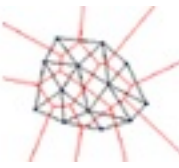


Triangulación de polígonos

Algoritmos iniciales



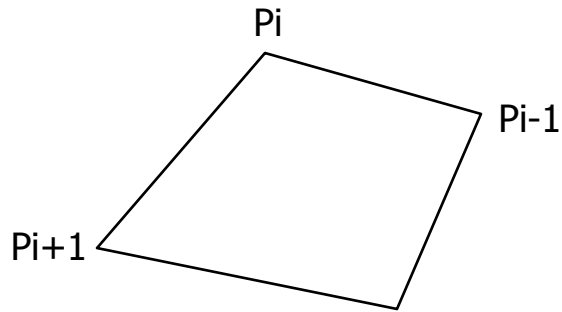
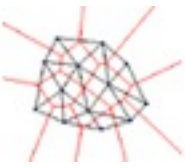
- Historia
- Conceptos previos
- Triangulación por recortado de orejas
- Triangulación de polígonos monótonos

Historia

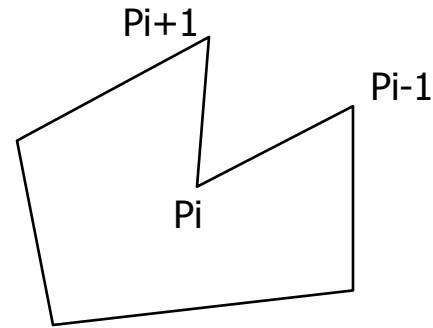


Complejidad	Autor	Año	Técnica
$O(n^3)$	Meisters	1975	Recortado de orejas
$O(n \log n)$	Preparata & Tarjan	1978	Descomposición
$O(n \log \log n)$	Tarjan	1987	Estructuras de datos complejas
$O(n)$	Chazelle	1990	
$O(n^2)$	ElGindy & Toussaint	1990	Búsqueda de oreja en $O(n)$
$O(kn)$	Kong & Toussaint	1990	Scan de Graham

Vértice convexo y cóncavo



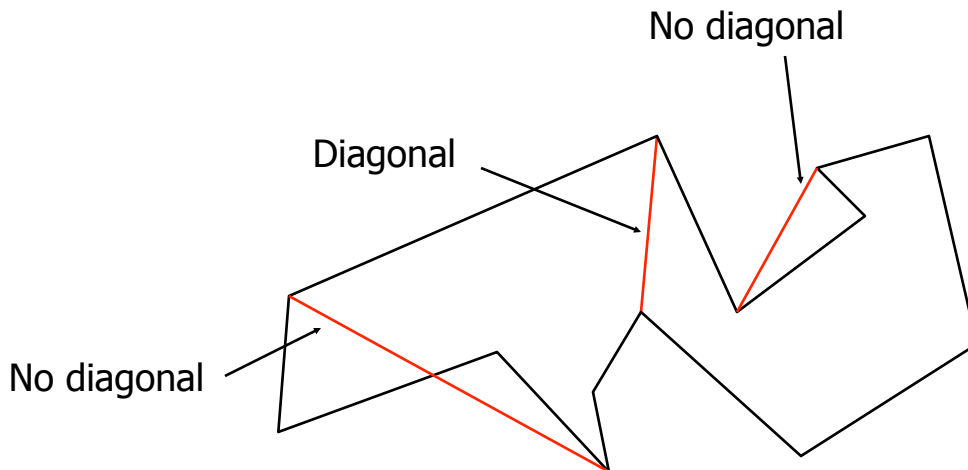
Vértice Convexo



Vértice Cóncavo

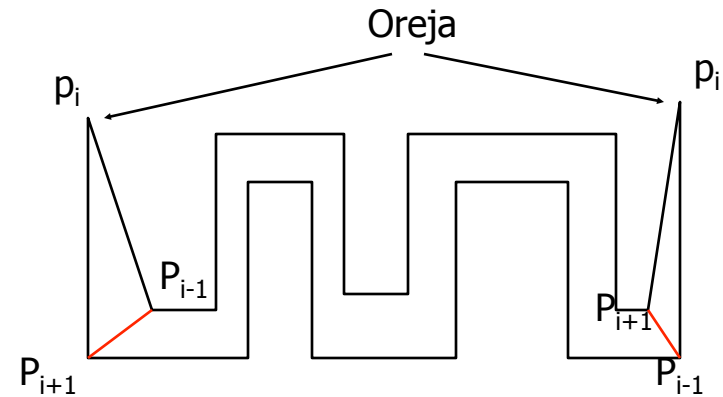
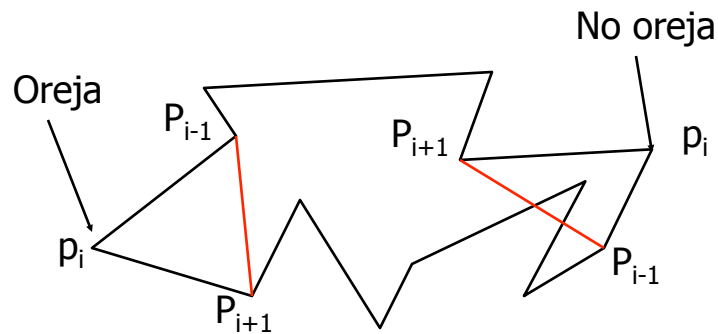
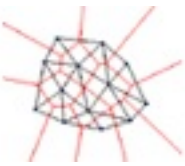
Si recorremos los vértices del polígono dejando el interior a nuestra izquierda (polígono antihorario), un vértice P_i será convexo si el punto P_{i+1} está a la izquierda del segmento orientado (P_{i-1}, P_i)

Diagonal



- Diagonal: segmento que está totalmente incluido en el polígono y que une dos vértices no consecutivos
- ¿Cómo comprobar que un segmento es diagonal?
¿Qué coste tendría el algoritmo?

Vértice oreja



Un vértice P_i se denomina oreja si el segmento (P_{i-1}, P_{i+1}) es una diagonal del polígono

Algoritmo de recortado de orejas



Triangular(P)

1. Mientras que P tenga más de 3 lados
2. Buscar vértice oreja P_i
3. Añadir arista (P_{i-1}, P_{i+1}) a la triangulación
4. Eliminar P_i de P // Cortar oreja

Coste total: depende de la búsqueda de vértices orejas (línea 2)

Búsqueda de oreja en $O(n^2)$

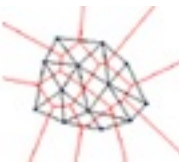


BuscarOreja(P)

1. Desde $i=1$ hasta n
2. Si (P_{i-1}, P_{i+1}) es una diagonal devolver P_i

- Coste de comprobar si un segmento es diagonal = $O(n)$
- Coste $O(n) * O(n) = O(n^2)$

Búsqueda de oreja en $O(kn)$

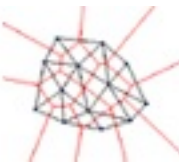


BuscarOreja(P)

1. $i := 0$
2. Mientras que P_i no es una oreja
3. Si P_i es convexo
4. Desde $j=0$ hasta k recorreremos todos los vértices cóncavos
5. Si no hay ningún P_j interior al triángulo (P_{i-1}, P_i, P_{i+1})
6. P_i es una oreja
7. Si P_i no es una oreja
8. $i := i+1$

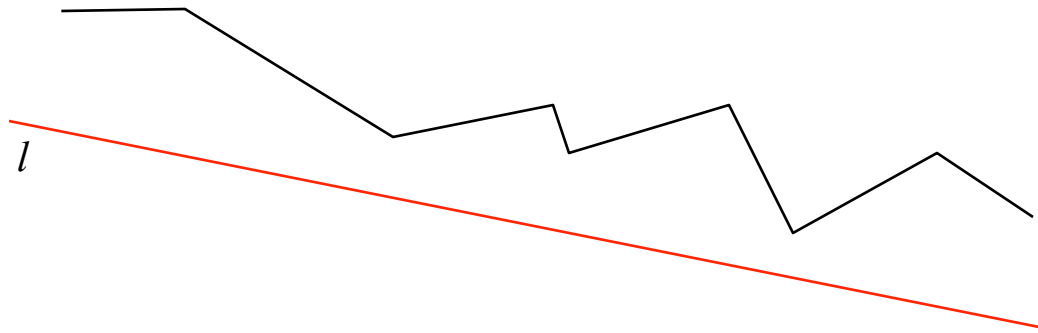
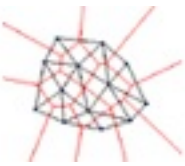
k = número de vértices cóncavos, calculados en preproceso

Polígonos monótonos



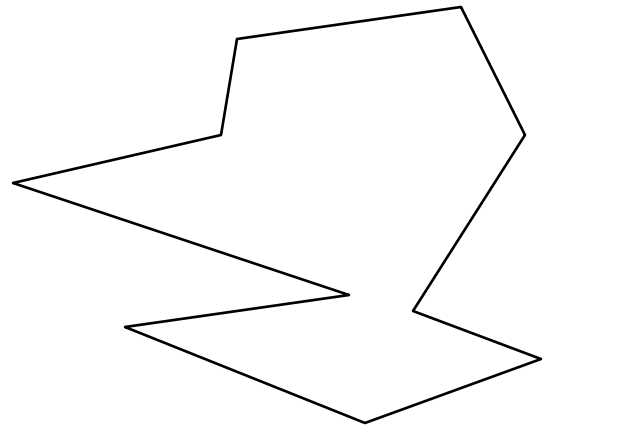
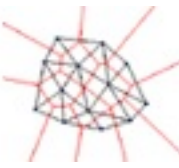
- Consideramos polígonos estrictamente monótonos en sentido vertical (no contiene aristas horizontales)
- Algoritmo basado en un barrido de vertical del plano
- Coste lineal $O(n)$

Cadena monótona



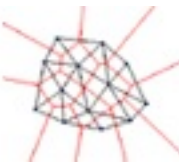
Una cadena $C = v_0v_1 \dots v_{n-1}$ es monótona con respecto a l cuando cualquier perpendicular a l corta a C únicamente en un punto

Polígono monótono



Un polígono es monótono con respecto a una línea l cuando podemos dividir el polígono en dos cadenas poligonales monótonas con respecto a l .

Algoritmo (pseudocódigo)

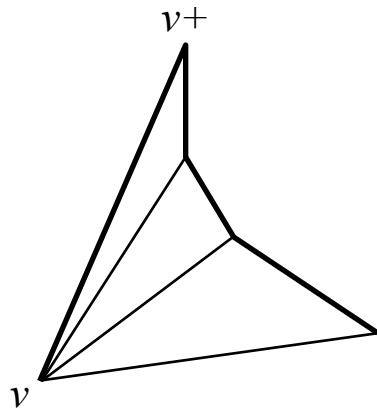
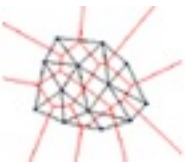


TriangularPolígono(P)

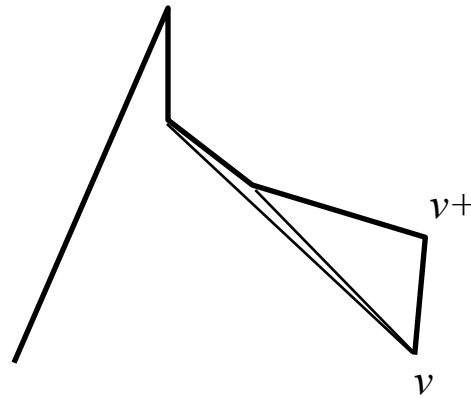
// P debe ser monótono con respecto al eje y

1. Ordenar los vértices de arriba abajo
2. ListaVérticesSuperiores = \emptyset
3. Mientras que queden vértices por tratar
4. Sea v el vértice con mayor coordenada y
5. Conectar v con todos los vértices de ListaVérticesSuperiores con los que sea posible definir una diagonal interior, y eliminar la parte del polígono triangulada. Si no fuera posible trazar ninguna diagonal, añadir v a ListaVérticesSuperiores.

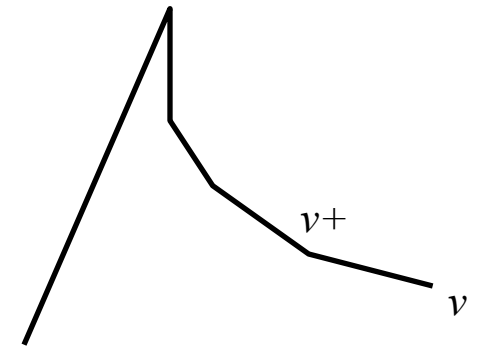
Casos a tratar en el paso 5



1



2a



2b

Algoritmo detallado



1. Ordenar los vértices de arriba abajo
2. Inicializar CadenaCóncava a los dos vértices sup. y v al siguiente
3. Mientras que $v \neq$ vértice inferior
4. Caso 1 (v está en la cadena opuesta a la CadenaCóncava)
 Trazar la diagonal de v al segundo vértice de cadena
 Eliminar el comienzo de la cadena
 Si la cadena tiene 1 elemento añadir v y avanzar v
5. Caso 2 (v es adyacente al último vértice de CadenaCóncava)
6. Caso 2a (v es convexo)
 Trazar la diagonal de v al segundo vértice de cadena
 Eliminar el final de la cadena
 Si la cadena tiene 1 elemento añadir v y avanzar v
7. Caso 2b (v es cóncavo)
 Añadir v al final de CadenaCóncava y avanzar v