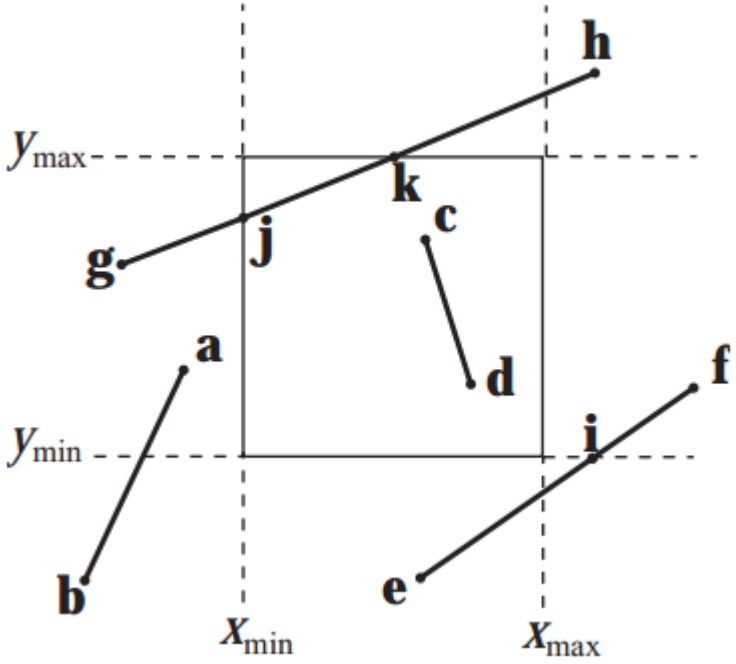
RECORTE

Prof. Dr. Bianchi Serique Meiguins
Prof. Dr. Carlos Gustavo Resque dos Santos

Recorte

- Surgiu da necessidade de <u>não</u> rasterizar elementos que estejam fora da tela (ou do framebuffer)

- Ex: para linhas avaliar três casos:
  - Elemento inteiramente dentro da tela
  - Elemento inteiramente fora da tela
  - Elemento possui parte dentro e parte fora da tela



Vantagens do Recorte

- Poupa recursos computacionais
  - especialmente em aplicações com muitos elementos gráficos que não aparecem em tela.
- Evita problemas com “OutOfBounds”
- Pode ser utilizado aplicado tanto em 2D quanto 3D
- A tela não precisa ser um retângulo (Para alguns algoritmos)

Tipos de Recorte

- Recorte de Pontos
- Recorte de Linhas
- Recorte de Polígonos

Recorte de Pontos

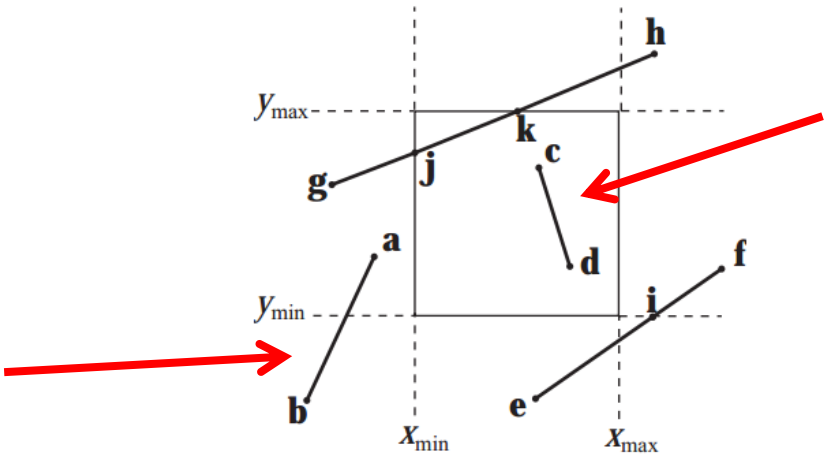
- O mais trivial:
- Basta verificar se o ponto está contido na tela.
- Ou seja:
  - $x_{min} \leq x \leq x_{max}$ E $y_{min} \leq y \leq y_{max}$

Recorte de Linhas

Recorte de Linhas

- A primeira vista seria apenas calcular as interseções das linhas com as bordas da tela

- Entretanto temos problemas com cálculos desnecessários nos casos triviais:
  - Totalmente fora
  - Totalmente dentro



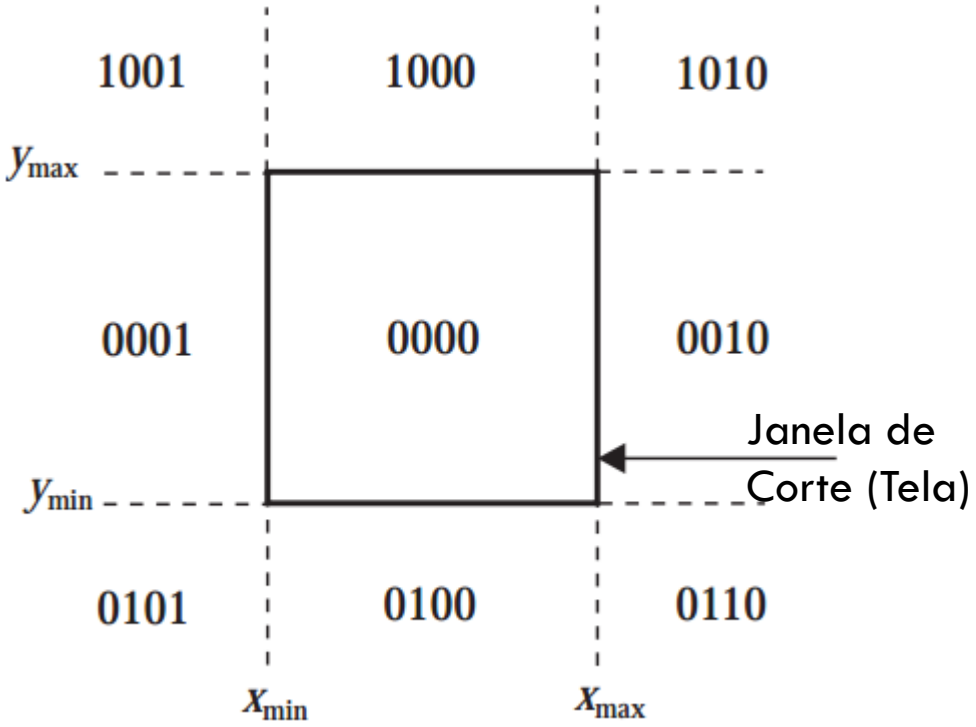
Algoritmo de Cohen-Sutherland

- Identifica, de forma eficiente, que linhas são trivialmente aceitas ou rejeitadas
- Dispensa cálculos de interseções nestes casos
- Utiliza operações binárias (apropriado para linguagens de baixo nível)

Algoritmo de Cohen-Sutherland

- Primeiro passo:
  - Definir um código binário para os pontos da reta
    - $sign(n)$ retorna 0 se $n >= 0$ e 1 se $n < 0$

- 1° bit: sign($y_{max} - y$)
- 2° bit: sign($y - y_{min}$)
- 3° bit: sign($x_{max} - x$)
- 4° bit: sign($x - x_{min}$)

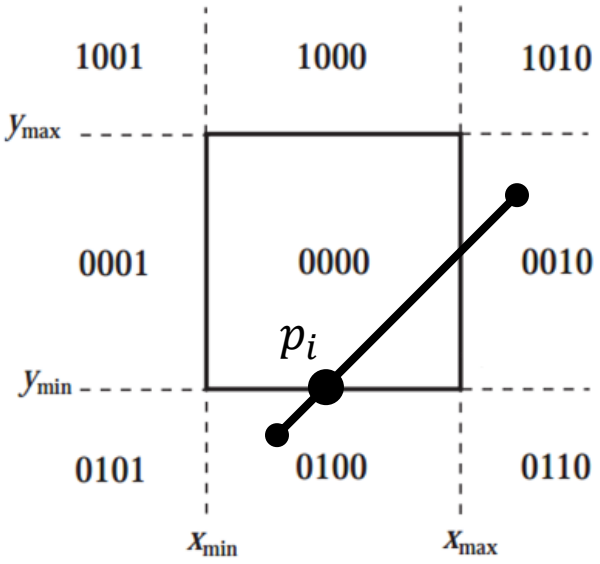


Algoritmo de Cohen-Sutherland

- Segundo Passo:
  - Verificar se a linha está totalmente dentro ou totalmente fora
  - Considerando c_1 e c_2 os códigos de uma reta
    - Se $c_1\ OU\ c_2 = 0000$, então linha está totalmente dentro
    - Se $c_1\ E\ c_2 \neq 0000$, então linha está totalmente fora

Algoritmo de Cohen-Sutherland

- Terceiro Passo:
  - Calcular a interseção (p_i) da reta com a linha da tela na qual ocorre a primeira diferença bits dos pontos
  - Considerar p_i e o ponto que tem zero na posição da diferença como uma nova reta
  - Usar essa reta recursivamente

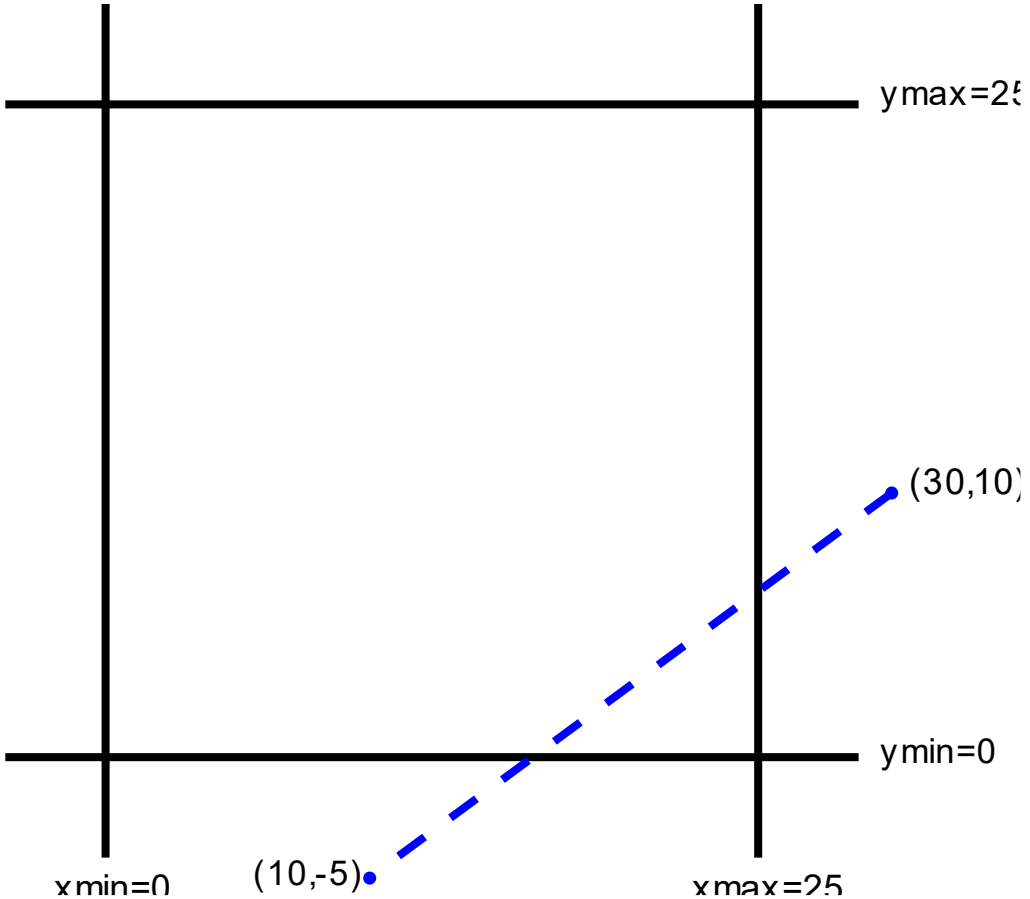


Algoritmo de Cohen-Sutherland

- CS_Clip(p1,p2,xmin,xmax,ymin,ymax):
  - int c1 = mkcode(p1) /*gera o código binário para o p1*/
  - int c2 = mkcode(p2) /*gera o código binário para o p2*/
  - Se ((c1|c2) == 0) desenhaLinha(p1,p2) /*totalmente dentro*/
  - Senão Se ((c1&c2) != 0) return null /*totalmente fora*/
  - Senão:
    - int difBit = findDifBit(c1,c2) /* encontra 1° bit com valor 1 */
    - /* calcula a intersecção i entre a reta e a borda do monitor (referente ao bit encontrado anteriormente) */
    - Point pi = intersect_lines(find_Window_Line(difBit), (p1,p2))
    - /* Usa o ponto que tem 0 nesse bit e a intersecção i recursivamente */
    - Se (getBit(c1, difBit) == 0):
      - CS_Clip(p1,pi,xmin,xmax,ymin,ymax)
    - Else:
      - CS_Clip(pi, p2,xmin,xmax,ymin,ymax)

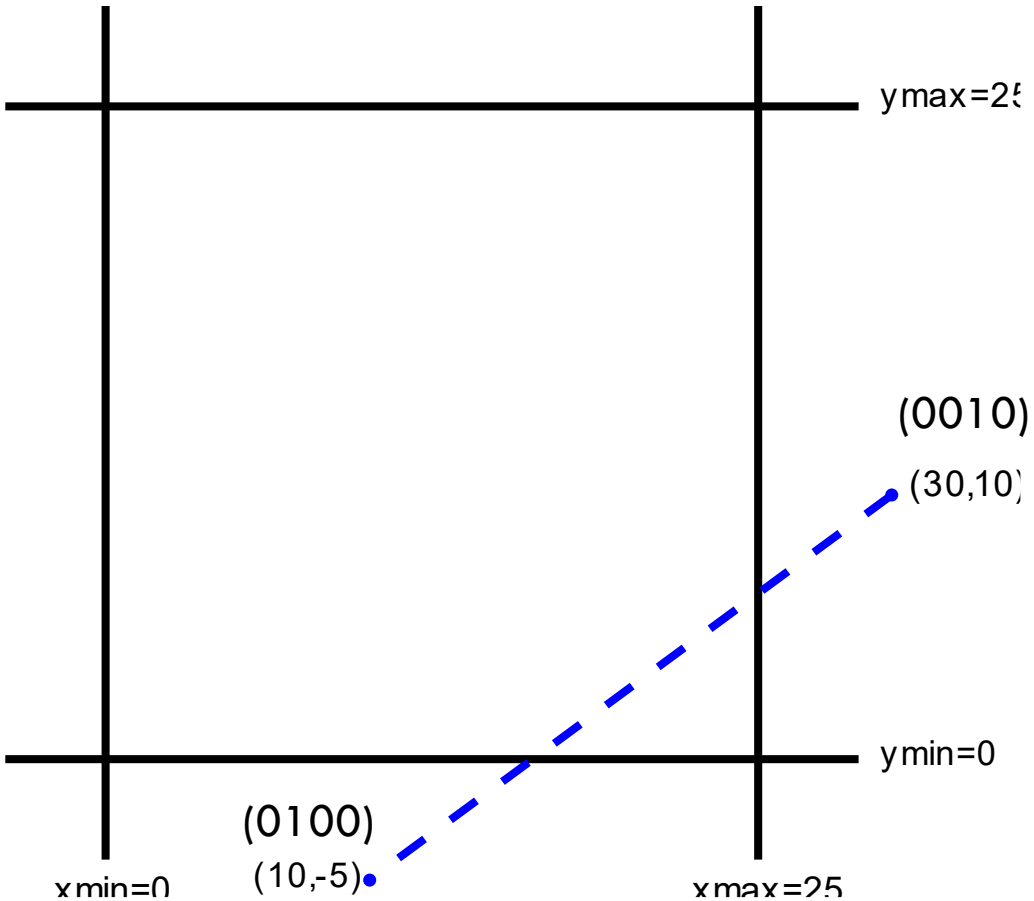
Exemplo do Alg. de CS

- Aplicar o algoritmo de Cohen-Sutherland no seguinte caso
  - p1=(10,-5)
  - p2=(30,10)
  - Xmin=0
  - Xmax=25
  - Ymin=0
  - Ymax=25



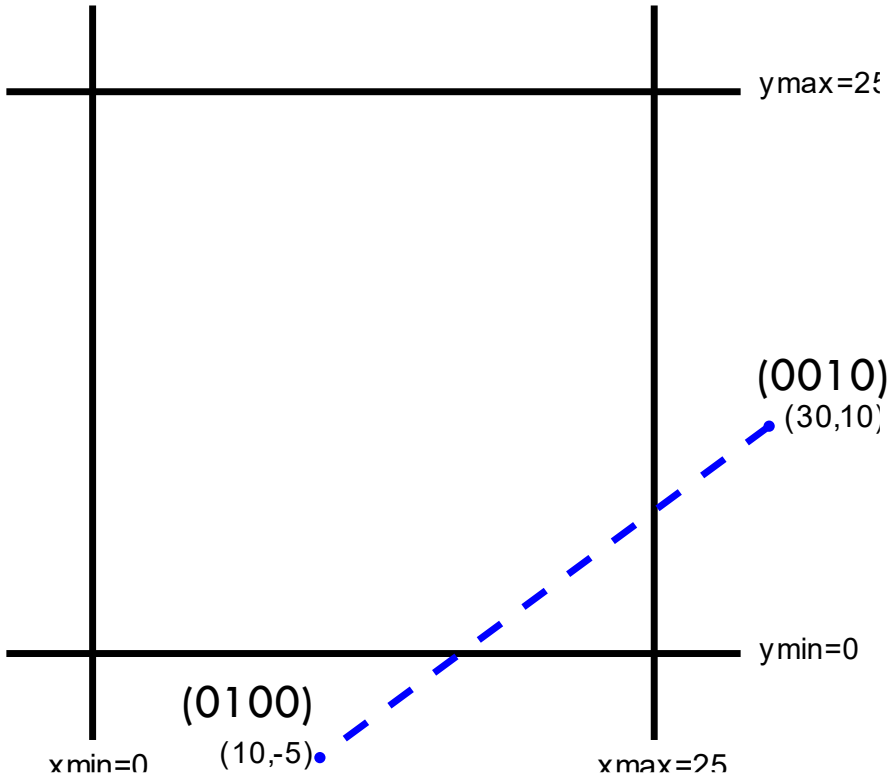
Exemplo do Alg. de CS

- 1° Passo: Encontrar os códigos binários para os pontos
  - C1 = 0100
  - C2 = 0010



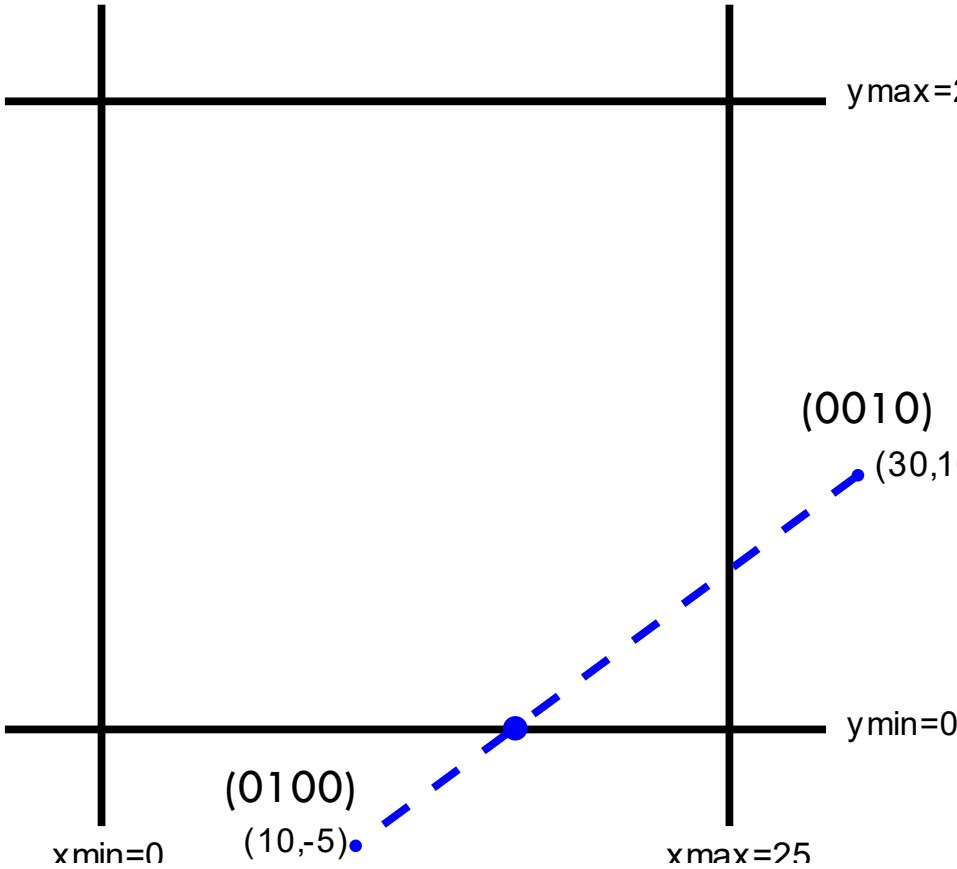
Exemplo do Alg. de CS

- 2° Passo: Verificar casos triviais
  - Trivialmente aceito?
  - C1 || C2 = 0000
    - 0100 || 0010 = 0110
  - Trivialmente rejetado?
  - C1 && C2 != 0000
    - 0100 && 0010 = 0000



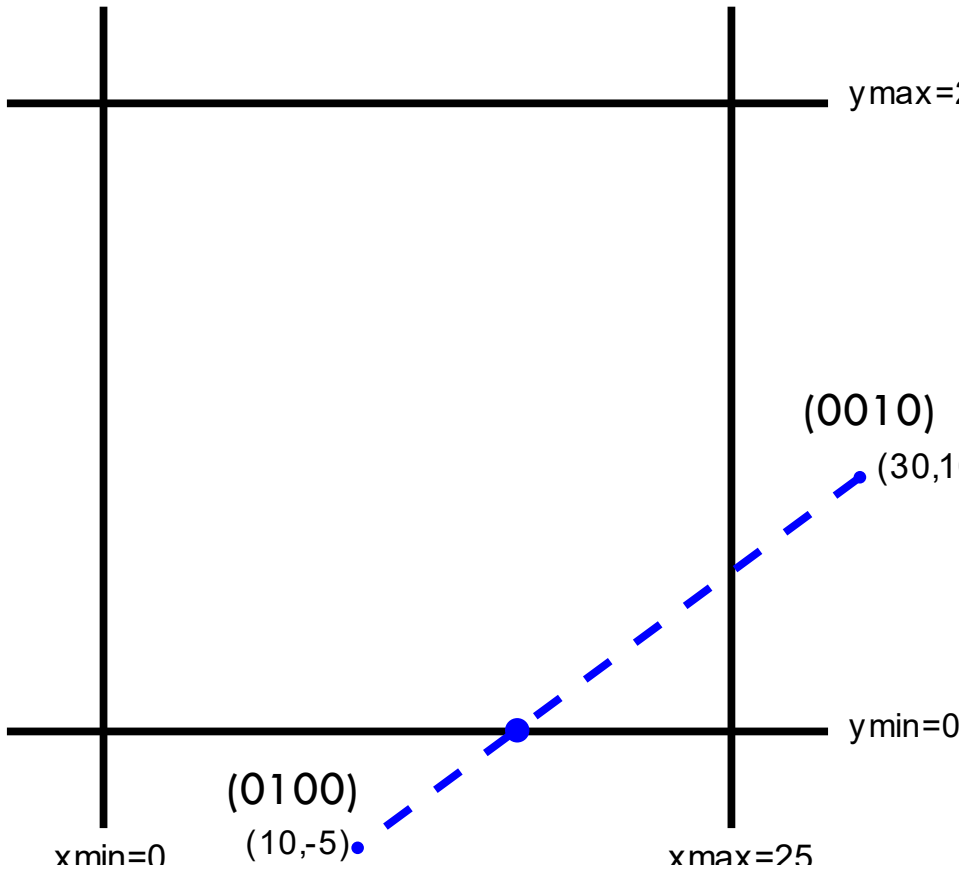
Exemplo do Alg. de CS

- 3° Passo: Encontrar a intersecção
  - $x_i = \frac{(y_i - y_1)(x_2 - x_1)}{(y_2 - y_1)} + x_1$
  - $y_i = \frac{(x_i - x_1)(y_2 - y_1)}{(x_2 - x_1)} + y_1$



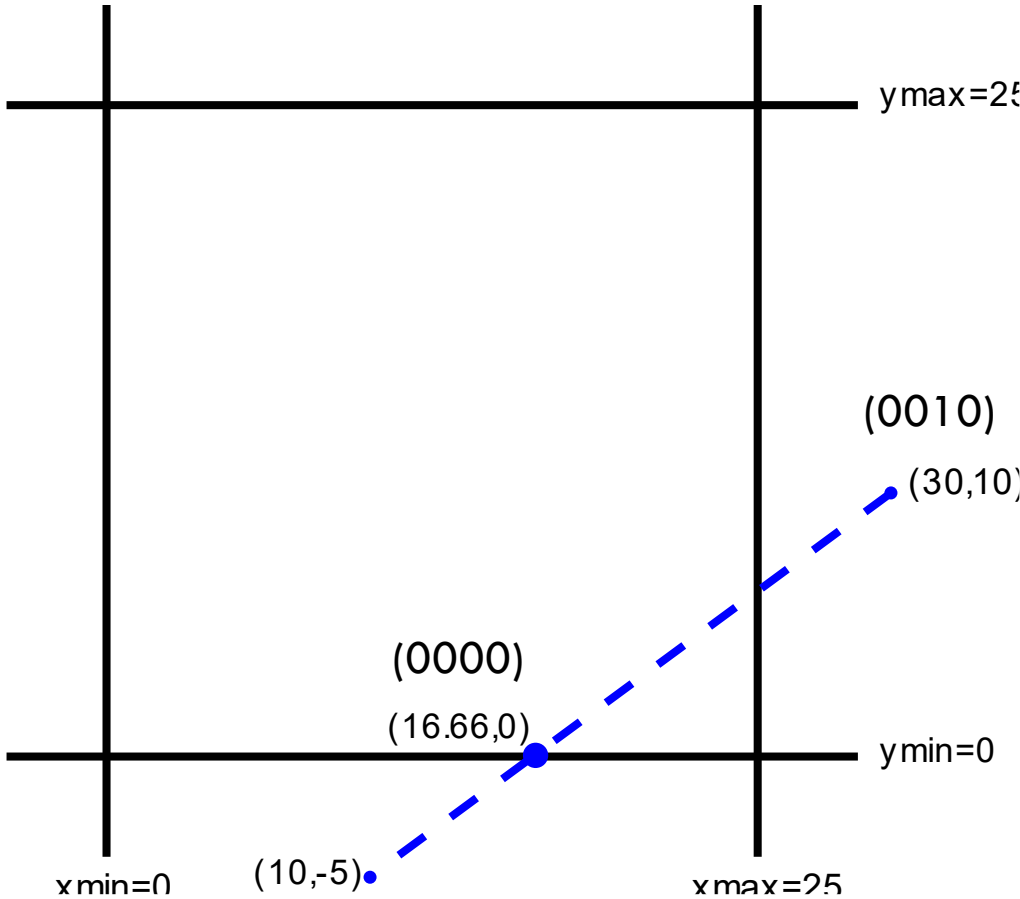
Exemplo do Alg. de CS

- 3° Passo: Encontrar a intersecção
  - $x_i = \frac{(y_i - y_1)(x_2 - x_1)}{(y_2 - y_1)} + x_1$
  - $x_i = \frac{(0-(-5))(30-10)}{(10-(-5))} + 10$
  - $x_i = \frac{5 \times 20}{15} + 10$
  - $x_i = 16.66$



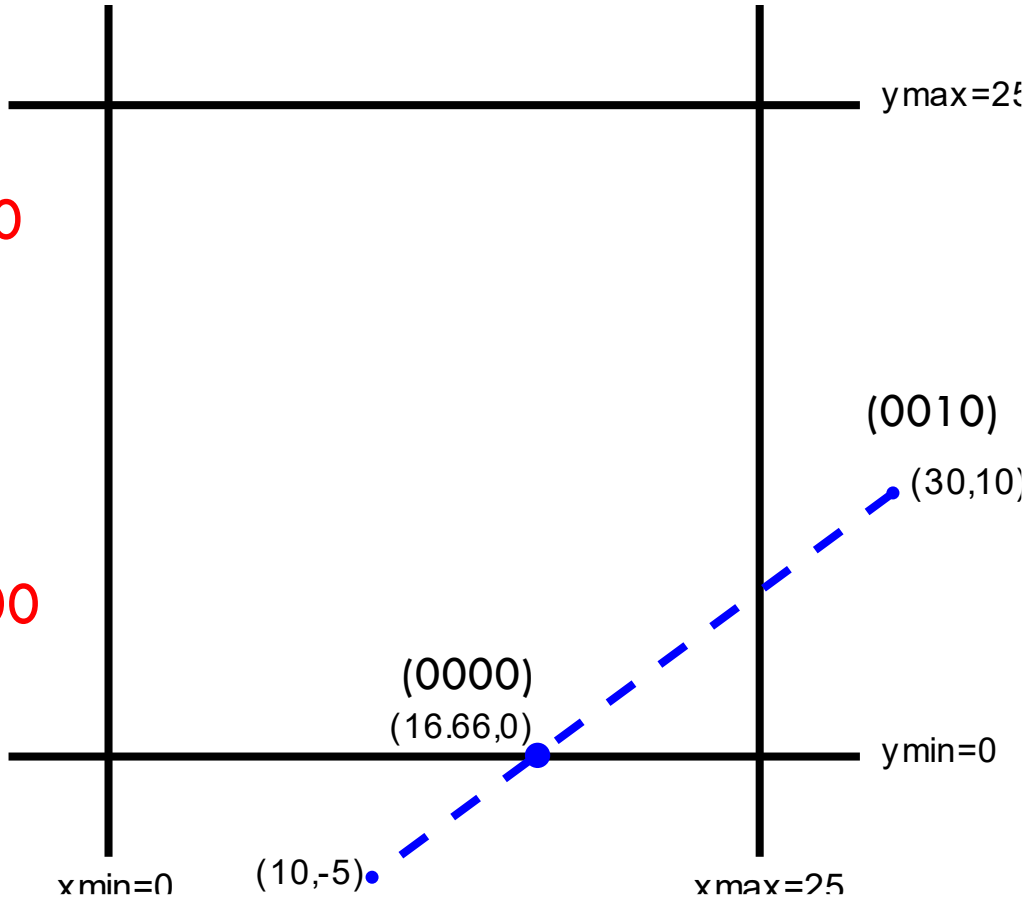
Exemplo do Alg. de CS

- 4° Passo: utiliza os novos pontos
  - P1=(16.66, 0)
  - P2=(30, 10)
  - C1=0000
  - C2=0010



Exemplo do Alg. de CS

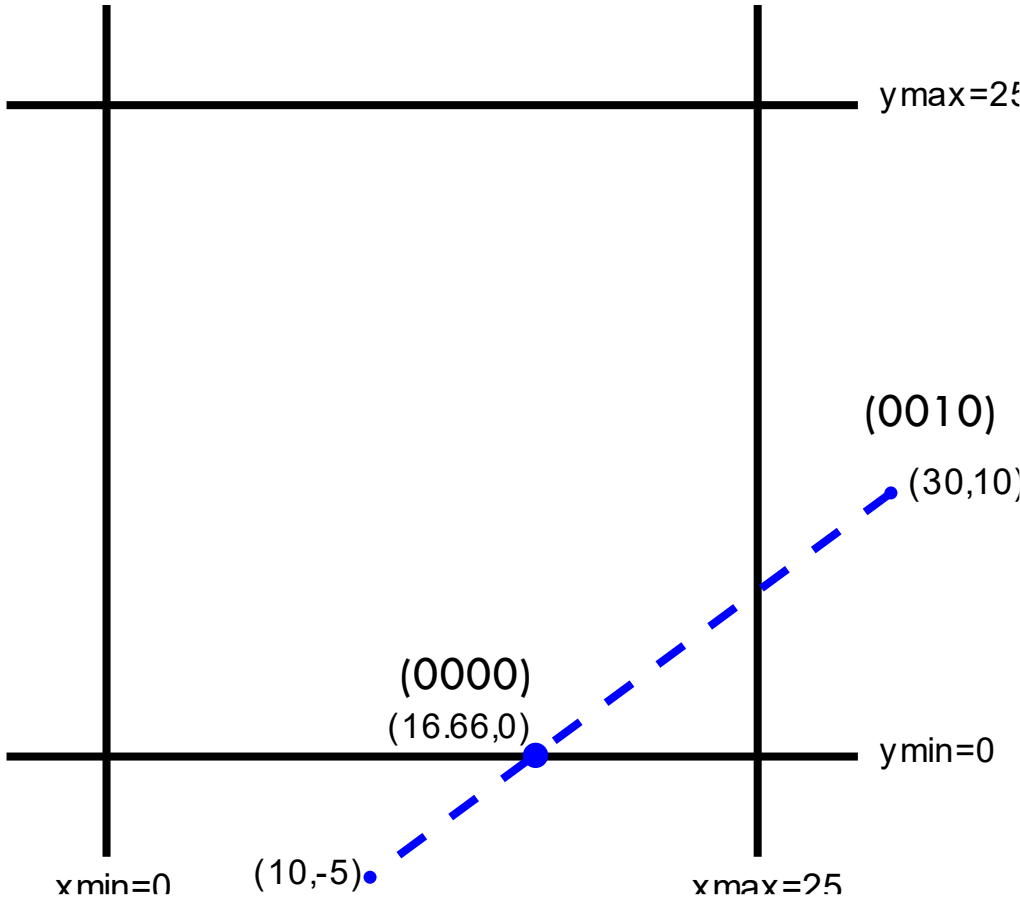
- 2°.1 Passo: Verificar casos triviais
  - Trivialmente aceito?
  - C1 || C2 = 0000
    - 0000 || 0010 = 0010
  - Trivialmente rejetado?
  - C1 && C2 != 0000
    - 0000 && 0010 = 0000



Exemplo do Alg. de CS

- 3°.1 Passo: Encontrar a intersecção
  - $y_i = \frac{(x_i - x_1)(y_2 - y_1)}{(x_2 - x_1)} + y_1$
  - $y_i = \frac{8.34 \times 10}{13.34}$
  - $y_i = 6.25$

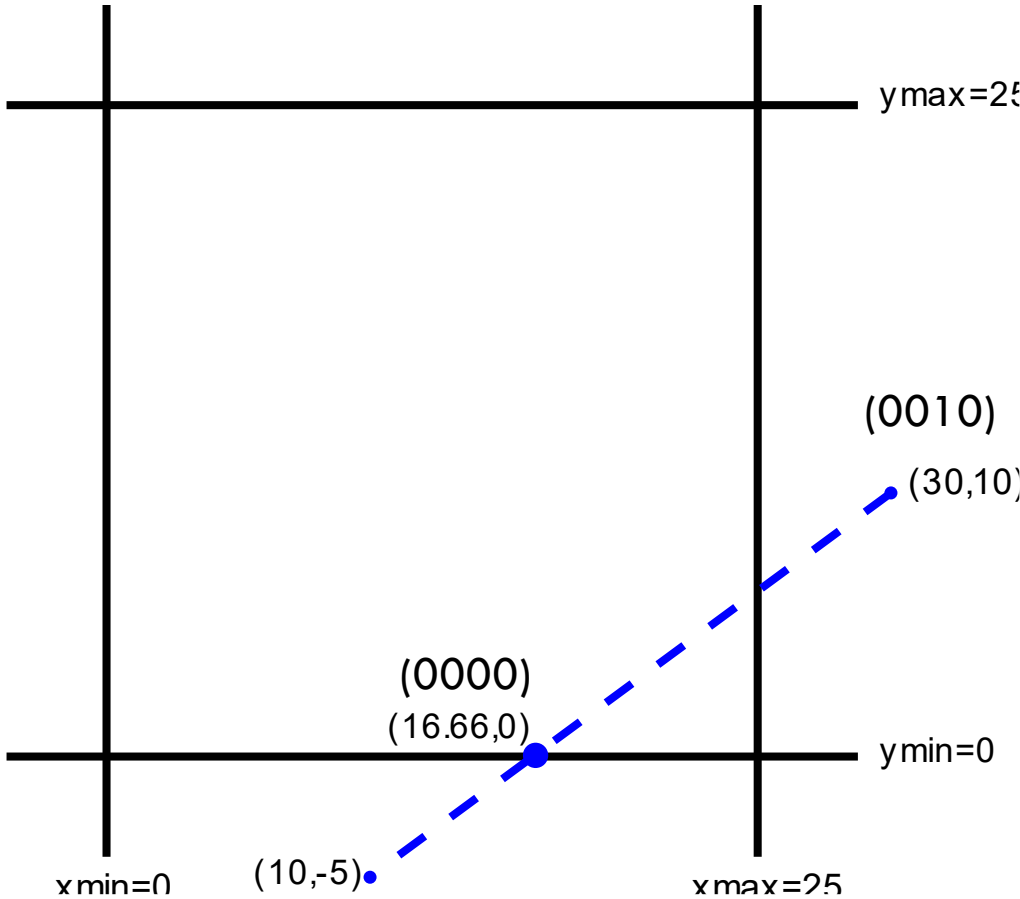
- 4°.1 Passo: Utiliza os novos pontos



Exemplo do Alg. de CS

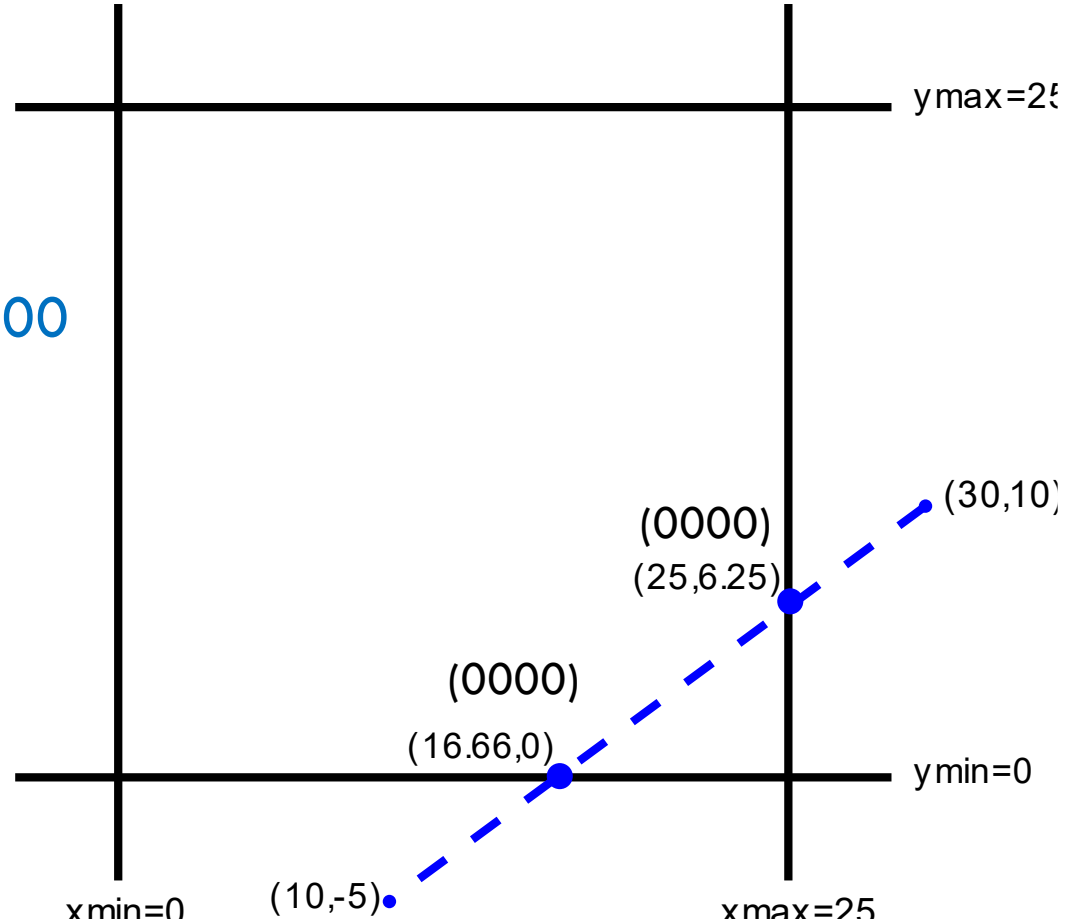
- 3°.1 Passo: Encontrar a intersecção
  - $y_i = \frac{(x_i - x_1)(y_2 - y_1)}{(x_2 - x_1)} + y_1$
  - $y_i = \frac{8.34 \times 10}{13.34}$
  - $y_i = 6.25$

- 4°.1 Passo: Utiliza os novos pontos



Exemplo do Alg. de CS

- 2°.2 Passo: Verificar casos triviais
  - Trivialmente aceito?
  - C1 || C2 = 0000
    - 0000 || 0000 = 0000
  - Então a linha está pronta para a rasterização!



Algoritmo de subdivisão por ponto médio

- Motivação:
  - Algoritmo anterior necessitava de cálculos para obtenção da interseção da linha com limites da janela

- Caso particular do Algoritmo anterior, proposto por Sproull e Sutherland

Algoritmo de subdivisão por ponto médio

- Utilizar apenas soma e divisão por 2 recursivamente.
  - um teste inicial é aplicado para detectar linhas trivialmente aceitas ou rejeitadas
  - linhas para as quais o teste inicial falha, são subdivididas em 2 partes iguais
    - $x_m = \frac{(x_1 + x_2)}{2}$
    - $y_m = \frac{(y_1 + y_2)}{2}$

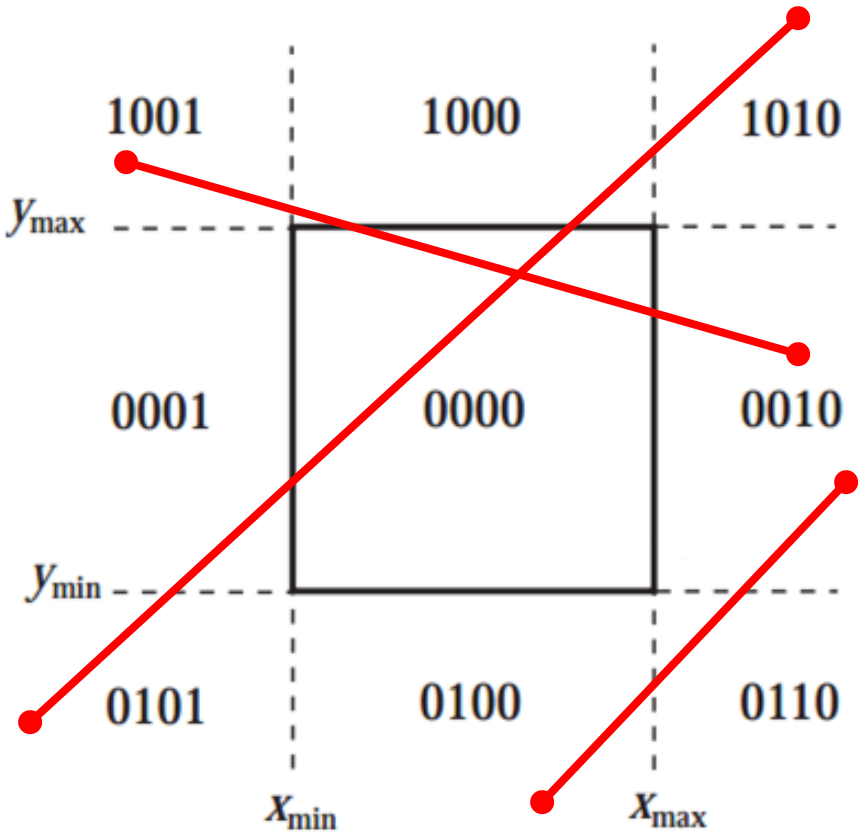
Algoritmo de subdivisão por ponto médio

- O teste é aplicado nas duas retas resultantes pela divisão (p_1, p_m) e (p_m, p_2)

- O Teste segue recursivamente:
  - Aceitando retas totalmente dentro
  - E rejeitando retas totalmente fora

Algoritmo de subdivisão por ponto médio

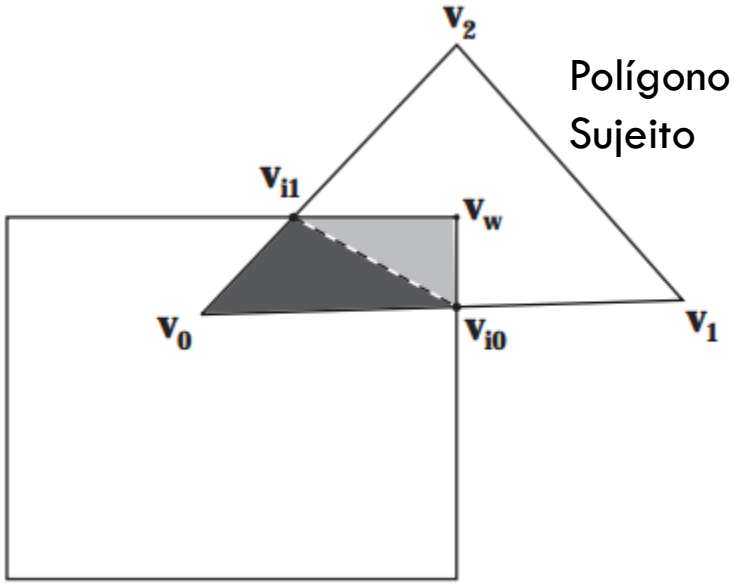
- Exemplos:



Recorte de Polígonos

Recorte de Polígonos

- A priori basta cortar as retas do polígono e ligar os pontos... Porém isso gera um problema

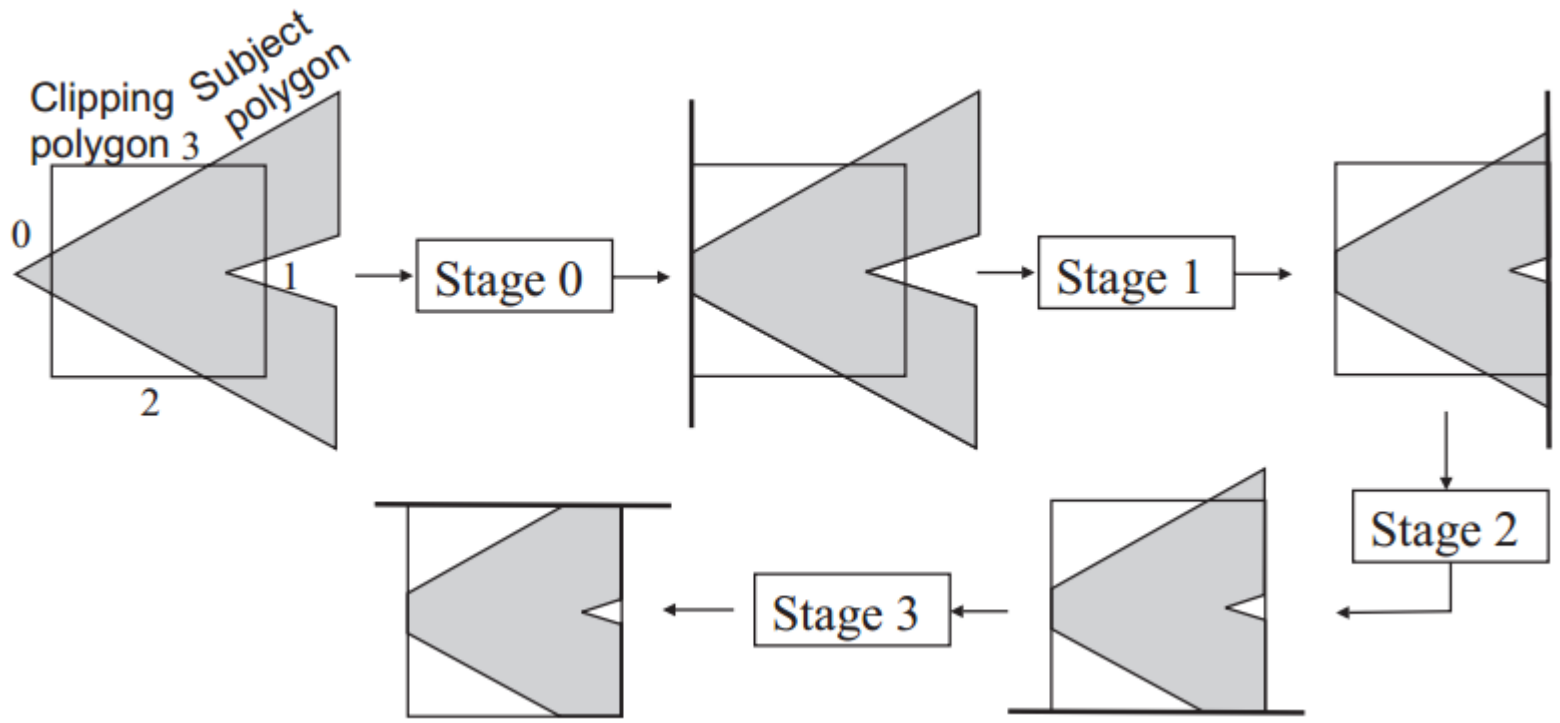


Algoritmo de Sutherland-Hodgman

- Marco no desenvolvimento da Computação Gráfica
  - Uma vez que os modelos tridimensionais geralmente são constituídos de malhas de polígonos
- Só funciona para quando o polígono de recorte é convexo
- Realiza as operações em etapas

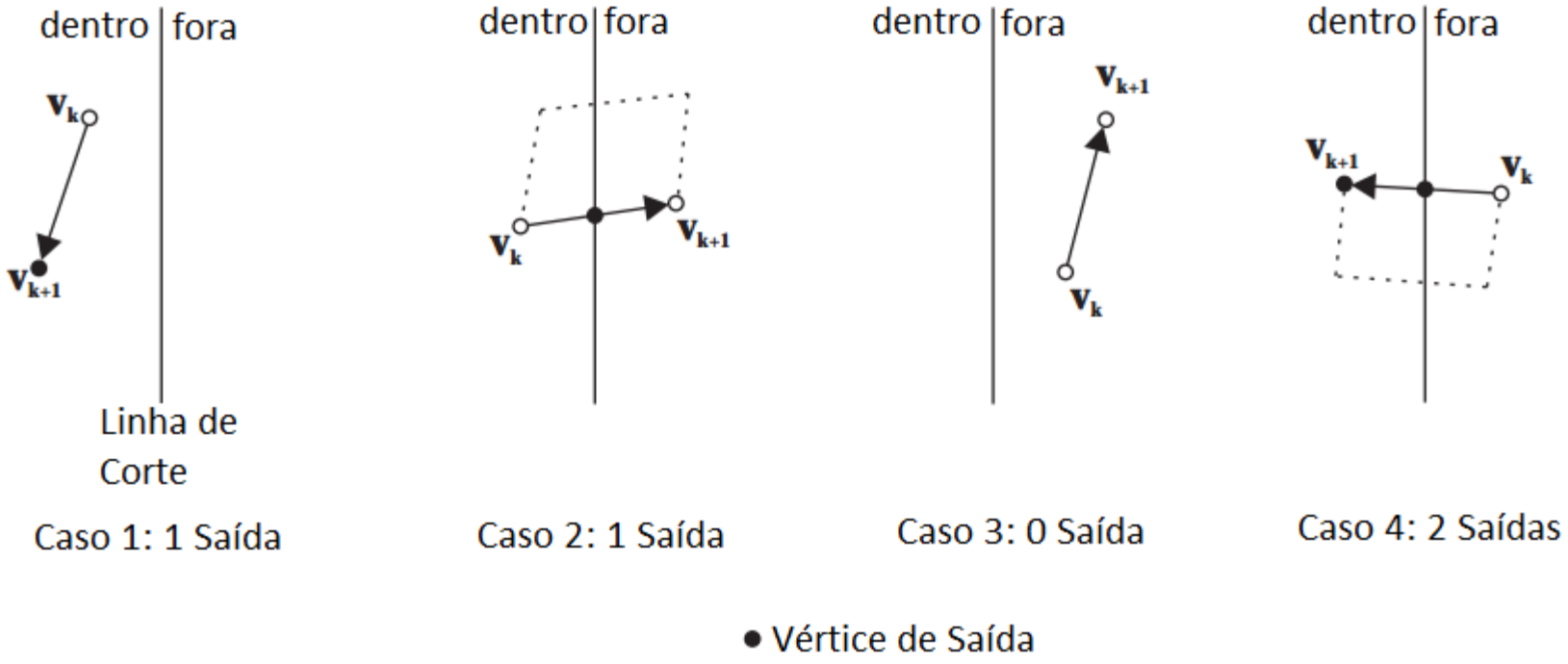
Algoritmo de Sutherland-Hodgman

- Recortar o Polígono sujeito para cada lado do polígono de recorte



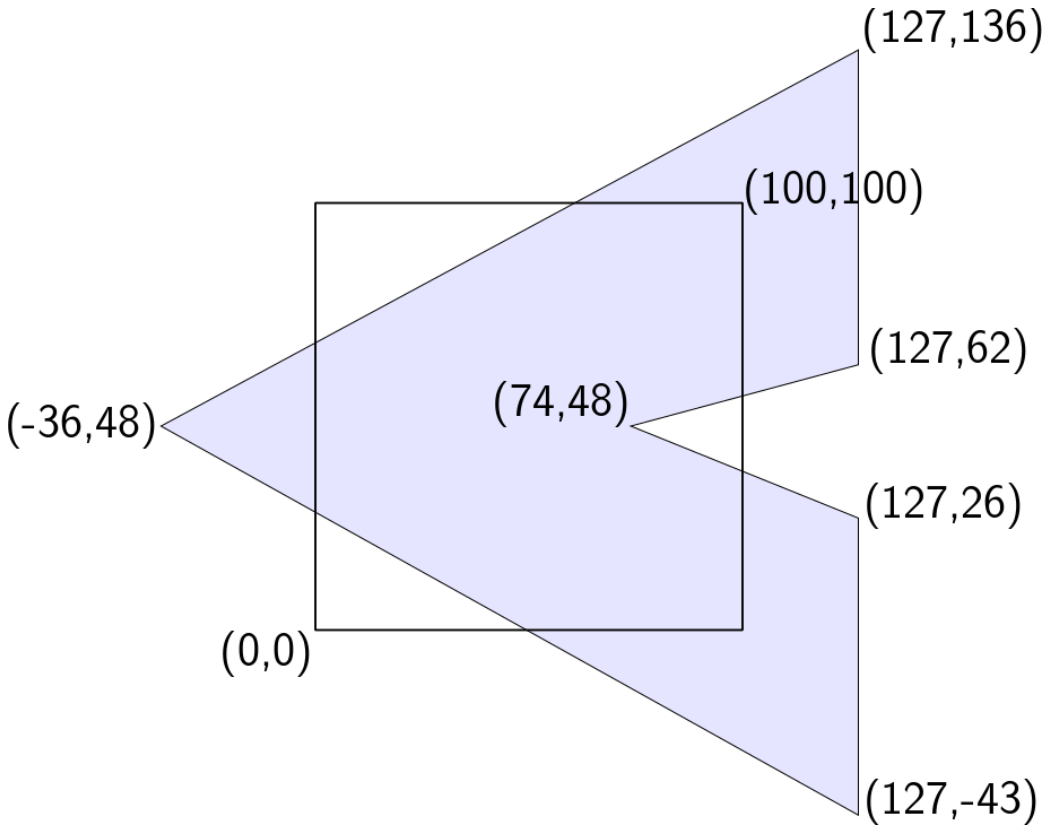
Algoritmo de Sutherland-Hodgman

- Casos a serem verificados:
  - Adiciona os vértices de saída em uma lista
  - Essa lista forma o polígono fechado



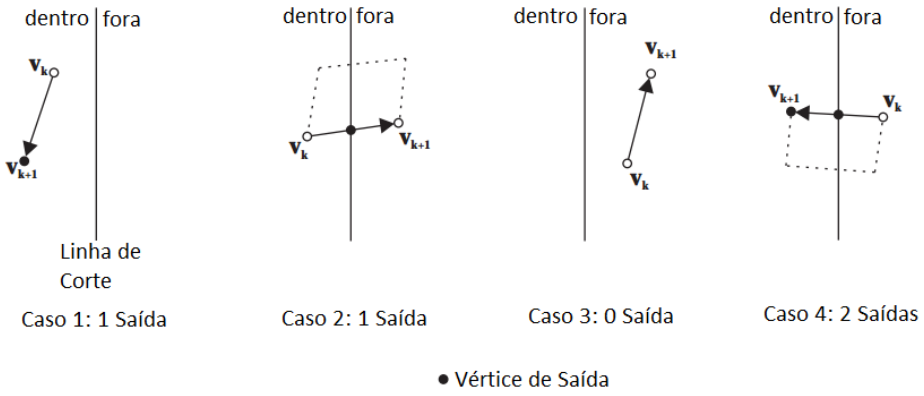
Exemplo

- Recortar o polígono abaixo utilizando o alg. de Sutherland-Hodgman



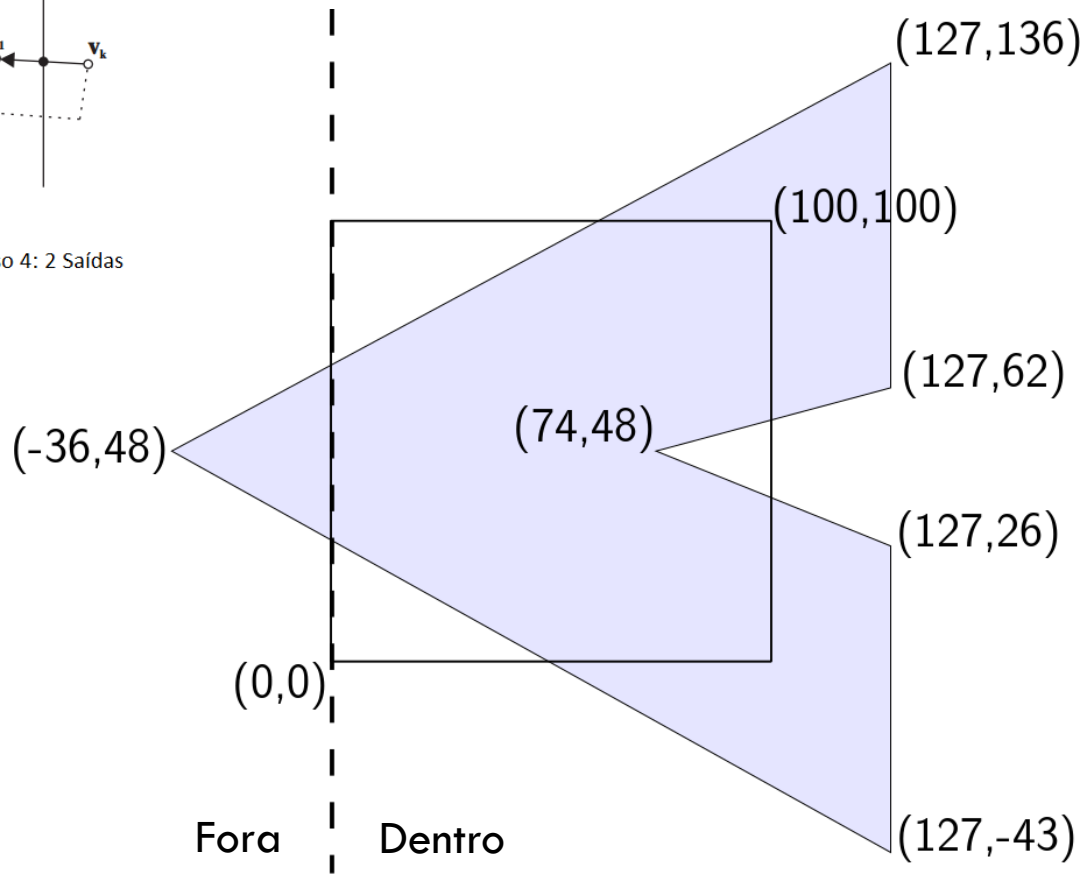
Exemplo

- 1° estágio: recortar com o lado esquerdo



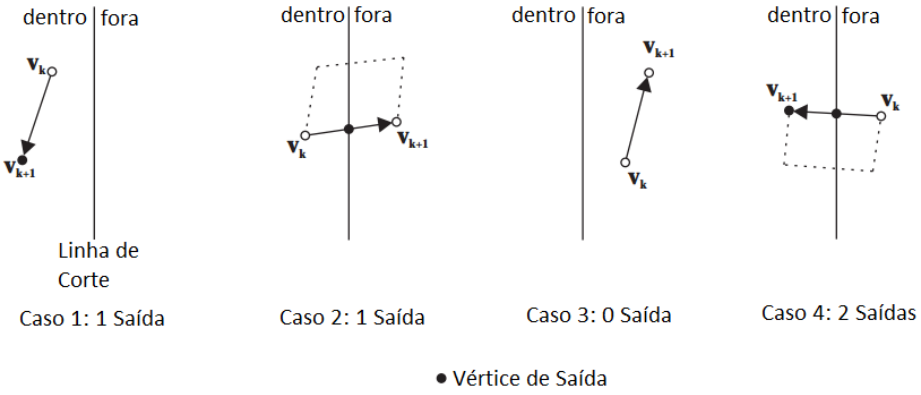
Entrada:
[(-36,48); (127,136); (127,62); (74,48); (127,26); (127,-43)]

Saída:
[(0,67); (127,136); (127,62); (74,48); (127,26); (127,-43); (0,28)]



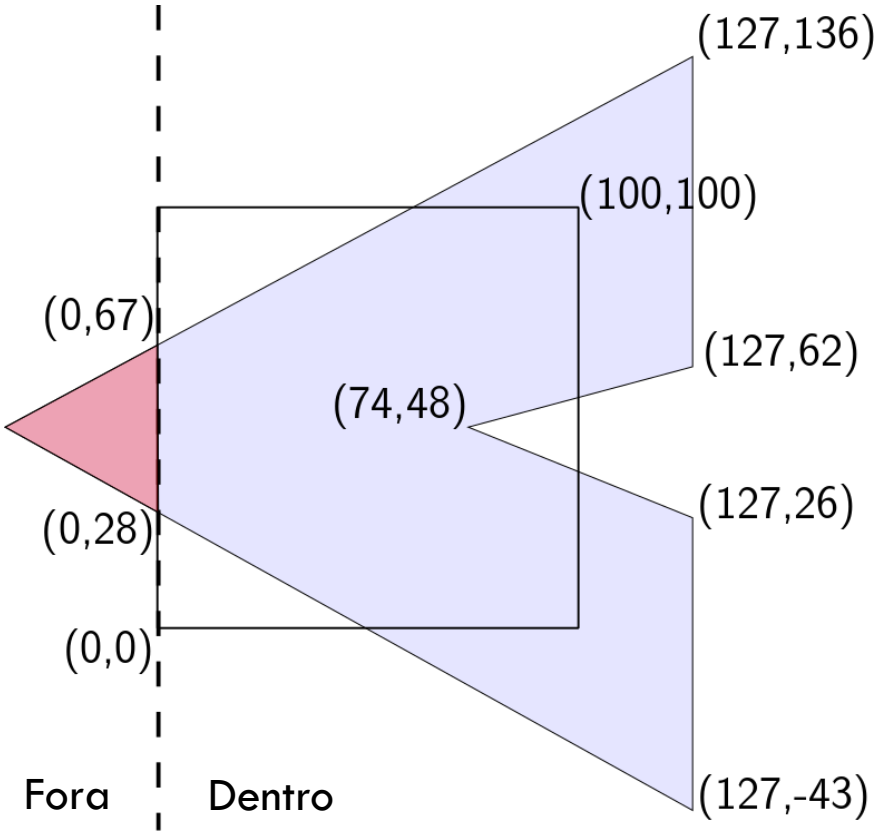
Exemplo

- 1° estágio: recortar com o lado esquerdo



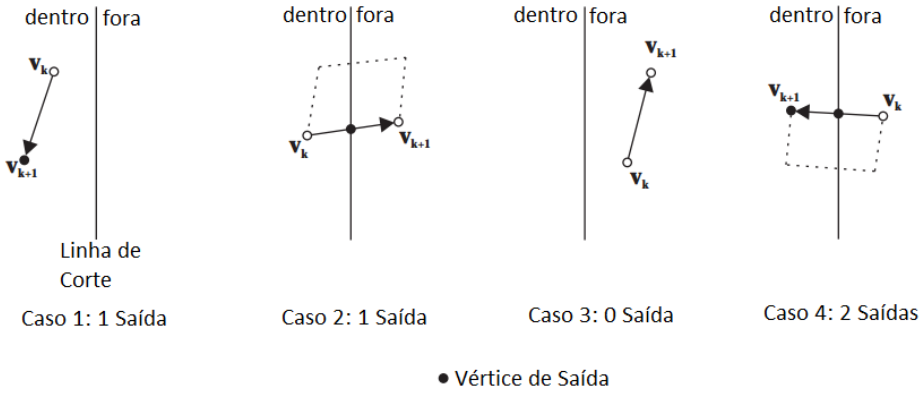
Entrada:
[(-36,48); (127,136); (127,62); (74,48); (127,26); (127,-43)]

Saída:
[(0,67); (127,136); (127,62); (74,48); (127,26); (127,-43); (0,28)]



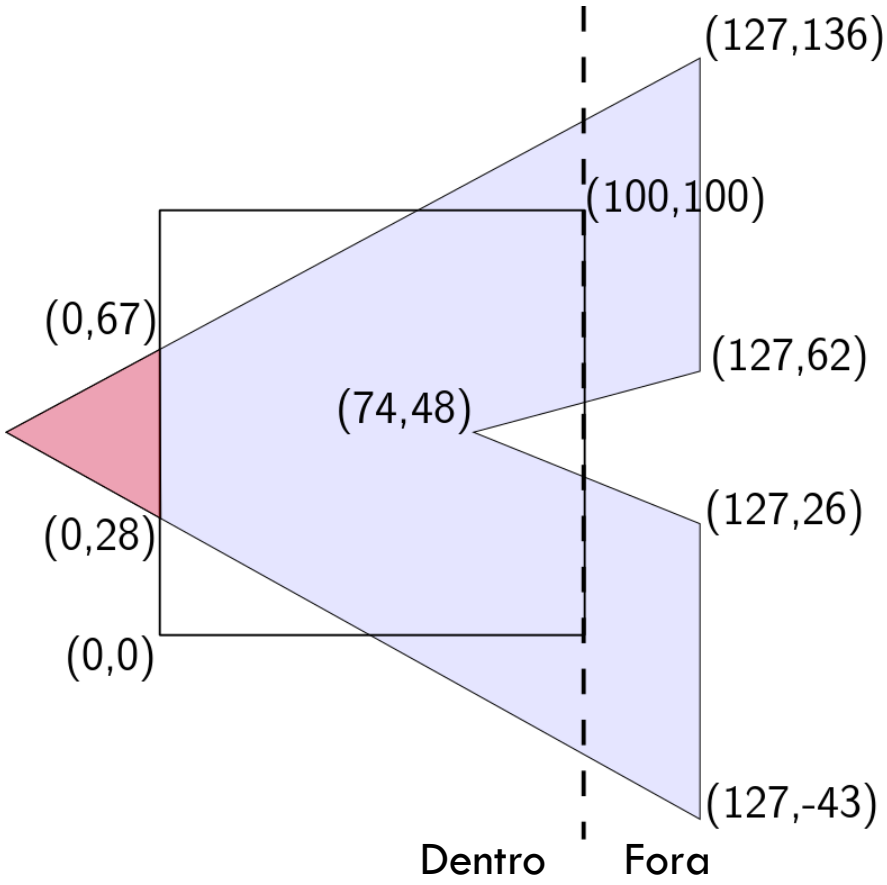
Exemplo

- 2° estágio: recortar com o lado direito



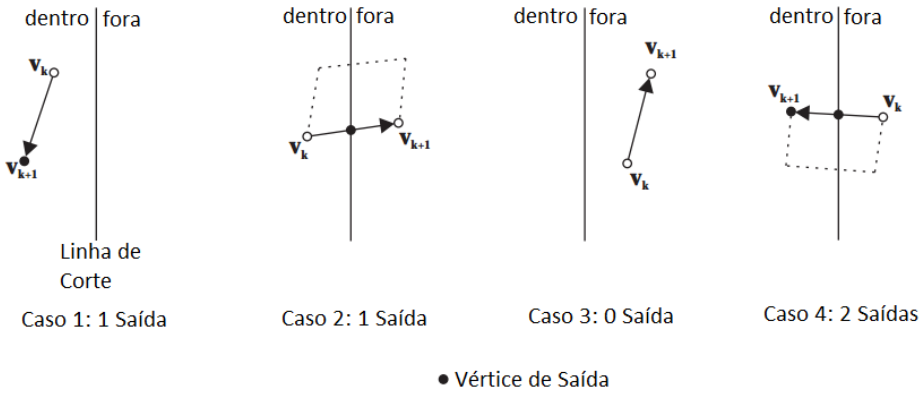
Entrada:
[(0,67); (127,136); (127,62);
(74,48); (127,26); (127,-43); (0,28)]

Saída:
[(100,121); (100,55); (74,48);
(100,37); (100,-28); (0,28); (0,67)]



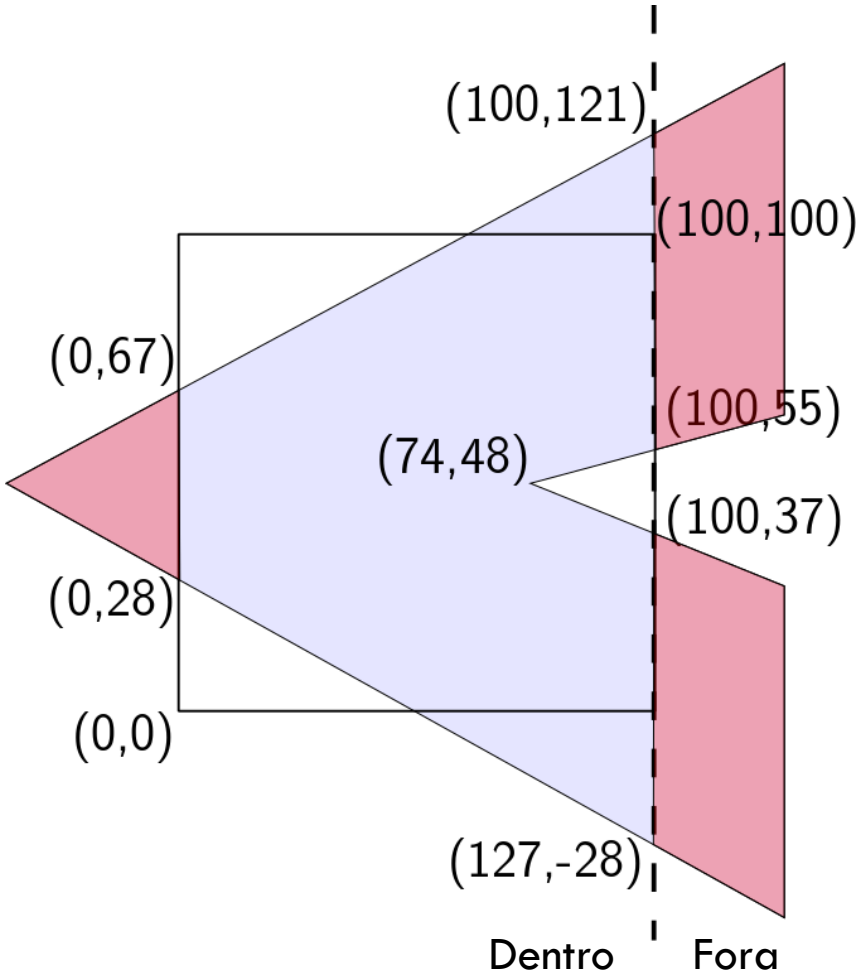
Exemplo

- 2° estágio: recortar com o lado direito



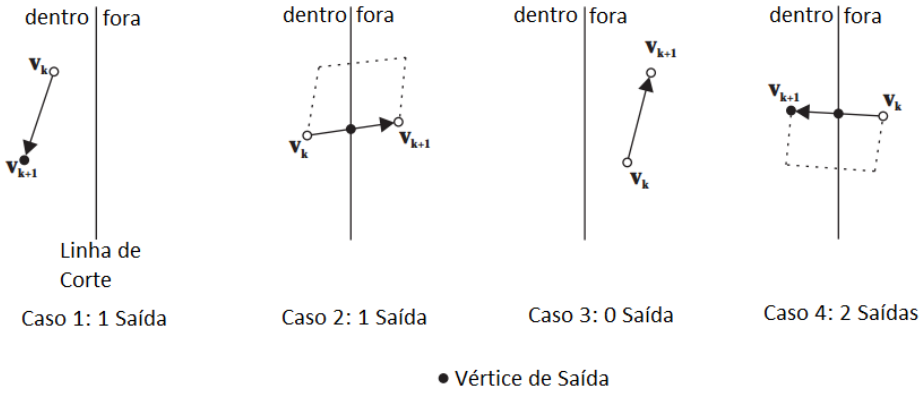
Entrada:
[(0,67); (127,136); (127,62);
(74,48); (127,26); (127,-43); (0,28)]

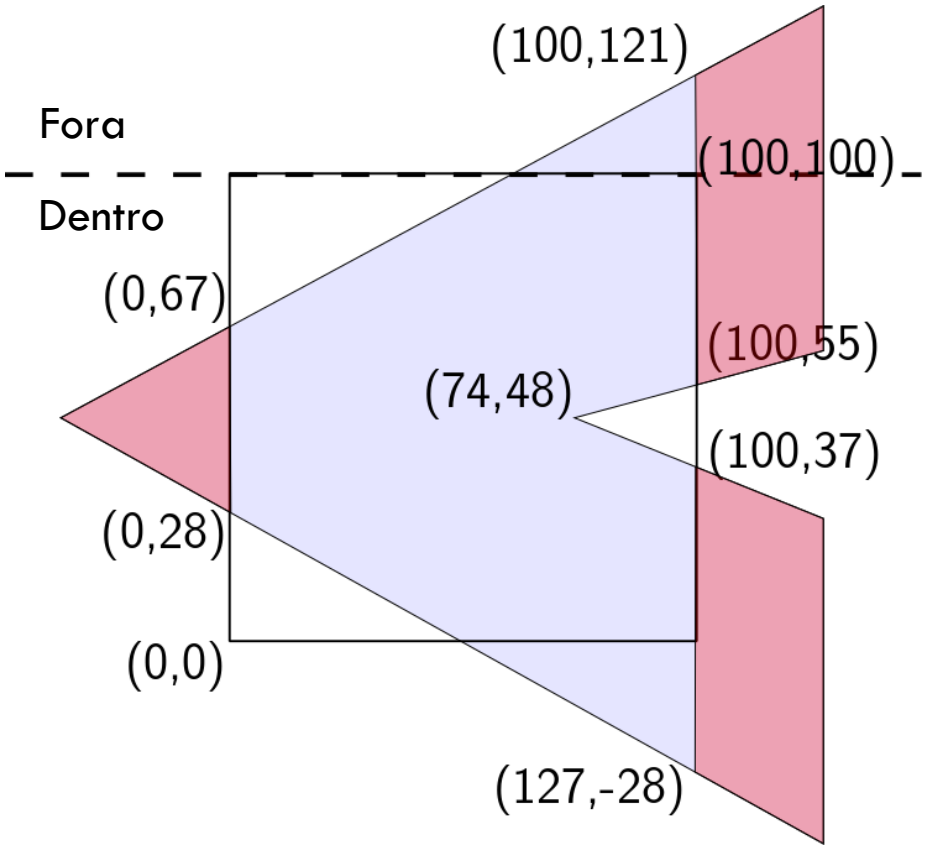
Saída:
[(100,121); (100,55); (74,48);
(100,37); (100,-28); (0,28); (0,67)]



Exemplo

- 3° estágio: recortar com o lado de cima



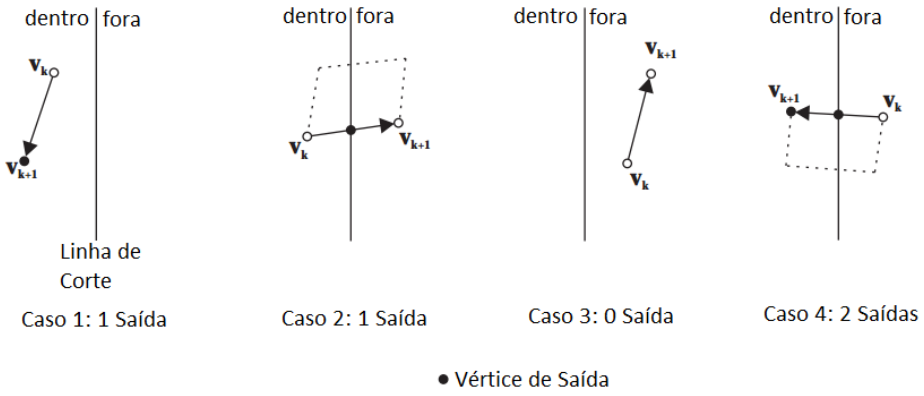


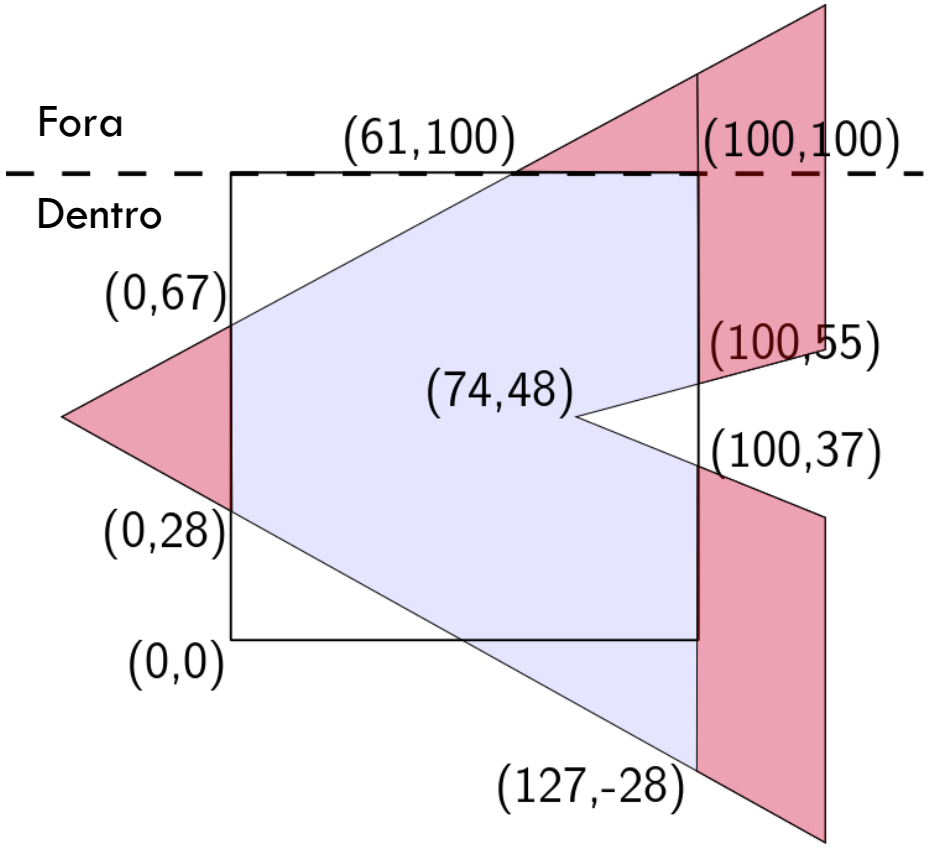
Entrada:
[(100,121); (100,55); (74,48);
(100,37); (100,-28); (0,28); (0,67)]

Saída:
[(100,100); (100,55); (74,48); (100,37);
(100,-28); (0,28); (0,67); (61,100)]

Exemplo

- 3° estágio: recortar com o lado de cima



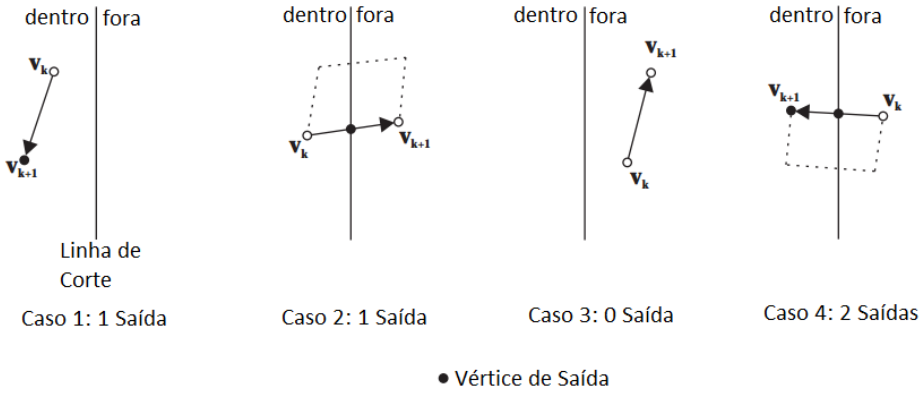


Entrada:
[(100,121); (100,55); (74,48);
(100,37); (100,-28); (0,28); (0,67)]

Saída:
[(100,100); (100,55); (74,48); (100,37);
(100,-28); (0,28); (0,67); (61,100)]

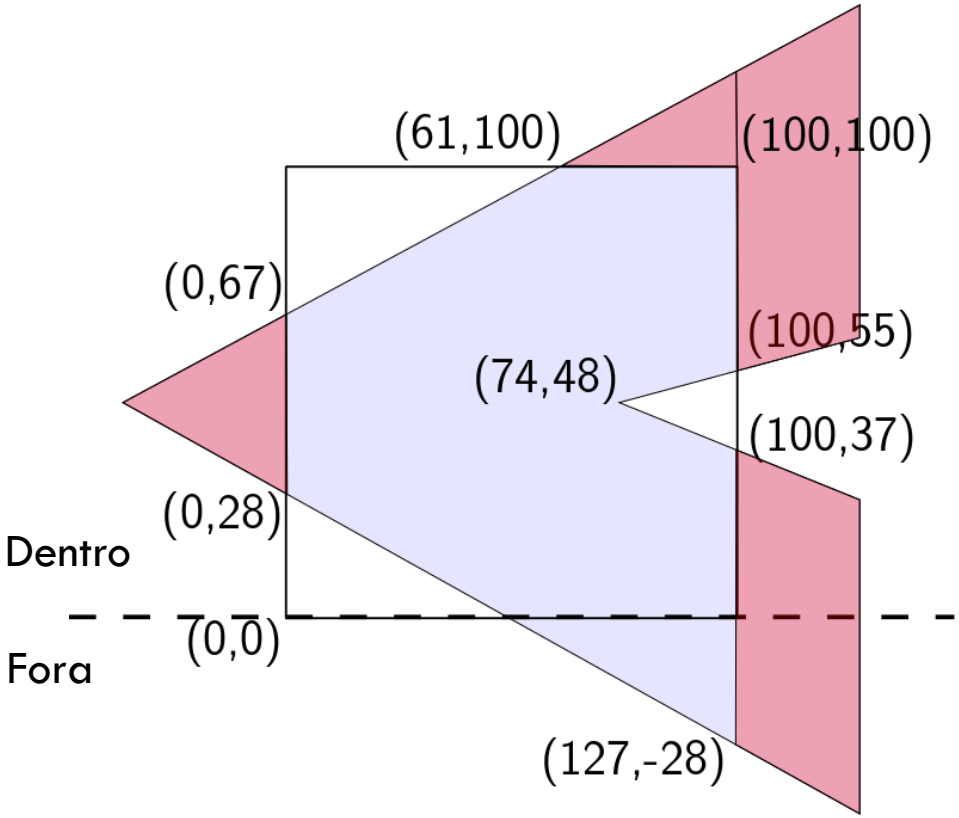
Exemplo

- 4° estágio: recortar com o lado de baixo



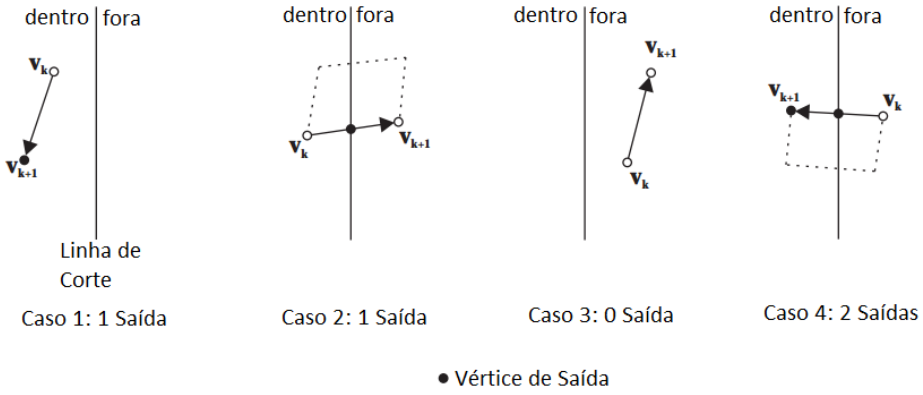
Entrada:
[(100,100); (100,55); (74,48); (100,37); (100,-28); (0,28); (0,67); (61,100)]

Saída:
[(100,55); (74,48); (100,37); (100,0); (50,0); (0,28); (0,67); (61,100); (100,100)]



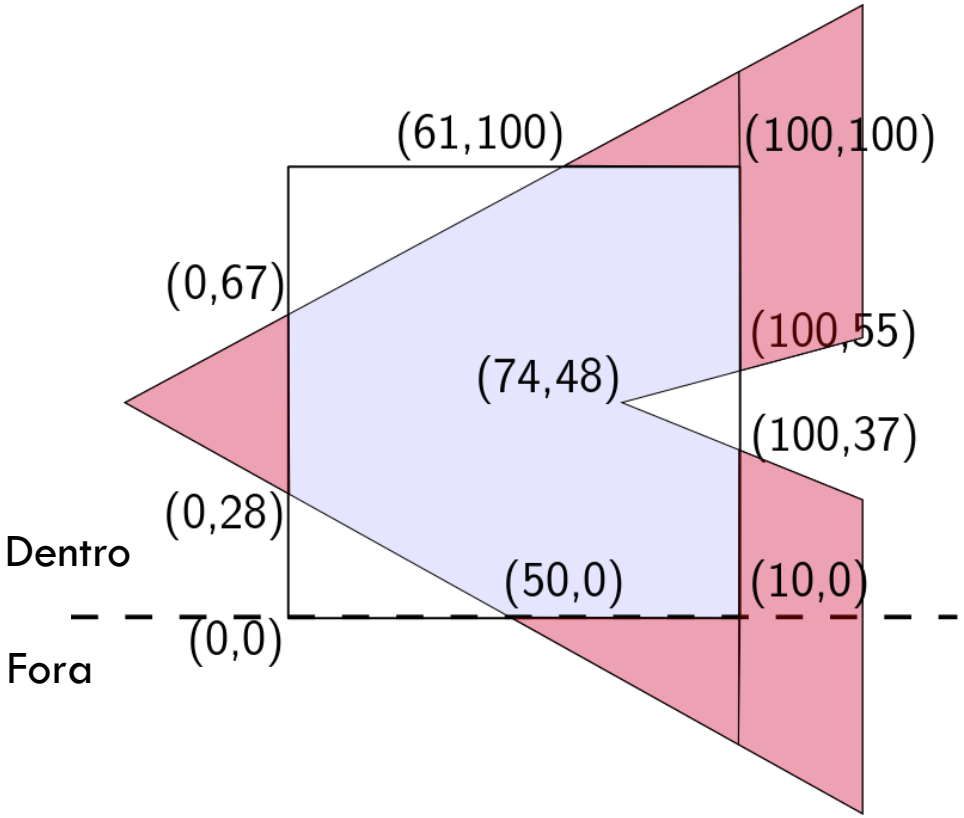
Exemplo

- 4° estágio: recortar com o lado de baixo



Entrada:
[(100,100); (100,55); (74,48); (100,37); (100,-28); (0,28); (0,67); (61,100)]

Saída:
[(100,55); (74,48); (100,37); (100,0); (50,0); (0,28); (0,67); (61,100); (100,100)]



Exemplo de Código

```
public static Point[] sutherland_hodgman(Point[] polygon, int left, int top, int right, int bottom) {

    ArrayList<Point> newPolygon = new ArrayList<>();
    for(int i=0;i<polygon.length;i++){
        Point p1 = polygon[i];
        Point p2 = polygon[(i+1)%polygon.length];
        if(p1.x>left){
            if(p2.x>left){
                //De dentro para dentro
                newPolygon.add(p2);
            }else{
                //De dentro para fora
                newPolygon.add(
                        new Point( left, Math.round((left-p1.x)*(p2.y-p1.y*1.f)/(p2.x*1.f-p1.x)+p1.y)));
            }
        }else{
            if(p2.x>left){
                //De fora para dentro
                newPolygon.add(
                        new Point( left, Math.round((left-p1.x)*(p2.y-p1.y*1.f)/(p2.x*1.f-p1.x)+p1.y)));
                newPolygon.add(p2);
            }else{
                //De fora para fora
                //Não adiciona nada
            }
        }
    }
```

Exemplo de Código

```
public static Point[] sutherland_hodgman(Point[] polygon, int left, int top, int right, int bottom) {

    ArrayList<Point> newPolygon = new ArrayList<>();
    for(int i=0;i<polygon.length;i++){
        Point p1 = polygon[i];
        Point p2 = polygon[(i+1)%polygon.length];
        if(p1.x>left){
            if(p2.x>left){
                //De dentro para dentro
                newPolygon.add(p2);
            }else{
                //De dentro para fora
                newPolygon.add(
                        new Point( left, Math.round((left-p1.x)*(p2.y-p1.y*1.f)/(p2.x*1.f-p1.x)+p1.y)));
            }
        }else{
            if(p2.x>left){
                //De fora para dentro
                newPolygon.add(
                        new Point( left, Math.round((left-p1.x)*(p2.y-p1.y*1.f)/(p2.x*1.f-p1.x)+p1.y)));
                newPolygon.add(p2);
            }else{
                //De fora para fora
                //Não adiciona nada
            }
        }
    }
```