Descomposicion en Polígonos Monótonos

comp-420

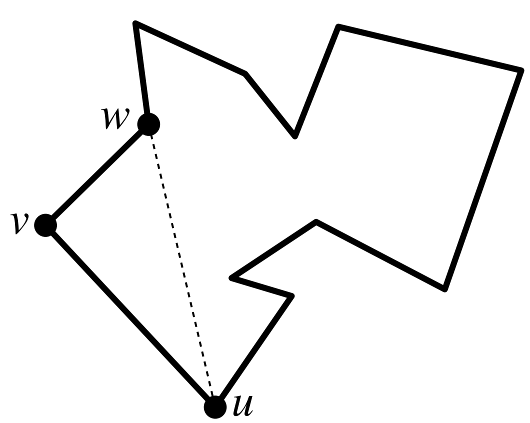
Triangulación de Polígonos

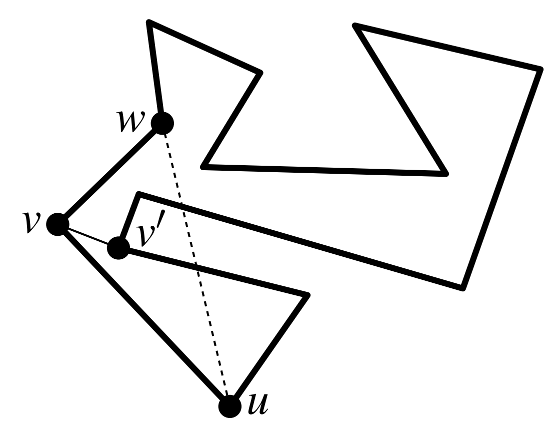
Teorema 1:

Todo polígono simple admite una triangulación, y cualquier triangulación de un polígono simple con *n* vértices consta de *n-2* triángulos exactamente.

Prueba por inducción:

- Para *n=3* el polígono es un triángulo y el teorema es trivialmente verdadero.
- Sea *n>3* y supongase el teorema cierto para toda *m<n* :
  - Empezamos por probar la existencia de una diagonal.

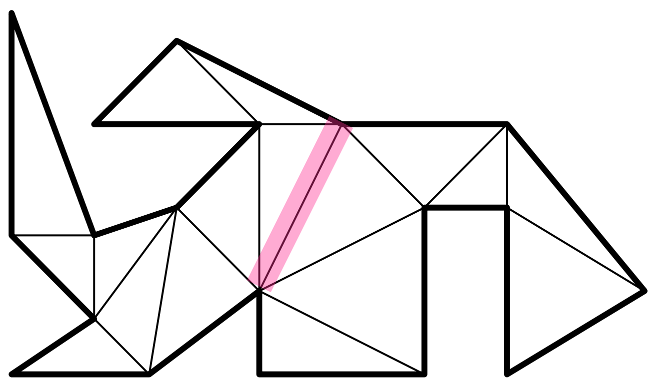




Triangulación de Polígonos

- cualquier diagonal corta $\mathcal{P}$ en dos polígonos simples $\mathcal{P}_1$ y $\mathcal{P}_2$.
- sea m_1 el número de vértices en $\mathcal{P}_1$ y m_2 el número de vértices en $\mathcal{P}_2$
- como $m_1, m2 < n$, por inducción $\mathcal{P}_1$ y $\mathcal{P}_2$ se pueden triangular, entonces $\mathcal{P}$ se puede triangular.

- Resta probar que cualquier triangulación de $\mathcal{P}$ tiene *n-2* triángulos.

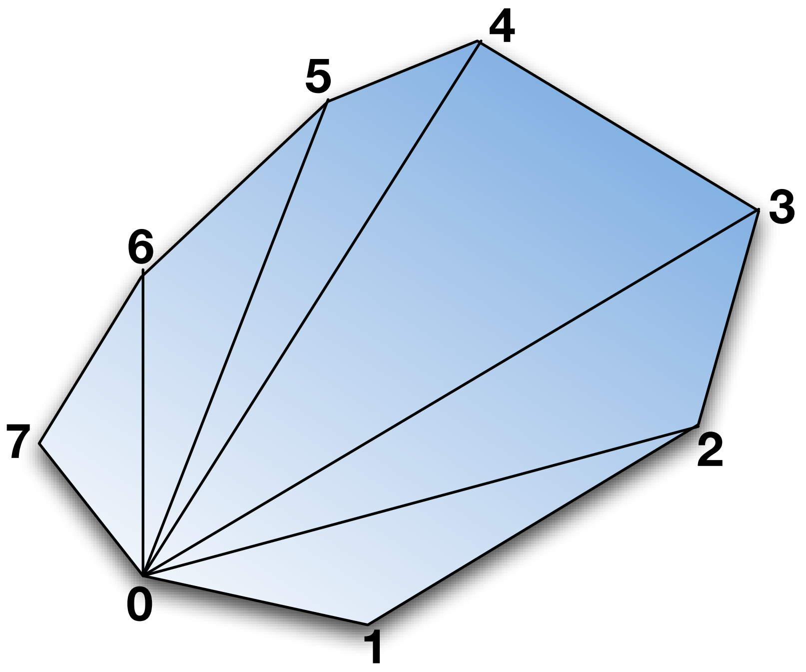


- cada triangulación de $\mathcal{P}_i$ tendrá $m_i - 2$ triángulos, lo que implica que consta de $(m_1 - 2) + (m_2 - 2) = n - 2$ triángulos.

■

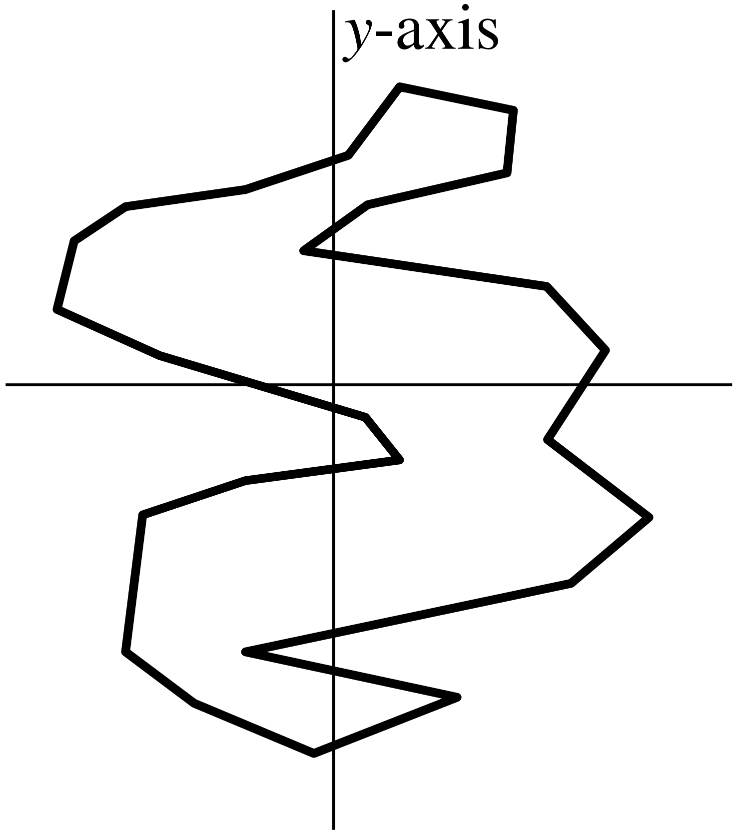
Triangulación de Polígonos

- Vimos un algoritmo recursivo de complejidad lineal para encontrar una diagonal en un polígono simple.
- Con esta estrategia la diagonal encontrada dividirá el polígono en dos, en un triángulo y en un polígono simple de $n-1$ vértices. Este algoritmo será de complejidad cuadrática en el peor caso.
- Para un polígono convexo podemos encontrar un algoritmo lineal:



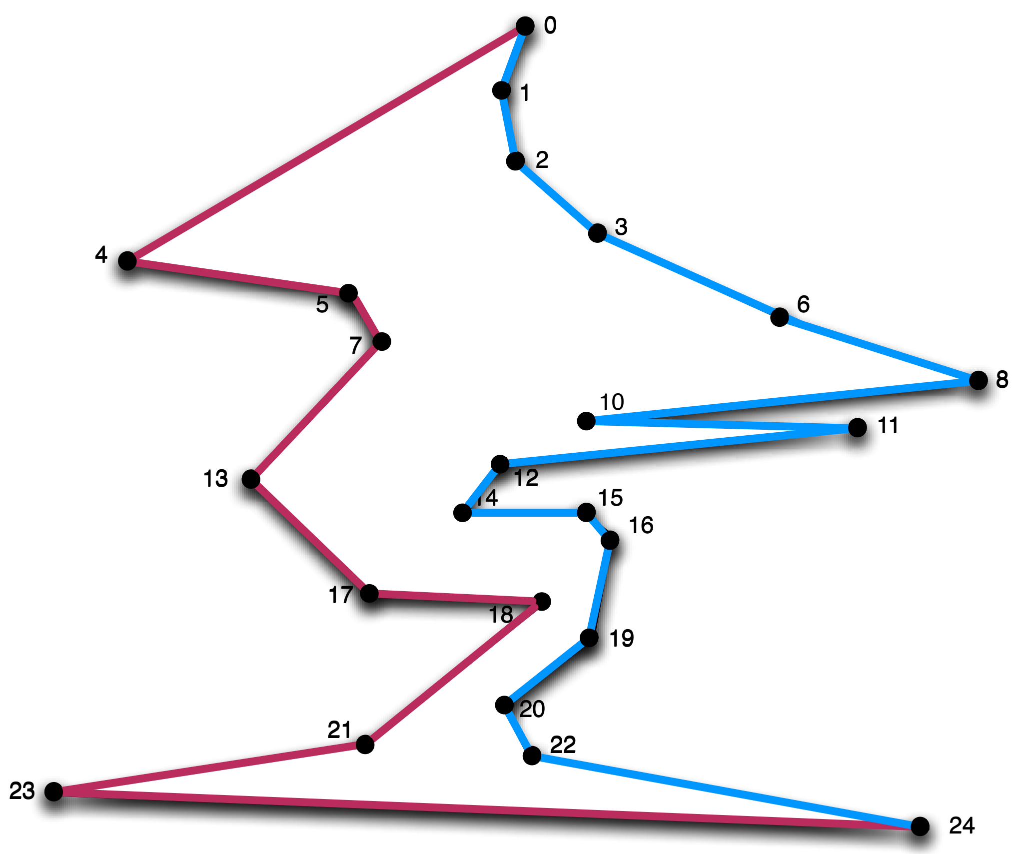
Polígonos Monótonos

- Una cadena polígonal $\mathcal{C}$ es *estrictamente monótona* respecto a una línea l si cada l' ortogonal a l intersecta a $\mathcal{C}$ en a lo más un punto:
- Esto es: $l' \cap C$ es vacío o un punto.
- Una cadena es *monótona* si $l' \cap C$ tiene a lo más un componente conectado: es vacío, un punto o un segmento de recta.
- Un polígono $\mathcal{P}$ es monótono respecto a la línea l, si $\partial\mathcal{P}$ se puede dividir en dos cadenas poligonales A y B tal que cada cadena sea monótona respecto a l. Ambas cadenas comparten un vértice en sus extremos.
- La estrategia para triangular el polígono $\mathcal{P}$ es primero dividir $\mathcal{P}$ en polígonos monótonos respecto a y y luego triangularlos.



Partición de un polígono en partes monótonas

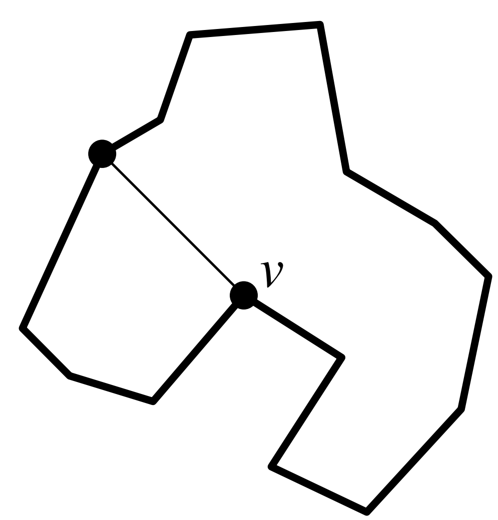
- Encontrar un vértice de giro (turn vertex) a partir del vértice más alto.



{0,4,5,7,13,17,18,21,23,24}

{0,1,2,3,6,8,10,11,12,14,15,18,19,20,22,24}

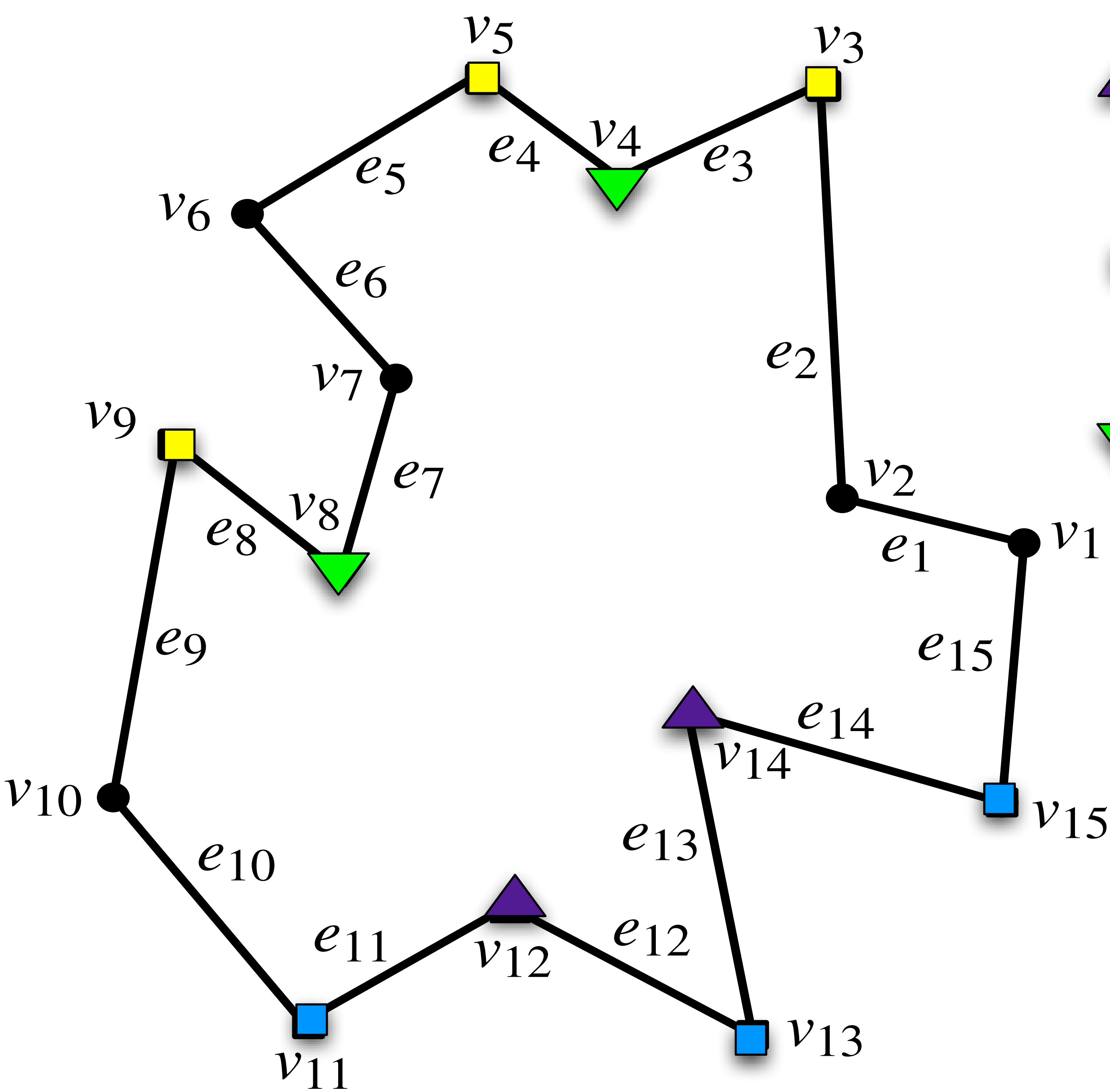
- Eliminar los vértices de giro agregando diagonales.
  - si las dos aristas adyacentes al vértice de giro bajan y el interior del polígono está arriba del vértice: agregar una diagonal hacia arriba.
  - la diagonal dividirá el polígono en dos.



Partición de un polígono en partes monótonas

- Para definir los diferentes tipos de vértices de giro hay que establecer un orden.
  - Un punto p está abajo de otro punto q si $p_y < q_y$ o $p_y = q_y$ y $p_x > q_x$.
  - Un punto p está arriba de otro punto q si $p_y > q_y$ o $p_y = q_y$ y $p_x < q_x$.
- Distinguimos 5 tipos de vértices, donde 4 son vértices de giro:
  - de giro: inicio(*start*), fin (*end*), división (*split*), unión (*merge*);
  - regulares.

Tipos de vértices en un polígono



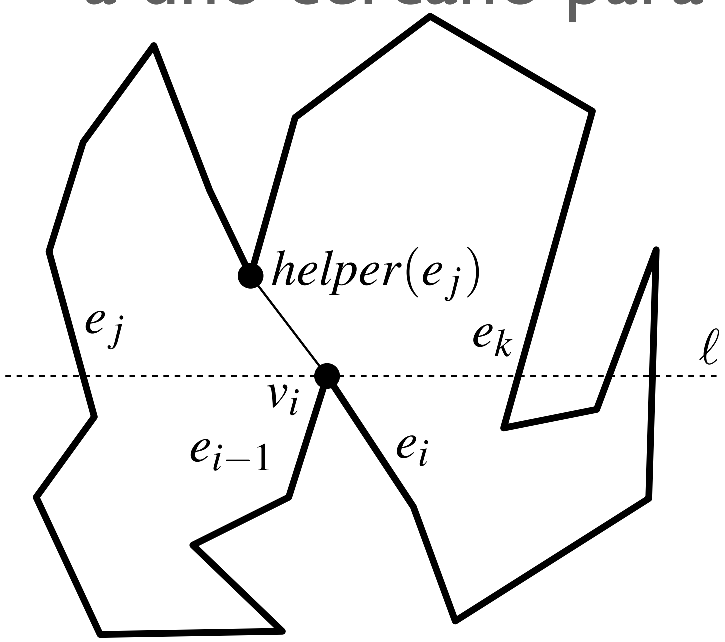
- *start* - sus vecinos están ambos abajo y el ángulo interior de *v* es inferior a π.
- *split* - sus vecinos están ambos abajo y el ángulo interior de *v* es superior a π.
- *end* - sus vecinos están ambos arriba y el ángulo interior de *v* es inferior a π.
- *merge* - sus vecinos están ambos arriba y el ángulo interior de *v* es superior a π.

Partición de un polígono en partes monótonas

- Un polígono es **monótono respecto al eje** y si no tiene vértices de división (*split*) ni de unión (*merge*).
- El polígono se dividirá en partes monótonas **insertando una diagonal hacia arriba** por cada vértice *split* y una **hacia abajo** en cada vértice *merge*.
- Sea $v_1, v_2, \ldots, v_n$ una enumeración en sentido contrario a las manecillas del reloj (ccw) de los vértices de $\mathcal{P}$.
- Sea $e_1, e_2, \ldots, e_n$ el conjunto de aristas de $\mathcal{P}$ donde $e_i = \overline{v_i v_{i+1}}$ para $1 \leq i < n$ y $e_n = \overline{v_n v_1}$.
- Una **línea de barrido** (*sweep line*) se moverá hacia abajo en el plano deteniendose en **puntos evento** (vértices de $\mathcal{P}$), no se crearán nuevos puntos evento durante el recorrido.

Partición de un polígono en partes monótonas

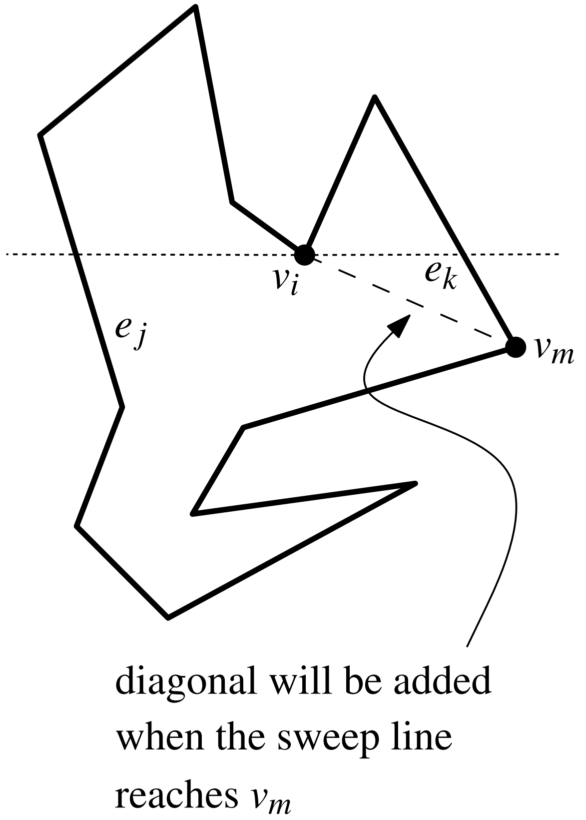
- Los puntos evento se almacenan en la cola de eventos $\mathcal{Q}$.
- Esta estructura será una cola de prioridad en su coordenada y. Si dos vértices tienen la misma coordenada y se tomará el que está a la izquierda como prioritario.
- La meta del barrido es agregar diagonales del vértice *split* a un vértice que se encuentre arriba de él. ¿A qué vertice nos conviene conectarle?
  - a uno cercano para evitar intersecciones con $\mathcal{P}$.



  - $\text{helper}(e_j)$ se define como el vértice más bajo sobre la línea de barrido tal que el segmento horizontal conectandolo con e_j está en el interior de $\mathcal{P}$.

Partición de un polígono en partes monótonas

- Una diagonal hacia abajo para eliminar vértices *merge* parece una tarea difícil, ¿por qué?
  - porque no se ha explorado el plano abajo de la línea de barrido.
  - cuando la línea llega al vértice v_i este se vuelve el nuevo $\text{helper}(e_j)$.
  - conectaremos v_i al primer vértice que aparezca sobre la línea entre e_j y e_k .



Partición de un polígono en partes monótonas

- Necesitamos encontrar las aristas a la izquierda de cada vértice por lo que almacenamos las aristas de $\mathcal{P}$ que intersecten a la línea de barrido en las hojas del árbol binario de búsqueda $\mathcal{T}$.
- Con cada arista en $\mathcal{T}$ almacenamos a su ayudante.
- El árbol $\mathcal{T}$ y sus ayudantes almacenados con las aristas forma el estado de la línea de barrido.
- El algoritmo divide a $\mathcal{P}$ en subpolígonos que deberán ser tratados en siguientes etapas. Para tener acceso a estos subpolígonos almacenaremos la subdivisión y las nuevas diagonales producidas en una lista doblemente ligada de aristas.
- $\mathcal{P}$ debe ser representado de la misma manera al inicio del algoritmo.

Partición de un polígono en partes monótonas

Algorithm MAKEMONOTONE($\mathcal{P}$)
Input. A simple polygon $\mathcal{P}$ stored in a doubly-connected edge list $\mathcal{D}$.
Output. A partitioning of $\mathcal{P}$ into monotone subpolygons, stored in $\mathcal{D}$.

1. Construct a priority queue $\mathcal{Q}$ on the vertices of $\mathcal{P}$, using their y-coordinates as priority. If two points have the same y-coordinate, the one with smaller x-coordinate has higher priority.
2. Initialize an empty binary search tree $\mathcal{T}$.
3. **while** $\mathcal{Q}$ is not empty
4.   **do** Remove the vertex v_i with the highest priority from $\mathcal{Q}$.
5.   Call the appropriate procedure to handle the vertex, depending on its type.

Partición de un polígono en partes monótonas

HANDLESTARTVERTEX(v_i)
1. Insert e_i in $\mathcal{T}$ and set $helper(e_i)$ to v_i.

HANDLEENDVERTEX(v_i)
1. **if** $helper(e_{i-1})$ is a merge vertex
2. **then** Insert the diagonal connecting v_i to $helper(e_{i-1})$ in $\mathcal{D}$.
3. Delete e_{i-1} from $\mathcal{T}$.

Partición de un polígono en partes monótonas

HANDLESPLITVERTEX(v_i)

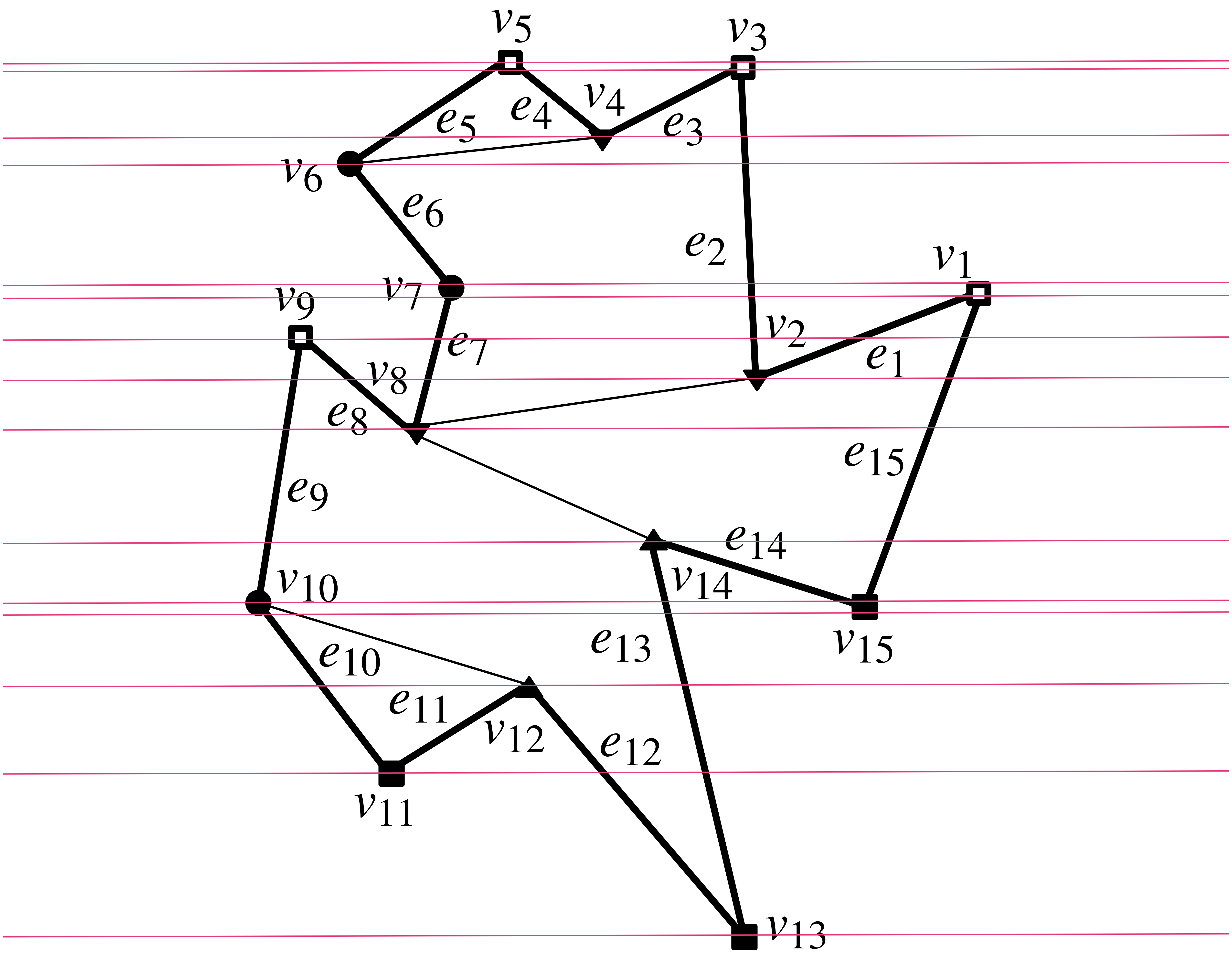
Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
Insert the diagonal connecting v_i to $helper(e_j)$ in $\mathcal{D}$.
$helper(e_j) \leftarrow v_i$
Insert e_i in $\mathcal{T}$ and set $helper(e_i)$ to v_i.

HANDLEMERGEVERTEX(v_i)

if $helper(e_{i-1})$ is a merge vertex
    then Insert the diagonal connecting v_i to $helper(e_{i-1})$ in $\mathcal{D}$.
Delete e_{i-1} from $\mathcal{T}$.
Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
if $helper(e_j)$ is a merge vertex
    then Insert the diagonal connecting v_i to $helper(e_j)$ in $\mathcal{D}$.
$helper(e_j) \leftarrow v_i$

Partición de un polígono en partes monótonas

HANDLEREGULARVERTEX(v_i)
1. **if** the interior of $\mathcal{P}$ lies to the right of v_i
2. **then if** $helper(e_{i-1})$ is a merge vertex
3. **then** Insert the diagonal connecting v_i to $helper(e_{i-1})$ in $\mathcal{D}$.
4. Delete e_{i-1} from $\mathcal{T}$.
5. Insert e_i in $\mathcal{T}$ and set $helper(e_i)$ to v_i.
6. **else** Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
7. **if** $helper(e_j)$ is a merge vertex
8. **then** Insert the diagonal connecting v_i to $helper(e_j)$ in $\mathcal{D}$.
9. $helper(e_j) \leftarrow v_i$



v5
v3
e4
v4
e5
e3
v6
e6
e2
v1
v7
v9
v2
e7
e1
v8
e8
e15
e9
e14
v14
v10
v15
e13
e10
e11
v12
e12
v11
v13

Partición de un polígono en partes monótonas: análisis

- ¿Tiempo de ejecución de MakeMonotone(P)?
  - Construir la cola de prioridad Q de eventos (cada vértice): $O(n)$
  - Inicializar el estado T de la línea de barrido $O(1)$
  - Para manejar un evento de Q usamos a lo más:
    - una operación en Q: $O(1)$
    - una búsqueda, una inserción y una eliminación de T
    - inserción de a lo más dos diagonales en D: $O(1)$
  - Las colas de prioridad y los árboles de búsqueda balanceados permiten búsquedas y actualizaciones en tiempo: $O(\log n)$
  - El manejo de eventos toma $O(\log n)$ y el algoritmo completo: $O(n \log n)$

Partición de un polígono en partes monótonas: análisis

- El tamaño de la memoria necesaria el lineal
  - cada vértice se almacena a lo más una vez en Q.
  - cada arista se almacena a lo más una vez en T.

Teorema 4:

Un polígono simple $\mathcal{P}$ con n vértices se puede dividir en polígonos monótonos respecto a y en tiempo $O(n \log n)$ y usando $O(n)$ memoria.