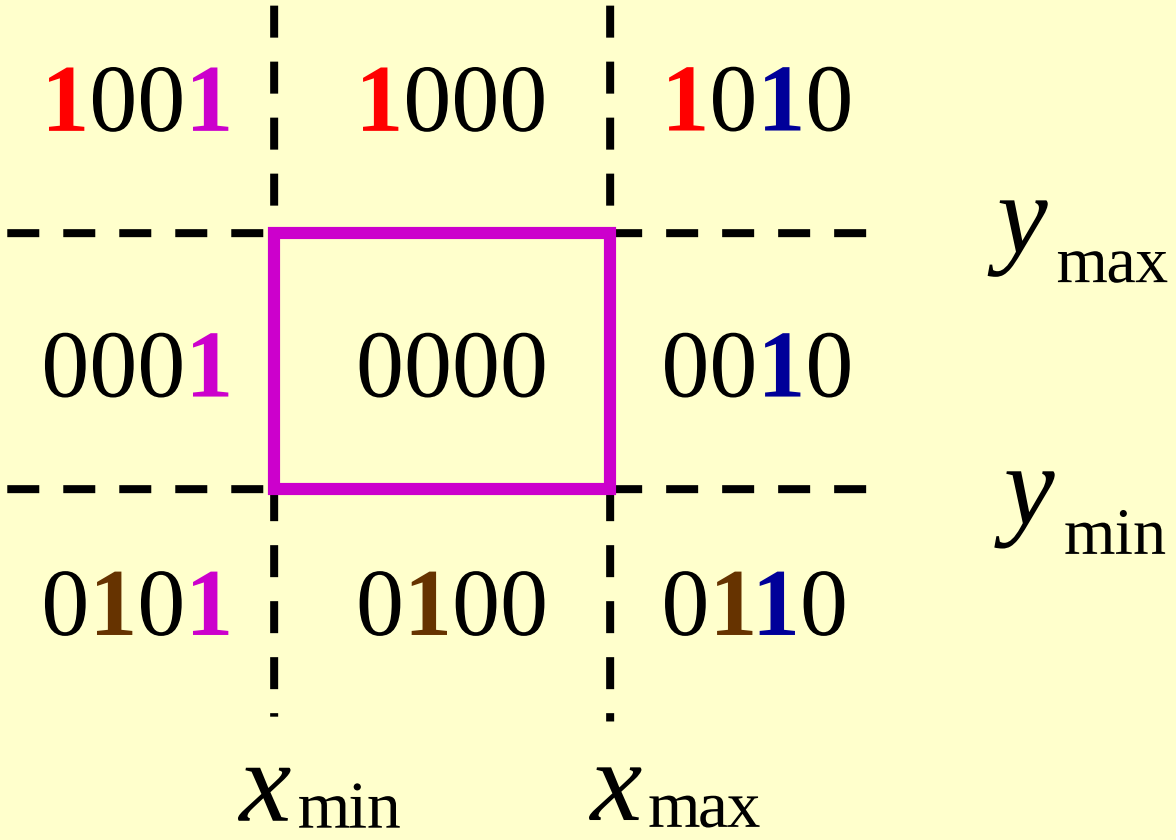
Cohen-Sutherland

- Metode ini mempercepat pemrosesan segmen garis dengan mengurangi jumlah perpotongan yang harus dihitung.
- Setiap endpoint dari garis dalam gambar dinyatakan dalam 4 digit kode biner disebut region code
- Nilai 1 pada setiap posisi bit menerangkan bahwa titik berada pada posisi region tersebut, jika tidak nilainya 0
- Nilai region code dapat ditentukan dengan 2 langkah:

  - hitung perbedaan antara koordinat endpoint dengan batas clipping

  - gunakan bit tanda resultan pada setiap perbedaan perhitungan untuk menentukan lokasi pada region

Region Outcode



Lihat pada Bit ($neg \Rightarrow 1$)

- Bit 1 $\leftarrow sign(y_{\max} - y)$
- Bit 2 $\leftarrow sign(y - y_{\min})$
- Bit 3 $\leftarrow sign(x_{\max} - x)$
- Bit 4 $\leftarrow sign(x - x_{\min})$

Butuh *Classify* Endpoint

- Lihat pada $C_0 \wedge C_1$
- Apakah yang bisa kita katakan?
- $C_0 \wedge C_1 \neq 0 \Rightarrow$ “trivial reject”
- Kedua ujung ada di dalam suatu baris atau kolom outside

Region Outcodes

Bit_1	Bit_2	Bit_3	Bit_4

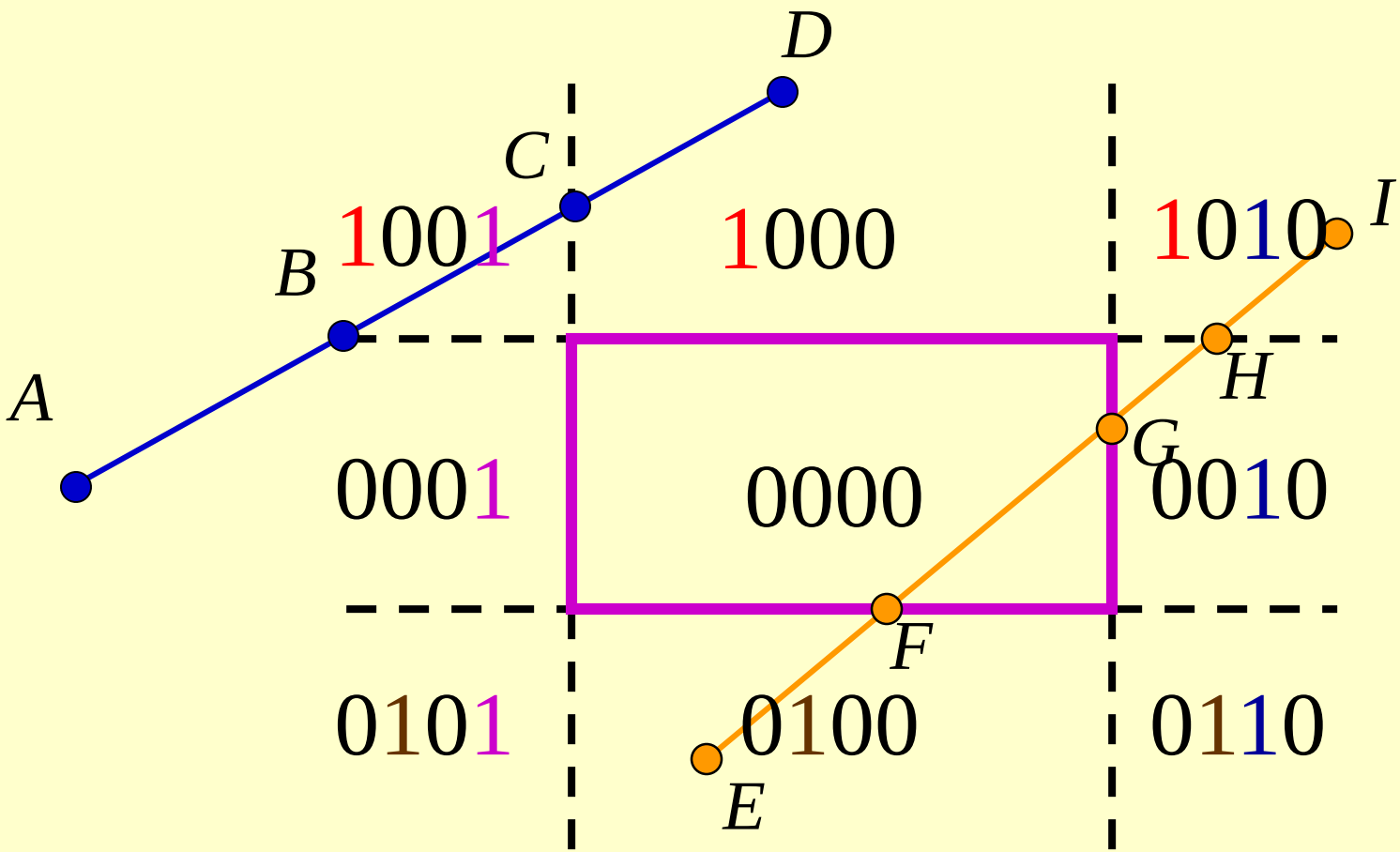
- $Bit\ 1 = t \Rightarrow$ *Atas window*
- $Bit\ 2 = t \Rightarrow$ *bawah window*
- $Bit\ 3 = t \Rightarrow$ *kanan window*
- $Bit\ 4 = t \Rightarrow$ *Kiri window*

Classify Endpoint

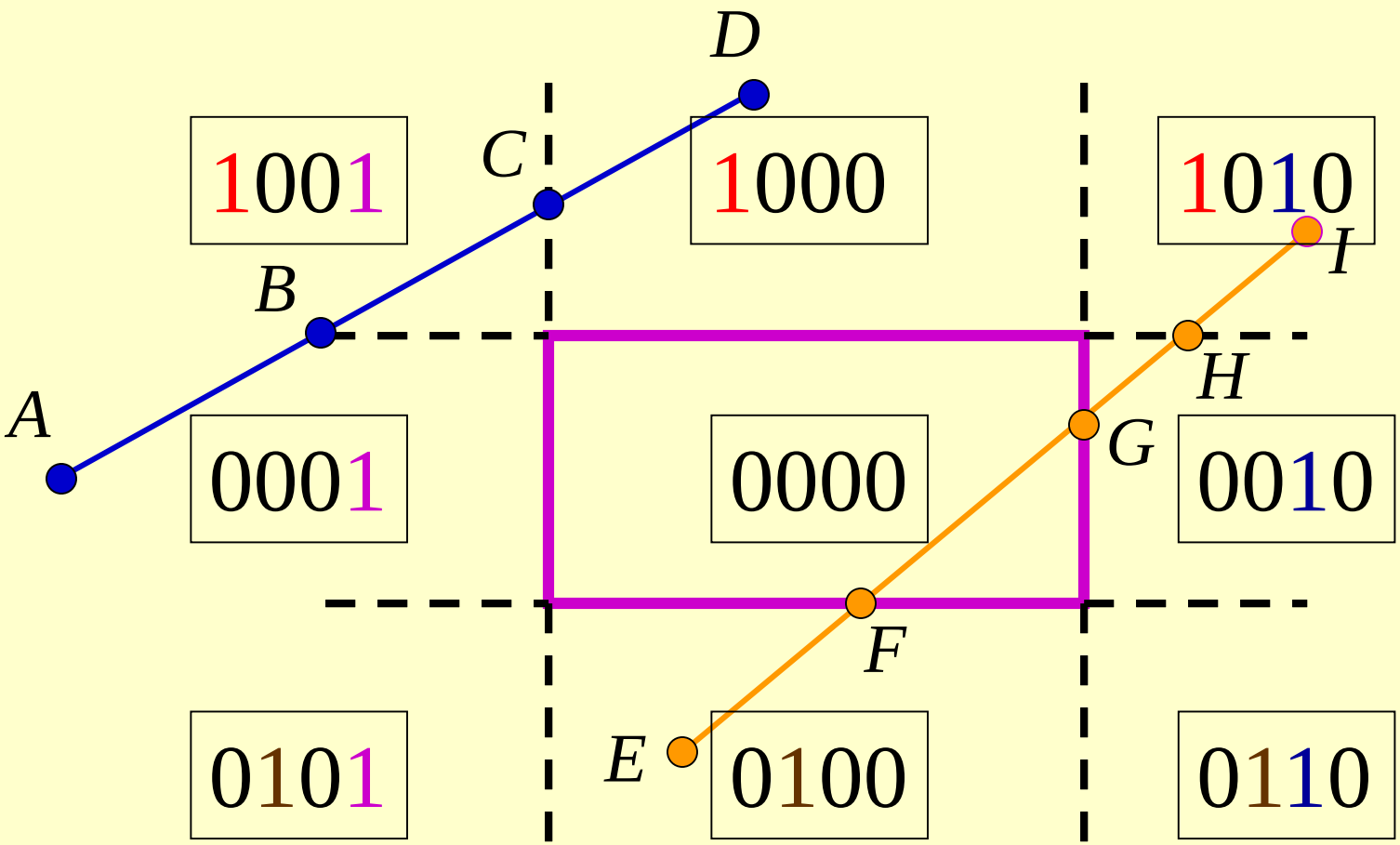
$\{C_0 \wedge C_1 = 0\} \Rightarrow$ Titik akhir mungkin saja tidak dalam window

Clip suatu akhir untuk $C_i \neq 0$

Cohen-Sutherland Line Clipping



Cohen-Sutherland Line Clipping



Penghitungan Outcode Awal

- $OC(D)=\mathbf{1}000$; $OC(A)=000\mathbf{1}$

$$\mathbf{1}000 \wedge 000\mathbf{1} = 0000$$

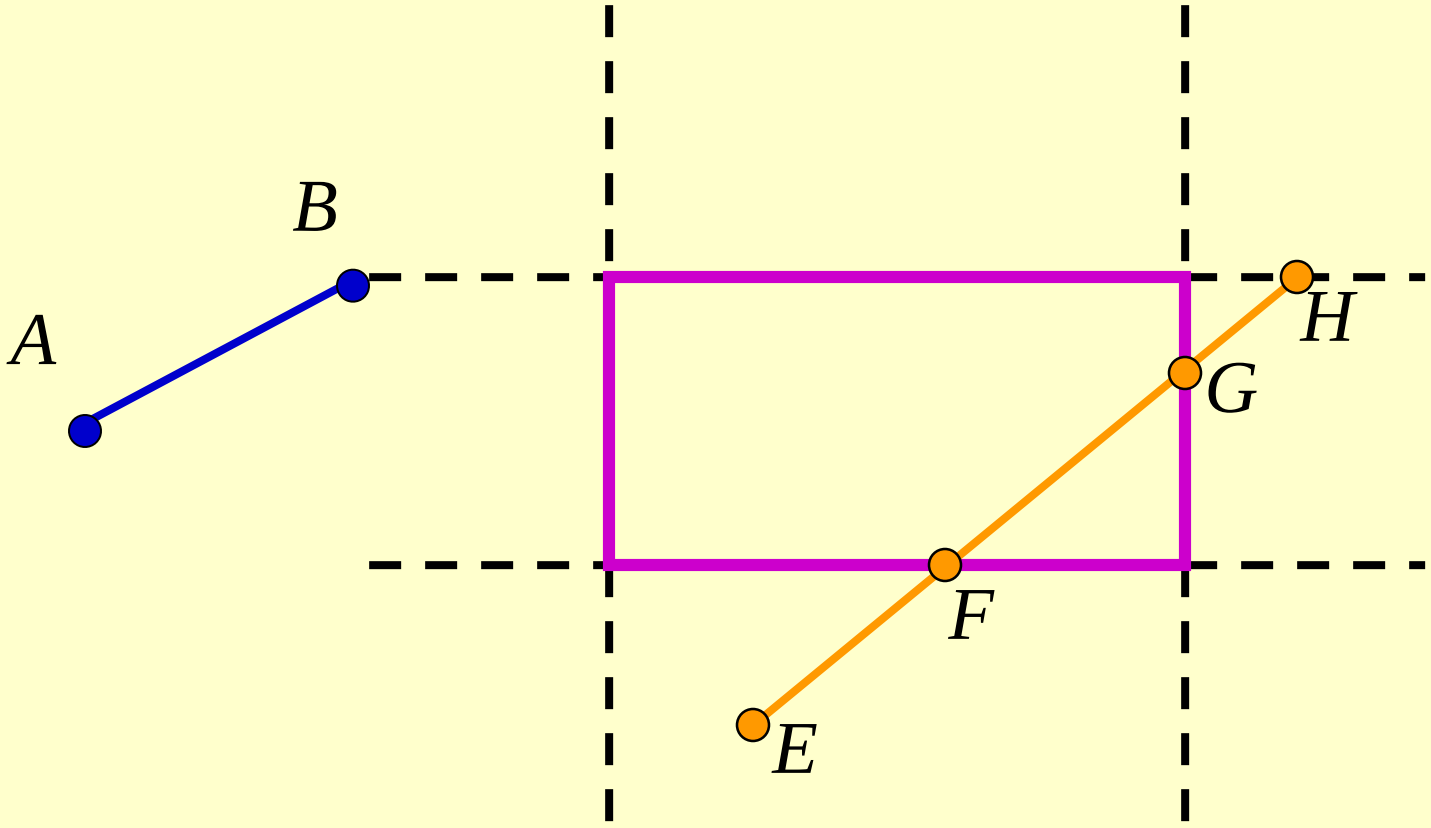
- $OC(E)=0\mathbf{1}00$; $OC(I)=\mathbf{1}0\mathbf{1}0$

$$0\mathbf{1}00 \wedge \mathbf{1}0\mathbf{1}0 = 0000$$

Clip dan lanjutkan

- Clip lagi batas atas $y = y_{\max}$
- Hitung B. Keep AB
- Hitung H . Keep EH

Cohen-Sutherland Line Clipping



Clip and Continue

- Clip lagi batas bawah $y = y_{\min}$
- Skr test dan tolak AB *karena*
- $OC(A)$=000**1** and $OC(B)$=000**1**;

$$000\mathbf{1} \wedge 000\mathbf{1} = 000\mathbf{1} \neq 0$$

- Tolak AB on outcode basis

Penghitungan Outcode

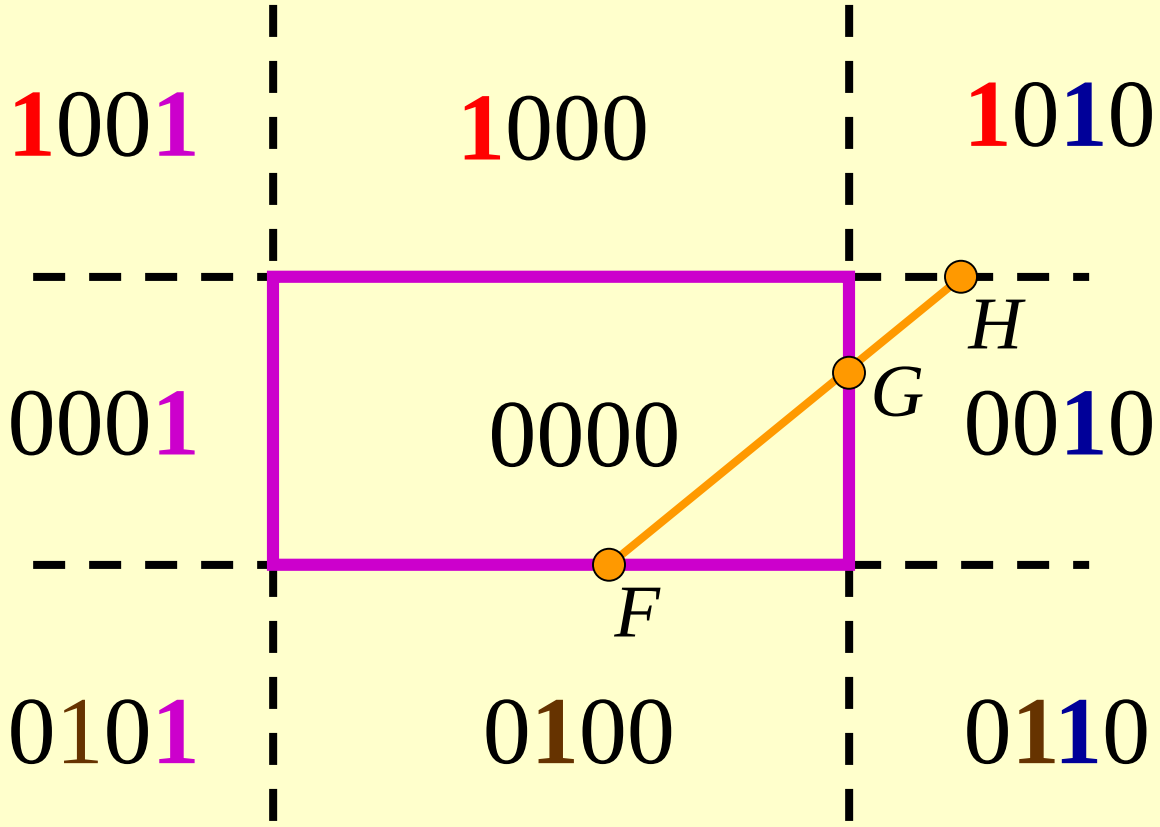
- $OC(H)$=00**1**0; $OC(E)$=0**1**00

  $00\mathbf{1}0 \wedge 0\mathbf{1}00 = 0000$

- Saat hasil adalah 0, proses

  HE untuk mendapatkan FH

Cohen-Sutherland Line Clipping



Outcode Calculations

- $OC(F)=0000$; $OC(H)=00\mathbf{1}0$

$$00\mathbf{1}0 \wedge 0\mathbf{1}00 = 0000$$

- Saat hasil adalah 0, proses HF untuk mendapatkan GF

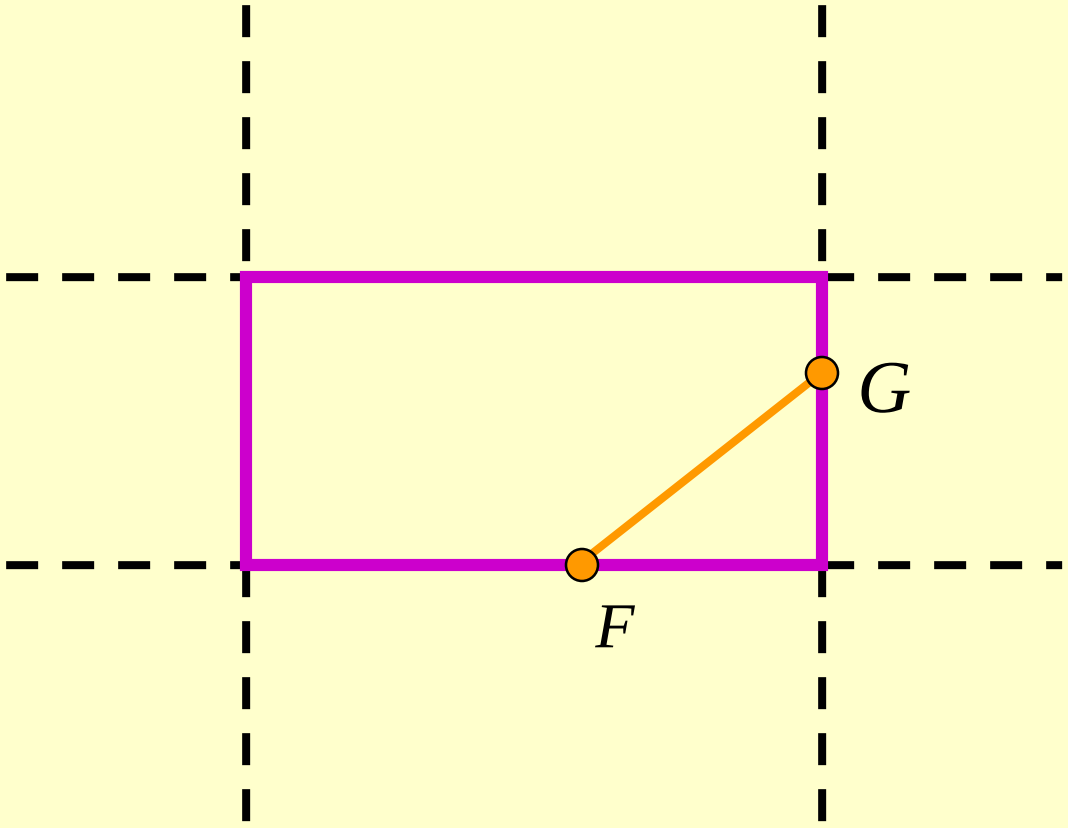
Clip and Continue

- Clip lagi batas kanan

$$x = x_{\max}$$

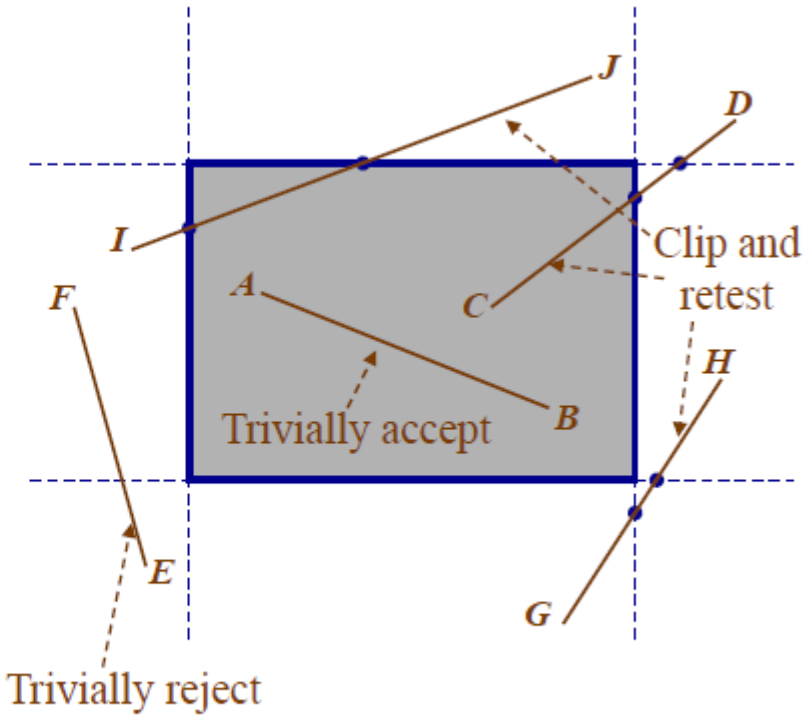
- Dapatkan GF
- kerjakan

Cohen-Sutherland Line Clipping



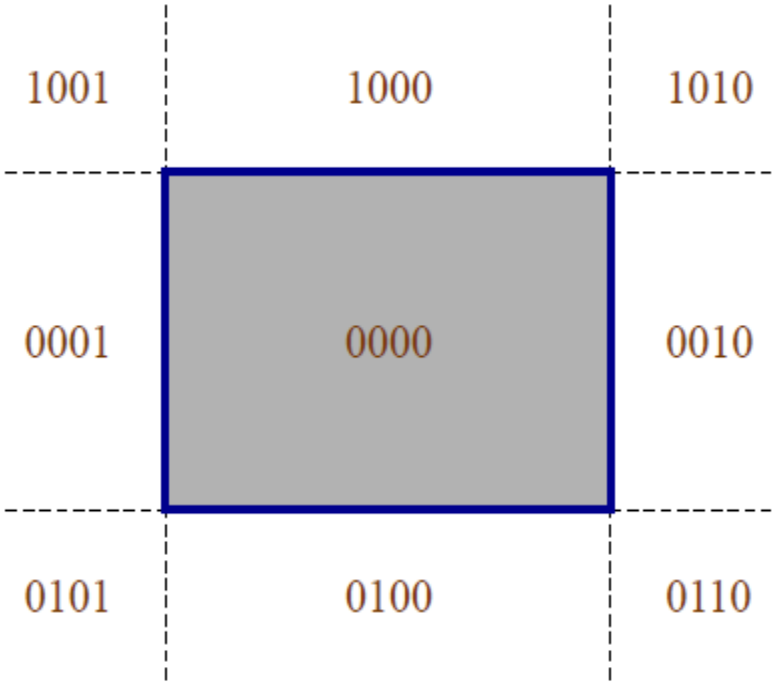
Cohen-Sutherland

- Basic algorithm:
  - Accept lines that have both endpoints inside the region.
  - Reject lines that have both endpoints less than x_{min} or y_{min} or greater than x_{max} or y_{max}.
  - Clip the remaining lines at a region boundary and repeat the previous steps on the clipped line segments.



Cohen-Sutherland: Accept/Reject Tests

- Assign a 4-bit code to each endpoint c_0, c_1 based on its position:
  - 1st bit (1000): if $y > y_{max}$
  - 2nd bit (0100): if $y < y_{min}$
  - 3rd bit (0010): if $x > x_{max}$
  - 4th bit (0001): if $x < x_{min}$

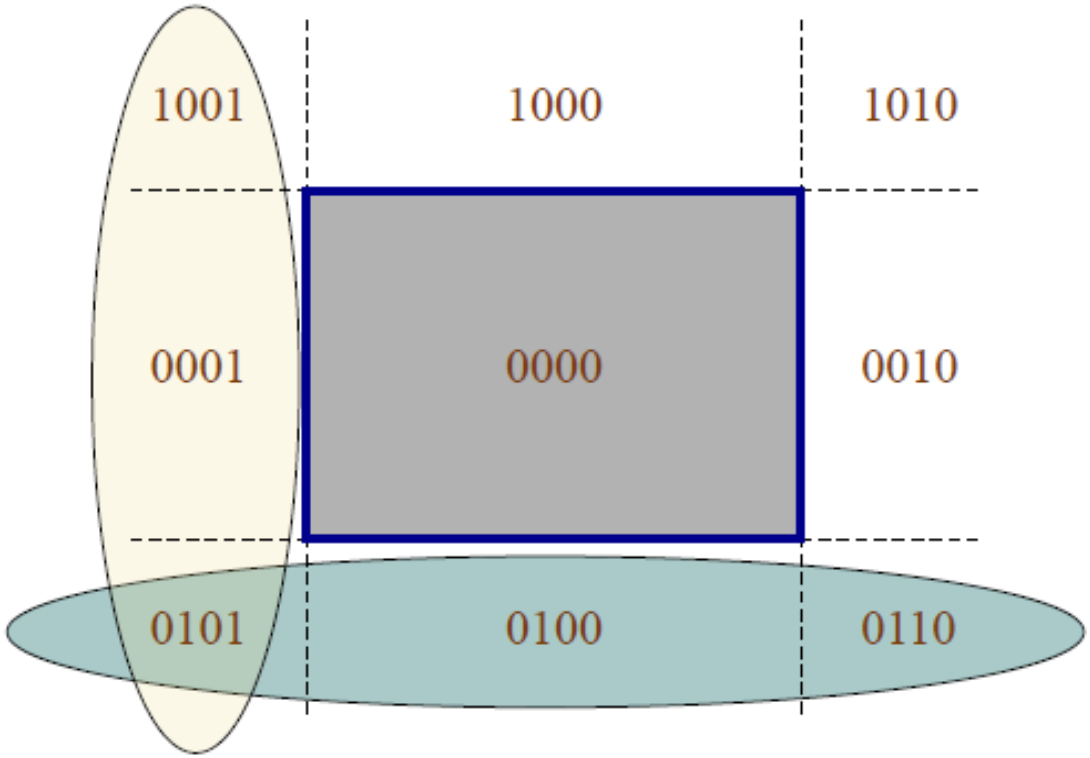


- Test using bitwise functions

if $c_0 \mid c_1 = 0000$
    accept (draw)
else if $c_0 \,\&\, c_1 \neq 0000$
    reject (don't draw)
else clip and retest

Cohen-Sutherland Accept/Reject

- Accept/reject/redo all based on bit-wise Boolean ops.



Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

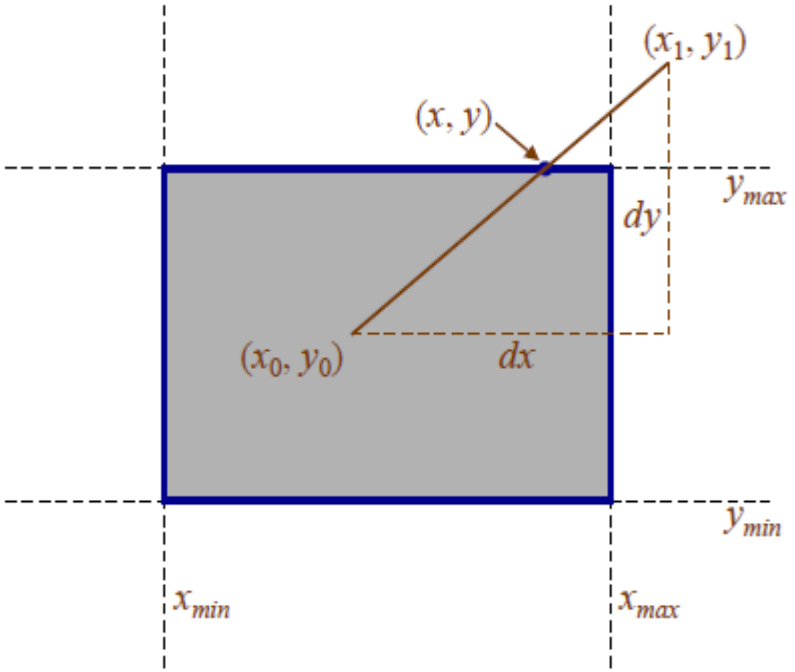
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

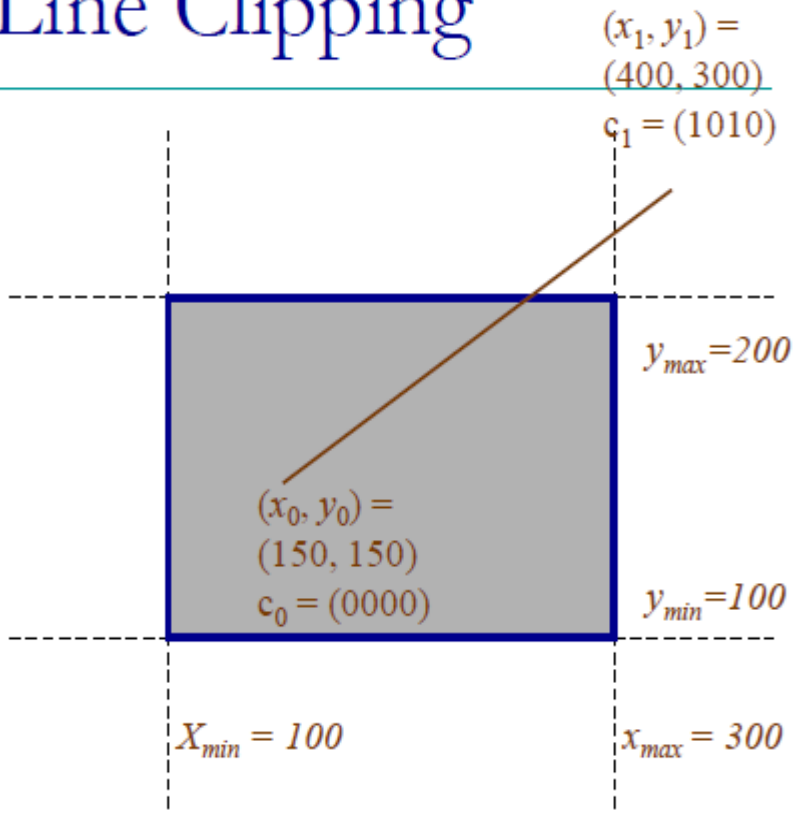
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



c dx dy x y

Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

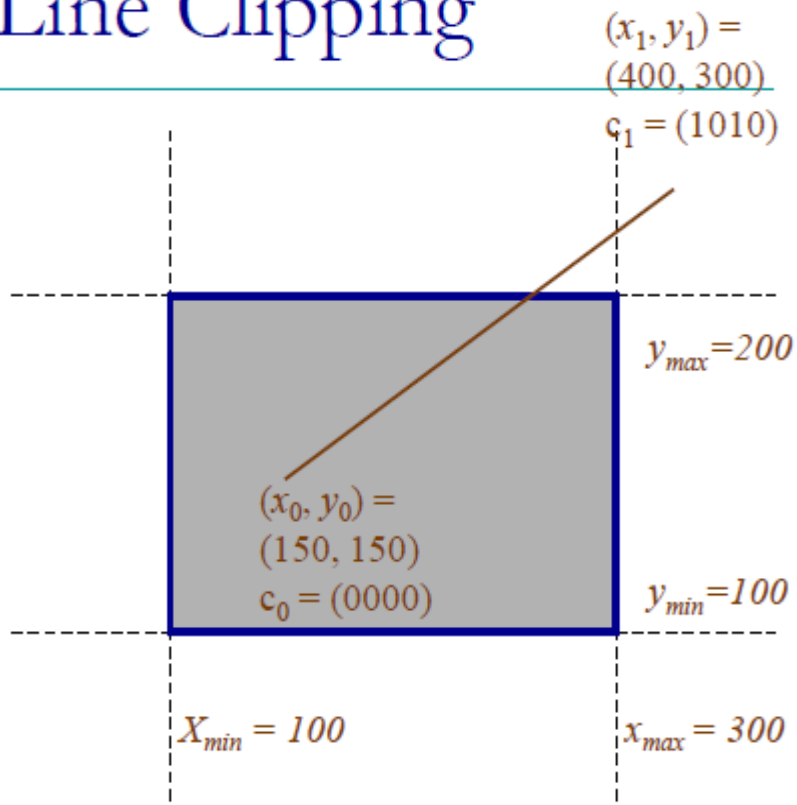
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



c	dx	dy	x	y
1010				

Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

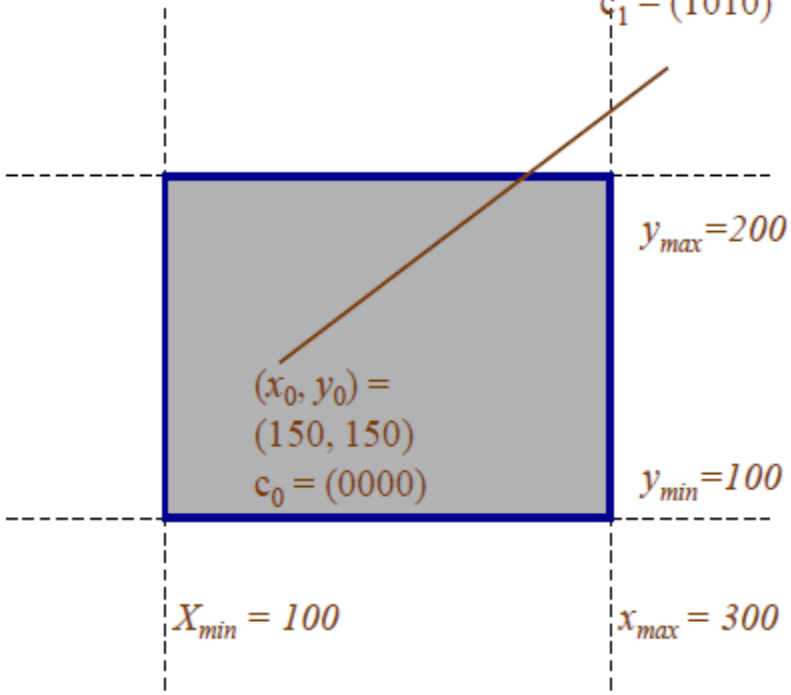
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



c	dx	dy	x	y
1010	250	150		

Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

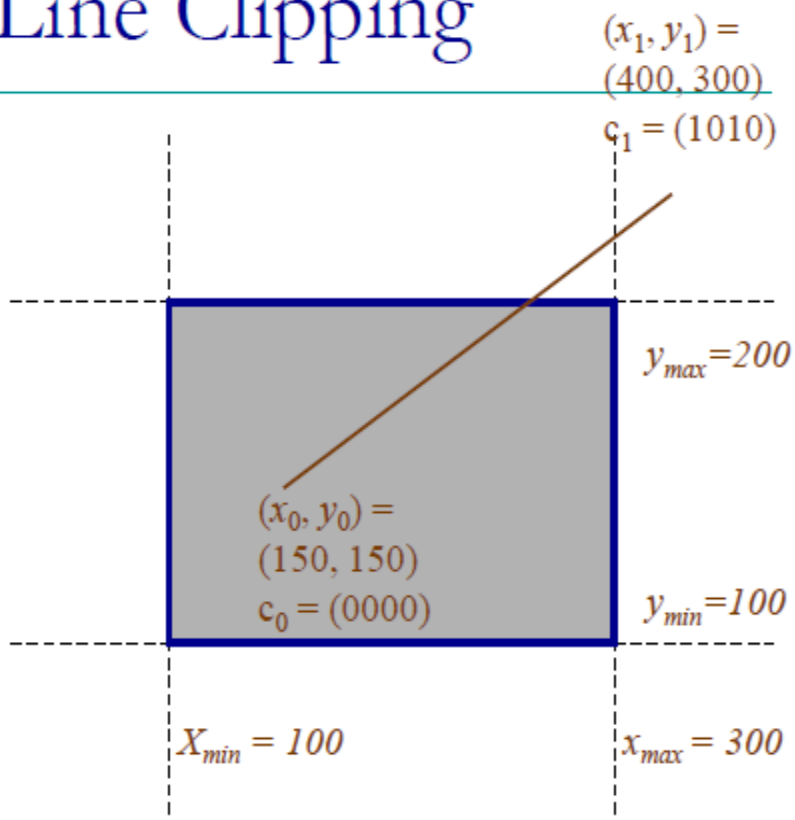
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



c	dx	dy	x	y
1010	250	150		

Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

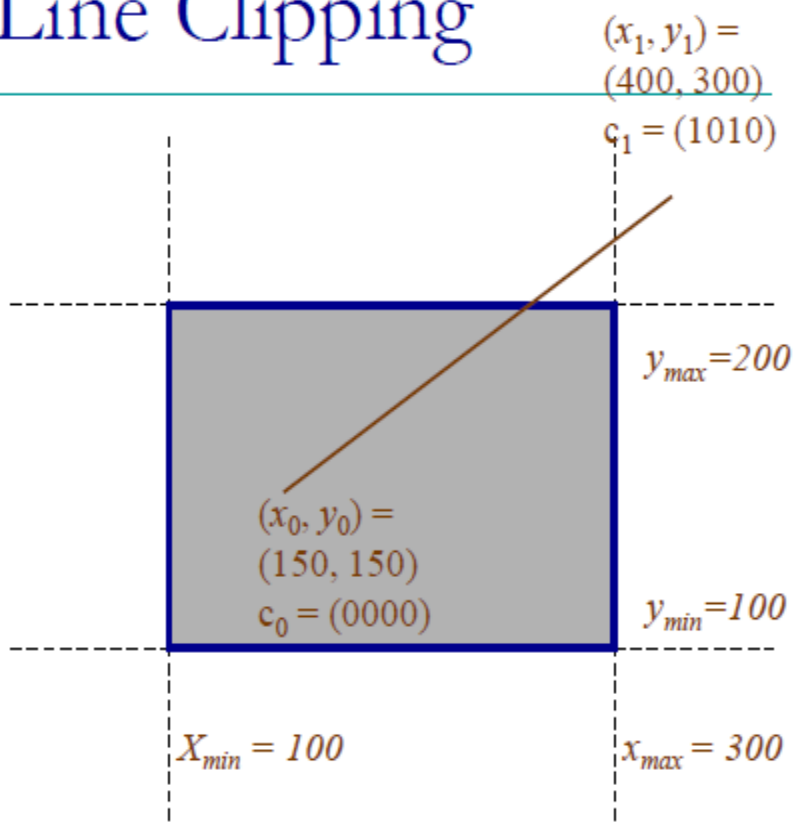
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



c	dx	dy	x	y
1010	250	150	233	200

Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

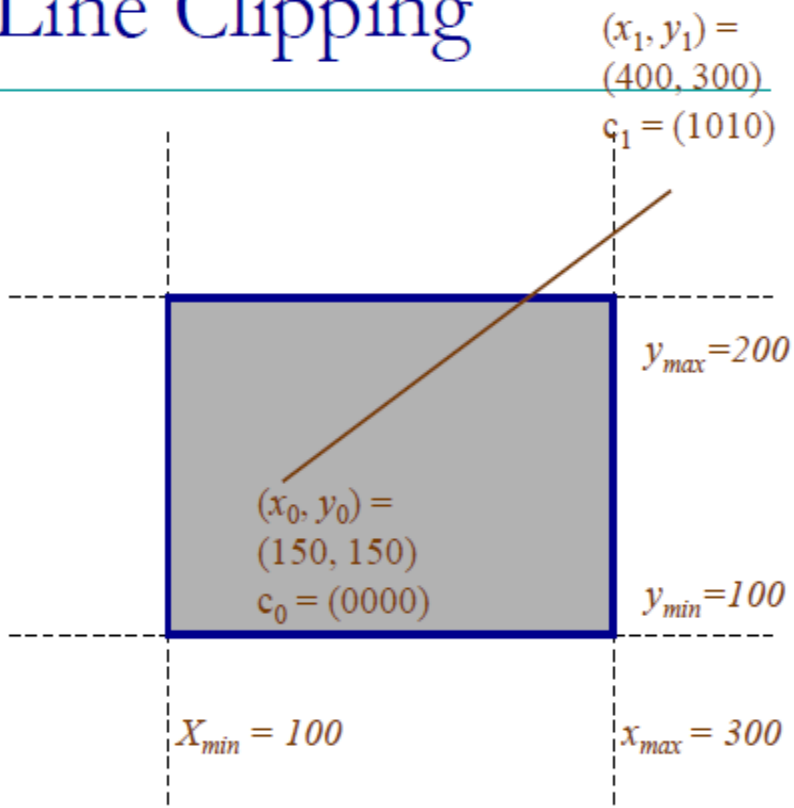
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



c	dx	dy	x	y
1010	250	150	233	200

Cohen-Sutherland: Line Clipping

Intersection algorithm:

if $c_0 \neq 0000$ then $c = c_0$;
else $c = c_1$;

$dx = x_1 - x_0$; $dy = y_1 - y_0$
if c & 1000 // y_{max}
$x = x_0 + dx * (y_{max} - y_0) / dy$; $y = y_{max}$;

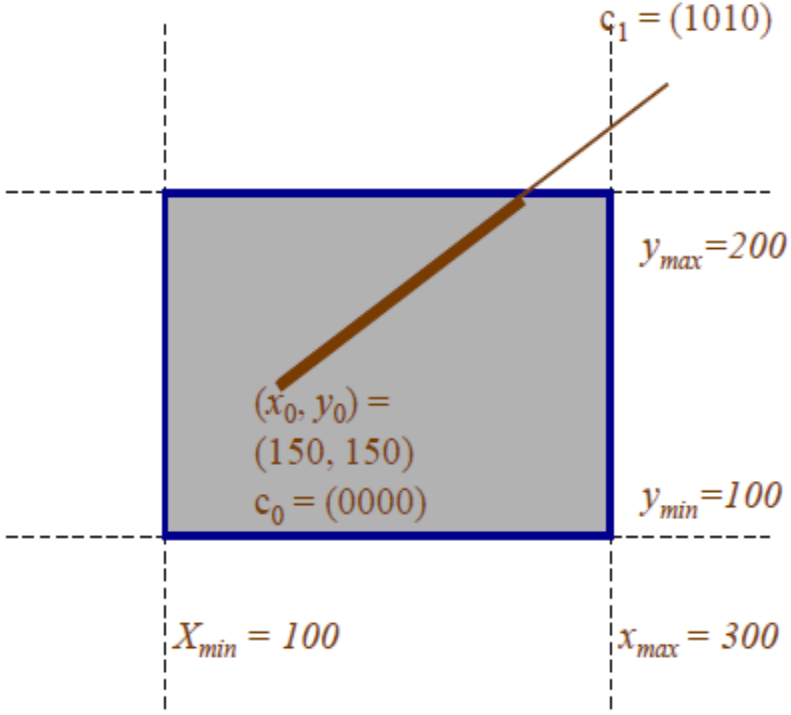
else if c & 0100 // y_{min}
$x = x_0 + dx * (y_{min} - y_0) / dy$; $y = y_{min}$;

else if c & 0010 // x_{max}
$y = y_0 + dy * (x_{max} - x_0) / dx$; $x = x_{max}$;

else // x_{min}
$y = y_0 + dy * (x_{min} - x_0) / dx$; $x = x_{min}$;

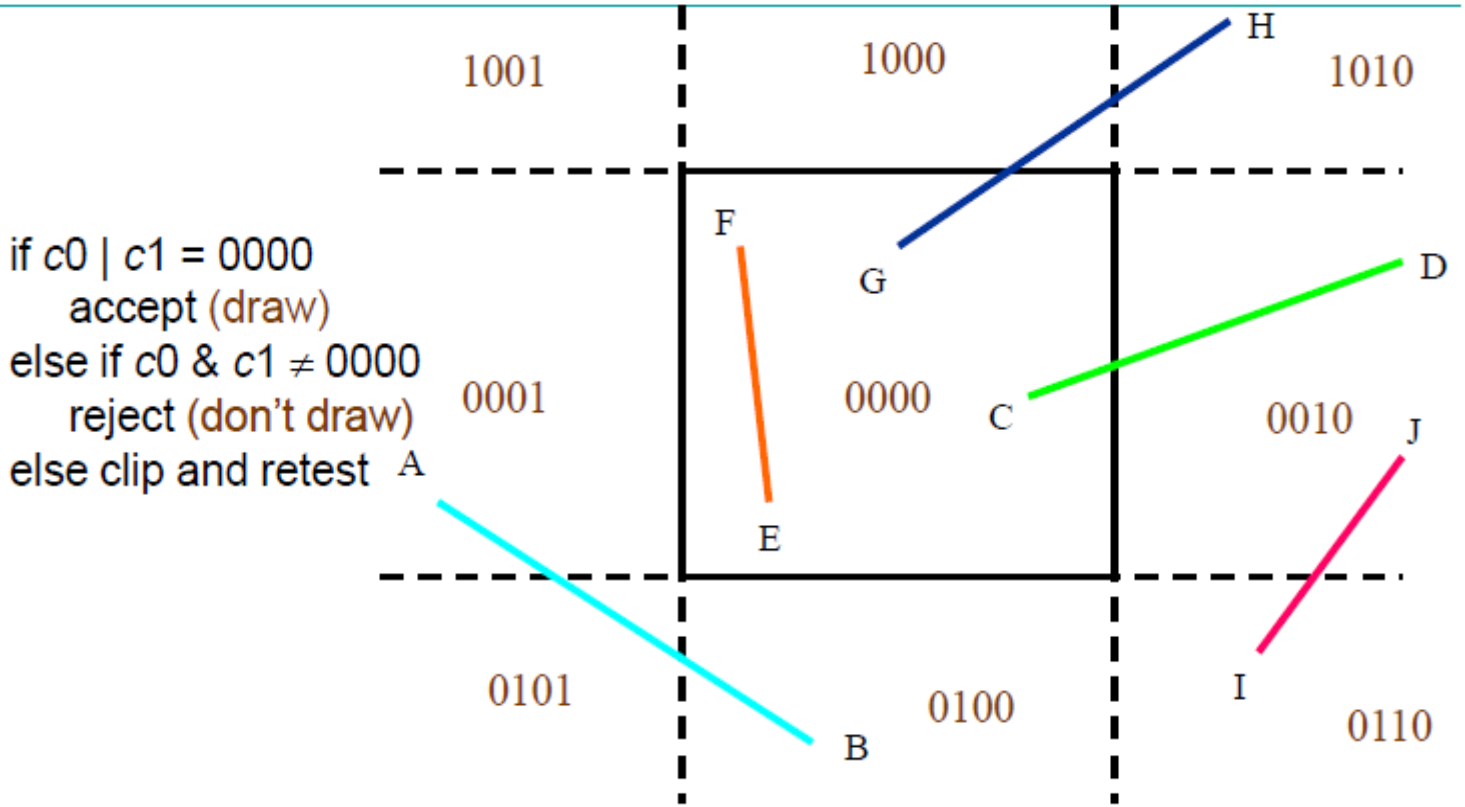
if $c = c_0$
$x_0 = x$; $y_0 = y$;

else
$x_1 = x$; $y_1 = y$;



c	dx	dy	x	y
1010	250	150	233	200

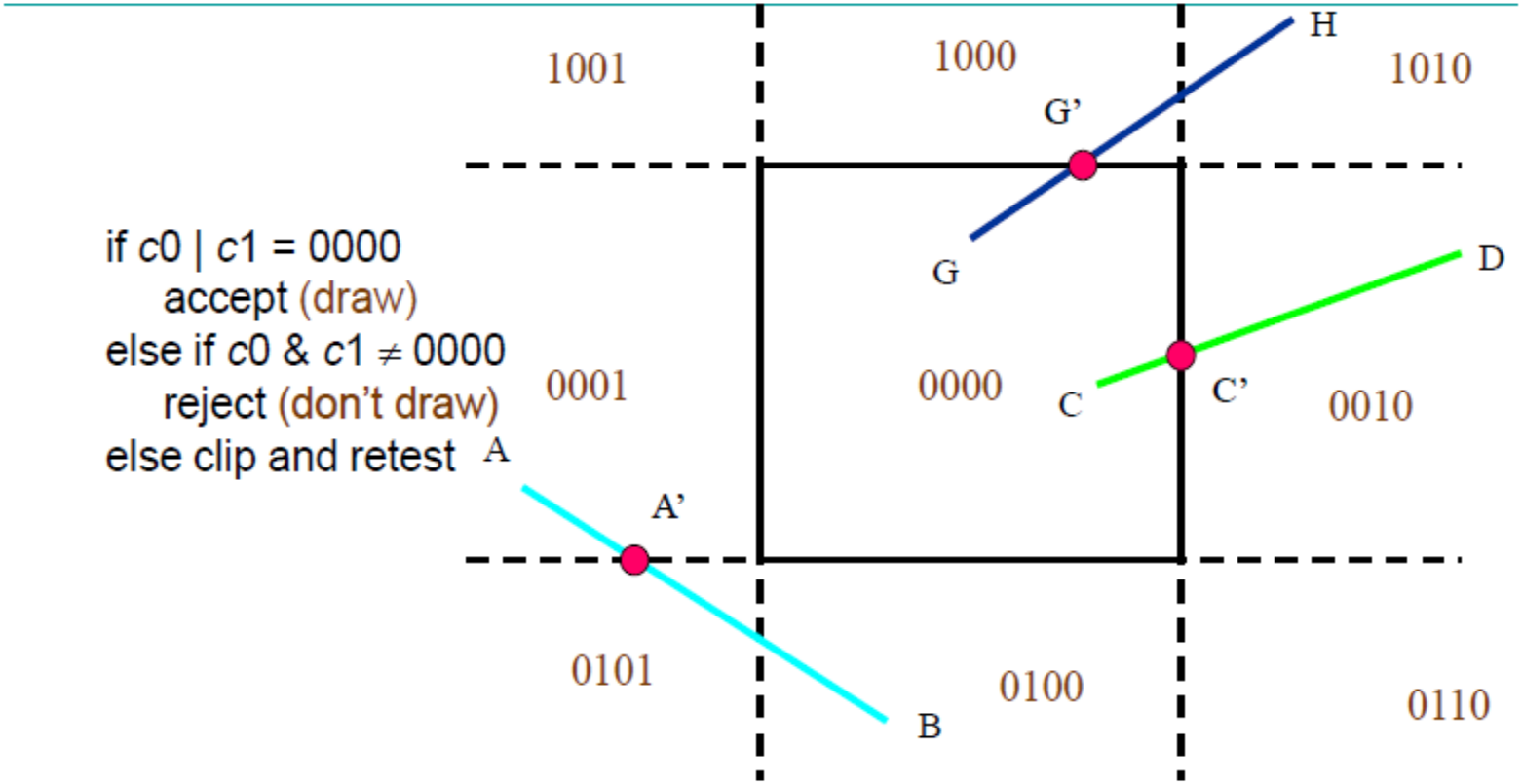
Cohen-Sutherland Line Clip Examples



if *c*0 | *c*1 = 0000
    accept (draw)
else if *c*0 & *c*1 ≠ 0000
    reject (don't draw)
else clip and retest

A 0001	C 0000	E 0000	G 0000	I 0110
B 0100	D 0010	F 0000	H 1010	J 0010
OR 0101	OR 0010	OR 0000	OR 1010	OR 0110
AND 0000	AND 0000	AND 0000	AND 0000	AND 0010
clip	**clip**	**accept**	**clip**	**reject**

Cohen-Sutherland Line Clip Examples

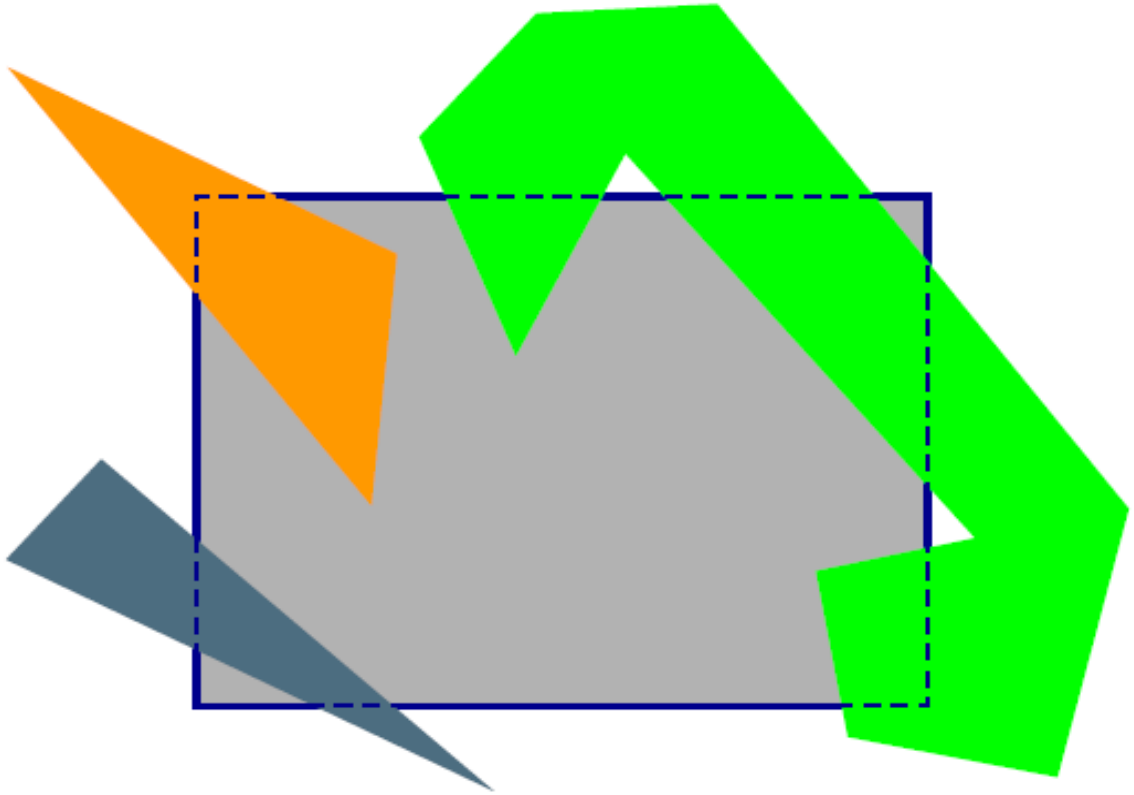


if *c*0 | *c*1 = 0000
accept (draw)
else if *c*0 & *c*1 ≠ 0000
reject (don't draw)
else clip and retest

remove A'B	remove C'D	remove G'H
A 0001	C 0000	G 0000
A' 0001	C' 0000	G' 0000
OR 0001	OR 0000	OR 0000
AND 0001	AND 0000	AND 0000
reject	**accept**	**accept**

Polygon Clipping

What about polygons?



Polygon Clipping: Algorithm

- Clip polygon to y_{min} and y_{max}:
  - Create empty output vertex list
  - Process input list $(\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ where $\boldsymbol{v}_0 = \boldsymbol{v}_n$
  - For each input vertex ($\boldsymbol{v}_i$ where $0 \le i \le n-1$):
    - If $\boldsymbol{v}_i$ is inside region → Add $\boldsymbol{v}_i$ to output list.
    - If the line between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$ intersects clipping boundaries → Add intersection point(s) to output list.
- Repeat: Clip to x_{min} and x_{max}
- Post-process:
  - Remove degenerate sections that have collapsed to region boundary.

Polygon Clipping: Example

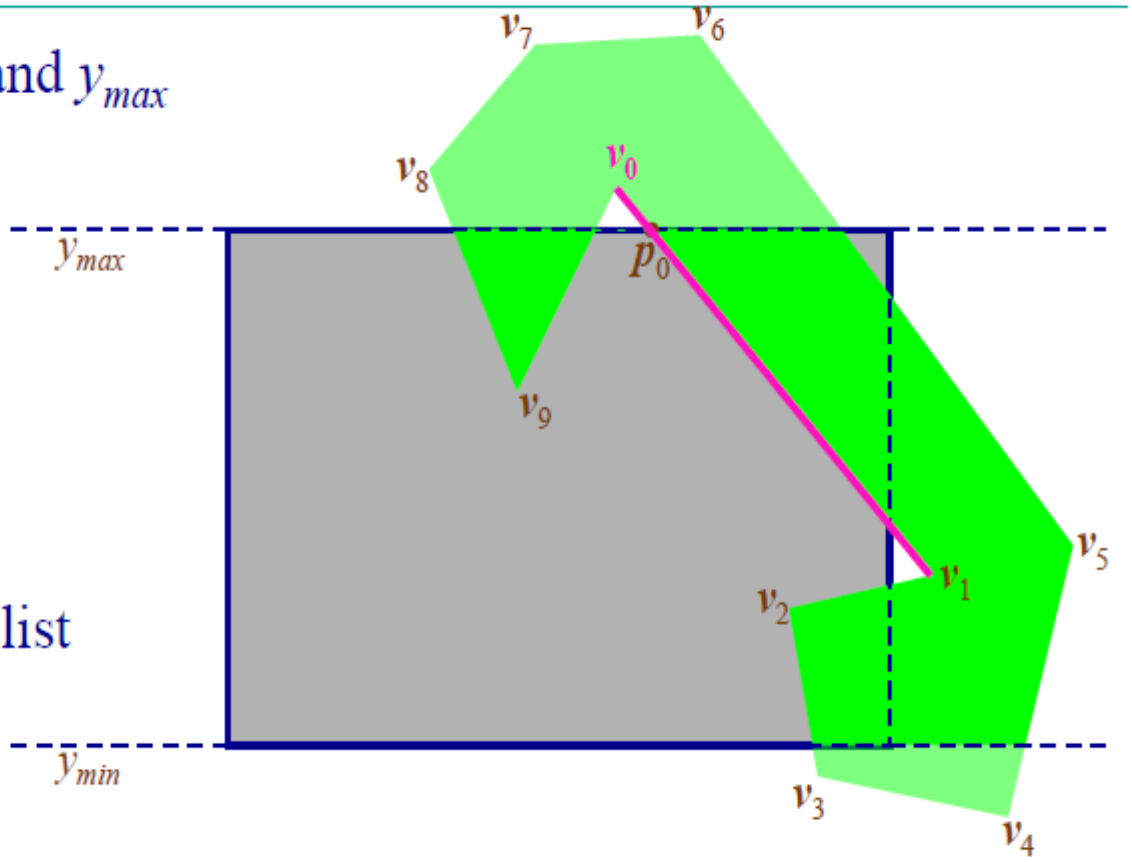
Clip first to y_{min} and y_{max}

vertex: v_0

Inside region: No

Line intersect boundary: Yes

Add p_0 to output list



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: p_0

Polygon Clipping: Example

Clip first to y_{min} and y_{max}

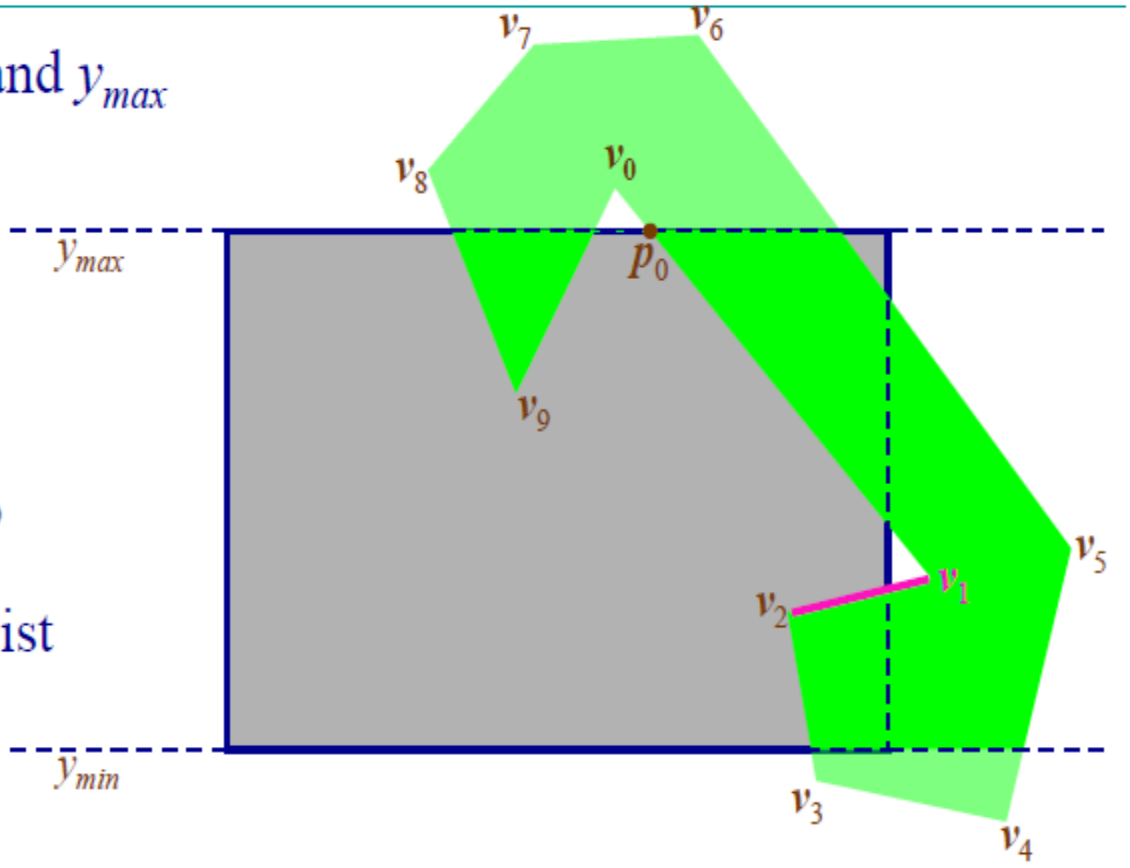
vertex: v_1

inside region: yes

line intersect

boundary: no

add v_1 to output list



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: p_0 , v_1

Polygon Clipping: Example

Clip first to y_{min} and y_{max}

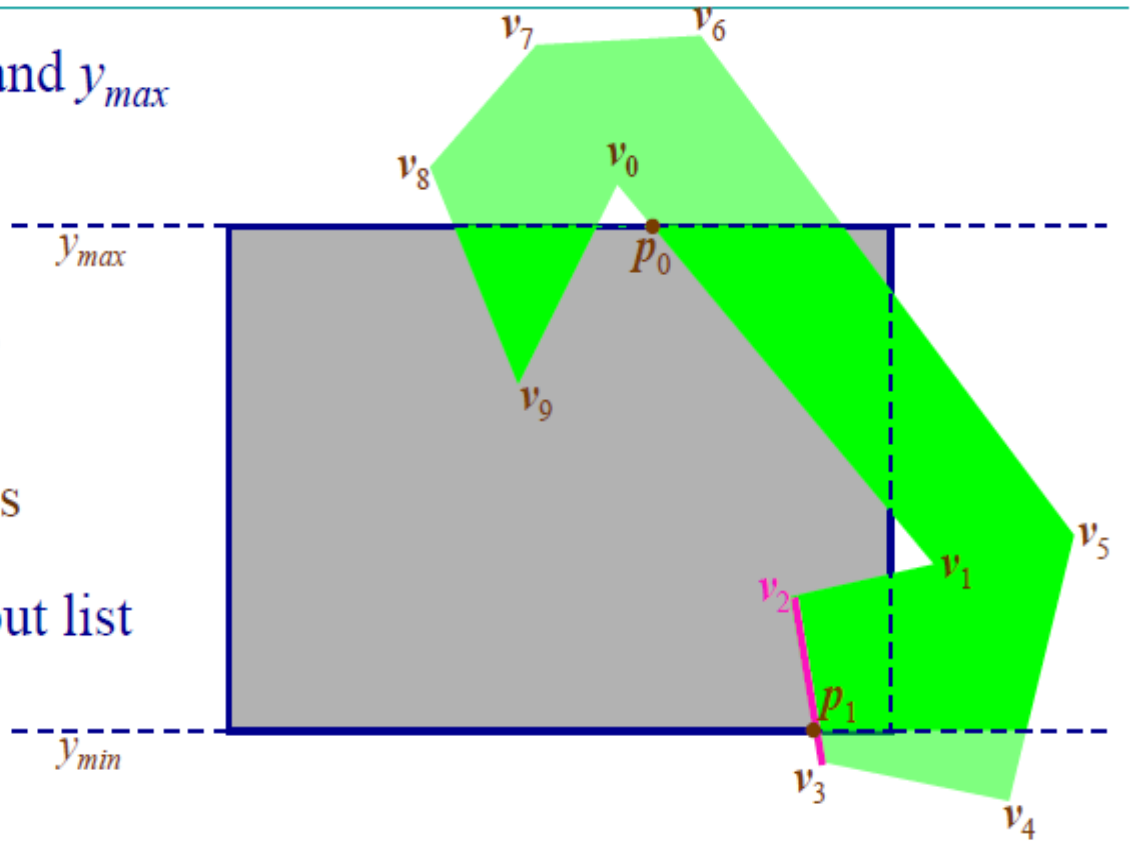
vertex: v_2

inside region: yes

line intersect

boundary: yes

add v_2, p_1 to output list



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: p_0, v_1, v_2, p_1

Polygon Clipping: Example

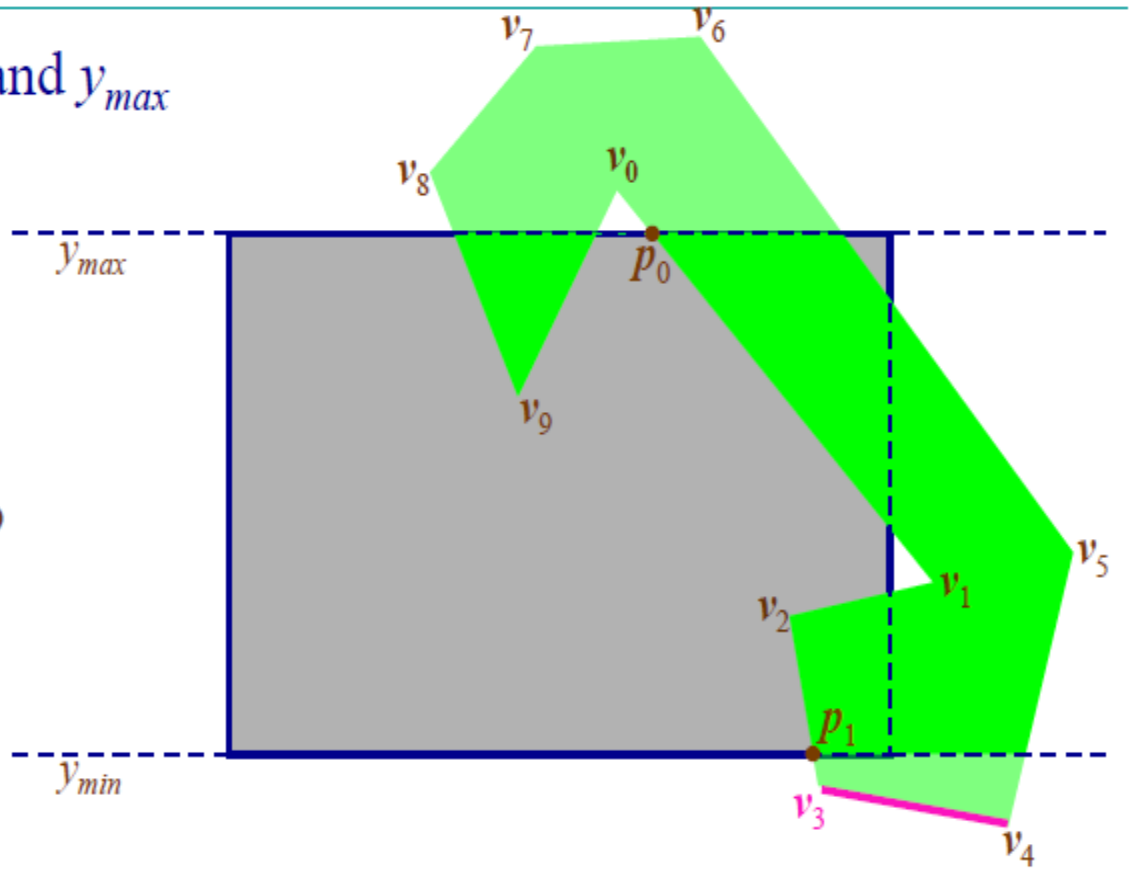
Clip first to y_{min} and y_{max}

vertex: v_3

inside region: no

line intersect

boundary: no



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: p_0, v_1, v_2, p_1

Polygon Clipping: Example

Clip first to y_{min} and y_{max}

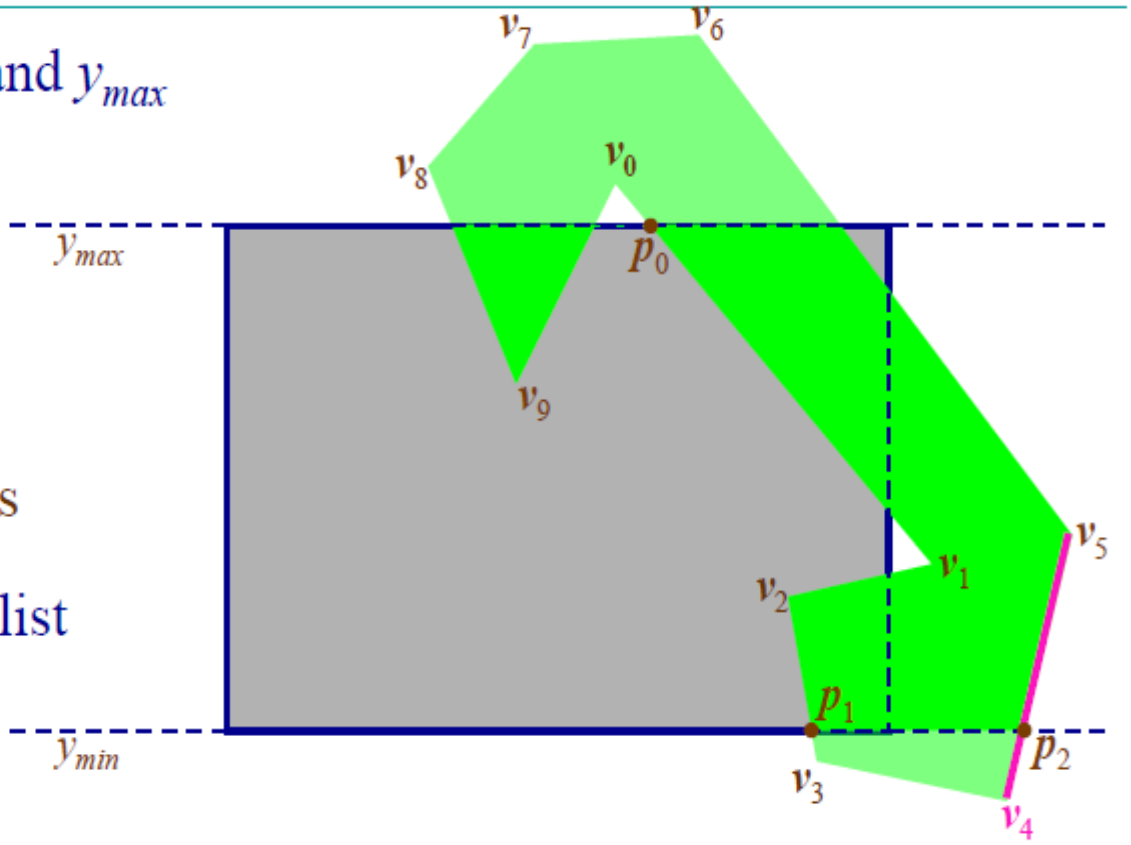
vertex: v_4

inside region: no

line intersect

boundary: yes

add p_2 to output list



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: p_0, v_1, v_2, p_1, p_2

Polygon Clipping: Example

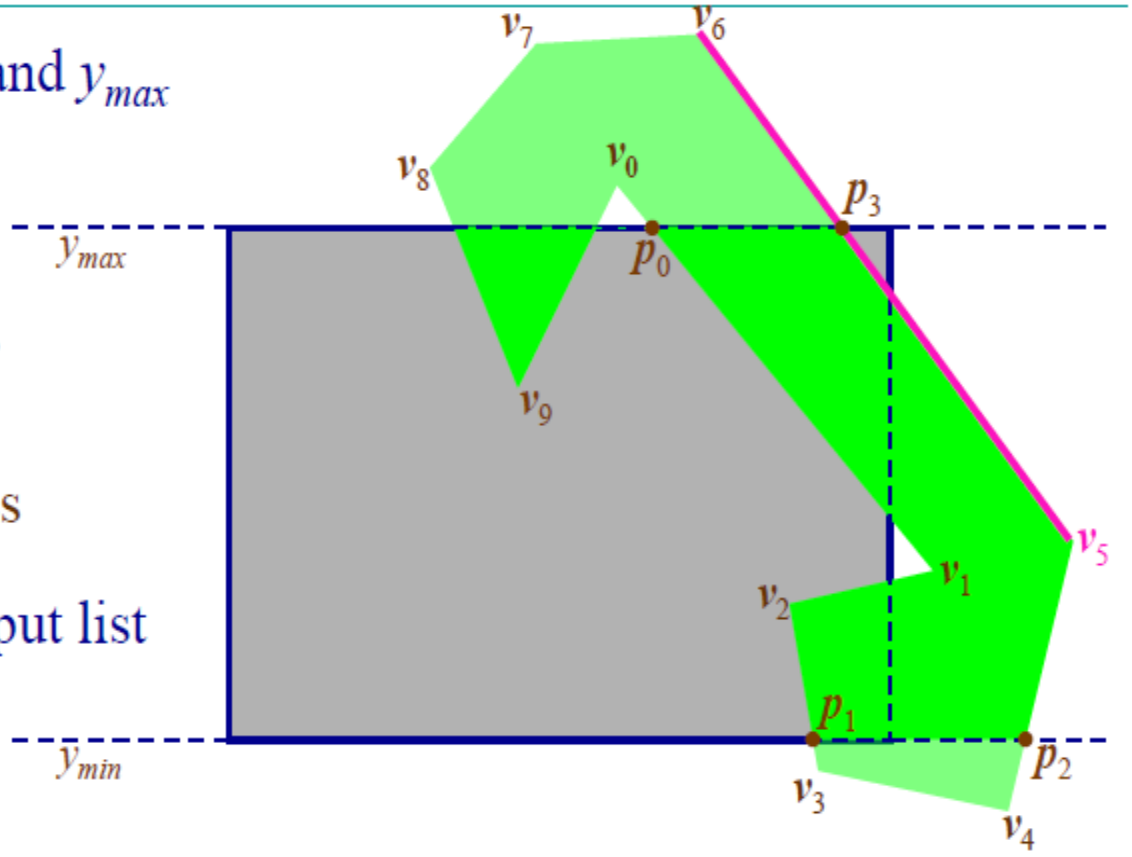
Clip first to y_{min} and y_{max}

vertex: v_5

inside region: yes

line intersect boundary: yes

add v_5, p_3 to output list



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: $p_0, v_1, v_2, p_1, p_2, v_5, p_3$

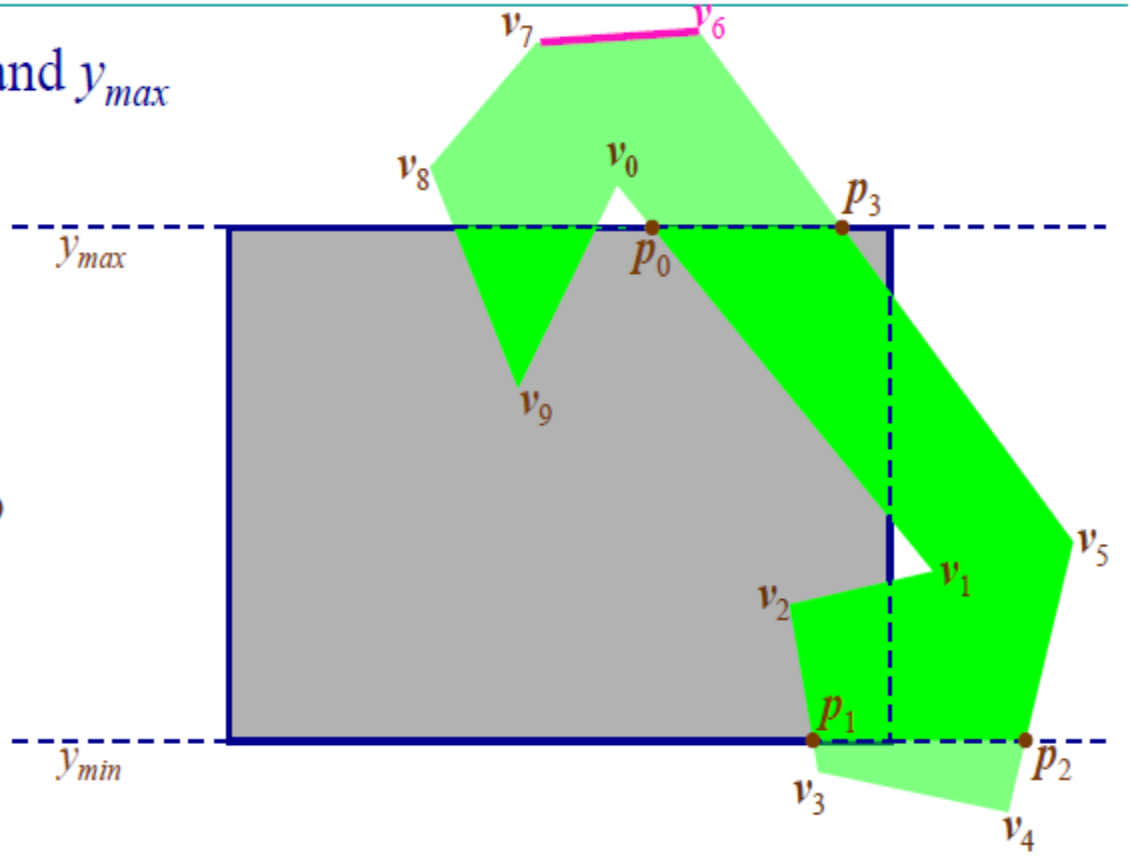
Polygon Clipping: Example

Clip first to y_{min} and y_{max}

vertex: v_6

inside region: no

line intersect boundary: no



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: $p_0, v_1, v_2, p_1, p_2, v_5, p_3$

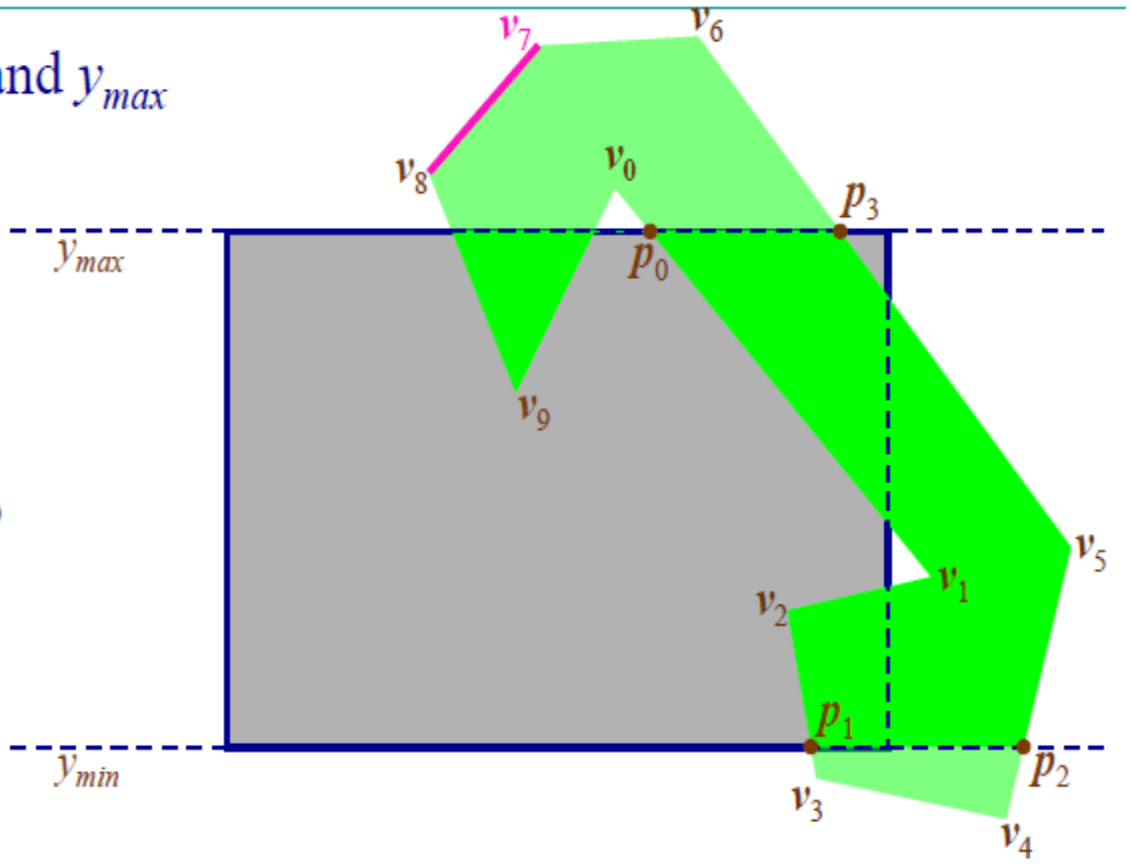
Polygon Clipping: Example

Clip first to y_{min} and y_{max}

vertex: v_7

inside region: no

line intersect boundary: no



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: $p_0, v_1, v_2, p_1, p_2, v_5, p_3$

Polygon Clipping: Example

Clip first to y_{min} and y_{max}

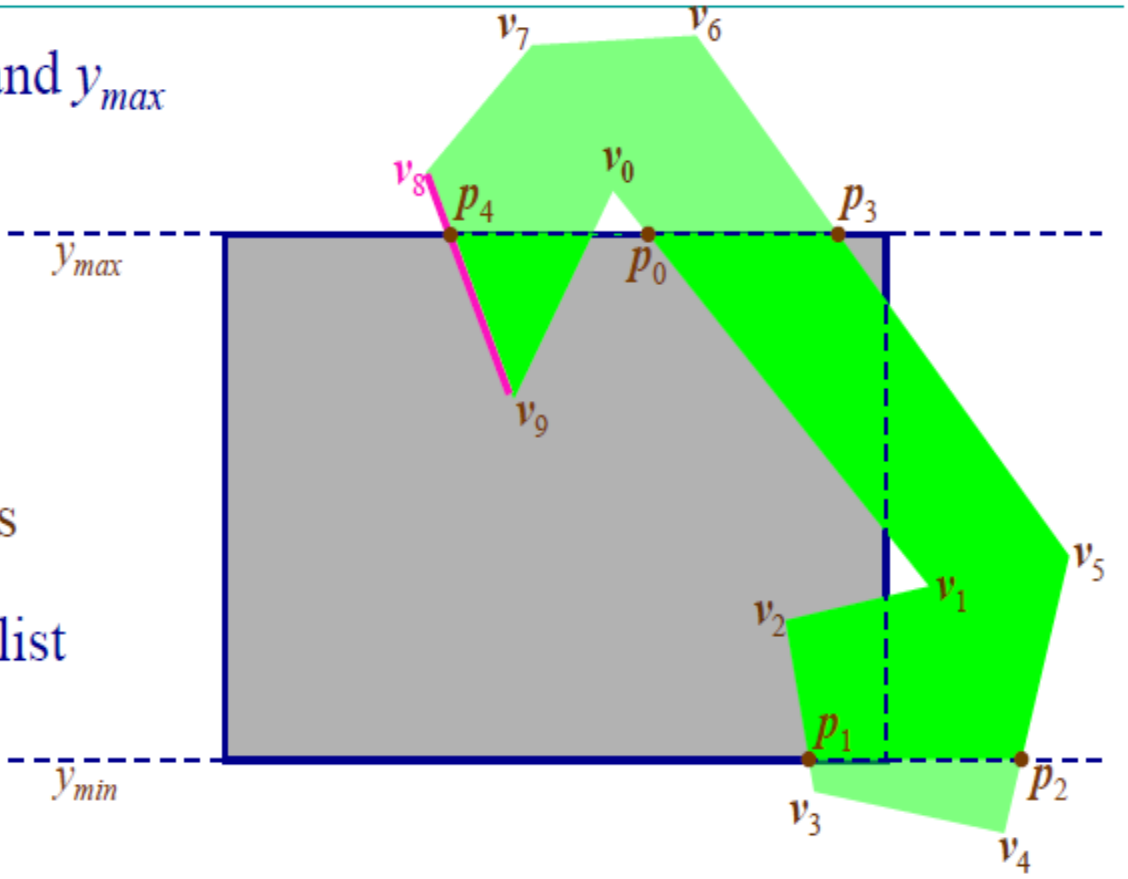
vertex: v_8

inside region: no

line intersect

boundary: yes

add p_4 to output list



Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: $p_0, v_1, v_2, p_1, p_2, v_5, p_3, p_4$

Polygon Clipping: Example

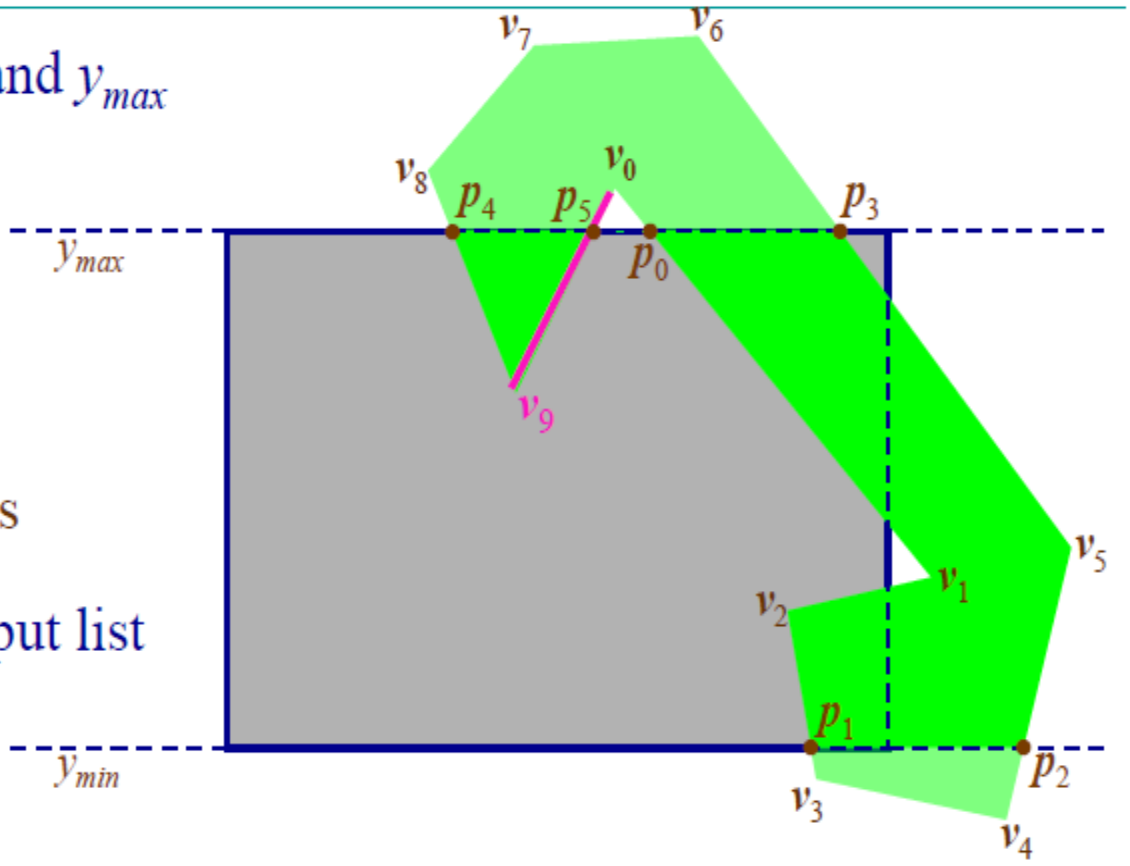
Clip first to y_{min} and y_{max}

vertex: v_9

inside region: yes

line intersect boundary: yes

add v_9, p_5 to output list

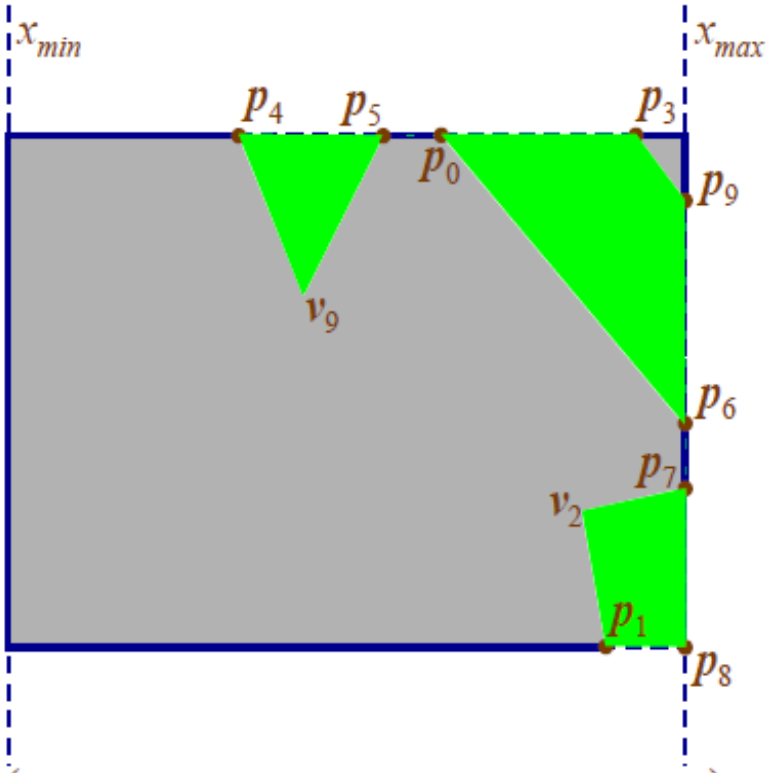


Input vertex list: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$

Output vertex list: $p_0, v_1, v_2, p_1, p_2, v_5, p_3, p_4, v_9, p_5$

Polygon Clipping: Example (cont.)

Now post-process



Output vertex list: $(p_0, p_6, p_7, v_2, p_1, p_8, p_9, p_3, p_4, v_9, p_5)$

Post-process: $(p_0, p_6, p_9, p_3,)$ and (p_7, v_2, p_1, p_8) and (v_4, v_9, p_5)