Computação Gráfica

Recorte

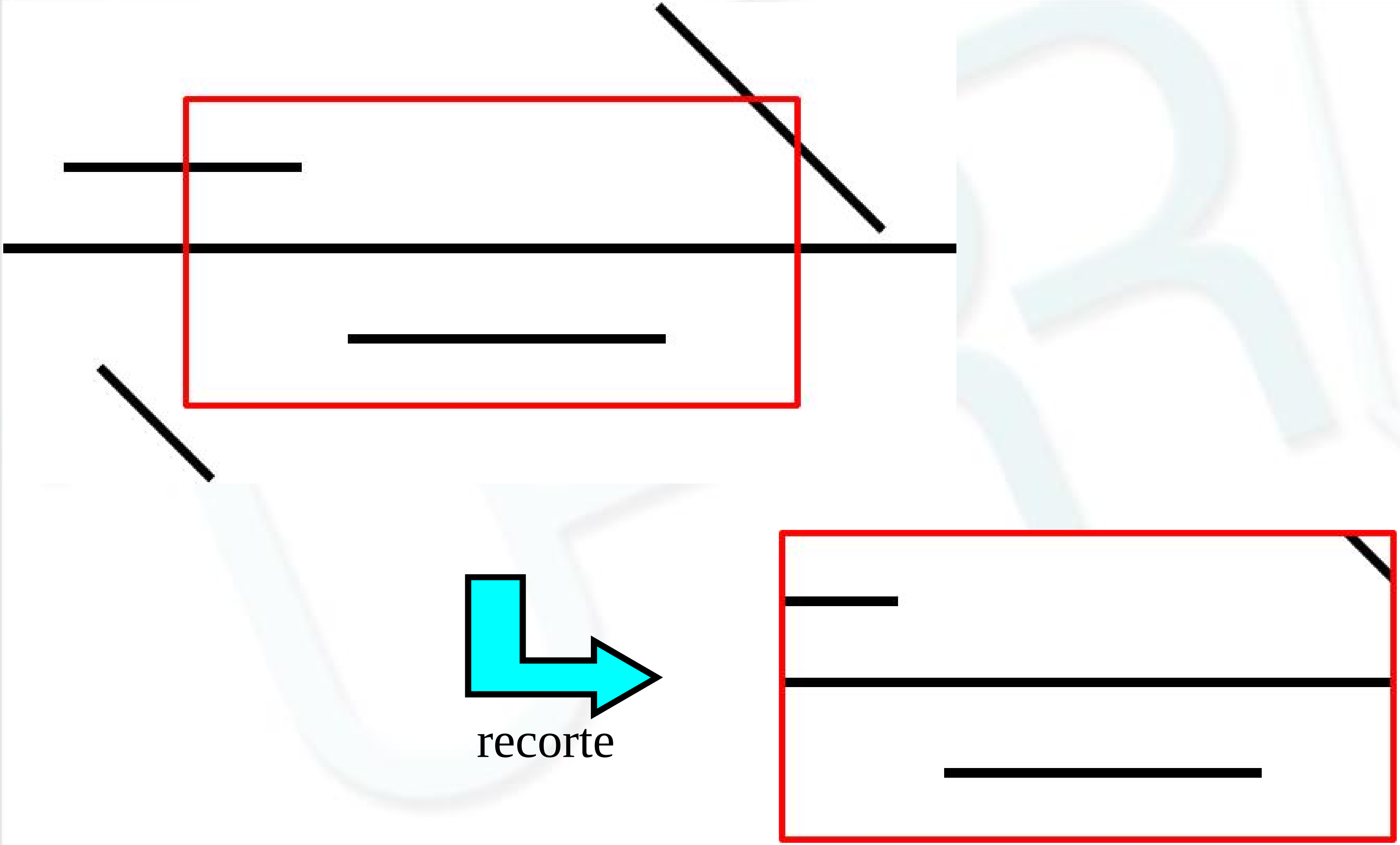
Professor: Luciano Ferreira Silva, Dr.

Aula 13

Recorte

- **Finalidade: eliminar objetos que não aparecem na Windows;**
- **Redefinir objetos que aparecem parcialmente na Windows;**
- **Algoritmos:**
  - Recorte de pontos
  - Recorte de linhas
  - Algoritmo de *Cohen-Sutherland*
  - Algoritmo de subdivisão - ponto médio
  - Algoritmo de *Sutherland-Hodgman*
  - Algoritmo de *Weiler-Atherton*

Recorte



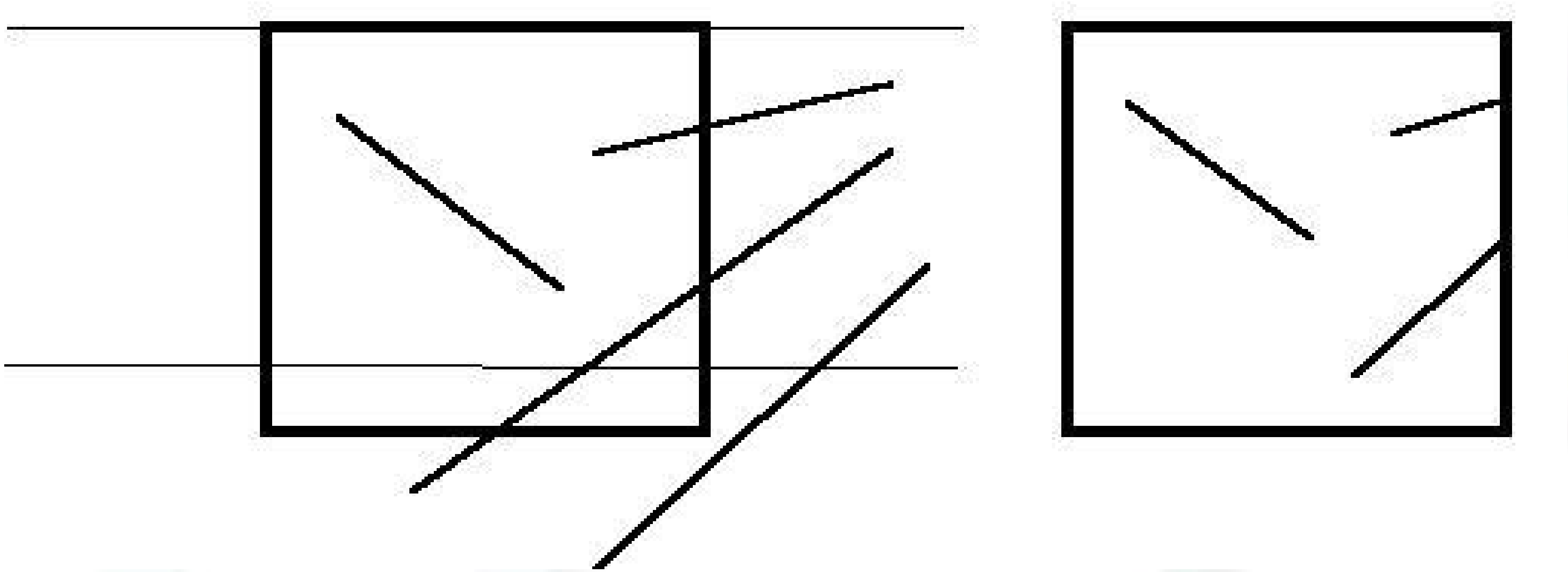
Recorte de pontos

- Processo rápido e simples;
- O ponto só será apresentado na *viewport* se:

$$x_{\min} < x < x_{\max} \qquad e \qquad y_{\min} < y < y_{\max}$$

- O ponto que não satisfaz todas as inequações não pode ser apresentado na *viewport*;

Recorte de Linhas



- O processo exige mais cálculos e testes;
- Estratégia: considerar os pontos extremos das linhas;

Recorte de Linhas

- **A principal finalidade de qualquer algoritmo de recorte de linha é minimizar os cálculos de interseção.**

- **1ª Solução:**

  - ✓ Checagem, utilizando equação paramétrica da reta;
  - ✓ Exige demasiada quantidade de cálculos e testes;
  - ✓ Estratégia: fazer testes iniciais de forma a detectar se cálculos de interseções são necessários;
    - Linhas aceitas ou rejeitadas trivialmente, checando pontos extremos;

Recorte - Linhas


```
if P1 = DENTRO and P2 = DENTRO
  ~ Desenha linha P1->P2
if P1 = DENTRO and P2 = FORA or
   P1 = FORA and P2 = DENTRO
      ~ Acha interseção da linha com a window
  (qual a borda?)
      ~ Redefine P2 (ou P1)
      ~ Desenha linha P1->P2 (ou P2->P1)
if P1 = FORA and P2 = FORA
```

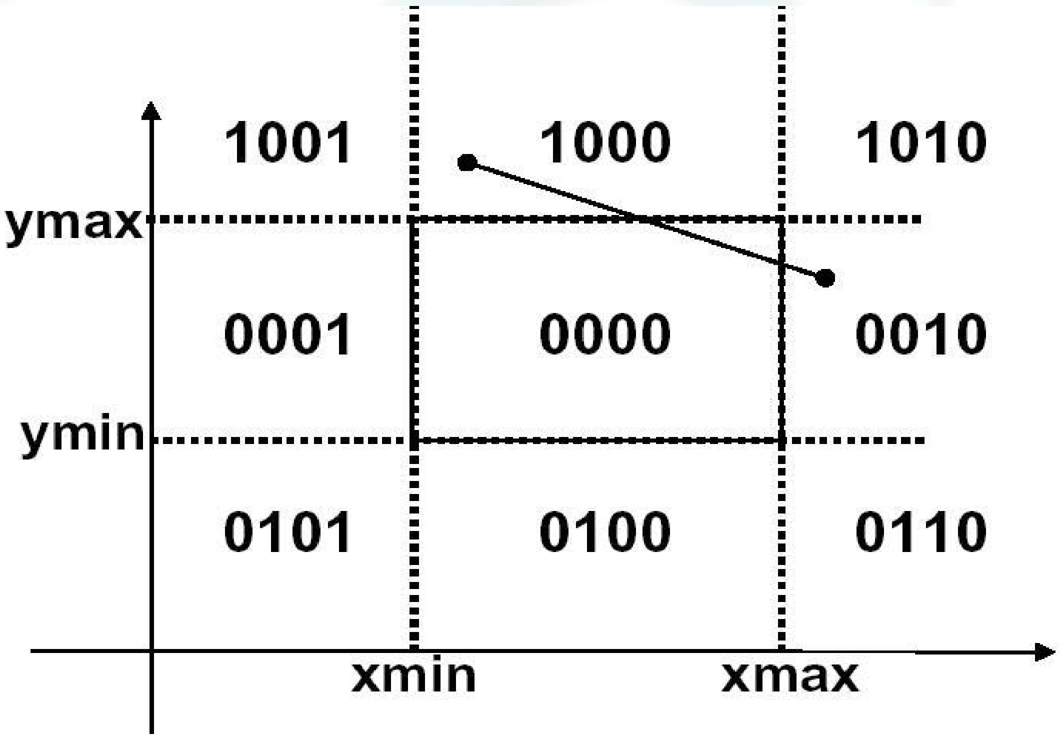
Algoritmo de Cohen Sutherland

- **Identifica, de forma eficiente, que linhas são trivialmente aceitas ou rejeitadas, por meio de regiões;**

- **Dispensa cálculos de interseções nos casos de linhas:**
  - ✓ Aceitas trivialmente
  - ✓ Rejeitadas trivialmente

Algoritmo de Cohen Sutherland

- Os pontos das extremidades da linha serão associados a um valor binário composto por 4 dígitos, chamado de código da região;



Mapa de bits:
(b3 b2 b1 b0)

4° bit - O ponto está à esquerda da janela - b0 - P(4)
3° bit - O ponto está à direita da janela - b1 - P(3)
2° bit - O ponto está à abaixo da janela - b2 - P(2)
1° bit - O ponto está à acima da janela - b3 - P(1)

Algoritmo de Cohen Sutherland

- 1º Passo: associar códigos aos pontos extremos, usando a regra:

$$\begin{cases} \text{se } x_1 < X_{min} \Rightarrow P_{code(4)} = 1; \text{do contrário}: P_{code(4)} = 0 \\ \text{se } x_1 > X_{max} \Rightarrow P_{code(3)} = 1; \text{do contrário}: P_{code(3)} = 0 \end{cases}$$

$$\begin{cases} \text{se } y_1 < Y_{mim} \Rightarrow P_{code(2)} = 1; \text{do contrário}: P_{code(2)} = 0 \\ \text{se } y_1 > Y_{max} \Rightarrow P_{code(1)} = 1; \text{do contrário}: P_{code(1)} = 0 \end{cases}$$

Algoritmo de Cohen Sutherland

- 2º Passo: verificar se a linha é totalmente visível:
  - ✓ Se os dois códigos associados às duas extremidades do segmento de reta forem zero: linha totalmente visível

```
soma : = 0 ;
        for i = 1 to 4
                soma : = soma + P1CODE (i) + P2CODE (i)
        if soma = 0  then
                linha totalmente visível
```

Algoritmo de Cohen Sutherland

- **3º Passo: verificar se a linha é invisível, estando totalmente à direita, esquerda, acima ou abaixo da janela:**

```
Inter : = 0
        for i = 1 to 4
                Inter : = P1.CODE(i) + P2.CODE(i)
        if  Inter = 2 then linha invisível
                                else next i
```

$$\text{Ex:} \begin{cases} \text{linha KL P1 (0 0 1 0) ; P2(0 1 1 0)} \Rightarrow \text{OK - linha invisivel} \\ \text{linha GH P1 (0 0 0 1) ; P2 (1 0 0 0)} \Rightarrow \text{Falha} \\ \text{linha IJ}_1 \text{ P1 (1 0 0 1) ; P2 (1 0 0 0)} \Rightarrow \text{OK - linha invisivel} \end{cases}$$

Algoritmo de Cohen Sutherland

- **4º Passo: Se a linha é parcialmente visível ou totalmente invisível para o terceiro passo, devem ser calculadas as interseções da mesma com a borda:**

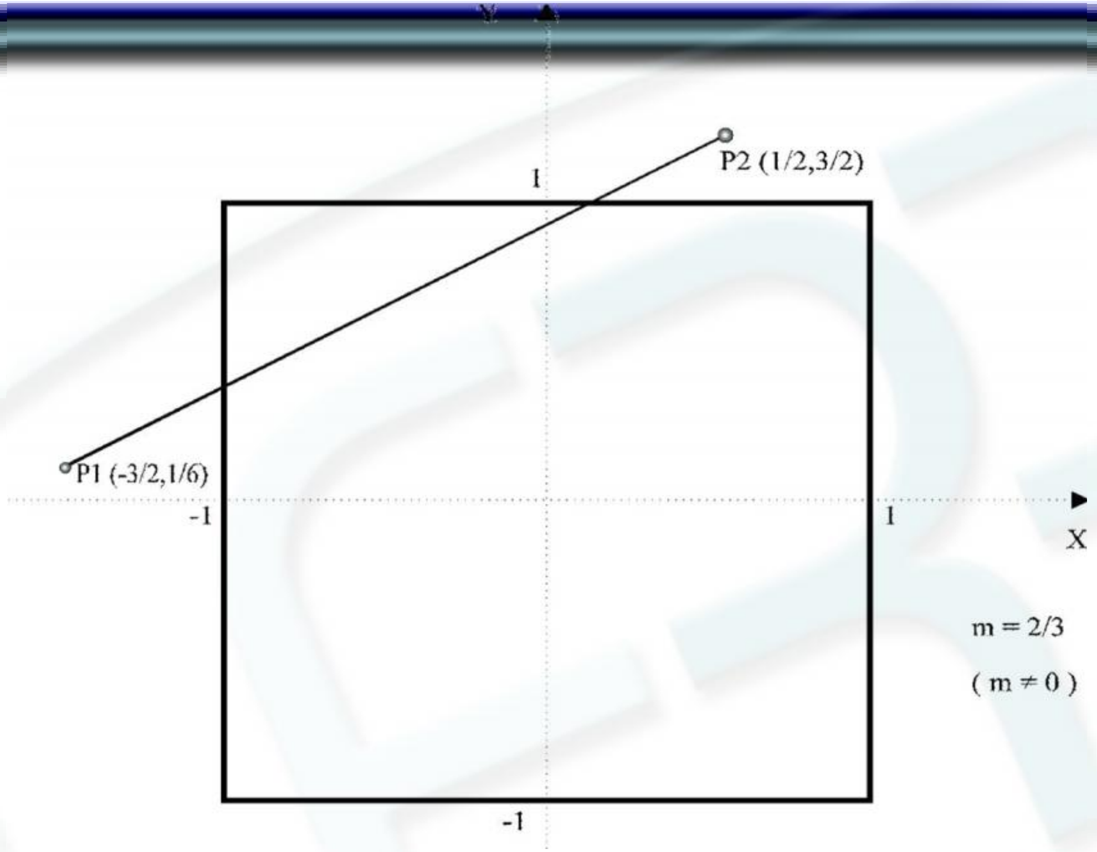
Esquerda: X_{mim}, $Y = M^* (X_{mim} - X_1) + Y_1$; $M \neq 0$;

Direita: X_{max}, $Y = M^* (X_{max} - X_1) + Y_1$; $M \neq 0$;

Top: Y_{max}, $X = X_1 + 1/M \,.\, (Y_{max} - Y_1)$; $M \neq 0$;

Botton: Y_{min}, $X = X_1 + 1/M \,.\, (Y_{min} - Y_1)$; $M \neq 0$;

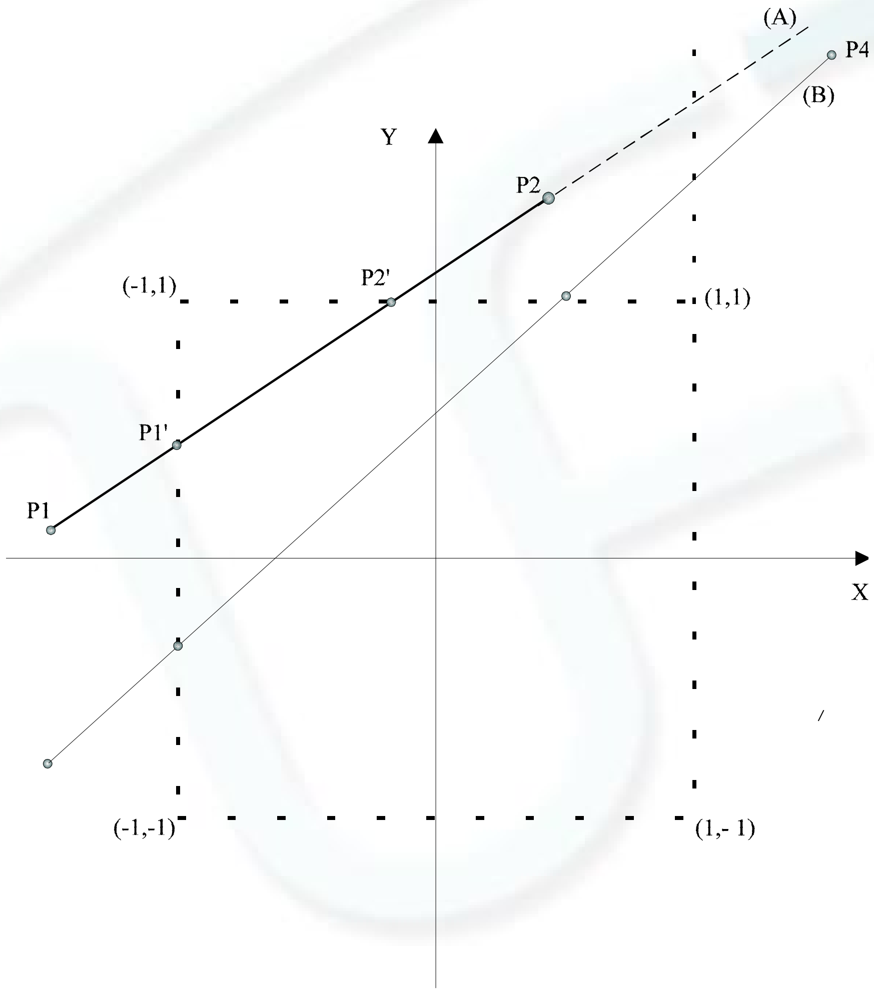
Exemplo



- Esquerda: $X_{min} = -1 \rightarrow Y = 2/3. [-1 - (-3/2)] + 1/6 = ½ \{Y_{min} \leq Y = ½ \leq Y_{max} \}$
- Direita: $X_{max} = 1 \rightarrow Y = 2/3. [1 - (-3/2)] + 1/6 = 11/6$ {FORA}
- Top: $Y_{max} = 1 \rightarrow X = -3/2 + 3/2. [1 - 1/6] = -1/4 \{X_{min} \leq -1/4 \leq X_{max} \}$
- Botton: $Y_{min} = -1 \rightarrow X = -3/2 + 3/2. [1 - 1/6] = -13/4$ {FORA}

Exercício

- Prove a eficiência do algoritmo anterior para os casos A e B abaixo:



P1 (-3/2 , 1/6)

P2 (1/2 , 3/2)

P3 (-1.5 , -0.75)

P4 (1.5 , 2)

Algoritmo de subdivisão por ponto médio

- **Motivação:**
  - ✓ Algoritmo anterior necessitava de cálculos para obtenção da interseção da linha com limites da janela;
  - ✓ Estratégia: usar a busca binária de forma a evitar o cálculo citado;
- **Caso particular do Algoritmo anterior, proposto por Sproull e Sutherland**
- **Implementação em hardware apresenta melhor desempenho:**
  - ✓ Uso de arquitetura paralela para divisão e multiplicação.

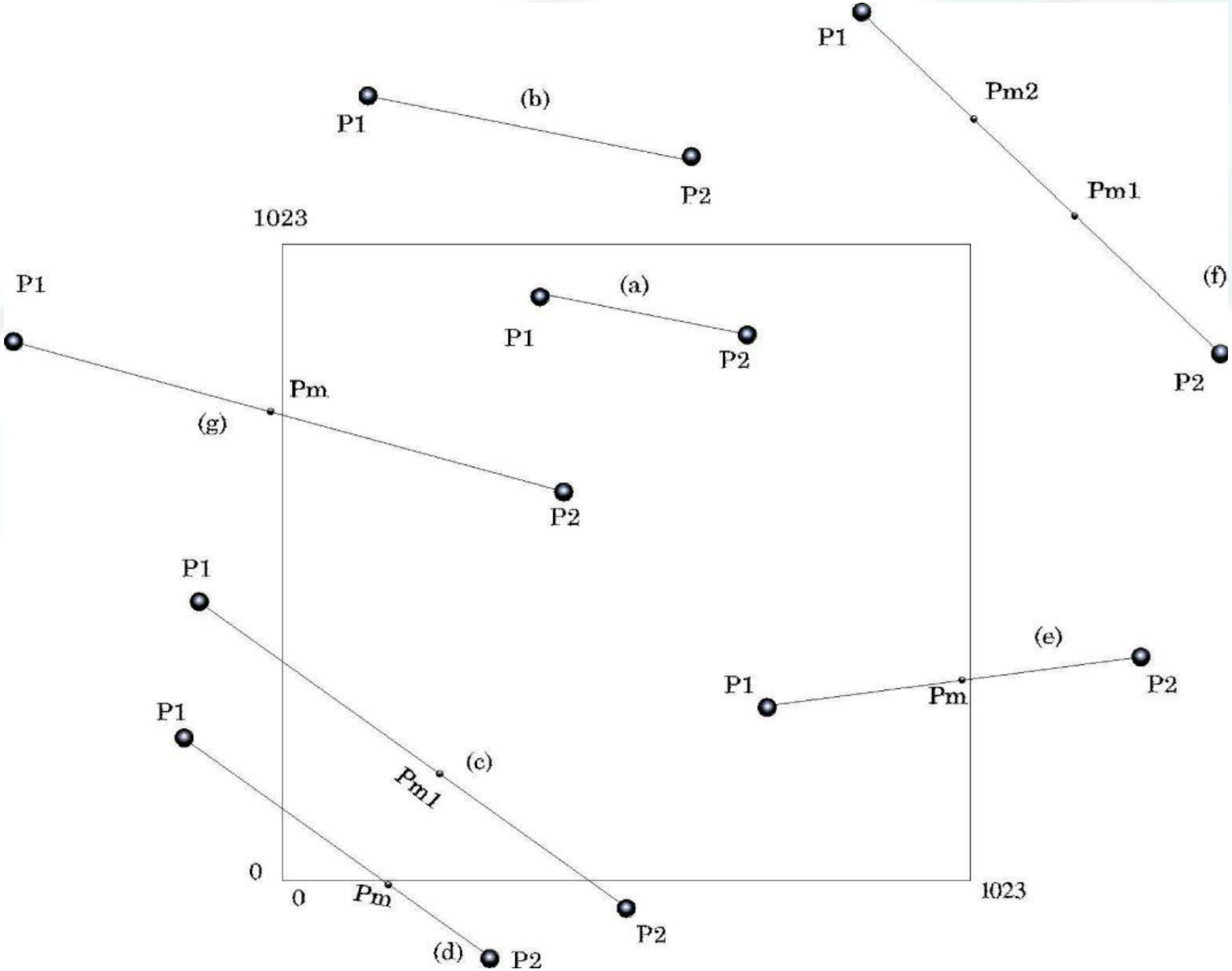
Algoritmo de subdivisão por ponto médio

- **Divisão por 2, em bits:**
  - ✓nº 6: 110, deslocando 1 bit para a direita: 011: nº 3
- **Estratégia:**
  - ✓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:
    - $Xm = (x_1 + x_2) / 2$
    - $Ym = (y_1 + y_2) / 2$

Algoritmo de subdivisão por ponto médio

- ✓O teste é aplicado a cada uma das metades até a obtenção da interseção com as bordas da janela - ou seja, até que o comprimento da parte resultante seja *infinitesimal*.
- ✓A visibilidade do ponto é, então, determinada - busca logarítmica;

Algoritmo de subdivisão por ponto médio



Algoritmo de subdivisão por ponto médio

- **Linha 'f': P1 - 1000 <−> P2 – 0010, necessário checar:**
  - ✓ Subdivisão P_{m1} - 1010 <−> P2 - 0010: trivialmente rejeitado;
  - ✓ Subdivisão P_{m1} - 1010 <−> P1 - 1000: trivialmente rejeitado;
  - ✓ Subdivisões sucessivas: rejeitar a linha
- **Linha 'c': P1 (0001) <−> P2 (0100), necessário checar:**
  - ✓ Ponto médio P_{m1} (0000): mesmo resultado para ambos os lados: parcialmente visível

*** Analisando P_{m1} P2 inicialmente:**

$$\text{dividimos em } P_{m2} \Rightarrow \begin{cases} P_{m1}(0000) \\ \\ P_{m2}\ (0000) \end{cases} \Rightarrow \text{ totalmente visivel}$$

Algoritmo de subdivisão por ponto médio

$$\left\{\begin{array}{l} P_{m2}\,(0000) \\ \\ P_2\,(0100) \end{array}\right. \Rightarrow \textit{Parcialmente} \quad \textit{Visivel}$$

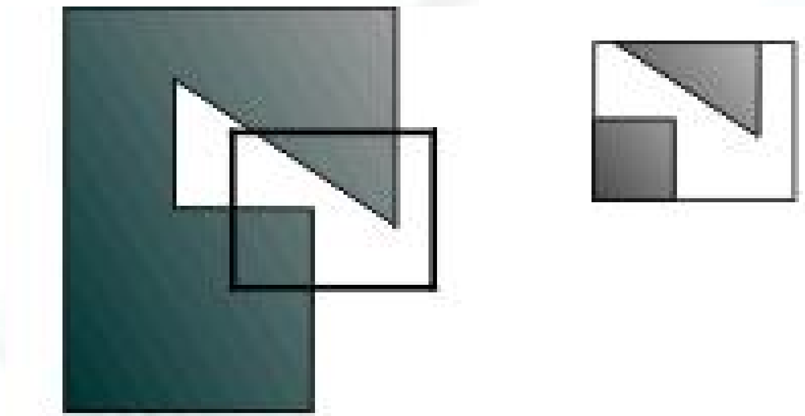
- **Traça-se $P_{m1}\,P_{m2}$; continuamos a divisão de $P_{m2}\,P_2$**
- **divisões sucessivas: segmentos menores serão obtidos e ao final, teremos a obtenção da interseção com a janela;**
- **O segmento $P_{m1}P_1$ sofrerá a mesma análise**
- **Ideal: obter os dois pontos e traçar a linha entre eles.**

Recorte de Polígonos

- **Vértices do polígono: armazenados em uma estrutura de dados conveniente;**
- **Exibição do polígono:**
  - ✓Operação de transformação de visualização;
  - ✓Recorte;
  - ✓Conversão nas coordenadas do equipamento;

Recorte de Polígonos

- **Algoritmo deve ser capaz de identificar situações distintas:**



- Polígono côncavo é recortado em 2 polígonos separados e distintos.

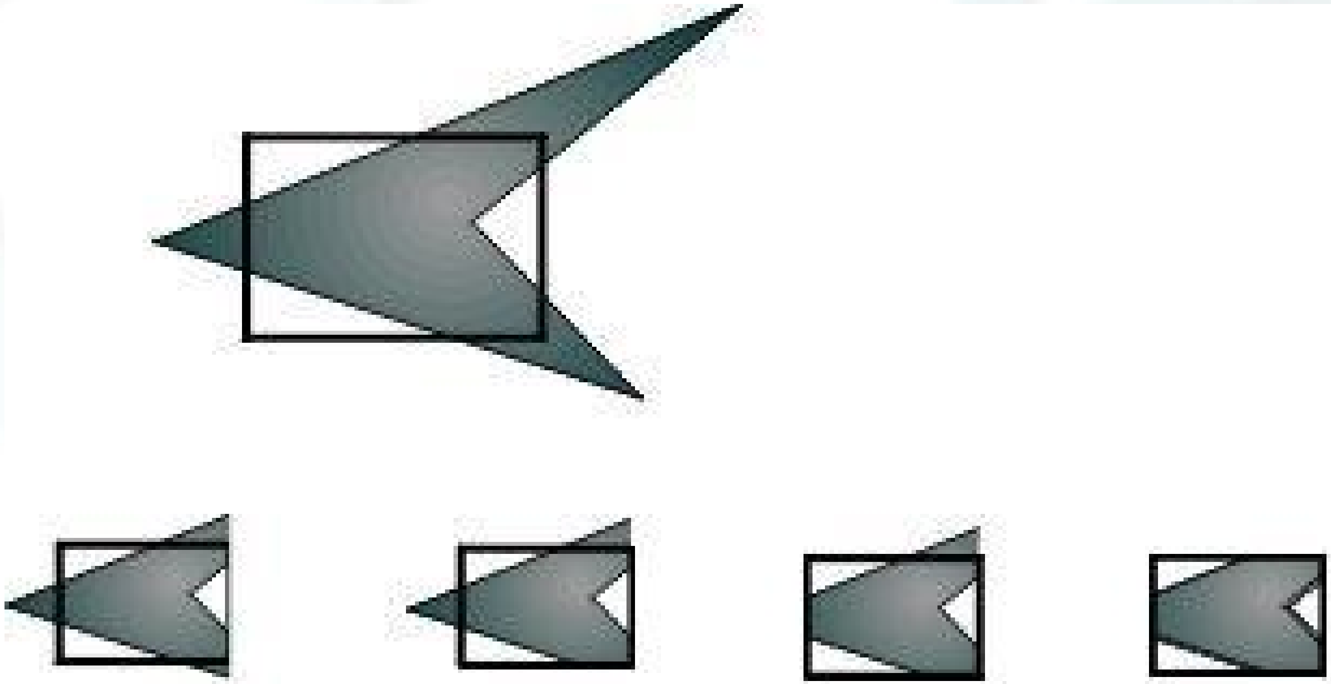
- Recorte de polígono (côncavo) que está quase integralmente na janela.

Algoritmo de Sutherland-Hodgman

- **Marco no desenvolvimento da Computação Gráfica;**
- **Até então: recorte de polígonos usava estratégias do recorte de retas:**
  - ✓ Cada aresta era analisada em relação à região de recorte como um todo;
  - ✓ Perda da conectividade do polígono recortado
- **Só funciona para regiões convexas**

Algoritmo de Sutherland-Hodgman

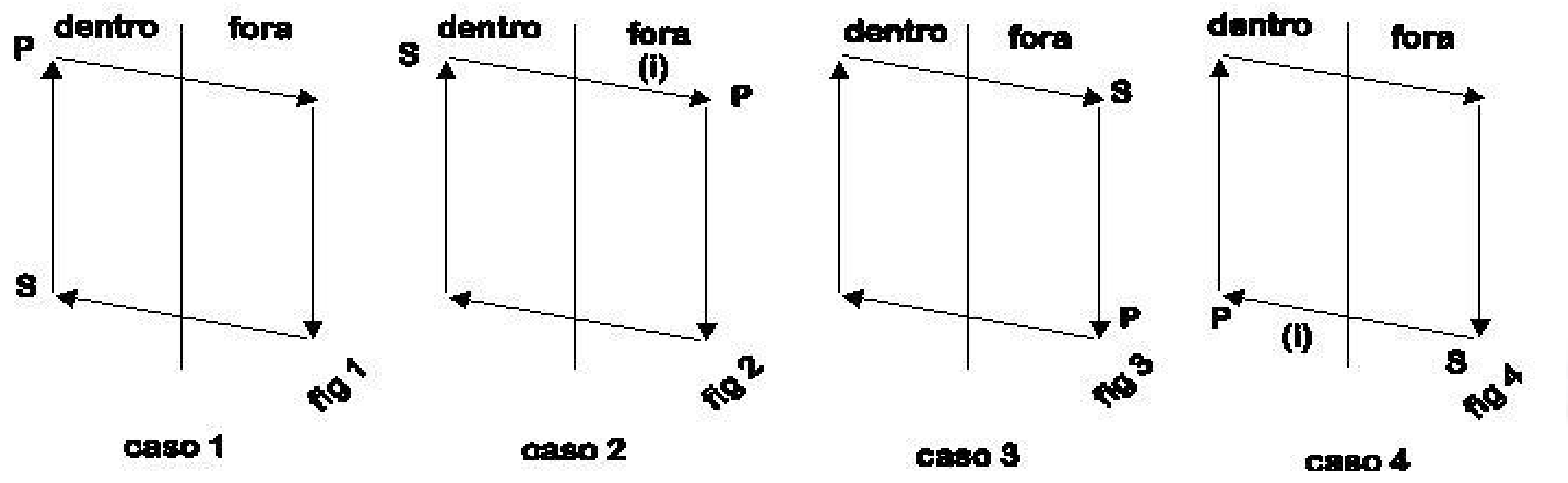
- Estratégia: recortar o polígono através do recorte de suas laterais. Serão efetuados quatro recortes:



Sutherland - Hodgman

- **Algoritmo:**
  - ✓Suponha um polígono de lados dados por vértices: $v_1, v_2, \ldots, v_n$;
  - ✓Para cada lado, observa-se a relação entre vértices sucessivos e as janelas (limites)
  - ✓Lados definidos pelos vértices da lista de saídas serão apresentados na tela.

Sutherland - Hodgman



CASO 1: um dos dois vértices é adicionado à lista de saídas (p, no caso)
CASO 2: o ponto "i" de interseção é tratado como um vértice de saída (a ser traçado)
CASO 3: os dois vértices são descartados.
CASO 4: os dois pontos "i" e "p" são colocados na lista de vértices de saída.

Sutherland - Hodgman

- **Algoritmo de obtenção das interseções:**

  ✓ O primeiro ponto não é colocado na lista de saídas (s, da fig. 1, anterior), já que o mesmo é o vértice inicial e já se encontra na mesma, pois o processo é seqüencial;

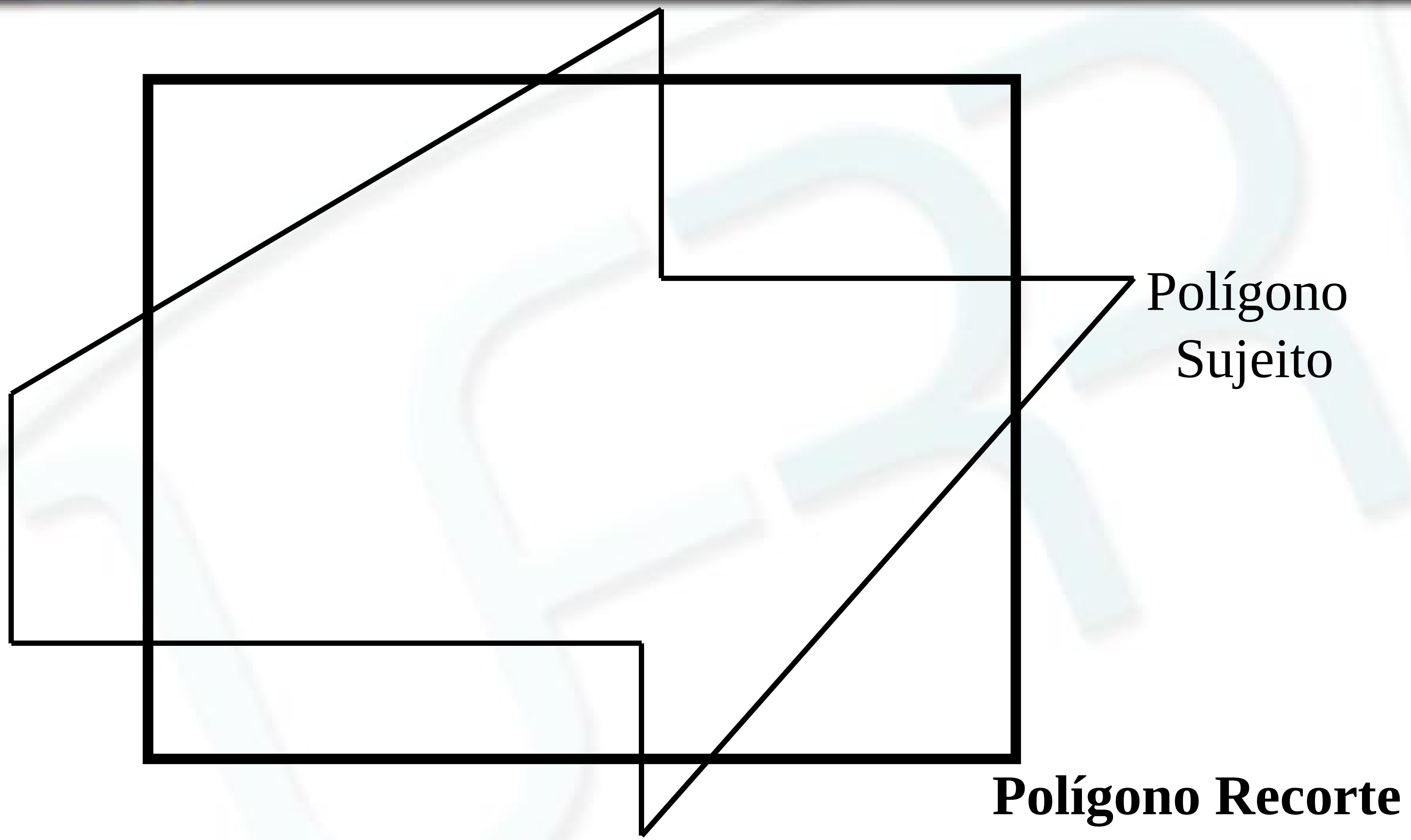
  ✓ Para linhas do polígono totalmente invisível, nenhum ponto é adicionado à lista de saídas (caso 3);

  ✓ Para casos 2 e 4, é necessário calcular as interseções;

Weiler Atherton

- **Solução para qualquer tipo de região, não só as convexas;**
- **Mais complexo que o anterior;**
- **Mais poderoso;**
- **Dois polígonos:**
  - ✓ Polígono a ser recortado: polígono sujeito;
    - Polígono que é definido na região de recorte: polígono a ser recortado;
  - ✓ Região de recorte: polígono recorte;

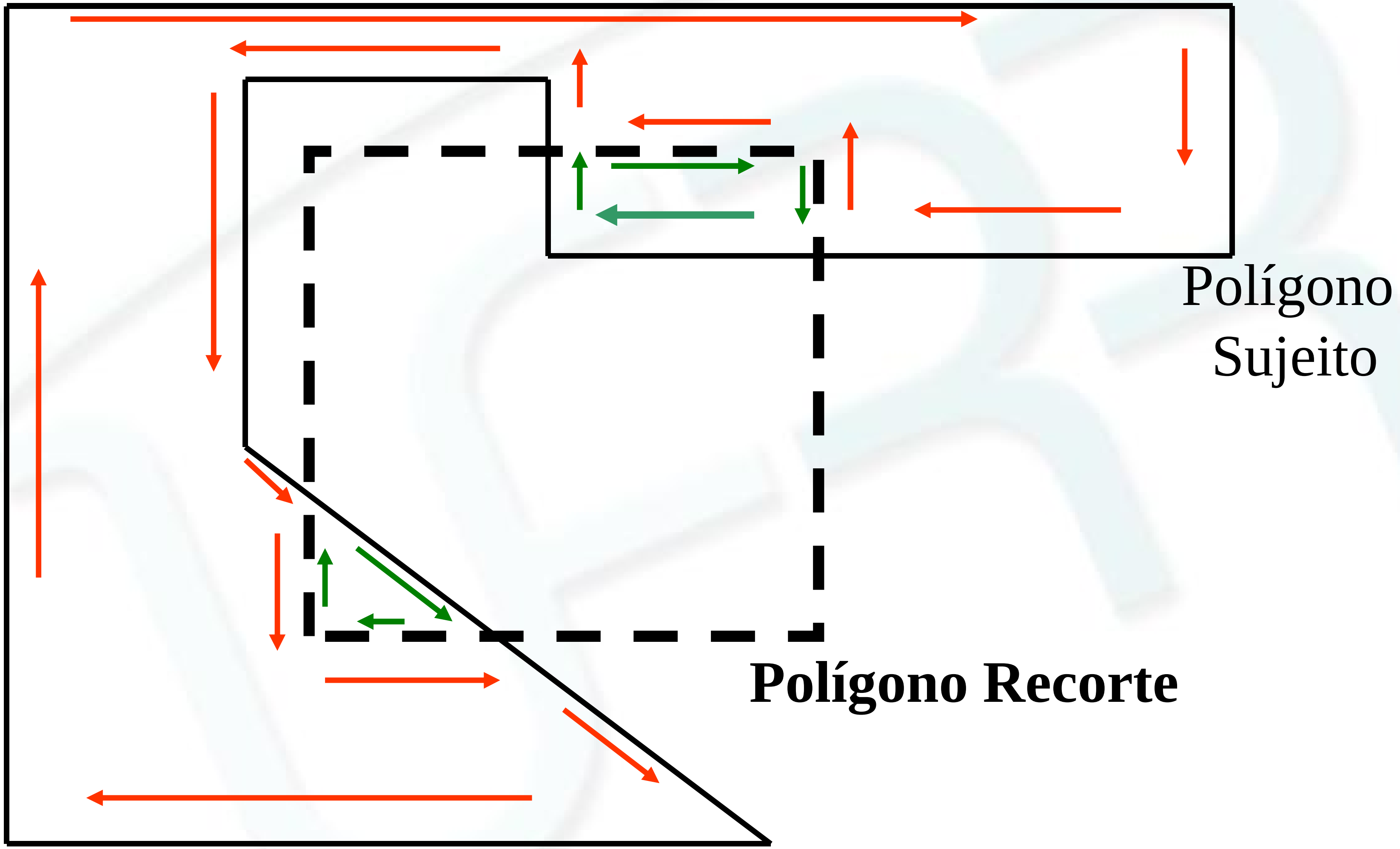
Weiler Atherton



Weiler - Atherton

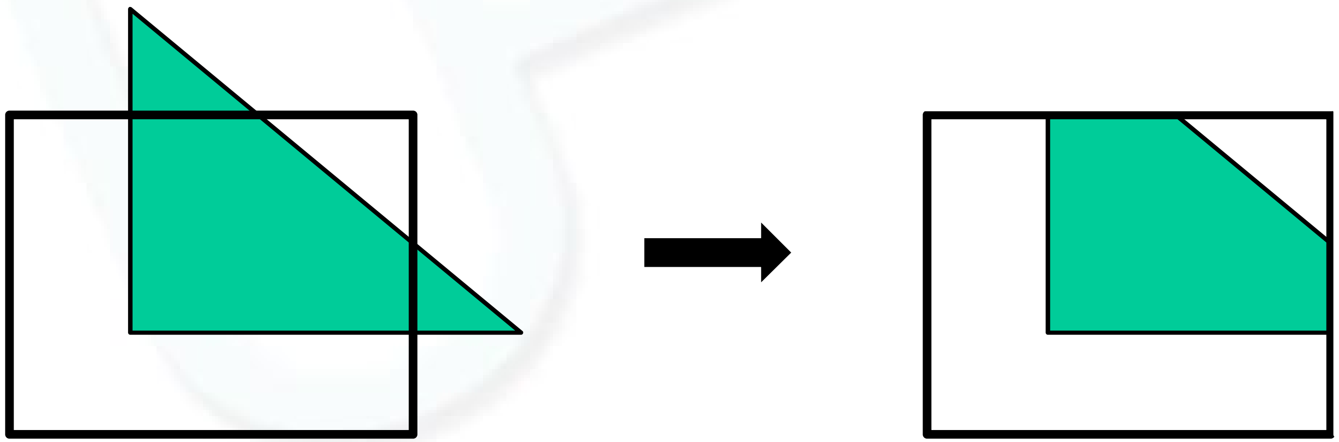
- **Estratégia:**
  - ✓ Descreve os dois polígonos por uma lista de vértices;
  - ✓ Percorre o polígono sujeito, na direção horária, até obter uma interseção com o polígono recorte;
  - ✓ Se o movimento indica entrar no polígono recorte, o algoritmo passa a varrer limites do polígono sujeito
  - ✓ Se o movimento indica sair do polígono recorte, uma varredura, por retorno à direita é feita, seguindo o polígono recorte no sentido horário até obter a interseção que originou o movimento.

Weiler - Atherton

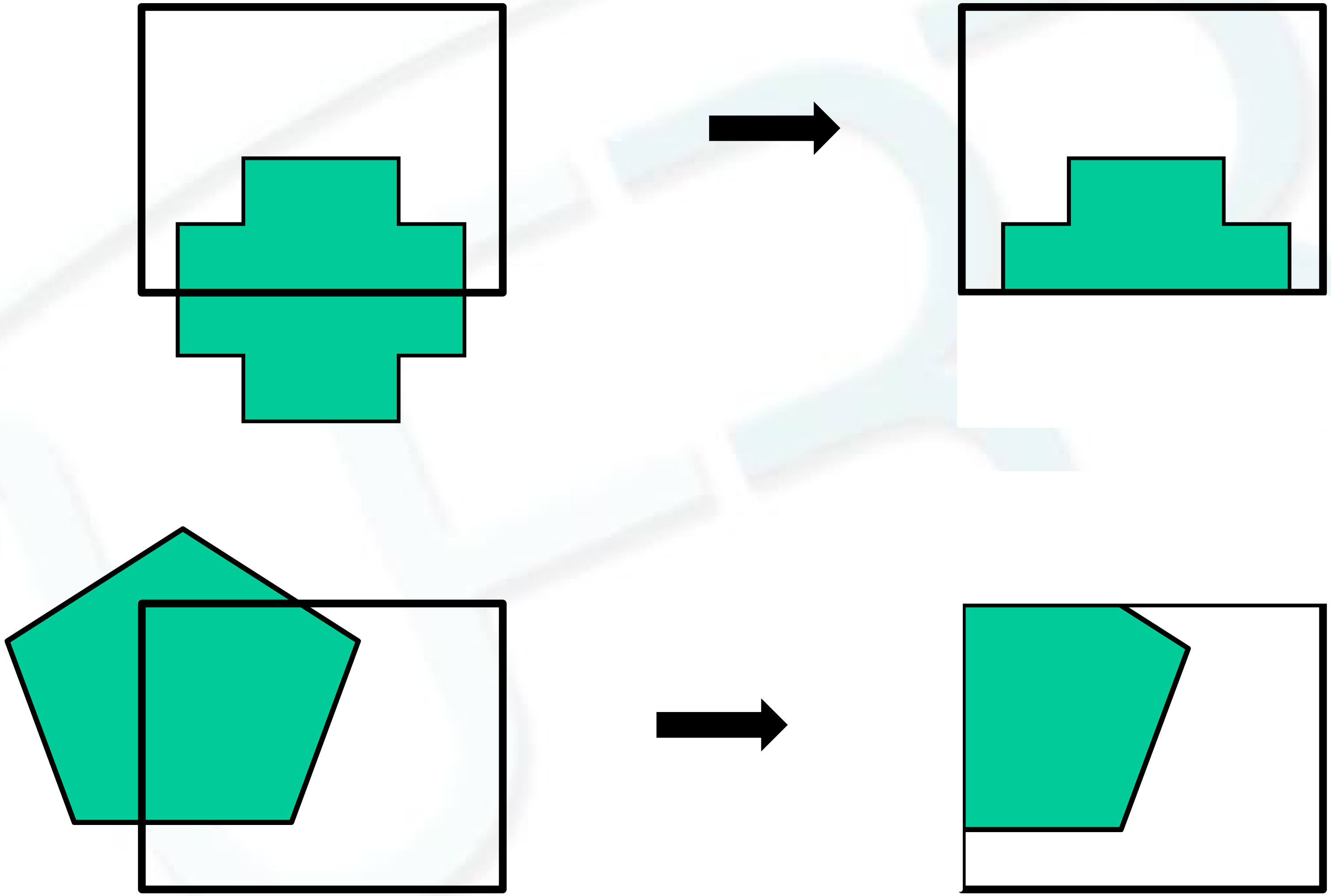


Trabalho 5

- Desenvolver um programa que realize recortes de polígonos por meio do algoritmo de Sutherland. Você deve realizar testes com as seguintes figuras:



Trabalho 5



UFRR – Departamento de Ciência da Computação
Computação Gráfica – Prof. Dr. Luciano F. Silva

Trabalho 5

UFRR – Departamento de Ciência da Computação
Computação Gráfica – Prof. Dr. Luciano F. Silva

Dúvidas