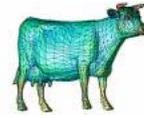




## 4 . Partição de Polígonos

António Leslie Bajuelos  
Departamento de Matemática  
Universidade de Aveiro



Mestrado em Matemática e Aplicações



*“Imagination is more important  
than knowledge”*

*A. Einstein*

## Algumas motivações ...



- A *decomposição* de um polígono, ou de qualquer outra região, em partes mais simples é útil em muitos problemas.
- Tipos mais comuns: *triângulos*, *quadriláteros* ou *peças convexas*

*Uma decomposição diz-se que é uma partição se as componentes do sub-polígono não se sobrepõem, excepto na sua fronteira. Se houver sobreposição de componentes, então dizemos que a decomposição é uma cobertura*

- A *partição de polígonos* é usada (por exemplo) na:
  - *Modelação de objectos em aplicações onde a geometria é importante*
  - *O reconhecimento de padrões*
  - *Compressão de dados*
  - *Processamento de imagens*

3

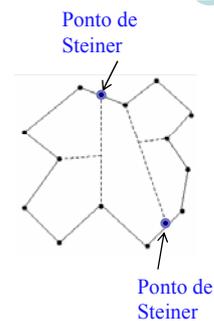
## Algumas motivações ...



- *Classes de polígonos* úteis para a decomposição de polígonos:

- *convexos*
- *estrelados*
- *espirais*
- *monótonos*
- *triângulos*
- *quadrados*
- *rectângulos e trapézios*

- A decomposição em componentes mais simples, pode ser feita (ou não) com a introdução de vértices adicionais, aos quais chamamos *pontos de Steiner*
- O uso de *pontos de Steiner* tornar a decomposição do polígono mais complexa
- Em geral pretende-se que a decomposição seja minimal.
  - **Exemplo:** *Decompor o polígono num número mínimo de componentes*

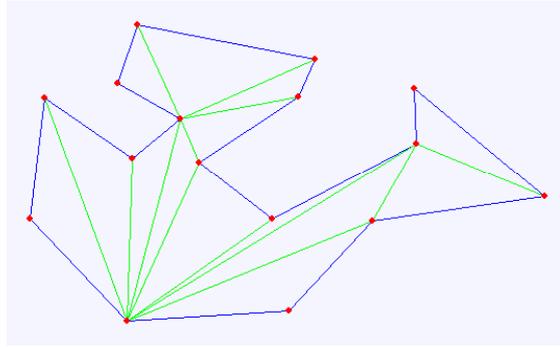


4

## Triangulação de Polígonos: O Problema



- *Dado um polígono  $P$ , com  $n$  vértices, encontrar: diagonais que particionem o polígono em triângulos.*
- Da definição da diagonal  $\Rightarrow$  uma diagonal *corta* um polígono simples em exactamente dois sub-polígonos.



5

## Triangulação de Polígonos



*Será sempre possível encontrarmos  
esta partição?*

**Sim!!!**



6

## Triangulação de Polígonos: Existência



**Teorema:** (Triangulação de polígonos, Lennes, 1911)

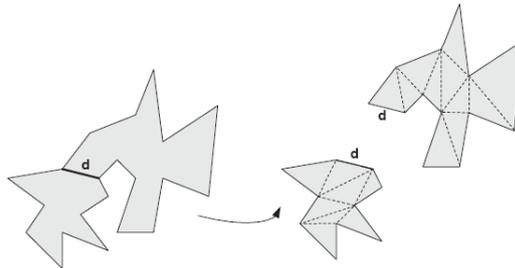
*Todo o polígono admite pelo menos uma triangulação*

**Prova:**

Seja  $P$  um polígono simples. A prova é feita por indução sobre o número de vértices do polígono.

Se  $n = 3$  o polígono é um triângulo ♦

Seja  $n \geq 4$ . Pelo lema de *Meister* todo o polígono com pelo menos 4 vértices possui uma diagonal, então  $P$  possui uma diagonal  $d$ . O segmento  $d$  particiona  $P$  em dois polígonos com menos do que  $n$  vértices; cada um tendo  $d$  como aresta.



7

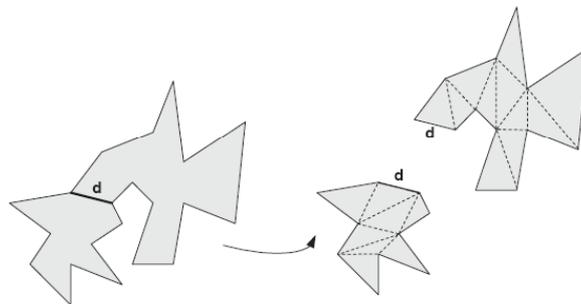
## Triangulação de Polígonos: Existência



**Teorema:** (Triangulação de polígonos) *Todo o polígono admite pelo menos uma triangulação*

**Prova (cont...):**

Aplicando a hipótese indutiva temos que cada um desses polígonos pode ser triangulado. Logo, combinando as triangulações de cada um dos polígonos e  $d$  obtemos uma triangulação de  $P$  ♦



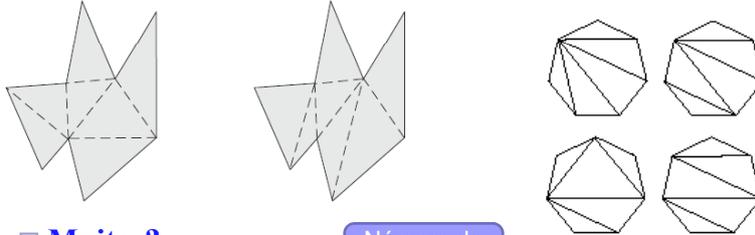
8

## Triangulação de Polígonos: não é única



- A triangulação de um polígono pode não ser única.

□ **Exemplos:**



□ **Muitas?**

$$1 \leq T_n \leq \frac{1}{n-1} \binom{2n-4}{n-2}$$

Número de Catalan

$$T_n = \frac{1}{n-1} \binom{2n-4}{n-2} - \text{número de triangulações de um polígono convexo}$$

9

## Triangulação de Polígonos: Quantas?



- **Teorema:** (Número de triângulos de uma triangulação)  
*Qualquer triangulação de um polígono simples P com n vértices tem exactamente n - 2 triângulos.*

**Prova:**

Consideremos uma diagonal qualquer.

Esta diagonal divide P em dois sub-polígonos,  $P_1$  e  $P_2$  com  $n_1$  e  $n_2$  vértices, respectivamente.

Cada vértice de P pertence exactamente a um dos dois sub-polígonos, excepto para os dois vértices que definem a diagonal, que pertence a ambos. Então,  $n_1 + n_2 = n + 2$ .

Pela indução, qualquer triangulação de  $P_1$  tem  $n_1 - 2$  triângulos o que implica que a triangulação de P tem  $(n_1 - 2) + (n_2 - 2) = n - 2$  triângulos. ♦

10

## Triangulação de Polígonos: Quantas?



### ■ Corolários:

- *Qualquer triangulação de um polígono simples  $P$  com  $n$  vértices tem exactamente  $n - 3$  diagonais.*

#### Prova:

Imediata do teorema anterior ♦

- *A soma dos ângulos internos de um polígono de  $n$  vértices é  $(n-2)\pi$*

#### Prova:

Pelo teorema anterior existem  $(n-2)$  triângulos numa triangulação de um polígono com  $n$  vértices.

Cada triângulo contribui com  $\pi$  para a soma dos ângulos internos ♦

11

## Triangulação de Polígonos: diagonais



Utilizando a definição de diagonal podemos provar facilmente que:

**Lema:** *Seja  $P = \{v_0, v_1, \dots, v_{n-1}\}$  um polígono simples. Então o segmento de recta  $s = v_i v_j$ ,  $i \neq j$  é uma diagonal de  $P$  sse:*

- *para cada aresta  $e$  de  $P$ , que não é incidente a  $v_i$  ou  $v_j$ , temos que  $s \cap e = \emptyset$ ;*
- *$s \subset P$  numa vizinhança de  $v_i$  ou de  $v_j$ .*

12

## Triangulação de Polígonos: diagonais



### Os predicados Left e LeftOn

- Uma recta pode ser determinado por o **segmento orientado**  $ab$ . Então:

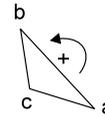
- Se um ponto  $c$  está à esquerda dessa recta  $\Rightarrow$  o terno  $(a, b, c)$  forma um circuito orientado positivamente (CCW)  
 $\Rightarrow A(T(a, b, c)) > 0$

- Predicado **Left(a, b, c)** – verdadeiro se  $c$  está à esquerda da recta determinada pelo segmento orientado  $ab$ , falso em caso contrario

```
bool Left (a, b, c)
{
    return Area (T(a, b, c)) > 0;
}
```

- Predicado **LeftOn(a, b, c)** – verdadeiro se  $c$  está à esquerda ou sobre a recta determinada pelo segmento orientado  $ab$ , falso em caso contrario

```
bool LeftOn (a, b, c)
{
    return Area (T(a, b, c)) ≥ 0;
}
```



13

## Triangulação de Polígonos: verificação de diagonal



- O seguinte segmento testa se um dado segmento é uma diagonal

### Algoritmo Diagonal(P, v[i], v[j])

**Entrada:** um polígono  $P = \{v[0], v[1], \dots, v[n-1]\}$  e dois vértices  $v[i]$  e  $v[j]$ ,  $i \neq j$

**Saída:** TRUE se  $s = v[i]v[j]$ , é uma diagonal de  $P$ , FALSE caso contrário

FOR cada aresta(e) de  $P$  não incidente a  $v[i]$  ou  $v[j]$  DO

IF  $e \cap s \neq \emptyset$  THEN return FALSE

IF  $v[j]$  é convexo ( $\text{LeftON}(v[i-1], v[i], v[i+1]) = \text{TRUE}$ ) THEN

return  $(\neg \text{Left}(v[j], v[i], v[i-1])) \wedge \text{Left}(v[j], v[i], v[i+1]))$

ELSE

return  $\neg(\text{Left}(v[i], v[j], v[i-1]) \wedge (\neg \text{Left}(v[j], v[i], v[i+1])))$

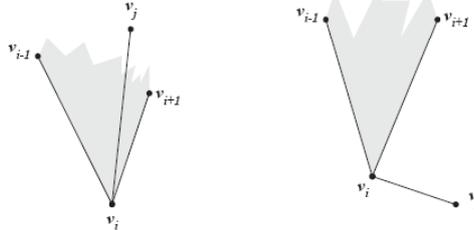
- **Análise:** A complexidade e tempo do **Algoritmo Diagonal** é determinada pela ciclo for i.e. é  $O(n)$ .
- **Teorema:** Seja  $P$  um polígono com  $n$  vértices e sejam  $u$  e  $v$  vértices de  $P$ . Então determinar se o segmento  $uv$  é diagonal leva tempo  $O(n)$ .

14

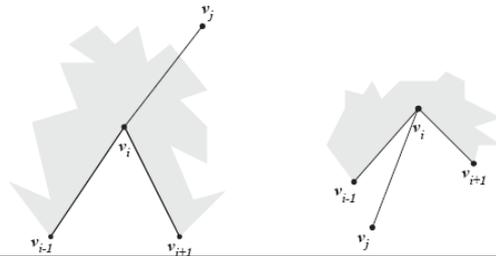
## Triangulação de Polígonos



.....  
IF  $v[i]$  é convexo ( $\text{LeftON}(v[i-1], v[i], v[i+1]) = \text{TRUE}$ ) THEN  
return  $(\neg \text{Left}(v[j], v[i], v[i-1])) \wedge \text{Left}(v[j], v[i], v[i+1])$



ELSE ( $v[j]$  é reflexo)  
return  $\neg(\text{Left}(v[i], v[j], v[i-1]) \wedge (\neg \text{Left}(v[j], v[i], v[i+1])))$



15

## Triangulação de Polígonos: Algoritmo Ingênuo N° 1



### Algoritmo T1(P)

**Entrada:** um polígono  $P = \{v_0, v_2, \dots, v_{n-1}\}$ ,  $n > 3$

**Saída:** uma triangulação de  $P$

WHILE não achou diagonal DO

Seja  $v[i]v[j]$  um segmento candidato a diagonal

IF Diagonal( $P, v[i], v[j]$ ) THEN

achou diagonal

Traçar a diagonal  $v[i]v[j]$

Particione  $P$  em dois subpolígonos

$P_1 = \{v[0], \dots, v[i], v[j], v[j+1], \dots, v[n-1]\}$

$P_2 = \{v[i], v[i+1], \dots, v[j-1], v[j]\}$

T1( $P_1$ )

T1( $P_2$ )

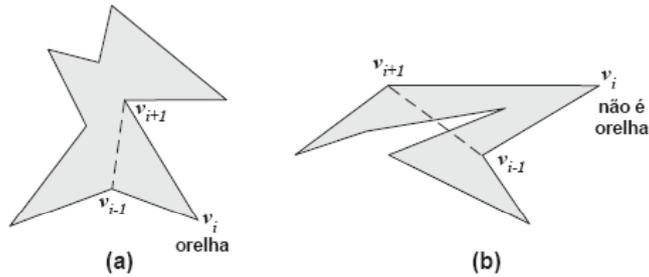
- **Análise:** Existem  $\binom{n}{2} \Rightarrow O(n^2)$  candidatas a diagonal. Testar se um segmento é diagonal é  $O(n)$ . Repetir estas  $O(n^3)$  operações elementares para cada uma das  $n-3$  diagonais será um custo de  $O(n^4)$

16

## Triangulação de Polígonos: Teoremas das duas orelhas



- Diremos que três vértices consecutivos  $u, v, w$  de um polígono formam uma **orelha** se o segmento de recta  $vw$  é uma diagonal.
- Duas orelhas não se sobrepõem se os seus interiores são disjuntos.



17

## Triangulação de Polígonos: Teoremas das duas orelhas



- **Teorema:** (Meister Two Ears Theorem) *Todo polígono com pelo menos 4 vértices possui pelo menos duas orelhas.*

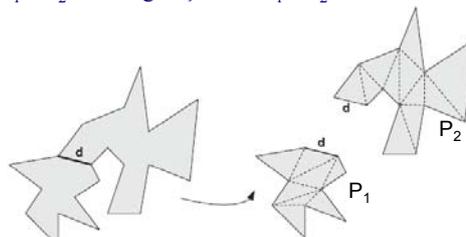


- **Teorema:** *Seja  $P$  um polígono com  $n \geq 4$  vértices e  $T$  uma triangulação de  $P$ . Então pelo menos dois triângulos de  $T$  formam orelhas de  $P$ .*

**Prova:** (por indução no número de vértices de  $P$ )

Se  $n = 4$ ,  $P$  – quadrilátero, os dois triângulos de  $T$  são orelhas de  $P$ .

Seja  $n \geq 5$ . Partimos  $P$  em dois polígonos  $P_1$  e  $P_2$  a través de uma diagonal  $d$  de  $T$ . Sejam  $T_1$  e  $T_2$  as triangulações de  $P_1$  e  $P_2$



18

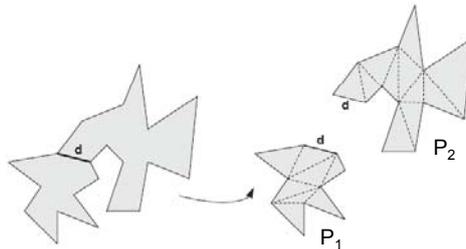
## Triangulação de Polígonos: Teoremas das duas orelhas



- **Teorema:** Seja  $P$  um polígono com  $n \geq 4$  vértices e  $T$  uma triangulação de  $P$ . Então pelo menos dois triângulos de  $T$  formam orelhas de  $P$ .

### Prova (cont...):

A diagonal  $d$  não faz parte de pelo menos um dos triângulos de  $T_1$  que formam orelhas de  $P_1$ , e que portanto, também forma uma orelha de  $P$ . Analogamente, pelo menos um dos triângulos de  $P_2$  forma uma orelha de  $P$ . Como estes triângulos são disjuntos, a prova do teorema está completa. ♦



19

## Triangulação de Polígonos: Algoritmo Ingénuo N° 2



- Podemos acelerar o Algoritmo T1(P) por um factor de  $n$  se explorarmos o Teorema das duas Orelhas.

### Algoritmo T2(P) – Triangulação por corte de orelhas, 1975

**Entrada:** um polígono  $P = \{v_0, v_2, \dots, v_{n-1}\}$ ,  $n > 3$

**Saída:** uma triangulação de  $P$

1. WHILE  $n > 3$  DO
  2.   Encontre  $v[i]$  tal que  $E := \Delta(v[i-1], v[i], v[i+1])$  é uma orelha
  3.   Traçar a diagonal  $v[i]v[j]$
  4.    $P \leftarrow P - E$  /\* corta orelha \*/
- **Análise:** Encontrar, na linha 2 do algoritmo, um vértice  $v[i]$  (ponta de uma orelha) é  $O(n)$  (precisamos fazer no máximo  $n$  testes de diagonalidade). Pelo teorema das duas orelhas de Meister sabemos que qualquer polígono possui pelo menos duas orelhas. Logo, cada iteração da linha 2 gasta tempo  $O(n^2)$ . Se o polígono estiver armazenado numa lista circular duplamente ligada, então a linha 3 é  $O(1)$ . Por tanto a complexidade de tempo deste algoritmo é  $O(n^3)$

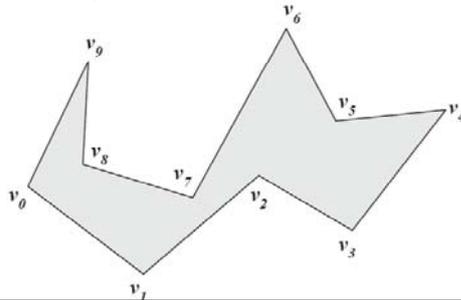
Utilizando estruturas de dados adequadas este algoritmo pode executar em  $O(n^2)$

20

## Triangulação de Polígonos: Exemplo de “corte de orelhas”



- Pela definição de orelha temos que:
  - **Orelha** – triângulo (T) formado por três vértices consecutivos de P tal que  $T \subset P$ .
  - Seja  $T(v_{i-1}, v_i, v_{i+1})$  uma orelha de P, então  $v_i$  é chamado **extremidade de uma orelha**
- Denotemos por:
  - C – conjunto dos **vértices convexos** de P
  - R – conjunto dos **vértices côncavos** de P
  - E – conjunto das **extremidades das orelhas** de P



$$C = \{v_0, v_1, v_3, v_4, v_6, v_9\}$$

$$R = \{v_2, v_5, v_7, v_8\}$$

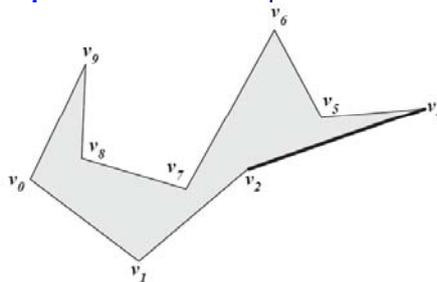
$$E = \{v_3, v_4, v_6, v_9\}$$

21

## Triangulação de Polígonos: Exemplo de “corte de orelhas”

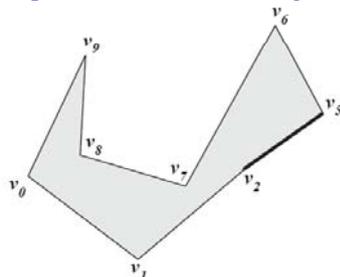


- **1º passo:** Removemos a primeira orelha:  $T(v_2, v_3, v_4)$



- Actualizamos C, R e E
  - $C = \{v_0, v_1, v_4, v_6, v_9\}$
  - $R = \{v_2, v_5, v_7, v_8\}$
  - $E = \{v_4, v_6, v_9\}$

- **2º passo:** Removemos a seguinte orelha:  $T(v_2, v_4, v_5)$



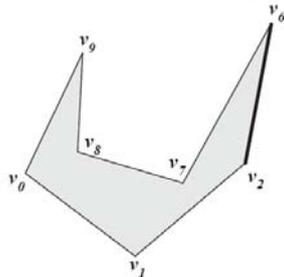
- Actualizamos C, R e E
  - $C = \{v_0, v_1, v_5, v_6, v_9\}$
  - $R = \{v_2, v_7, v_8\}$
  - $E = \{v_5, v_6, v_9\}$

22

## Triangulação de Polígonos: Exemplo de “corte de orelhas”

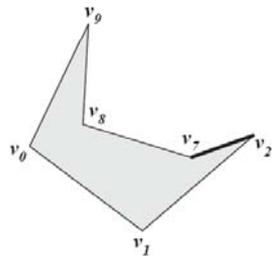


- 3º passo: Removemos a seguinte orelha:  $T(v_2, v_5, v_7)$



- Actualizamos C, R e E
  - $C = \{v_0, v_1, v_2, v_6, v_9\}$
  - $R = \{v_2, v_7, v_8\}$
  - $E = \{v_4, v_6, v_9\}$

- 4º passo: Removemos a seguinte orelha:  $T(v_2, v_6, v_7)$



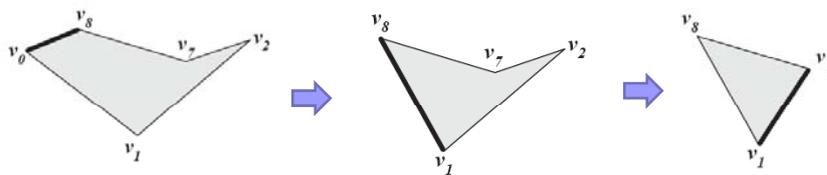
- Actualizamos C, R e E
  - $C = \{v_0, v_1, v_2, v_9\}$
  - $R = \{v_7, v_8\}$
  - $E = \{v_9, v_2\}$

23

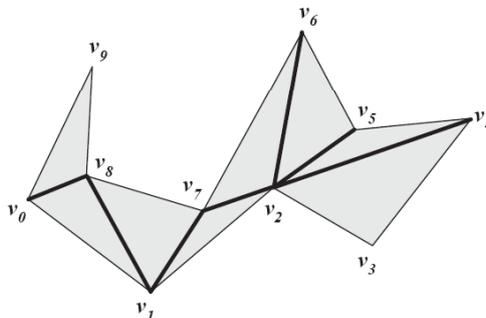
## Triangulação de Polígonos: Exemplo de “corte de orelhas”



- Próximos passos ...



- Resultado final:

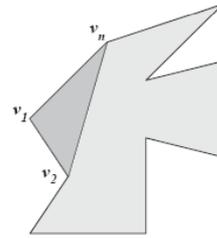


24

## Triangulação de Polígonos: Algoritmo de Lennes, 1911



- Utiliza a técnica de **inserção recursiva de diagonais**
- **Ideias gerais:**
  - Procurar um vértice  $v_i$  que defina com os seus adjacentes ( $v_{i-1}$  e  $v_{i+1}$ ) um ângulo interno convexo. **Então:**
    - Se o  $T(v_{i-1}, v_i, v_{i+1})$  não contém outros vértices de  $P$  no seu interior  $\Rightarrow$  o segmento  $v_{i-1}v_{i+1}$  define uma diagonal
    - Se existe pelo menos um vértice de  $P$  no interior de  $T(v_{i-1}, v_i, v_{i+1})$   $\Rightarrow$  o segmento que liga  $v_i$  ao vértice mais próximo dele (no interior de  $T(v_{i-1}, v_i, v_{i+1})$ ) define uma diagonal.
  - Após esta verificação o polígono fica dividido em dois: um triângulo e um polígono de  $(n-2)$  lados ou dois polígonos de lados maiores ou iguais a 3  $\Rightarrow$  **podemos aplicar o algoritmo recursivamente até obter somente triângulos**



25

## Triangulação de Polígonos: Algoritmo de Lennes, 1911



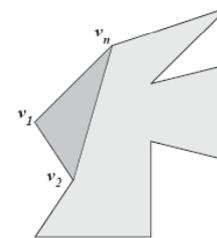
### Algoritmo T3(P)

#### Triangulação por inserção recursiva de diagonais (Lennes, 1911)

**Entrada:** um polígono  $P = \{v_1, v_2, \dots, v_n\}$ ,  $n > 3$

**Saída:** uma triangulação de  $P$

1. Se  $P$  for triângulo, **Stop**
  2. Senão, determinar um vértice (suponha sem perda da generalidade que seja  $v_1$ ) tal  $\angle v_n v_1 v_2$  seja convexo. Então
  3. Se o conjunto  $V$  dos vértices de  $P$  que estão dentro do  $T(v_n, v_1, v_2)$  for vazio então:
    4. Faça  $V_1 = \{v_1, v_2, v_n\}$  e  $V_2 = \{v_2, v_3, \dots, v_n\}$
  5. Senão, determinar o vértice  $v_k$  ( $\in V$ ) mais próximo de  $v_1$
  6. Faça  $V_1 = \{v_1, v_2, \dots, v_k\}$  e  $V_2 = \{v_1, v_k, v_{k+1}, \dots, v_n\}$
  7. Aplique recursivamente o algoritmo a  $V_1$  e  $V_2$
- **Análise:** Cada execução do passo 3 é  $O(n)$  é gera uma das  $(n-3)$  diagonais da triangulação. Logo o algoritmo é  $O(n^2)$

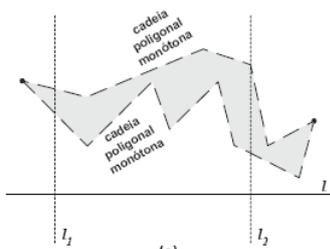


26

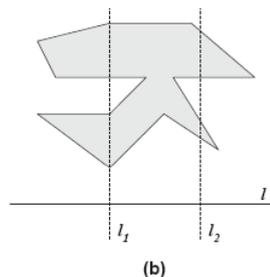
## Triangulação de Polígonos Monótonos



- Um **polígono** diz-se **monótono** em relação à recta  $l$  se, para toda a recta  $l'$  perpendicular a  $l$ , a intersecção do polígono com  $l'$  é conexa, isto é, a intersecção é um segmento de recta, um ponto ou o conjunto vazio.
  - Ou ainda, um **polígono** diz-se **monótono** em relação à recta  $l$  se a sua fronteira for composta por duas curvas poligonais (ou cadeias) monótonas em relação à recta  $l$ .



Polígono **monótono** (em relação a  $l$ )



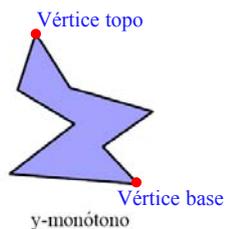
Polígono **não monótono** (em relação a  $l$ )

27

## Triangulação de Polígonos Monótonos



- Um **polígono monótono** em relação ao eixo dos  $xx$ 's e em relação ao eixo dos  $yy$ 's diz-se **x-monótono** e **y-monótono**, respectivamente.
- Na prática a monotonia é usada apenas em relação aos eixos coordenados.
- A fronteira de um **polígono y-monótono** é composta por duas cadeias monótonas: a cadeia direita e a cadeia esquerda.
  - Cada cadeia tem como extremos o vértice mais baixo (**base**) e o vértice mais alto (**topo**) do polígono, e contém zero ou mais vértices entre estes extremos.

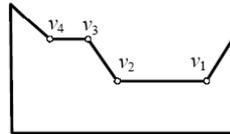
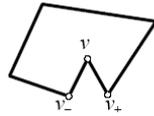


28

## Triangulação de Polígonos Monótonos



- Uma ponta interior de um polígono é um vértice reflexo  $v$  cujos vértices adjacentes  $v_-$  e  $v_+$  estão acima ou abaixo de  $v$  (pelo menos um deles tem de estar estritamente acima ou abaixo).



- **Lema:** Um polígono é y-monótono se não possui pontas interiores.
- **Teorema:** Um polígono com  $n$  vértices pode ser particionado em polígonos y-monótonos em tempo  $O(n \log n)$  por um algoritmo que usa espaço  $O(n)$

**Nota:** o algoritmo pode ser encontrado, por exemplo, em:

<http://www.ime.usp.br/~coelho/mac747/material.html>

29

## Triangulação de Polígonos Monótonos



- **Importante!!!**
  - Um facto fundamental do ponto de vista de algoritmos envolvendo polígonos y-monótonos é que os seus vértices podem ser ordenados em relação às suas y-coordenadas em tempo linear.

### Justificação:

- Para se obter uma ordenação  $u_1, \dots, u_n$  dos vértices de  $P$  em relação às suas y-coordenadas, podemos proceder da seguinte forma:
  - Encontrar um vértice mais alto e um vértice mais baixo de  $P$ . [Pode-se executar este passo em tempo  $O(n)$ , simplesmente percorrendo a lista de vértices de  $P$ ]
  - Particionar  $fr(P)$  em duas cadeias y-monótonas (a cadeia esquerda e a cadeia direita). [Este passo gasta tempo  $O(1)$ ]
  - Intercalar (*merge*) as duas cadeias para formar uma lista dos vértices ordenados em relação às suas y-coordenadas. [O conhecido algoritmo de intercalação\* gasta tempo  $O(n)$ ]

\* ver <http://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html>

30

## Triangulação de Polígonos Monótonos



- Seja  $P$  um polígono **y-monótono** com  $n$  vértices. Assumimos que  $P$  é estritamente y-monótono.
  - Iremos sempre para baixo quando percorremos a cadeia poligonal esquerda ou direita de  $P$ , desde o vértice de maior  $y$  até ao de menor  $y$ .
- Propriedade importante dos polígonos monótonos que permitem simplificar a triangulação:
  - *podemos trabalhar em  $P$  desde cima até baixo em ambas as cadeias, adicionando diagonais sempre que seja possível.*
- O algoritmo de triangulação de polígonos monótonos utiliza uma estrutura de dados em forma de **pilha**
- Quando processamos um vértice adicionamos o maior número possível de deste vértice até os vértices que se encontram na pilha. Estas diagonais dividem  $P$  em triângulos

31

## Triangulação de Polígonos Monótonos



### Algoritmo T4(P) - Triangulação de polígonos monótonos

**Entrada:** um polígono  $P = \{v_1, v_2, \dots, v_n\}$ ,  $n > 3$ , y-monótono

**Saída:** uma triangulação de  $P$

1. Ordenar os vértices  $v[1], \dots, v[n]$  por ordem decrescente de y-coordenadas.  
Sejam  $u[1], \dots, u[n]$  os vértices nessa ordenação.  
[Este passo gasta tempo  $O(n)$ , como vimos anteriormente]
2. Iniciar com a cadeia reflexa  $S$  com os dois vértices mais altos, ou seja,  $S$  (pilha)  $\leftarrow (u[1]$  (base),  $u[2]$  (topo))
3. FOR  $i = 3, \dots, n$  DO  
Sejam  $s[1], \dots, s[t]$  os vértices em  $S$  da base até ao topo da pilha ( $s[1]$  é o vértice mais alto da cadeia reflexa e  $s[t]$  é o mais baixo).  
Caso (a):  $u[i]$  é adjacente a  $s_t$ , mas não é adjacente a  $s[1]$ .  
Enquanto  $t > 1$  e o ângulo  $\angle u[i]s[t]s[t-1] < \pi$ , traçar a diagonal  $u[i]s[t-1]$ , desempilhar  $s[t]$  e diminuir  $t$  de uma unidade. O algoritmo usa o facto de que  $\angle u[i]s[t]s[t-1] < \pi$  somente se:
  - (1)  $u[i]$  está à direita do segmento  $s[t-1]s[t]$  se  $u[i]$  pertence à cadeia direita;
  - (2)  $u[i]$  está à esquerda do segmento  $s[t-1]s[t]$  se  $u[i]$  pertence à cadeia esquerda.

32

## Triangulação de Polígonos Monótonos

### Algoritmo T4(P) - Triangulação de polígonos monótonos

#### Exemplo:

Caso (b):  $u[i]$  é adjacente a  $s_1$ , mas não é adjacente a  $s[t]$ .  
Traçar as diagonais  $u[i]s[2]$ ,  $u[i]s[3]$ ,  $\dots$ ,  $u[i]s[t]$ . Desempilhe todos os vértices em  $S$  e empilhe  $s[t]$  seguido de  $u[i]$ . (Note-se que o polígono determinado pelos vértices  $u[i]$ ,  $s[1], \dots$ ,  $s[t]$  possui uma única triangulação).

Caso c):  $u_i$  é adjacente a  $s[1]$  e a  $s[t]$ .

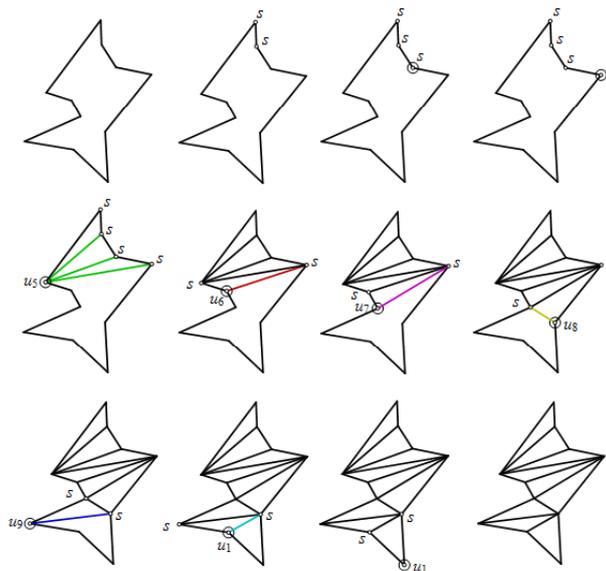
Neste caso  $u[i] = u[n]$  e o polígono  $P_{i-1}$  determinado pelos vértices  $u[i]$ ,  $s[1], \dots, s[t]$  possui uma única triangulação. Traçar as diagonais  $u[i]s[2]$ ,  $u[i]s[3], \dots, u[i]s[t-1]$  e parar.

33

## Triangulação de Polígonos Monótonos

### Algoritmo T4(P) - Triangulação de polígonos monótonos

#### Exemplo:



34

## Triangulação de Polígonos Monótonos

### **Algoritmo T4(P) - Triangulação de polígonos monótonos**



#### ■ Análise da complexidade do algoritmo:

- O ciclo **for** é executado  $n - 2$  vezes.
- Observação Importante:
  - Em cada execução do ciclo **for** são executadas no máximo duas operações **empilha**. Assim, o número total de operações **empilha** executadas pelo algoritmo durante toda a sua execução é limitado por  $2n - 2$  incluindo os dois **empilha** executados no passo 2.
  - Portanto, como o número de execuções da operação **desempilha**, ao longo da execução do algoritmo, não pode exceder o número de operações **empilha**, o tempo total gasto pelo ciclo **FOR** é  $O(n)$ .

- **Teorema:** *Um polígono y-monótono com  $n$  vértices pode ser triangulado em tempo linear*

35

## Triangulação de Polígonos Simples em $O(n \log n)$



### Ideias do algoritmo:

- **Decompor o polígono dado em polígonos y-monótonos**, aplicando o algoritmo para partição de polígonos em partes monótonas;
- **Triangular cada um dos polígonos y-monótonos** obtidos, aplicando o algoritmo triangular polígono y-monótonos.
- **Teorema:** *Um polígono simples com  $n$  vértices pode ser triangulado em tempo  $O(n \log n)$  por um algoritmo que usa espaço  $O(n)$*

36

## O estudo das triangulações não fica por aqui ...



Running Time	Designers	Year	Technique
$O(n^2)$	<i>Lenne</i>	1911	Recursive diagonal insertion
$O(n^3)$	<i>Meisters</i>	1975	Ear cutting
$O(n \log n)$	<i>Garey, Johnson, Preparata &amp; Tarjan</i>	1978	Decomposition into monotone pieces
$O(n \log n)$	<i>Chazelle</i>	1982	Divide & Conquer
$O(n + r \log r)$ where $r$ is the # of reflex vertices of the Polygon	<i>Hertel &amp; Mehlhorn</i>	1983	-
$O(n \log s)$ where $s$ is the sinuosity of $P$	<i>Chazelle</i>	1983	-
$O(n \log \log n)$	<i>Tarjan &amp; Van Wyk</i>	1987	Using involved data structures
$O(n(1 + \epsilon))$ where $\epsilon$ is the # of free triangles in the output triangulation.	<i>Toussaint</i>	1988	Sleeve-searching (Output sensitive)
$O(n^2)$	<i>ElGindy, Everett &amp; Toussaint</i>	1990	Finding an ear in linear time via prune & search
$O(n)$	<i>Chazelle</i>	1990	-
$O(kn)$	<i>Kong, Everett &amp; Toussaint</i>	1990	Graham Scan

37