

# PARTICIONAMENTO DE POLÍGONOS

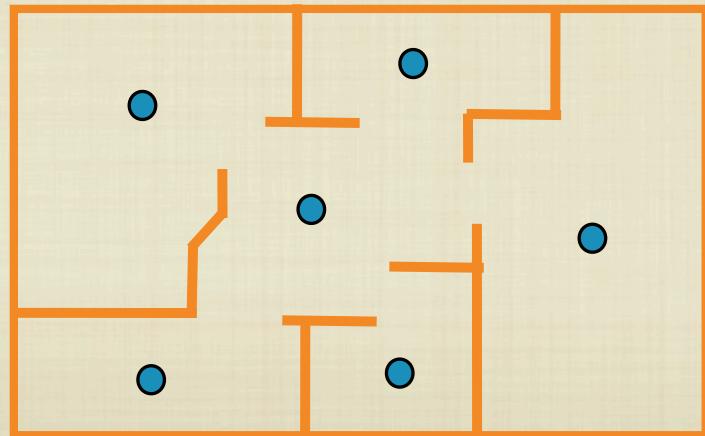
JOÃO COMBA

## PROBLEMA DA GALERIA DE ARTE



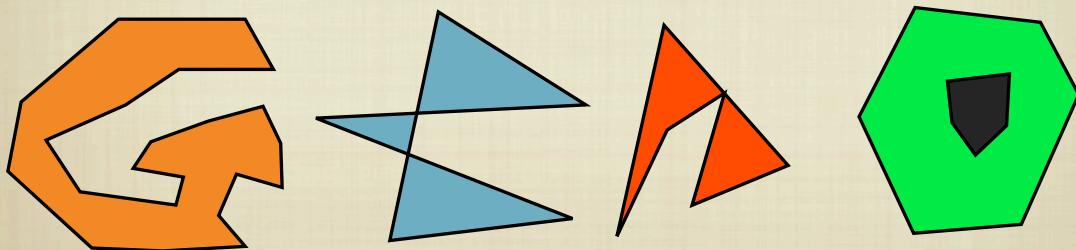
THE WALTERS ART MUSEUM,  
BALTIMORE, EUA

# PROBLEMA DA GALERIA DE ARTE



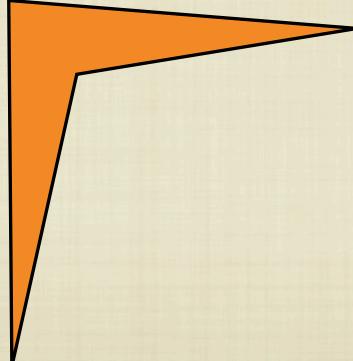
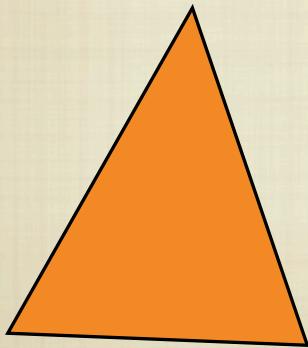
## POLÍGONOS SIMPLES

- **POLÍGONO SIMPLES:** REGIÃO DO PLANO LIMITADA POR UMA COLEÇÃO FINITA DE SEGMENTOS DE LINHA FORMANDO UMA CURVA FECHADA SIMPLES.



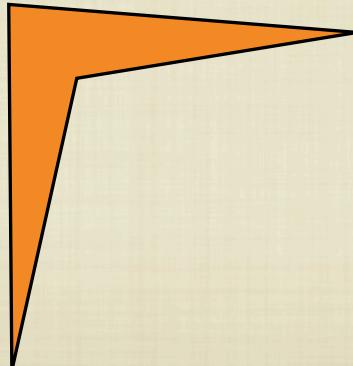
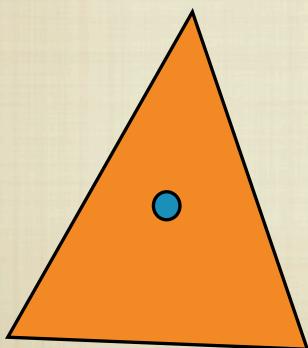
# PROBLEMA DA GALERIA DE ARTE

Qual é o menor número  $g$  de **guardas estáticos** que guardam uma galeria de arte modelada por um polígono simples com  $n$  vértices ?



# PROBLEMA DA GALERIA DE ARTE

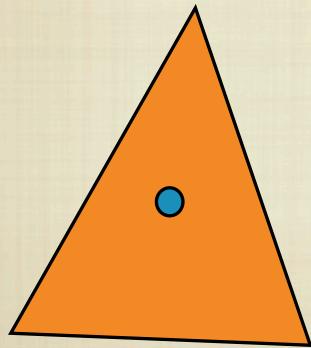
Qual é o menor número  $g$  de **guardas estáticos** que guardam uma galeria de arte modelada por um polígono simples com  $n$  vértices ?



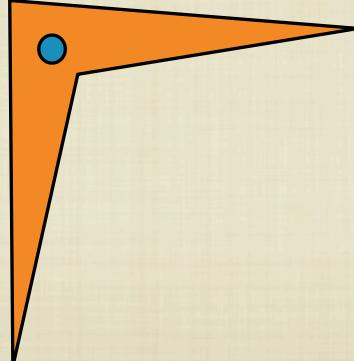
$$n = 3, g = 1$$

# PROBLEMA DA GALERIA DE ARTE

Qual é o menor número **g** de **guardas estáticos** que guardam uma galeria de arte modelada por um polígono simples com **n** vértices ?

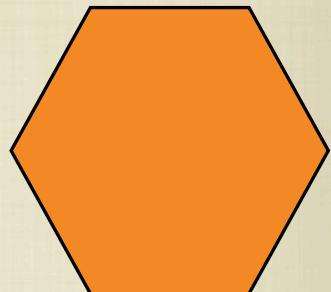
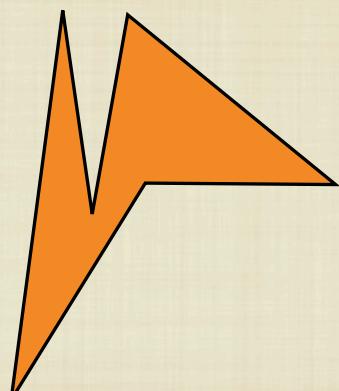
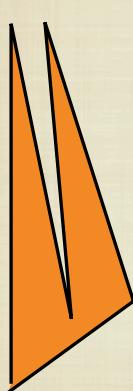


$n = 3, g = 1$

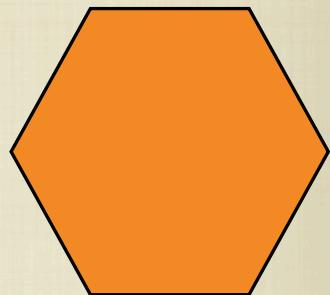
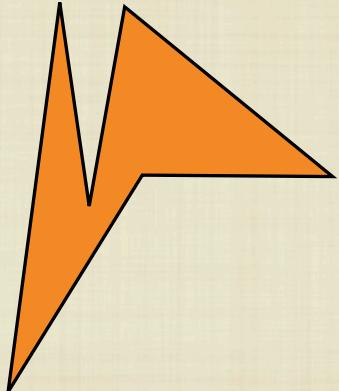
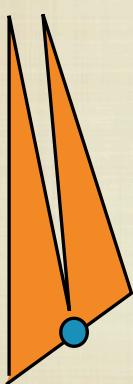


$n = 4, g = 1$

# PROBLEMA DA GALERIA DE ARTE

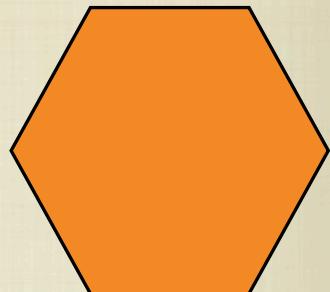
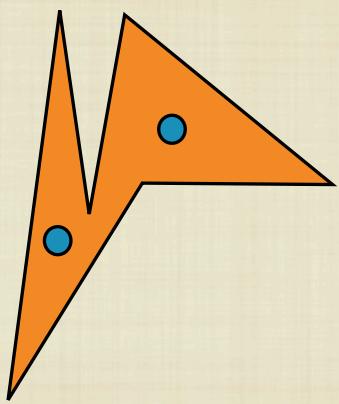
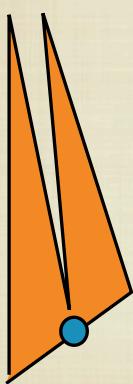


# PROBLEMA DA GALERIA DE ARTE



$n = 5, g = 1$

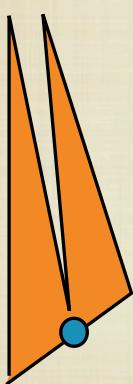
# PROBLEMA DA GALERIA DE ARTE



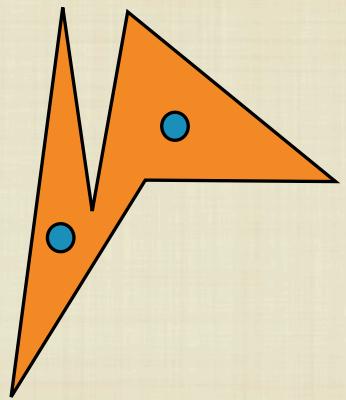
$n = 5, g = 1$

$n = 6, g = 2$

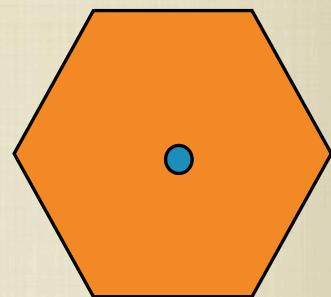
# PROBLEMA DA GALERIA DE ARTE



$n = 5, g = 1$

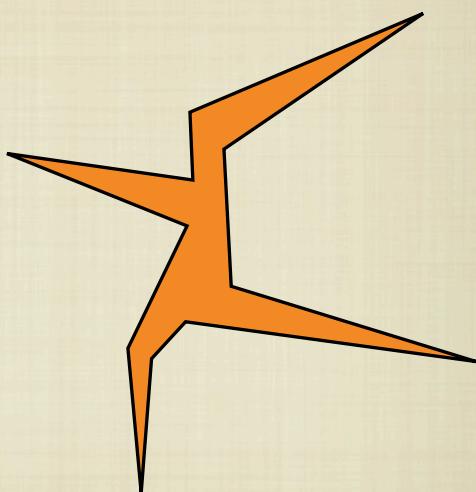
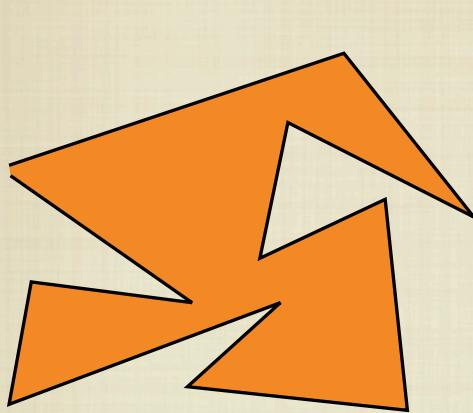


$n = 6, g = 2$

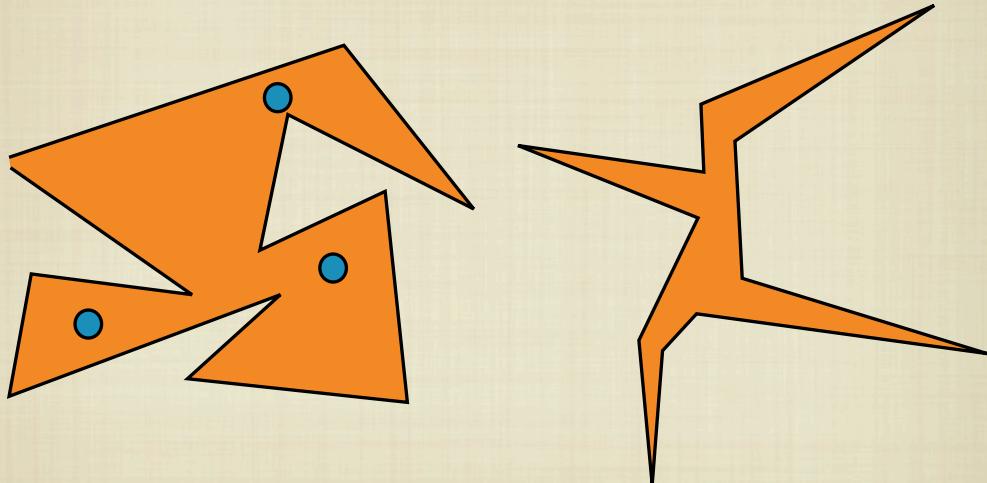


$n = 6, g = 1$

# PROBLEMA DA GALERIA DE ARTE

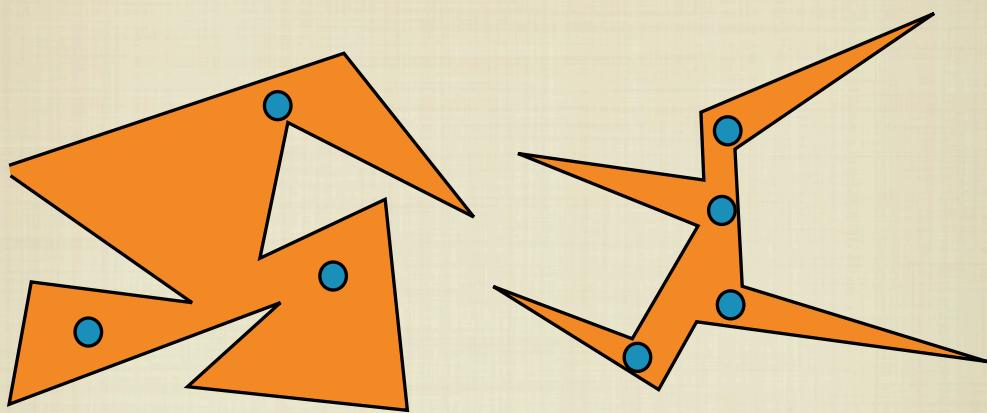


# PROBLEMA DA GALERIA DE ARTE



$n = 12, g = 3$

# PROBLEMA DA GALERIA DE ARTE



$n = 12, g = 3$

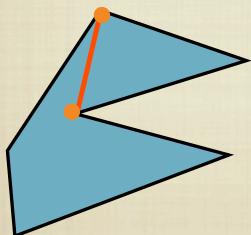
$n = 12, g = 4$

# PROBLEMA DA GALERIA DE ARTE

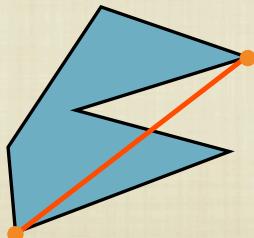
- $G(3) = 1$
- $G(4) = 1$
- $G(5) = 1$
- $G(6) = 2$
- $G(12) = 4$
- $G(N) = \text{FLOOR}(N/3)$  ?

## TRIANGULAÇÕES

- **DIAGONAL: SEGMENTO ABERTO DE LINHA QUE CONECTA DOIS VÉRTICES DE UM POLÍGONO P E PERTENCE AO INTERIOR DE P**



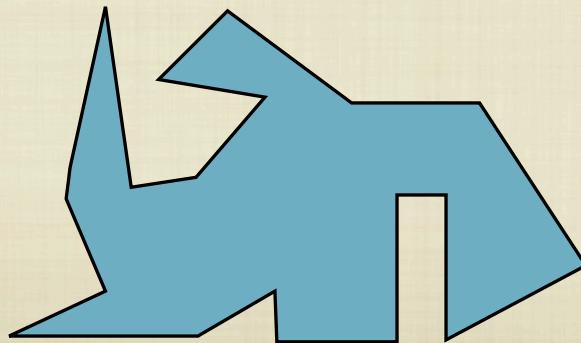
diagonal



não é diagonal

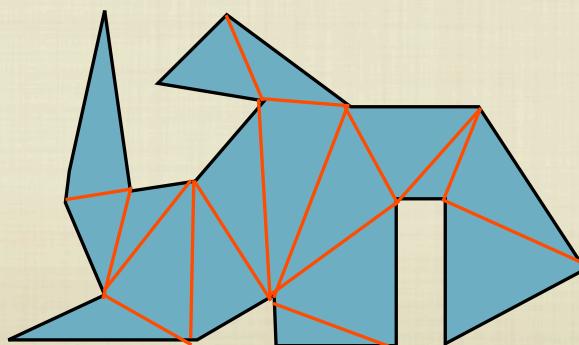
# TRIANGULAÇÕES

- **TRIANGULAÇÃO: DECOMPOSIÇÃO DE UM POLÍGONO  $P$  EM TRIÂNGULOS POR UM CONJUNTO MAXIMAL DE DIAGONAIS QUE NÃO SE INTERCEPTAM.**



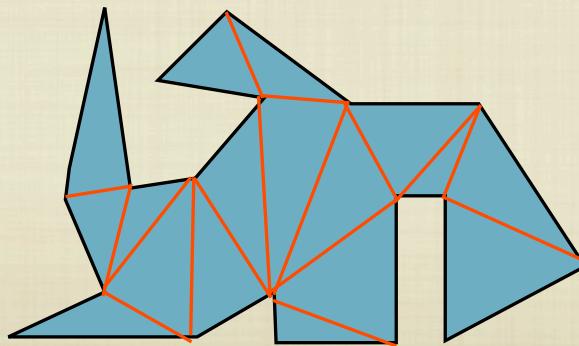
# TRIANGULAÇÕES

- **TRIANGULAÇÃO: DECOMPOSIÇÃO DE UM POLÍGONO  $P$  EM TRIÂNGULOS POR UM CONJUNTO MAXIMAL DE DIAGONAIS QUE NÃO SE INTERCEPTAM.**



# TRIANGULAÇÕES

- **TEOREMA: TODO POLÍGONO SIMPLES P ADMITE UMA TRIANGULAÇÃO**



PROVA POR INDUÇÃO:

- **CASO BÁSICO: N=3 - TRIVIAL**



## PROVA POR INDUÇÃO:

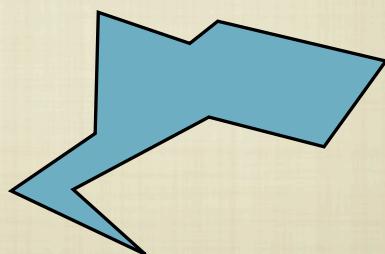
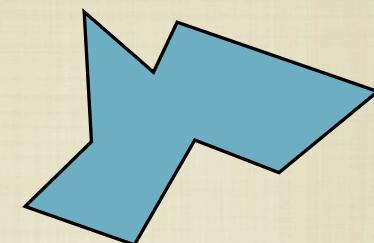
- CASO BÁSICO:  $N=3$  - TRIVIAL



- PASSO DE INDUÇÃO ( $N > 3$ )

- ASSUME VALIDO PARA  $M < N$

- SEJA UM POLÍGONO DE  $N$  VÉRTICES



## PROVA POR INDUÇÃO:

- CASO BÁSICO:  $N=3$  - TRIVIAL



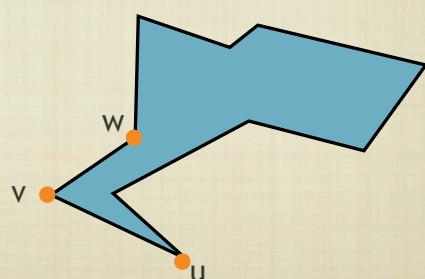
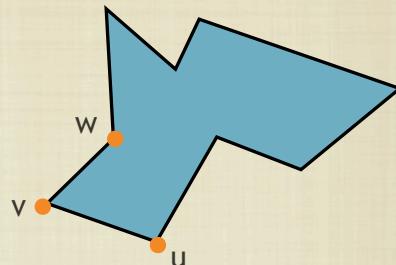
- PASSO DE INDUÇÃO ( $N > 3$ )

- ASSUME VALIDO PARA  $M < N$

- SEJA UM POLÍGONO DE  $N$  VÉRTICES

- SEJA  $V$  O VÉRTICE MAIS A ESQUERDA

- SEJA  $U$  E  $W$  VIZINHOS DE  $V$

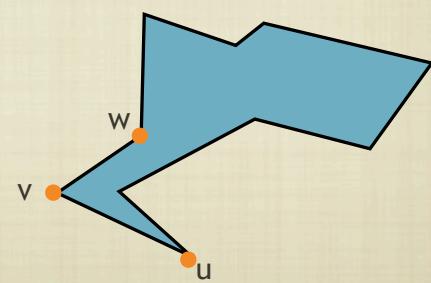
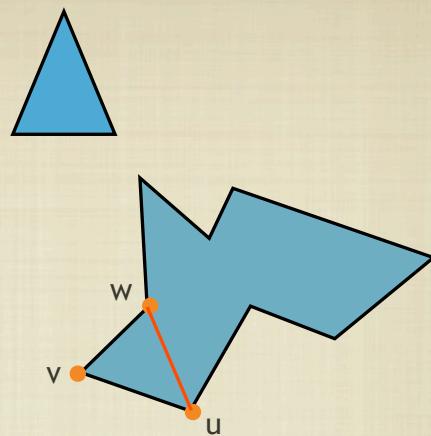


## PROVA POR INDUÇÃO:

- **CASO BÁSICO:  $N=3$  - TRIVIAL**

- **PASSO DE INDUÇÃO ( $N > 3$ )**

- ASSUME VALIDO PARA  $M < N$
- SEJA UM POLÍGONO DE  $N$  VÉRTICES
- SEJA  $V$  O VÉRTICE MAIS A ESQUERDA
- SEJA  $U$  E  $W$  VIZINHOS DE  $V$
- SE  $UW$  ESTÁ NO INTERIOR É UMA DIAGONAL

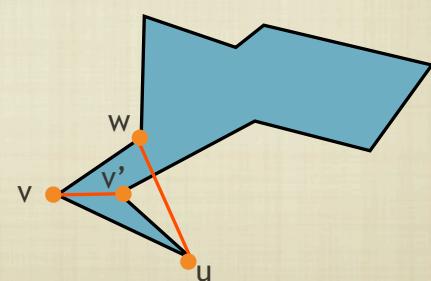
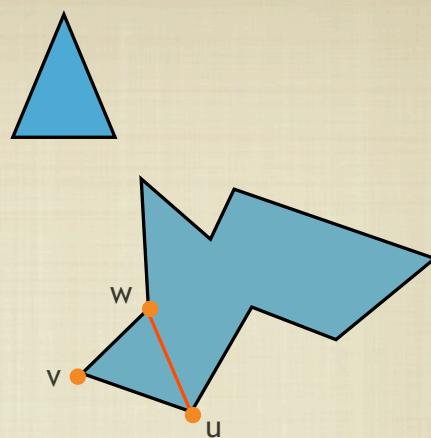


## PROVA POR INDUÇÃO:

- **CASO BÁSICO:  $N=3$  - TRIVIAL**

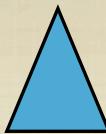
- **PASSO DE INDUÇÃO ( $N > 3$ )**

- ASSUME VALIDO PARA  $M < N$
- SEJA UM POLÍGONO DE  $N$  VÉRTICES
- SEJA  $V$  O VÉRTICE MAIS A ESQUERDA
- SEJA  $U$  E  $W$  VIZINHOS DE  $V$
- SE  $UW$  ESTÁ NO INTERIOR É UMA DIAGONAL
- SENÃO,  $V'$  O VÉRTICE MAIS LONGE DE  $UW$ , E  $V'V$  É UMA DIAGONAL



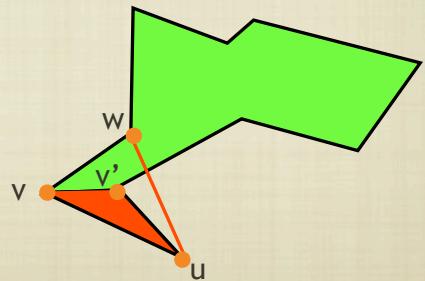
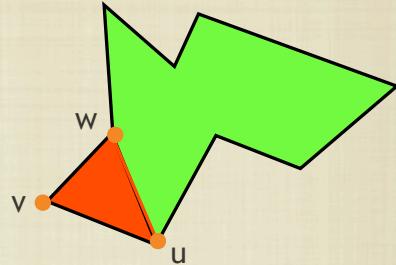
## PROVA POR INDUÇÃO:

- **CASO BÁSICO:  $N=3$  - TRIVIAL**



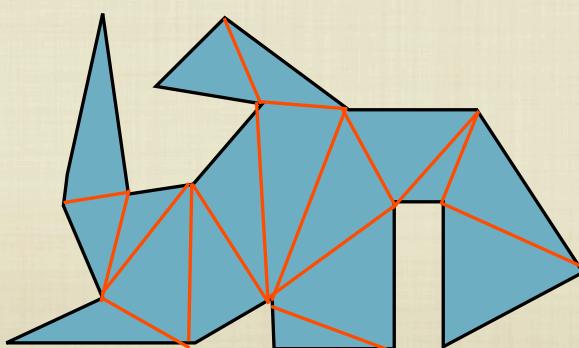
- **PASSO DE INDUÇÃO ( $N > 3$ )**

- ASSUME VALIDO PARA  $M < N$
- SEJA UM POLÍGONO DE  $N$  VÉRTICES
- SEJA  $V$  O VÉRTICE MAIS A ESQUERDA
- SEJA  $U$  E  $W$  VIZINHOS DE  $V$
- SE  $UW$  ESTÁ NO INTERIOR É UMA DIAGONAL
- SENÃO,  $V'$  O VERTICE MAIS LONGE DE  $UV$ , E  $V'V$  É UMA DIAGONAL
- CORTE O POLÍGONO USANDO A DIAGONAL, PRODUZINDO  $P_1$  E  $P_2$  COM  $M_1$  E  $M_2$  VÉRTICES ( $M_1 < N \leq M_2 < N$ )



## TRIANGULAÇÕES

- **TEOREMA: TODA TRIANGULAÇÃO DE UM POLÍGONO SIMPLES COM  $n$  VÉRTICES CONSISTE DE  $n-2$  TRIÂNGULOS**



## PROVA POR INDUÇÃO DO TEOREMA:

- CASO BÁSICO:  $N=3$  - TRIVIAL



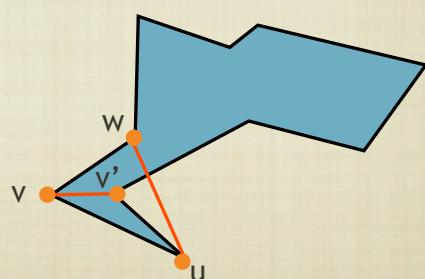
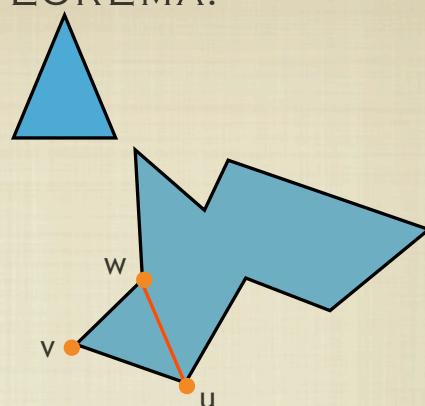
## PROVA POR INDUÇÃO DO TEOREMA:

- CASO BÁSICO:  $N=3$  - TRIVIAL

- PASSO DE INDUÇÃO ( $N > 3$ )

■ ASSUME UMA TRIANGULAÇÃO  $T_P$

■ CONSIDERE UMA DIAGONAL QUE CORTA  $P$  EM DOIS SUB-POLÍGONOS  $P_1$  E  $P_2$  COM  $M_1$  E  $M_2$  VÉRTICES

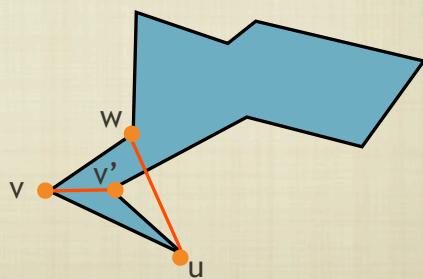
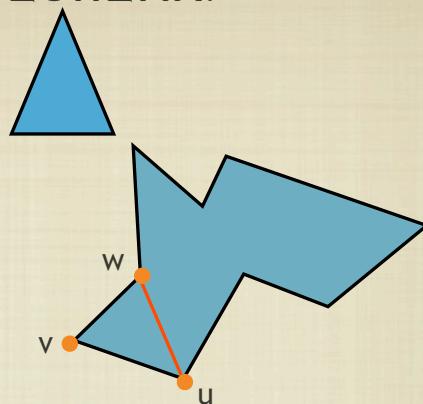


## PROVA POR INDUÇÃO DO TEOREMA:

- **CASO BÁSICO:  $N=3$  - TRIVIAL**

- **PASSO DE INDUÇÃO ( $N > 3$ )**

- ASSUME UMA TRIANGULAÇÃO TP
- CONSIDERE UMA DIAGONAL QUE CORTA P EM DOIS SUB-POLÍGONOS  $P_1$  E  $P_2$  COM  $M_1$  E  $M_2$  VÉRTICES
- CADA VÉRTICE DE P OCORRE UMA VEZ NOS DOIS SUB-POLÍGONOS, COM EXCEÇÃO DOS DOIS VÉRTICES QUE DEFINEM A DIAGONAL, PORTANTO:
- $M_1 + M_2 = N + 2$

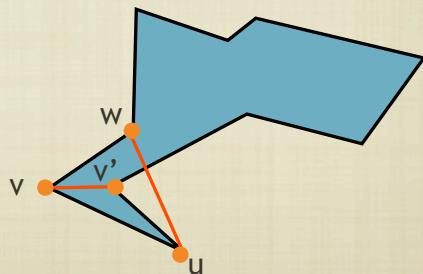
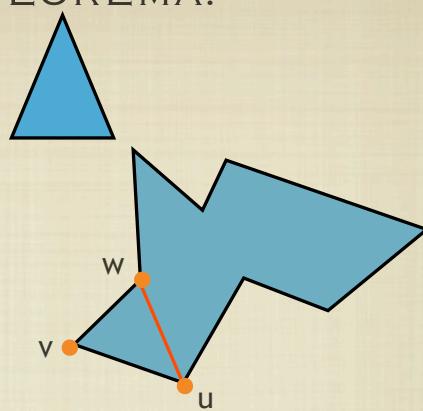


## PROVA POR INDUÇÃO DO TEOREMA:

- **CASO BÁSICO:  $N=3$  - TRIVIAL**

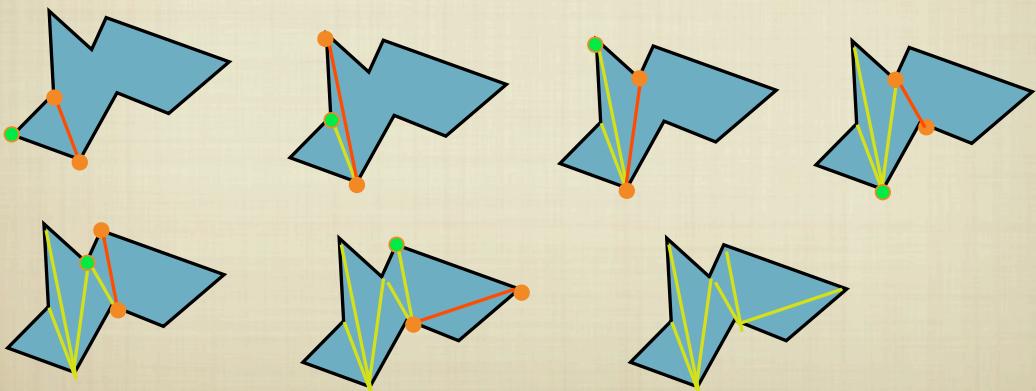
- **PASSO DE INDUÇÃO ( $N > 3$ )**

- ASSUME UMA TRIANGULAÇÃO TP
- CONSIDERE UMA DIAGONAL QUE CORTA P EM DOIS SUB-POLÍGONOS  $P_1$  E  $P_2$  COM  $M_1$  E  $M_2$  VÉRTICES
- CADA VÉRTICE DE P OCORRE UMA VEZ NOS DOIS SUB-POLÍGONOS, COM EXCEÇÃO DOS DOIS VÉRTICES QUE DEFINEM A DIAGONAL, PORTANTO:
- $M_1 + M_2 = N + 2$
- $P_1$  E  $P_2$  POSSUEM  $M_1-2$  E  $M_2-2$  TRIÂNGULOS (HI):
- $TP = M_1 - 2 + M_2 - 2$   
 $= M_1 + M_2 - 4$   
 $= N + 2 - 4$   
 $= N - 2$



# ALGORITMO SIMPLES

- 1. ESCOLHER O VÉRTICE MAIS A ESQUERDA**
- 2. ENCONTRAR A DIAGONAL**
- 3. TRIANGULAR OS DOIS POLÍGONOS**



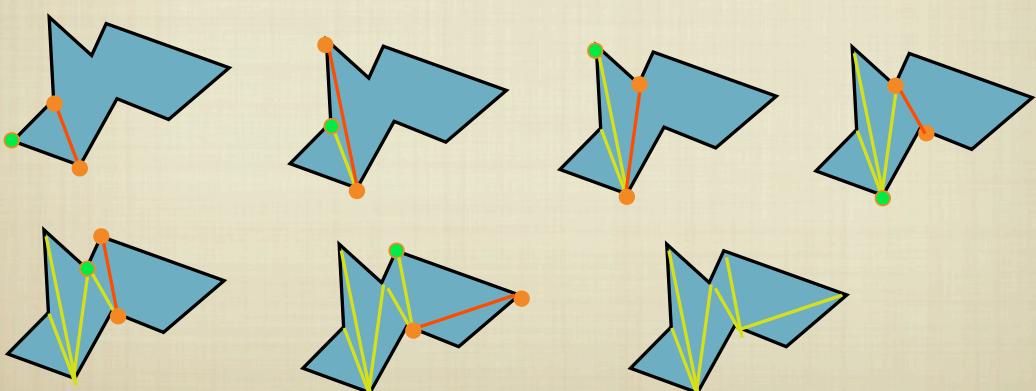
# ALGORITMO SIMPLES

- 1. ESCOLHER O VÉRTICE MAIS A ESQUERDA**

$O(n)$  encontrar a diagonal

Diagonal divide polígono de  $n-1$  vértices  
 $\rightarrow O(n^2)$

- 2. ENCONTRAR A DIAGONAL**
- 3. TRIANGULAR OS DOIS POLÍGONOS**



# PROBLEMA DA GALERIA DE ARTE

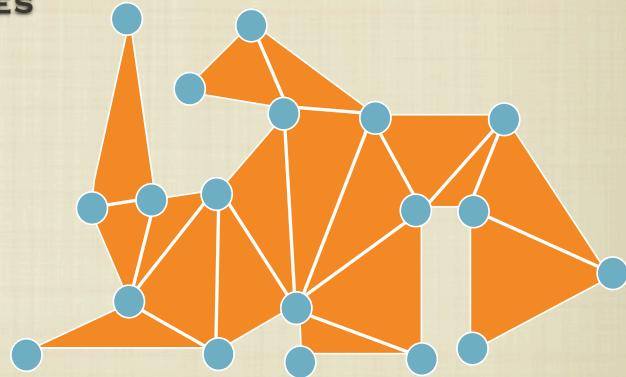
- USANDO TEOREMA, COLOCAR **N-2** GUARDAS PARA CADA TRIÂNGULO DA TRIANGULAÇÃO DE P
- MELHOR QUE ISSO ?

# PROBLEMA DA GALERIA

- USANDO TEOREMA, COLOCAR **N-2** GUARDAS PARA CADA TRIÂNGULO DA TRIANGULAÇÃO DE P
- MELHOR QUE ISSO ?
  - UMA DIAGONAL GUARDA DOIS TRIÂNGULOS, TALVEZ REDUZIR PARA  $\sim N/2$
  - COLOCAR CÂMERAS SOBRE VÉRTICES, JÁ QUE ALGUNS VÉRTICES SÃO ADJACENTES A VÁRIOS TRIANGS

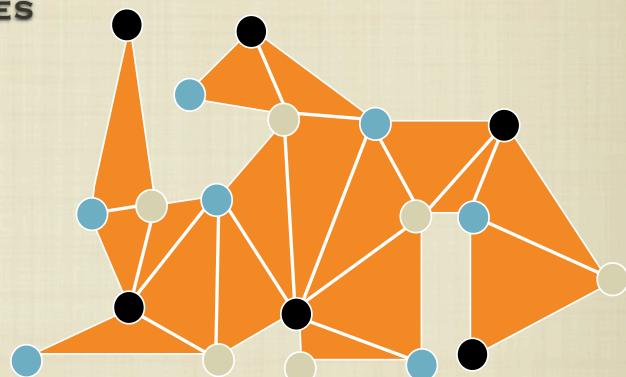
# PROBLEMA DA GALERIA DE ARTE

- SELEÇÃO UM  
SUBCONJUNTO DE VÉRTICES  
COMO GUARDAS



# PROBLEMA DA GALERIA

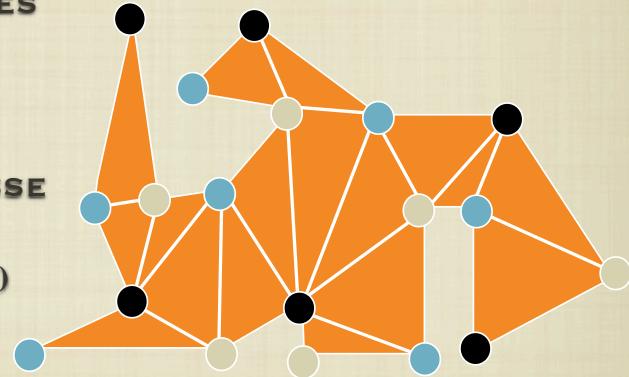
- SELEÇÃO UM  
SUBCONJUNTO DE VÉRTICES  
COMO GUARDAS
- USE ALGORITMO DE  
COLORAÇÃO-3 DE GRAFOS



# PROBLEMA DA GALERIA

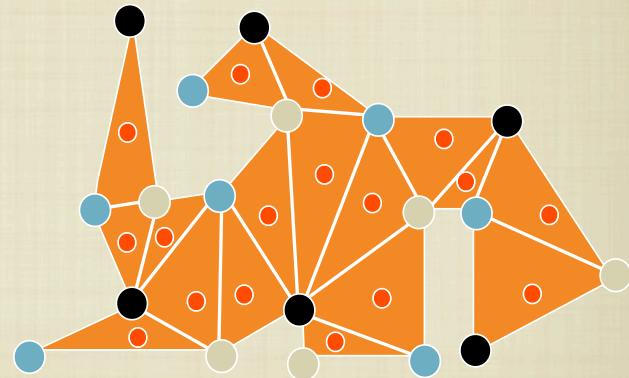
- SELEÇÃO UM SUBCONJUNTO DE VÉRTICES COMO GUARDAS
- USE ALGORITMO DE COLORAÇÃO-3 DE GRAFOS
- ESCOLHA A MENOR CLASSE COMO GUARDAS
- RESULTADO:  $\text{FLOOR}(N/3)$  GUARDAS
- COLORAÇÃO SEMPRE EXISTE ?

SIM



## COLORAÇÃO SEMPRE EXISTE ?

- $G(TP)$  É O DUAL DE  $TP$
- UM TRIÂNGULO POR NODO  $V$ :  $T(V)$



# COLORAÇÃO SEMPRE

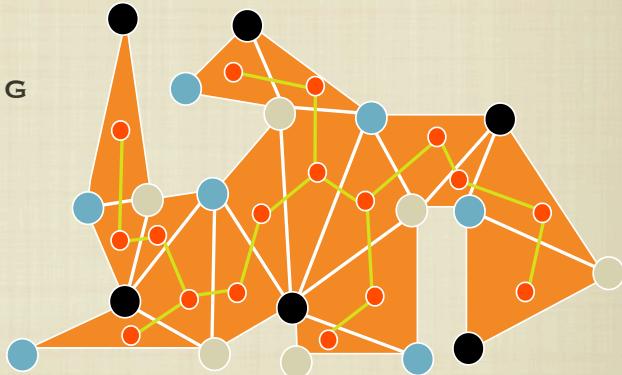
## • DIAGONAL CORTA $G(T_P)$ EM DOIS:

- REMOVER UMA ARESTA DE  $G(T_P)$  CORTA O GRAFO EM 2

- $G(T_P)$  É UMA ÁRVORE

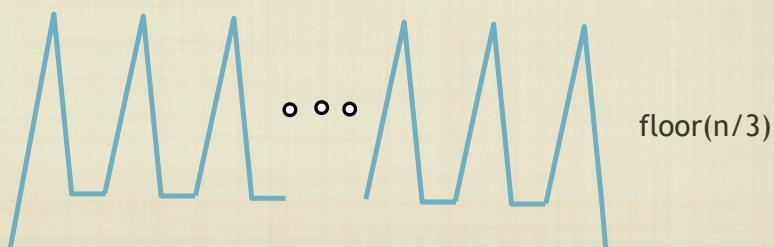
## • 3-COLORAÇÃO

- BUSCA EM PROFUNDIDADE NA ÁRVORE



# PROBLEMA DA GALERIA

**Teorema:** Para um polígono simples com  $n$  vértices,  $\text{floor}(n/3)$  guardas são ocasionalmente necessários e sempre suficientes para ter todos os pontos do polígono visível por uma dos guardas



Pente de Chvátal (1975)

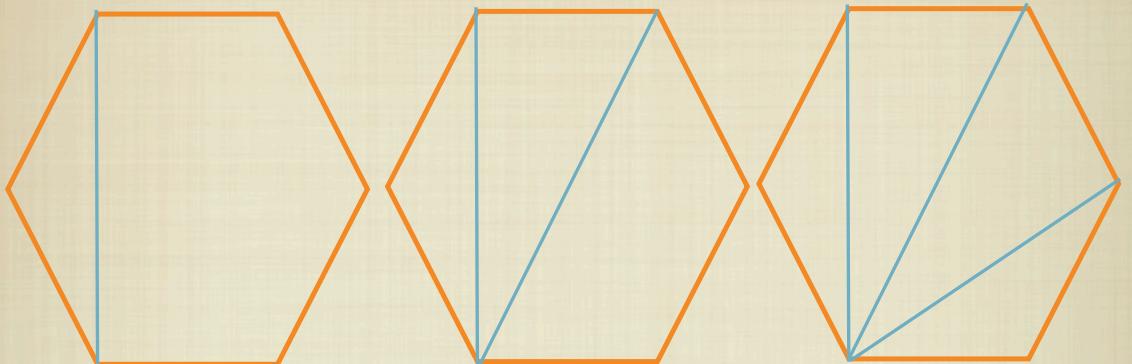
# COMO COLOCAR AS

- **FLOOR[N/3] CAMÉRAS SUFICIENTES**
- **TRIANGULAR UM POLÍGONO:**
  - ALGORITMO RÁPIDO:
    - ENTRADA: POLÍGONO SIMPLES
    - SAÍDA: TRIANGULAÇÃO (LISTA DE ARESTAS DUPLAMENTE CONECTADA)

# COMO COLOCAR AS

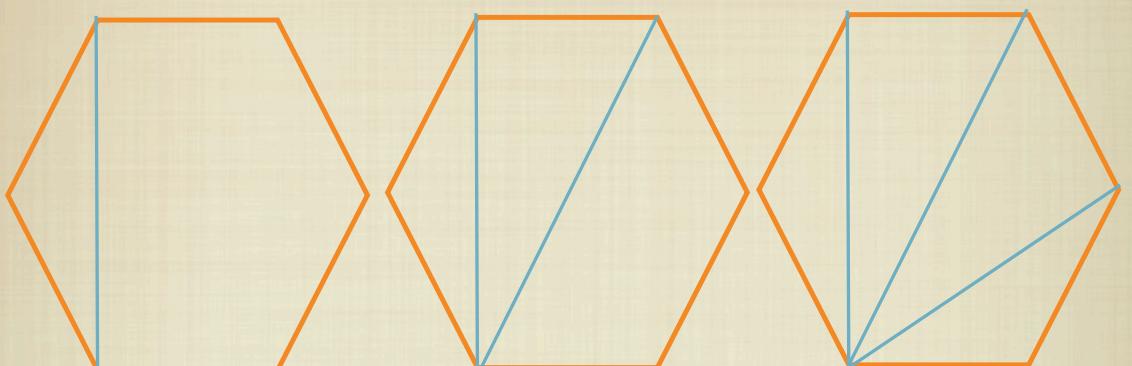
- **FLOOR[N/3] CAMÉRAS SUFICIENTES**
- **TRIANGULAR UM POLÍGONO:**
  - ALGORITMO RÁPIDO:
    - ENTRADA: POLÍGONO SIMPLES
    - SAÍDA: TRIANGULAÇÃO (LISTA DE ARESTAS DUPLAMENTE CONECTADA)
  - PROVA QUE EXISTE UMA TRIANGULAÇÃO SUGERE UM ALGORITMO. QUAL SUA COMPLEXIDADE ?

## IDÉIA: QUEBRAR EM PARTES CONVEXAS



1. Decompor  $P$  em partes convexas
2. Triangular as partes convexas

## IDÉIA: QUEBRAR EM PARTES CONVEXAS

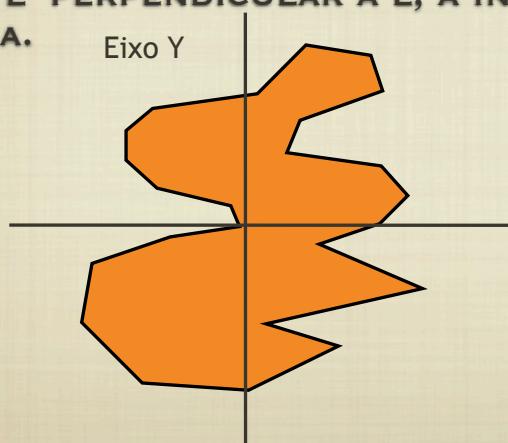


1. Decompor  $P$  em partes convexas
2. Triangular as partes convexas

1. Decompor  $P$  em partes monotônicas
2. Triangular as partes

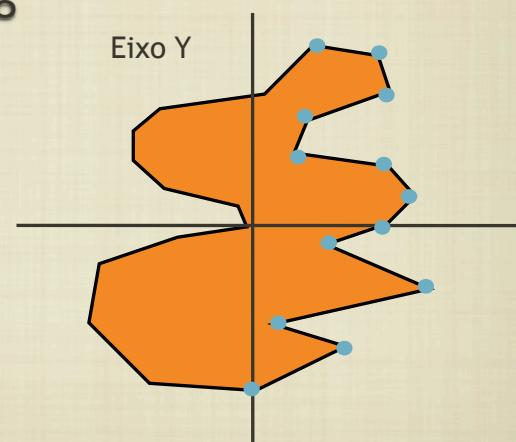
# PARTICIONAMENTO MONOTÔNICO DE POLÍGONOS

- **DEF:** UM POLÍGONO SIMPLES  $P$  É CHAMADO MONOTÔNICO COM RESPEITO A UMA LINHA  $L$  SE PARA QUALQUER LINHA  $L'$  PERPENDICULAR A  $L$ , A INTERSEÇÃO DE  $L'$  E  $P$  É CONEXA.

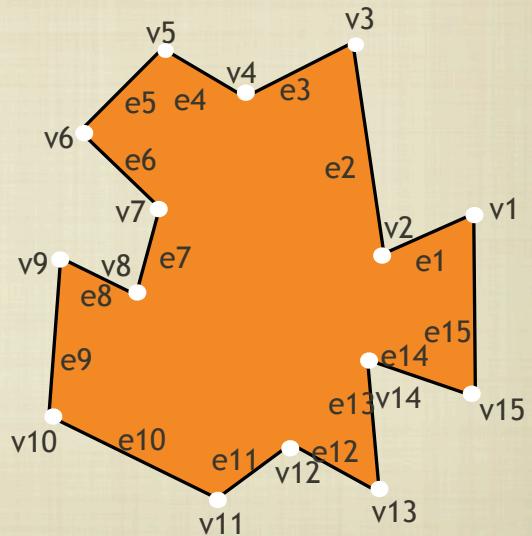


# PARTICIONAMENTO MONOTÔNICO DE POLÍGONOS

1. **PARTICIONAR O POLÍGONO EM PARTES Y-MONOTÔNICAS**
2. **TRIANGULAR AS PARTES MONOTÔNICAS**

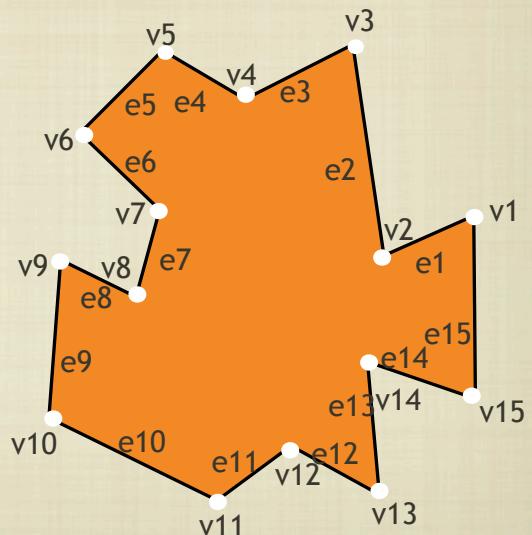


# PARTICIONAMENTO MONOTÔNICO DE POLÍGONOS



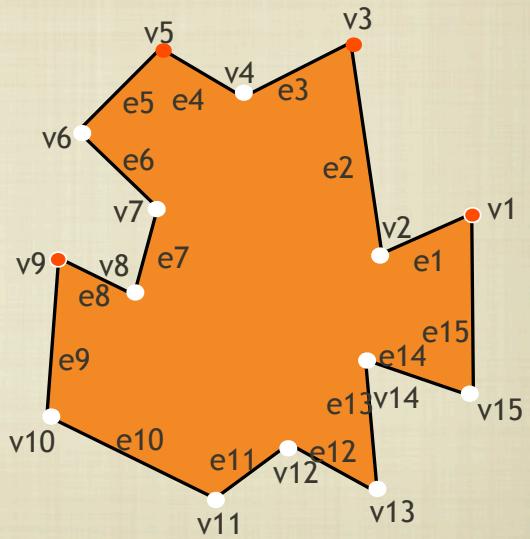
# PARTICIONAMENTO MONOTÔNICO DE POLÍGONOS

**START:** ambos vizinhos estão abaixo,  
e o ângulo interno < 180



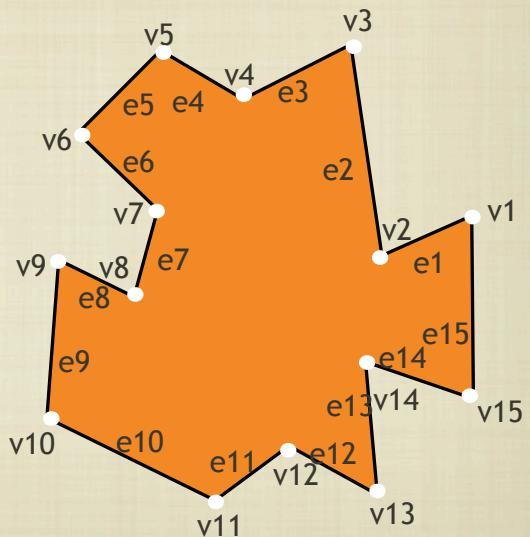
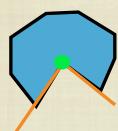
# PARTICIONAMENTO MONOTÔNICO DE POLÍGONOS

**START:** ambos vizinhos estão abaixo, e o ângulo interno < 180



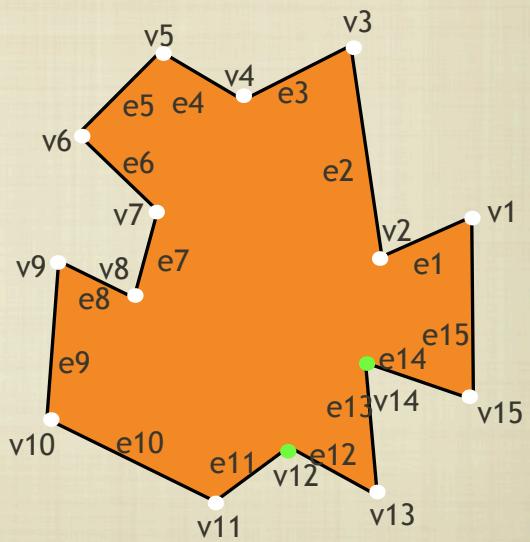
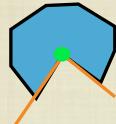
# PARTICIONAMENTO

**SPLIT:** ambos vizinhos estão abaixo, e o ângulo interno > 180



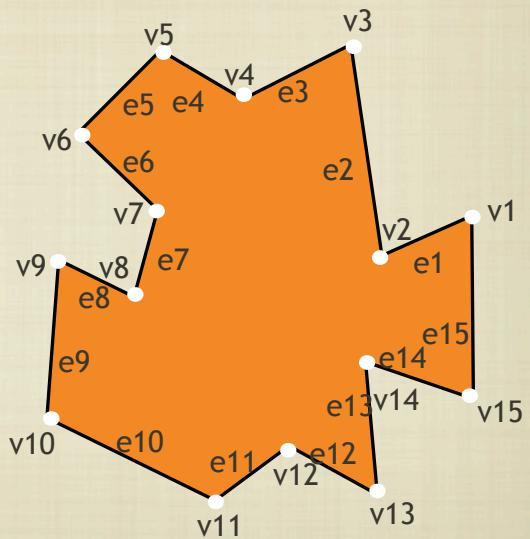
# PARTICIONAMENTO

**SPLIT:** ambos vizinhos estão abaixo, e o ângulo interno > 180



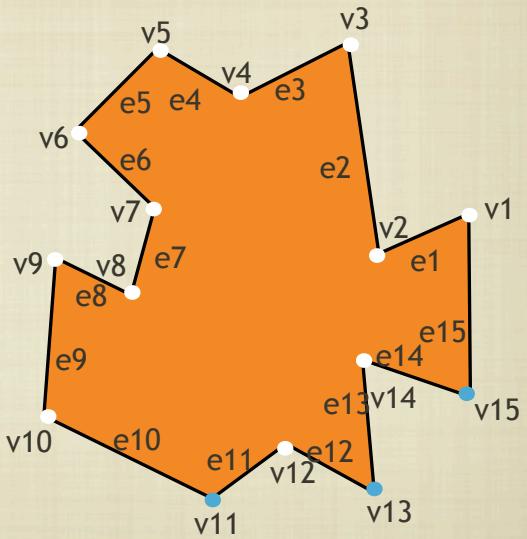
# PARTICIONAMENTO

**END:** ambos vizinhos estão acima, e o ângulo interno < 180



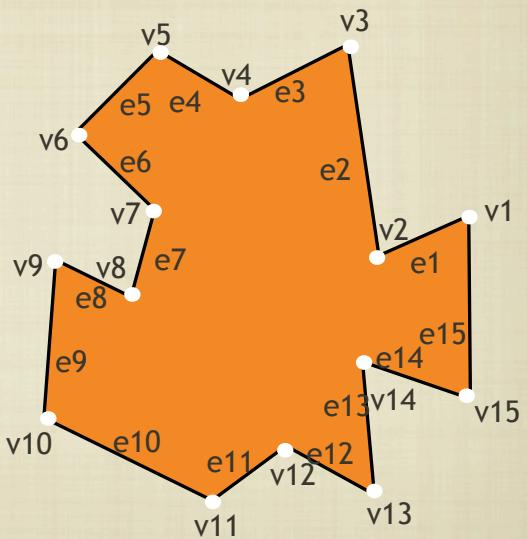
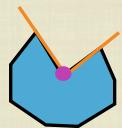
# PARTICIONAMENTO

**END:** ambos vizinhos estão acima, e o ângulo interno < 180



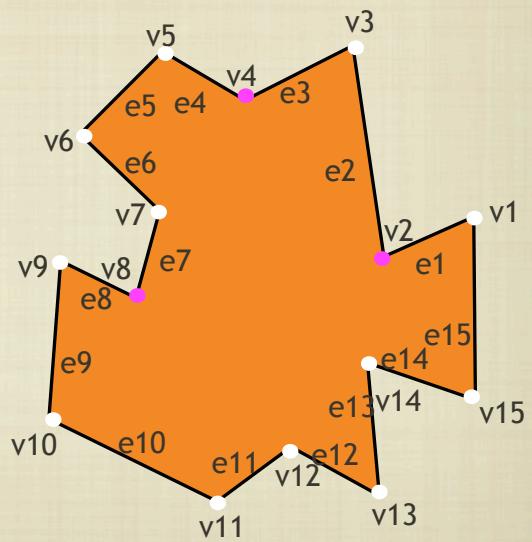
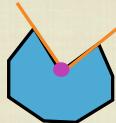
# PARTICIONAMENTO

**MERGE:** ambos vizinhos estão acima, e o ângulo interno > 180



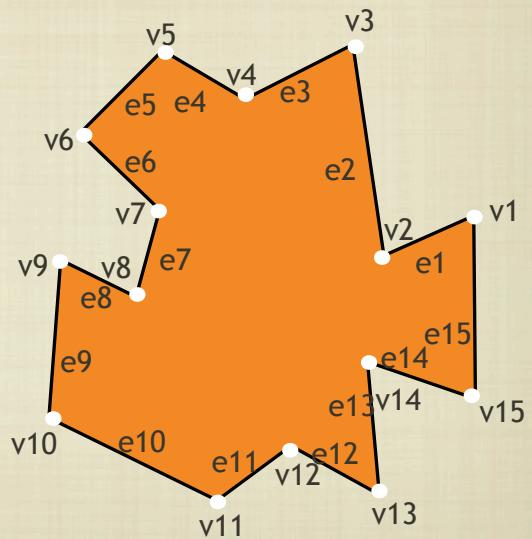
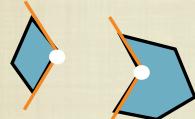
# PARTICIONAMENTO

MERGE: ambos vizinhos estão acima,  
e o ângulo interno > 180



# PARTICIONAMENTO

REGULAR: uma acima, um abaixo



# PARTICIONAMENTO

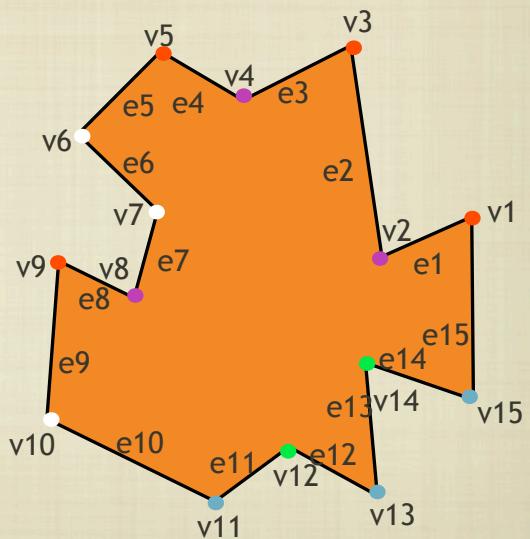
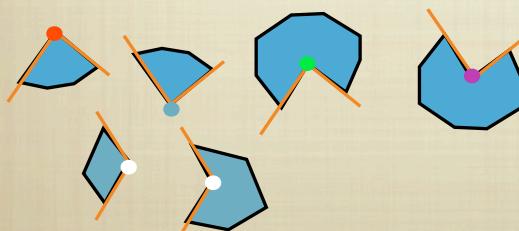
**START:** ambos vizinhos estão abaixo, e o ângulo interno < 180

**SPLIT:** ambos vizinhos estão abaixo, e o ângulo interno > 180

**END:** ambos vizinhos estão acima, e o ângulo interno < 180

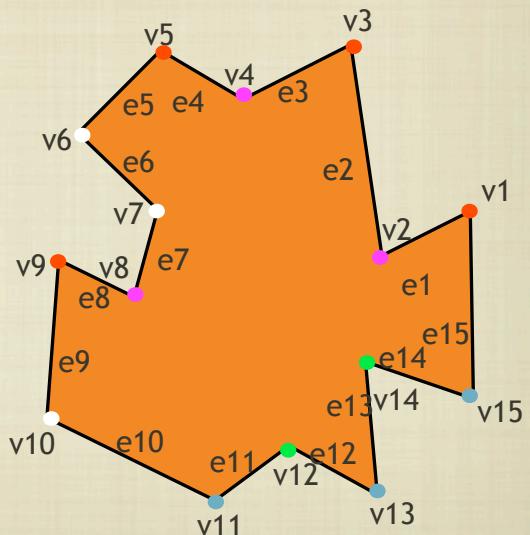
**MERGE:** ambos vizinhos estão acima, e o ângulo interno > 180

**REGULAR:** uma acima, um abaixo



## PARTICIONAMENTO MONOTÔNICO DE POLÍGONOS

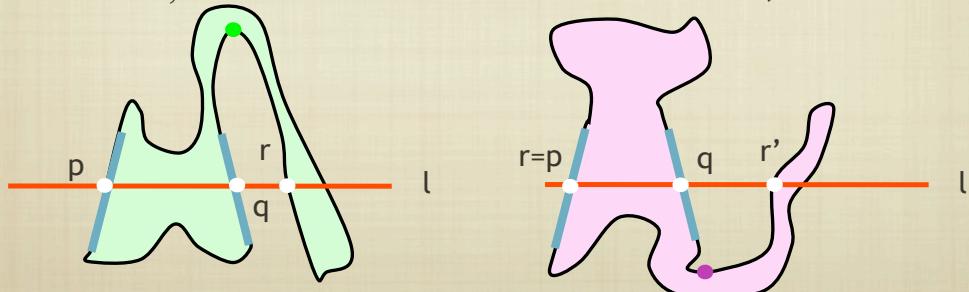
**LEMA:** Um polígono é y-monotônico se não possui vértices SPLIT ou MERGE



## PROVA

- $P$  não é y-monotônico (existe linha horizontal  $l$ ):
  - intersecta  $P$  em mais de um componente conexo
  - componente mais à esquerda seja um segmento de reta
- De  $q$  em direção a  $p$  intercepta  $l$  em um ponto  $r$

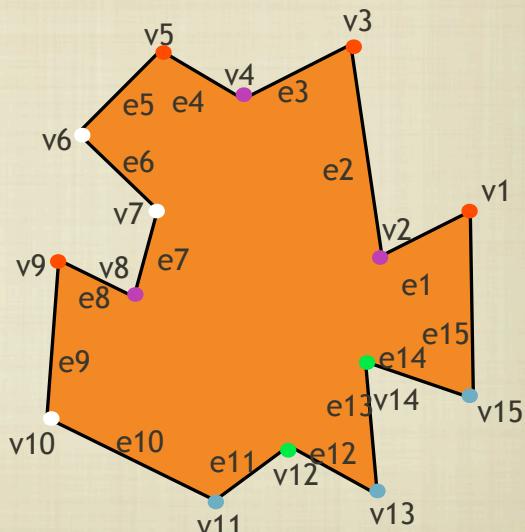
■ SE  $R \neq P$ , O MAIS ALTO VÉRTICE NO TRAJETO DE  $Q$  A  $R$  É SPLIT



## PARTICIONAMENTO

Adicionar diagonais:

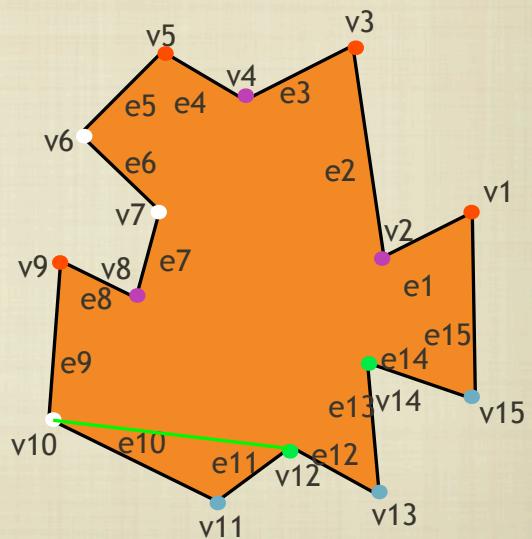
- **SPLIT**: cima
- **MERGE**: baixo



# PARTICIONAMENTO

Adicionar diagonais:

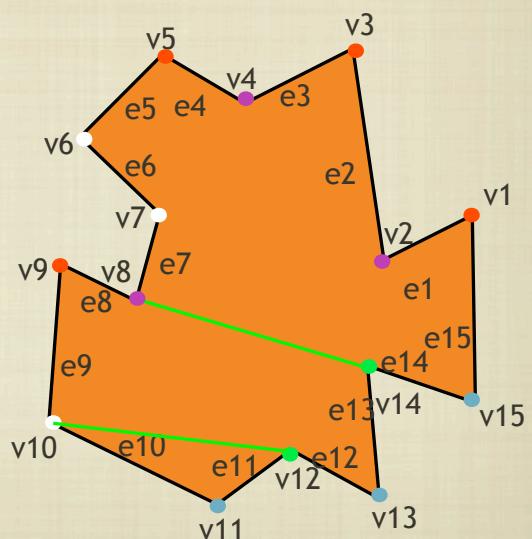
- **SPLIT**: cima
- **MERGE**: baixo



# PARTICIONAMENTO

Adicionar diagonais:

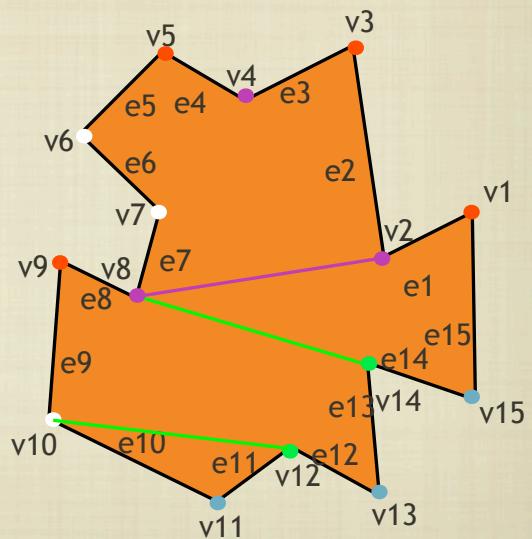
- **SPLIT**: cima
- **MERGE**: baixo



# PARTICIONAMENTO

Adicionar diagonais:

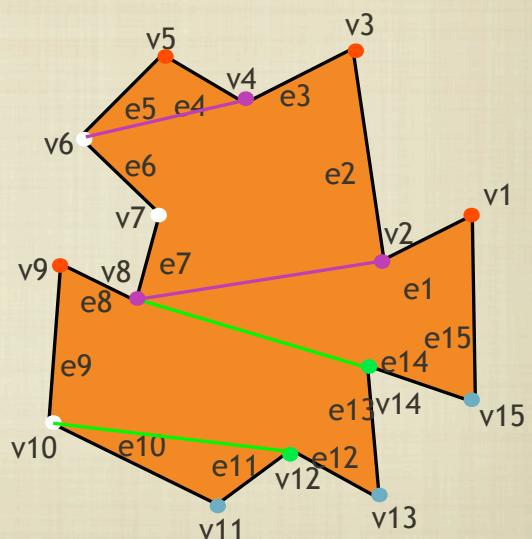
- **SPLIT**: cima
- **MERGE**: baixo



# PARTICIONAMENTO

Adicionar diagonais:

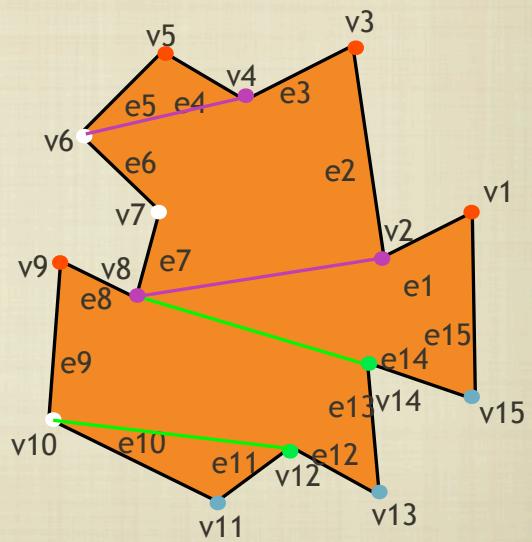
- **SPLIT**: cima
- **MERGE**: baixo



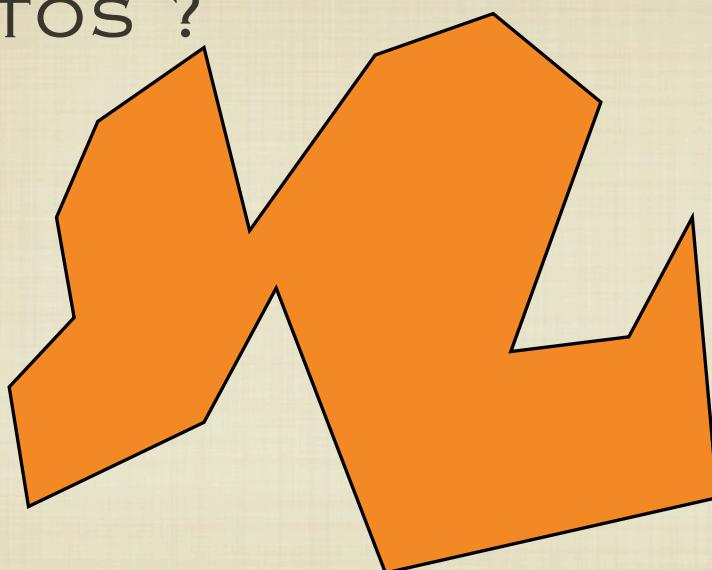
# PARTICIONAMENTO

Adicionar diagonais:

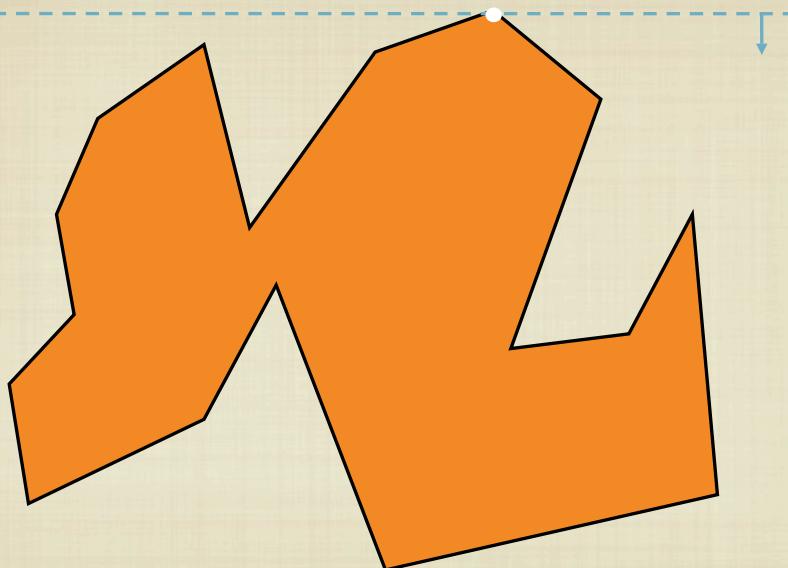
- **SPLIT**: cima
- **MERGE**: baixo



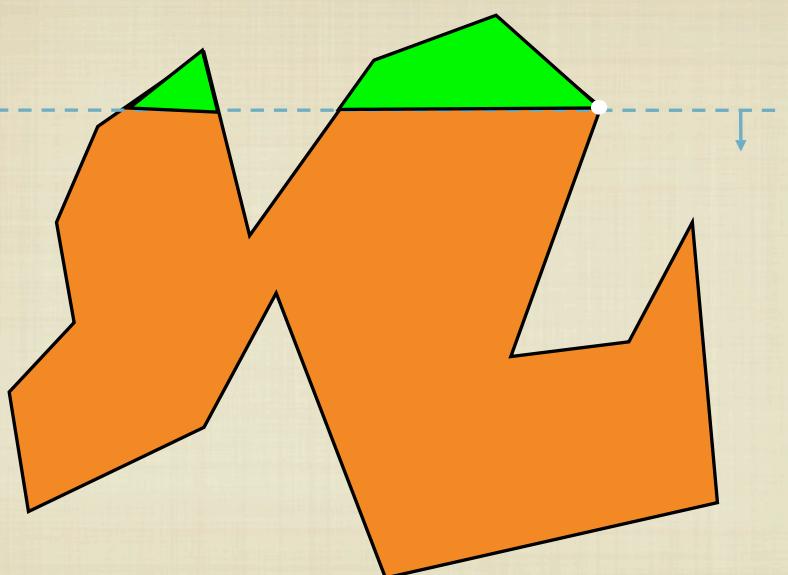
## COMO PROCESSAR OS PONTOS ?



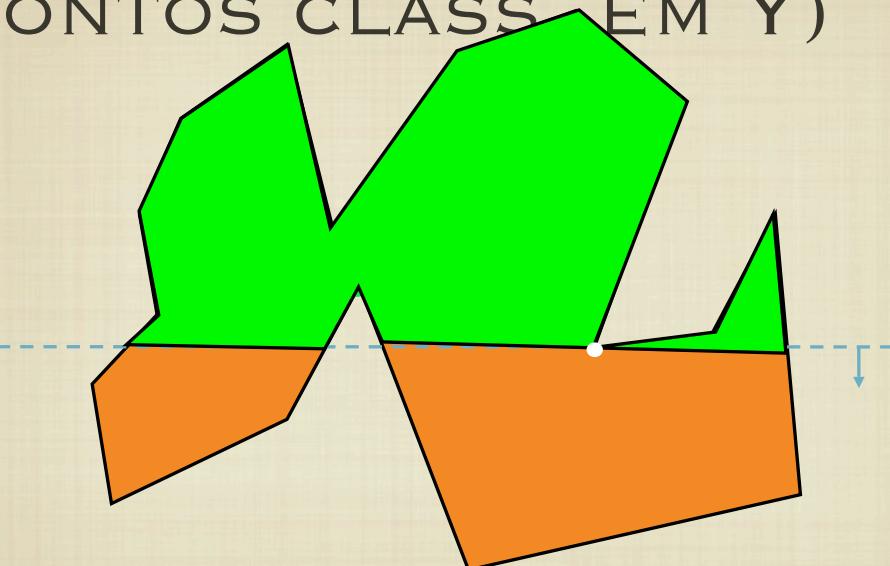
**SWEET VERTICAL**



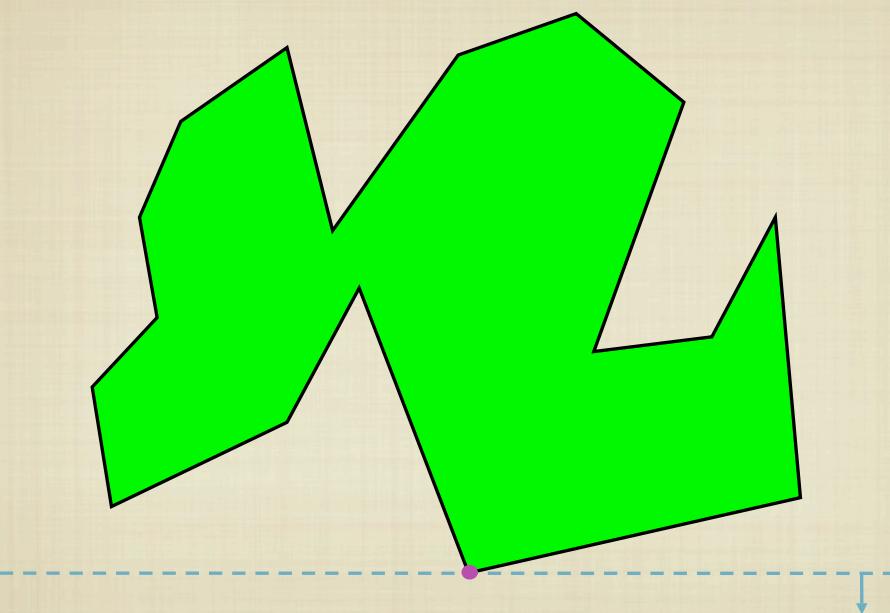
**SWEET VERTICAL**



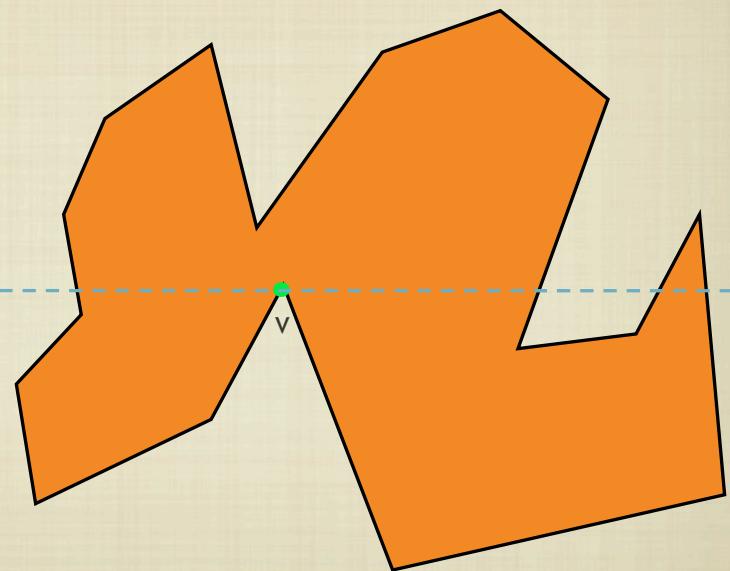
# SWEET VERTICAL (PONTOS CLASS EM Y)



# SWEET VERTICAL

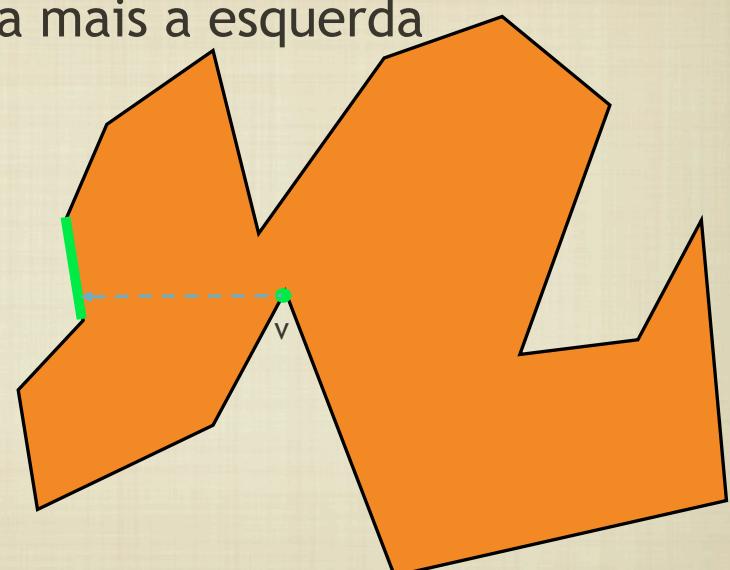


## REMOVENDO SPLIT



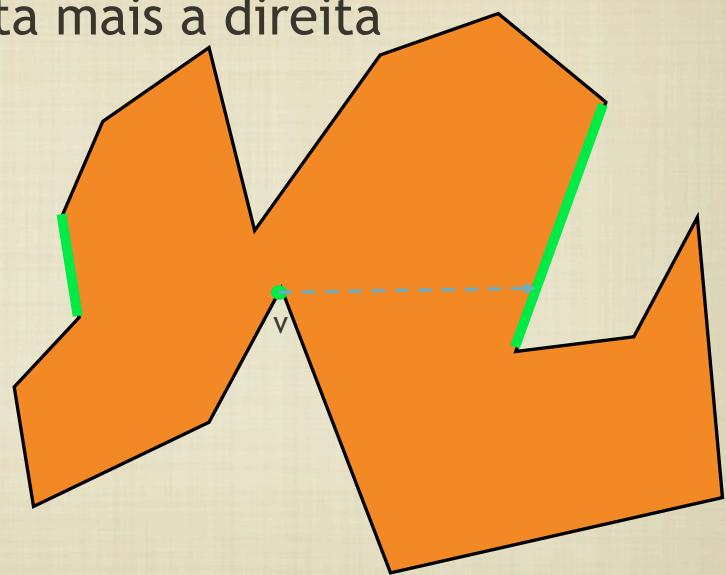
## REMOVENDO SPLIT

Encontrar aresta mais a esquerda



## REMOVENDO SPLIT

Encontrar aresta mais a direita



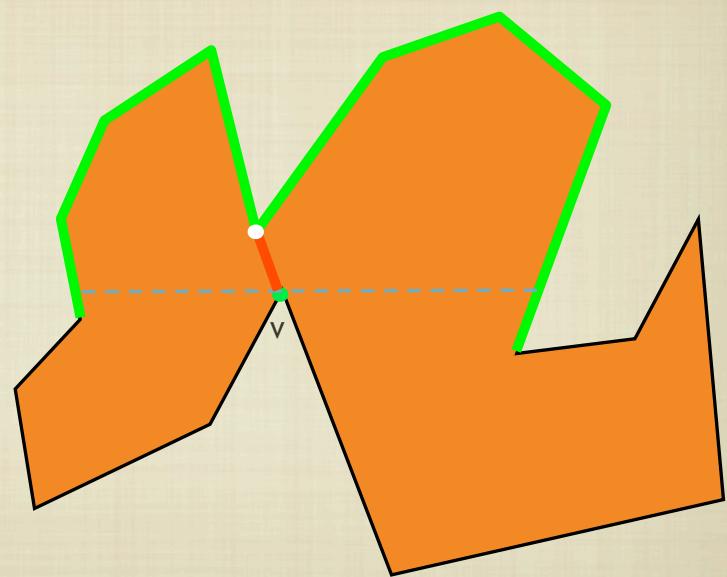
## REMOVENDO SPLIT

Procurar pelo ponto mais baixo  
do segmento, mas acima de v

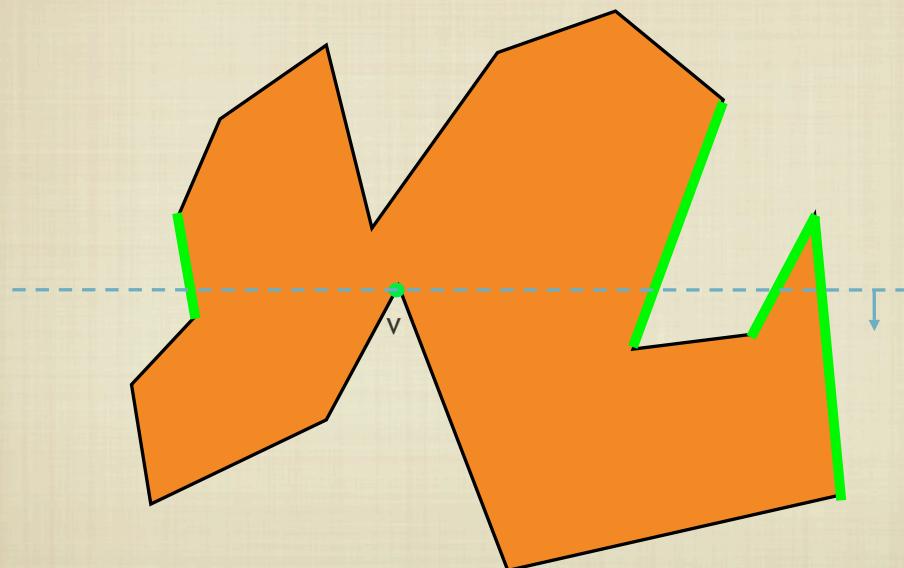


# REMOVENDO SPLIT

Criar diagonal

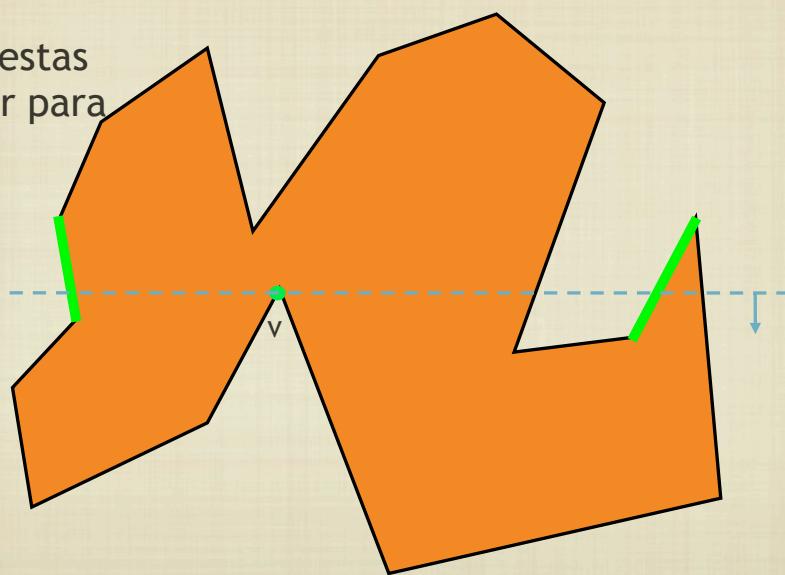


# ARMAZENAR ARESTAS



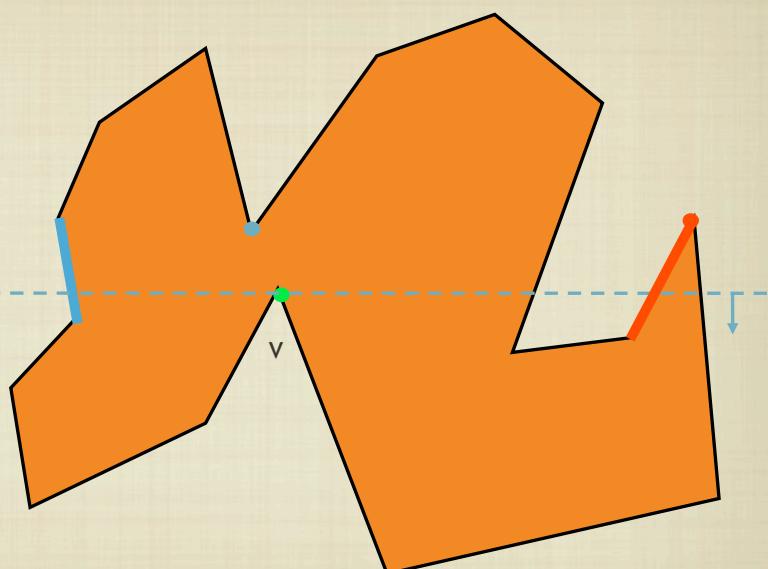
## ARMAZENAR ARESTAS

Remover arestas com interior para esquerda



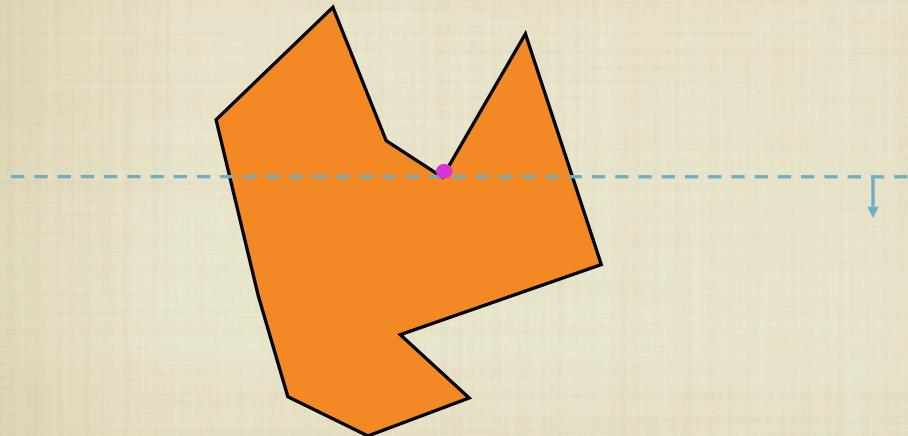
Armazenar numa arvore binaria T, ordenada da esquerda para direita

## ARMAZENAR **HELPER**

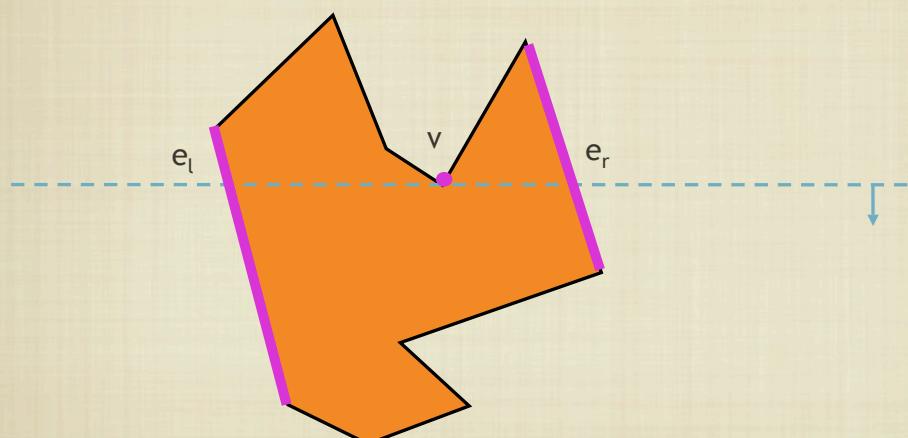


**Helper(*e*)** = vertice *h* mais baixo acima da linha de sweep de forma que o segmento horizontal que o conecta a *e* esta dentro do polígono

# REMOVENDO MERGE

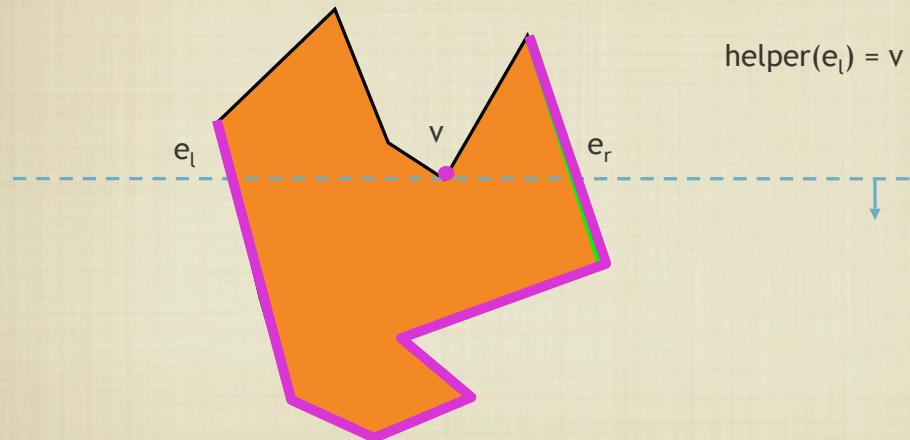


# REMOVENDO MERGE



# REMOVENDO MERGE

Procurar pelo ponto mais alto  
do segmento, mas **abaixo** de v

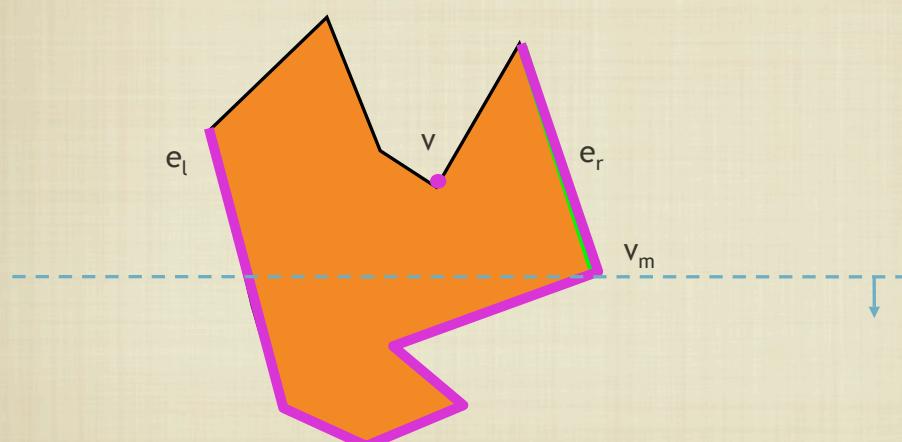


Problema: Este ponto ainda não foi encontrado pela sweep line !!!

Achar o vertice mais tarde !

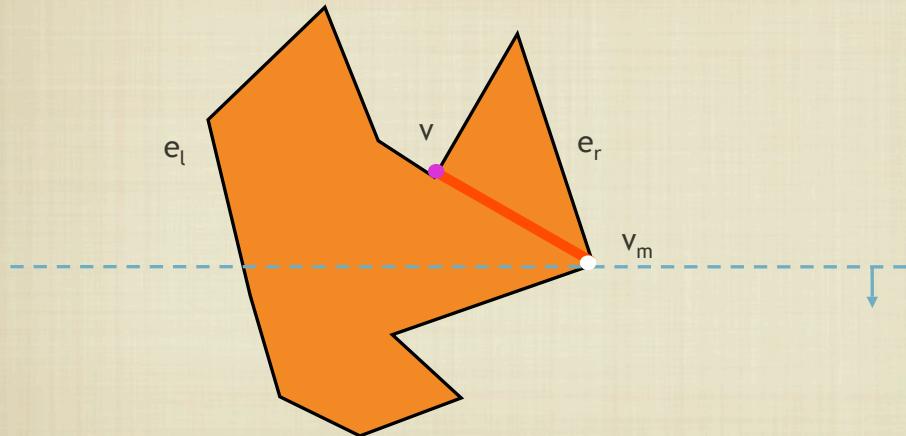
# REMOVENDO MERGE

O vertice procurado substitui v como helper de  $e_l$



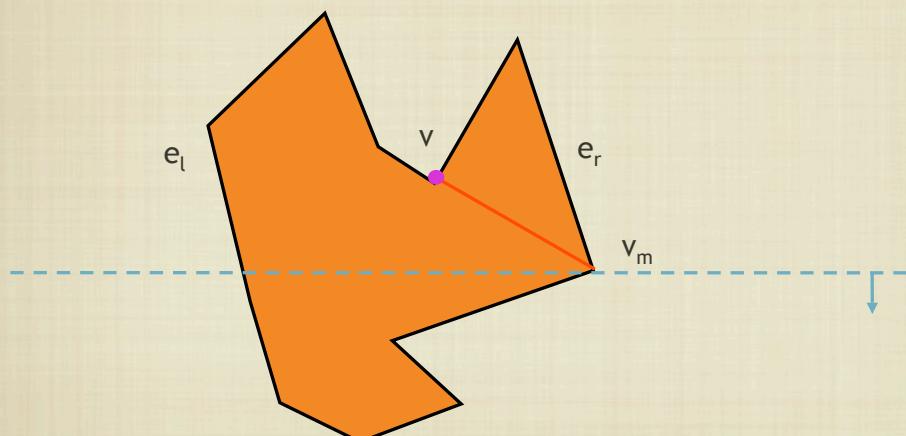
# REMOVENDO MERGE

Criar a diagonal se o antigo helper for um MERGE vertex



# REMOVENDO MERGE

O vertice procurado substitui  $v$  como helper de  $e_l$



# EVENTOSTARTVERTEX, EVENTOENDVERTEX

EventoStartVertex( $v_i$ )

1. Inserir  $e_i$  em T e atribuir helper( $e_i$ ) para  $v_i$

EventoEndVertex( $v_i$ )

1. IF helper( $e_{i-1}$ ) é um MERGE vertice, inserir a diagonal conectando  $v_i$  ao helper( $e_{i-1}$ )
2. Deletar  $e_{i-1}$  de T

# EVENTOSPLITVERTEX

EventoSplitVertex( $v_i$ )

1. Busque em T a aresta mais a esquerda de  $v_i$
2. Inserir a diagonal conectando  $v_i$  ao helper  $e_j$
3. helper( $e_j$ ) =  $v_i$
4. Inserir  $e_i$  em T e atribuir helper( $e_i$ ) =  $v_i$

# EVENTOMERGEVERTEX

EventoMergeVertex( $v_i$ )

1. IF helper( $e_{i-1}$ ) é um MERGE vértice
2. THEN inserir diagonal conectando  $v_i$  ao helper  $e_{j-1}$
3. Deletar  $e_{i-1}$
4. Buscar em T a aresta  $e_j$  a esquerda de  $v_i$
5. IF helper( $e_j$ ) é um MERGE vertex
6. THEN inserir a diagonal conectando  $v_i$  ao helper( $e_j$ )
7. helper( $e_j$ ) =  $v_i$

# EVENTOREGULARVERTEX

EventoRegularVertex( $v_i$ )

1. IF o interior está a direita de  $v_i$
2. THEN IF helper( $e_{i-1}$ ) é um MERGE vértice
3. THEN inserir diagonal conectando  $v_i$  ao helper  $e_{j-1}$
4. Deletar  $e_{i-1}$
5. Inserir  $e_i$  em T e helper( $e_i$ )= $v_i$
6. ELSE Buscar em T a aresta  $e_j$  a esquerda de  $v_i$
7. IF helper( $e_j$ ) é um MERGE vertex
8. THEN inserir uma diagonal conectando  $v_i$  ao helper( $e_j$ )
9. helper( $e_j$ ) =  $v_i$

# PARTICIONAMENTO

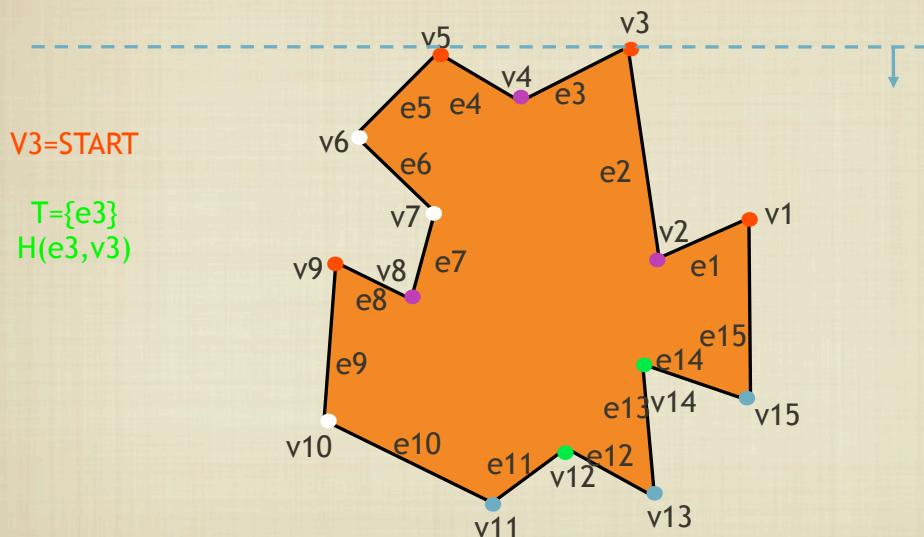
## ALGORITMO: MakeMonotone(P)

**Entrada:** Polígono simples P armazenado em lista de arestas duplamente encadeada D

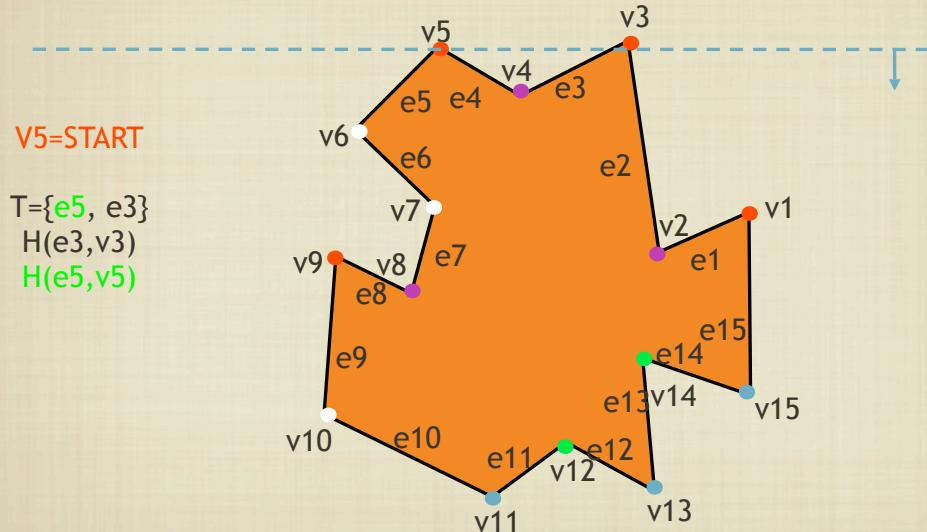
**Saida:** Particionamento de P em subpolígonos monotônicos

1. Construir uma priority queue Q com os vértices de P, usando a coordenada y como prioridade. Se dois pontos possuem mesma coordenada y, aquele com menor coordenada x possui a maior prioridade
2. Inicializar uma árvore binária de busca T
3. WHILE Q não está vazio
4. DO Remove o vértice  $v_i$  com prioridade mais alta de Q
  - Chame o procedimento apropriado para lidar com o vértice, dependendo do tipo

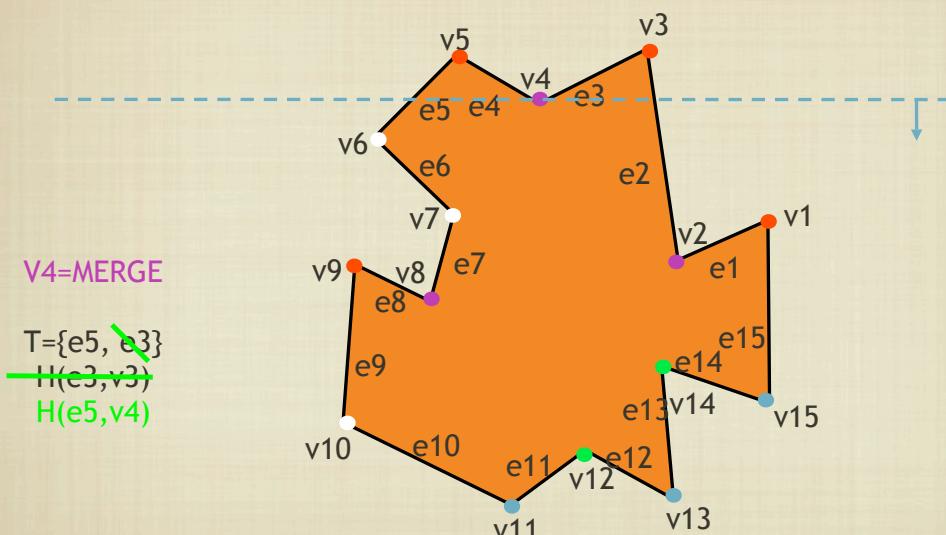
## EXEMPLO



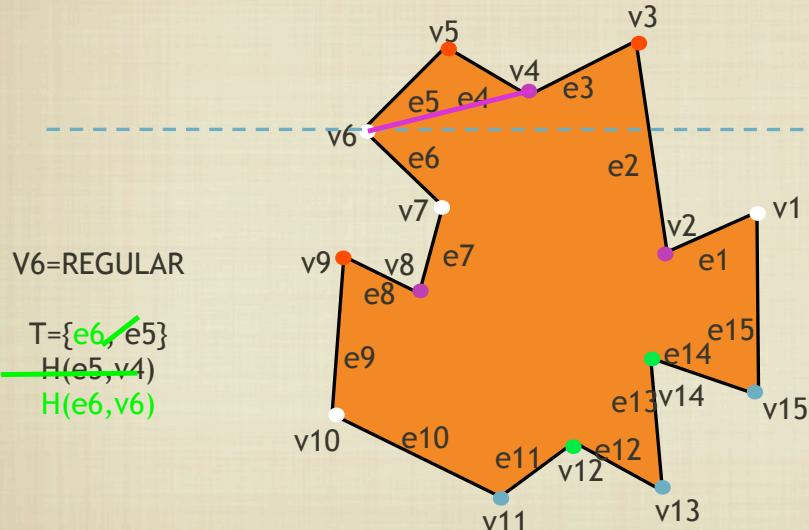
## EXEMPLO



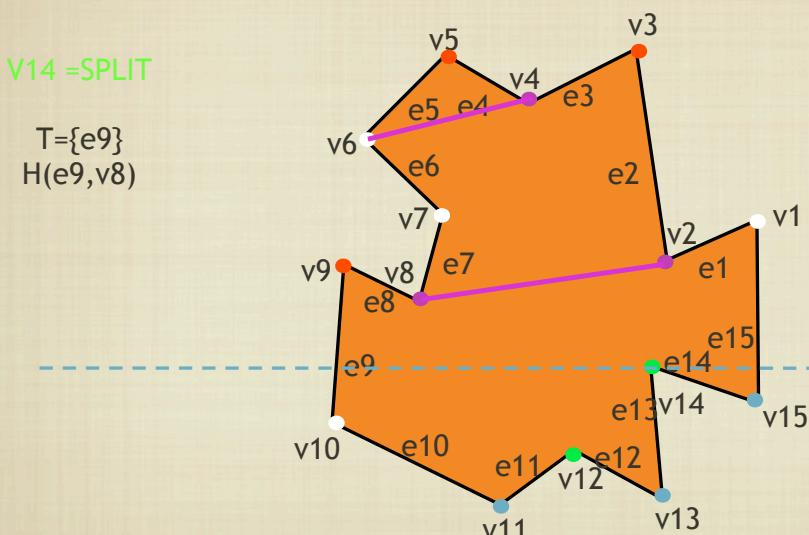
## EXEMPLO



## EXEMPLO



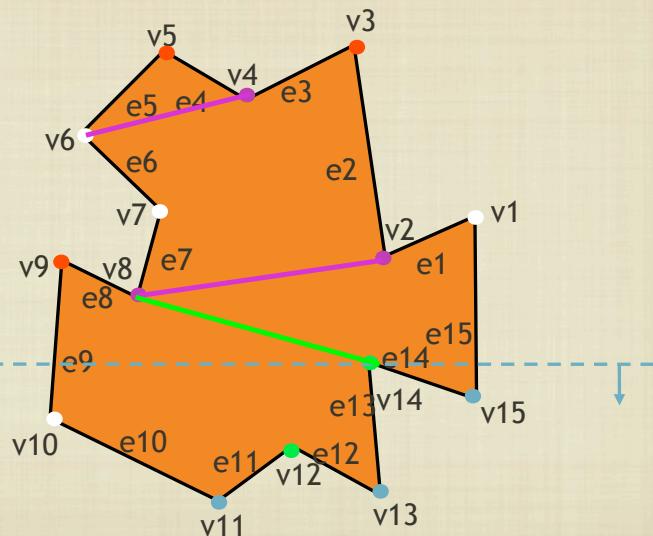
## EXEMPLO



## EXEMPLO

V14 =SPLIT

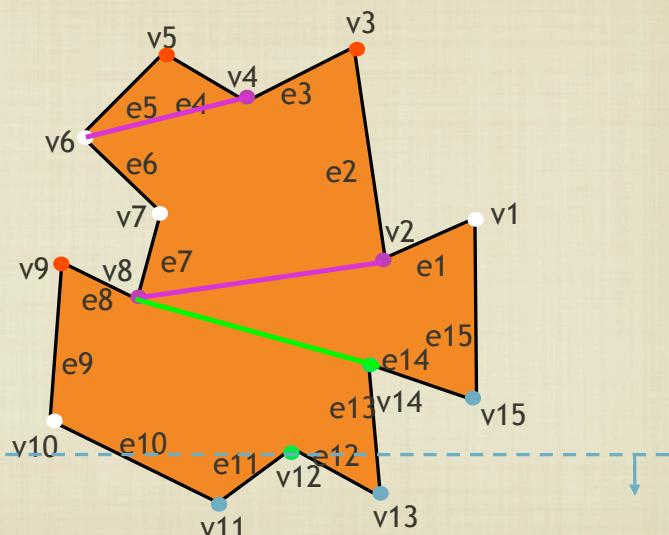
T={e9, e14}  
H(e9,v14)  
H(e14,v14)



## EXEMPLO

V12 =SPLIT

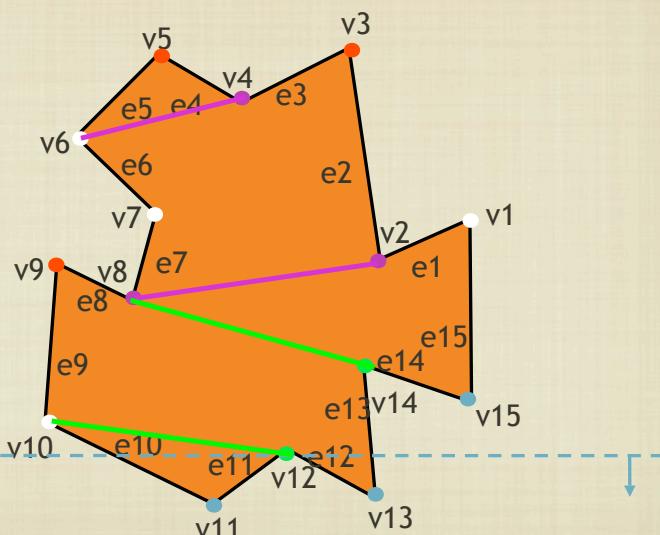
T={e10}  
H(e10,v10)



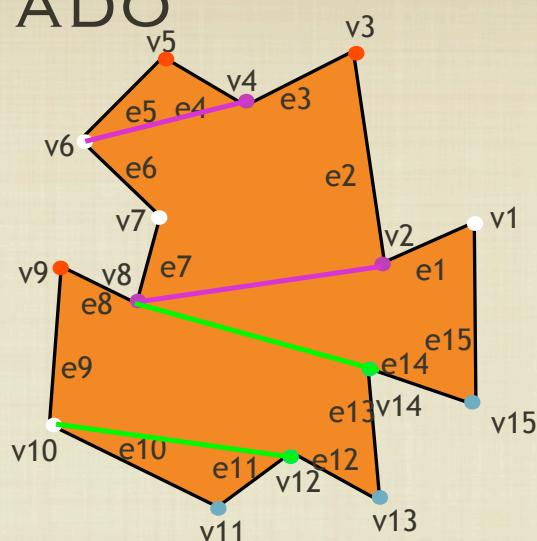
# EXEMPLO

V12 =SPLIT

$T=\{e10\}$   
 $H(e10, v10)$

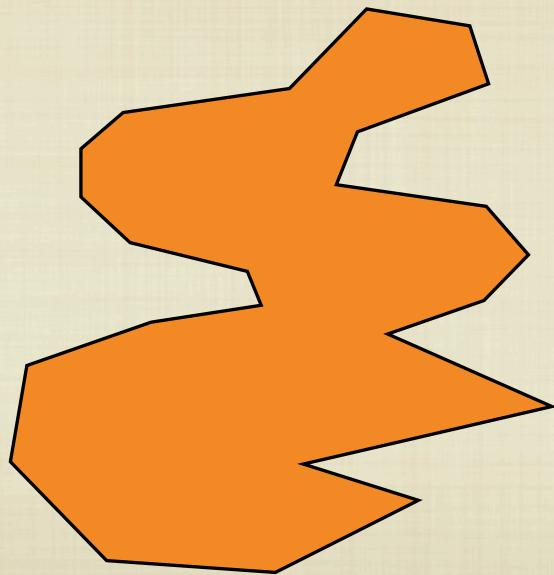


# RESULTADO

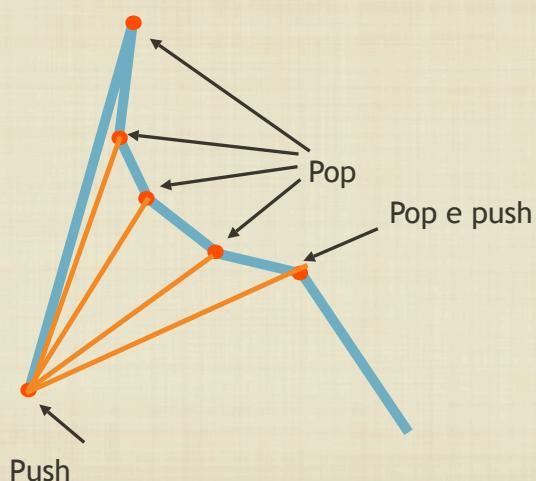


**Teorema:** Um polígono simples com  $n$  vértices pode ser triangulado em polígonos  $y$ -monotônicos em  $O(n \log n)$  com  $O(n)$  armazenamento

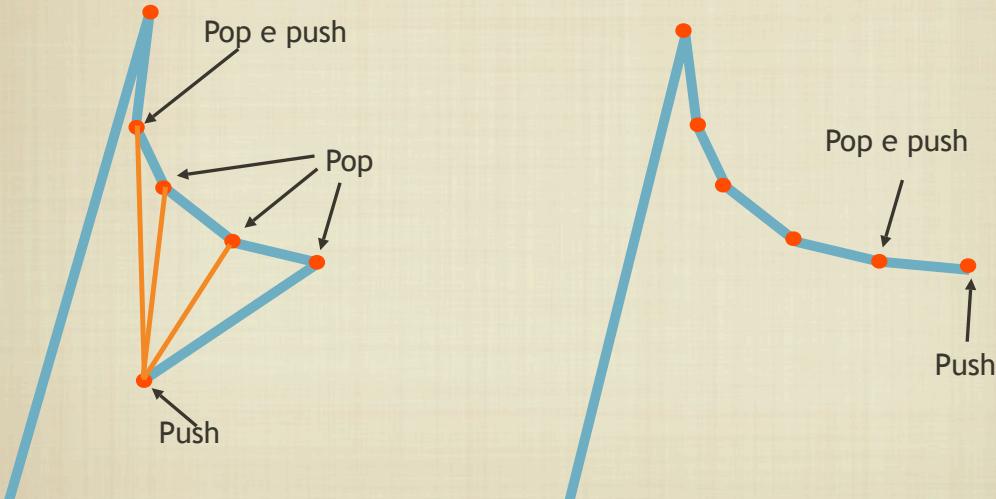
# TRIANGULAR UM POLÍGONO MONOTÔNICO



## CADEIAS DIFERENTES



# MESMA CADEIA



## TRIANGULAR UM POLÍGONO MONOTÔNICO

1. Processar os vertices em ordem decrescente de y ( $u_1, u_2, \dots, u_n$ )
2. Colocar  $u_1$  e  $u_2$  em uma pilha S
3. **FOR**  $j=3$  TO  $n-1$
4. **DO IF**  $u_j$  e o vertice no topo de S estao em cadeias diferentes
5.     **THEN** Pop todos vertices de S
6.         Insere uma diagonal de  $u_j$  para cada vertice 'popped', exceto o ultimo
7.         Push  $u_{j-1}$  e  $u_j$  em S
8.     **ELSE** Pop um vertice de S
9.         Pop outros vertices de S desde que as diagonais de  $u_j$  estejam dentro de P. Inserir as diagonais. Push o ultimo vertice popped de volta
10.         Push  $u_j$  em S
11. Adicione diagonais de  $u_n$  para todos os vertices em S, execto o primeiro e o ultimo

# RESULTADOS

- **TEOREMA: UM POLIGONO Y-MONOTONICO COM N VERTICES PODE SER TRIANGULADO EM  $O(N)$**
- **TEOREMA: UM POLIGONO SIMPLES COM N VERTICES PODE SER TRIANGULADO EM  $O(N \log N)$  COM  $O(N)$  DE MEMORIA**
- **CHAZELLE(1990): ALGORITMO LINEAR PARA TRIANGULAR UM POLIGONO SIMPLES**