

# COT 4521: INTRODUCTION TO COMPUTATIONAL GEOMETRY

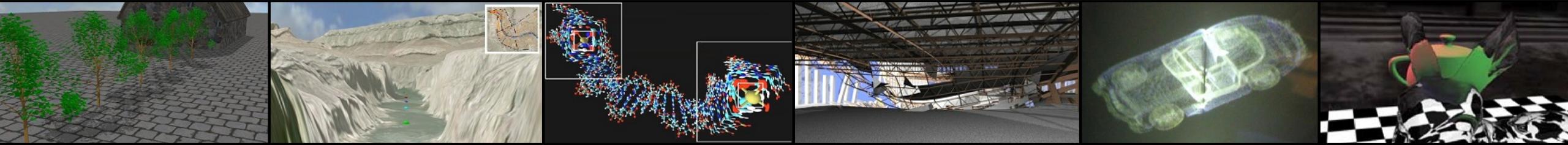
---



## Polygon Partitioning

Paul Rosen  
Assistant Professor  
University of South Florida

Some slide from Valentina Korzhova



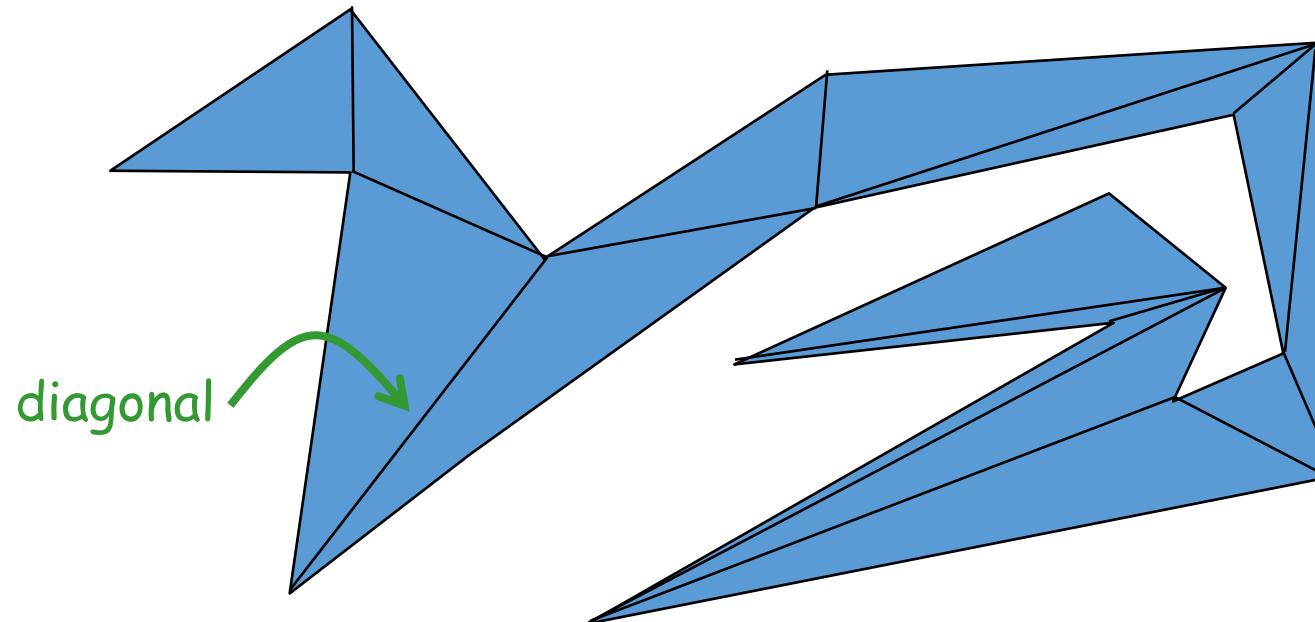
# OBJECTIVES

- TRIANGULATION ALGORITHMS
  - Diagonal-based triangulation algorithm
  - Ear-based triangulation algorithm
- COMPLEXITY OF THE ALGORITHMS
- DEFINITION OF MONOTONE POLYGON
  - Triangulation of a monotone polygon
- PARTITIONING INTO MONOTONE POLYGONS
- TRIANGULATION OF A POLYGON IN  $N \log N$  TIME



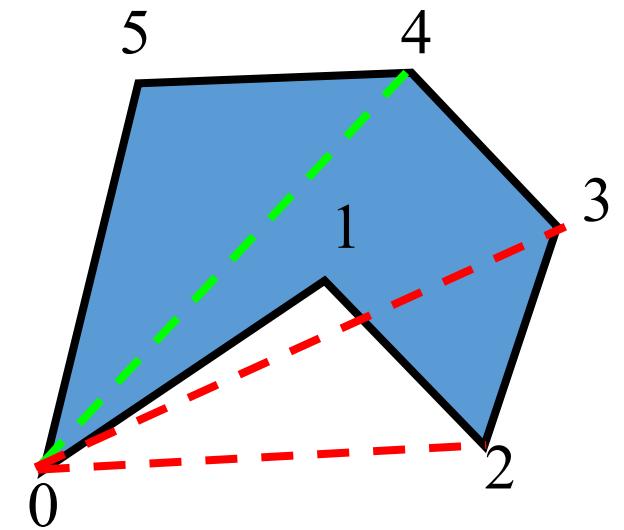
# TRIANGULATION OF POLYGONS

- DECOMPOSE THE POLYGON INTO SHAPES THAT ARE EASIER TO HANDLE: TRIANGLES
- A TRIANGULATION OF A POLYGON  $P$  IS A DECOMPOSITION OF  $P$  INTO TRIANGLES WHOSE VERTICES ARE VERTICES OF  $P$ . IN OTHER WORDS, A TRIANGULATION IS A MAXIMAL SET OF NON-CROSSING DIAGONALS.



# DIAGONAL-BASED TRIANGULATION

- DIAGONAL TEST
- THE SEGMENT  $s = v_i v_j$  IS A DIAGONAL OF  $P$  IFF
  - for all edges  $e$  of  $P$  that are not incident to either  $v_i$  and  $v_j$ ,  $s$  and  $e$  do not intersect.
  - $s$  is internal to  $P$  in the neighborhood of  $v_i$  and  $v_j$ .
- ALGORITHM: DIAGONAL TRIANGULATION  
REPEAT  $N-3$  TIMES
  - FOR EACH CANDIDATE DIAGONAL
  - TEST EACH OF NEIGHBORHOODS
  - OUTPUT PROPER DIAGONAL



# BRUTE FORCE TRIANGULATION

- **THEOREM:** EVERY POLYGON  $P$  OF  $N$  VERTICES CAN BE PARTITIONED INTO TRIANGLE BY THE ADDITION OF (ZERO OR MORE) DIAGONALS.
  - Complexity of diagonal-based algorithm:
    - $O(n^2)$  - # of diagonal candidates
    - $O(n)$  testing each of neighborhoods
    - Repeating this  $O(n^3)$  computation for each of the  $n - 3$  diagonals yields  $O(n^4)$



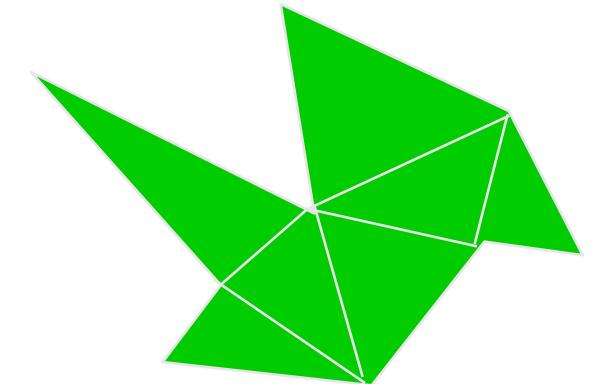
## EAR BASED IDEA....

- LOCATE AN EAR
- OUTPUT DIAGONAL
- CLIP THE EAR
- REPEAT UNTIL A TRIANGLE IS LEFT



## EAR BASED IDEA....

- **DEFINITION OF EAR:** THREE CONSECUTIVE VERTICES, A, B, C FORM AN EAR IF AC IS A DIAGONAL
- **MEISTERS' TWO EARS THEOREM:** EVERY POLYGON ( $N \geq 4$ ) HAS AT LEAST TWO NON-OVERLAPPING EARS.



## EAR BASED IDEA....

- PROOF: CONSIDER A TRIANGULATION OF AN  $n$ -POLYGON, WITH  $n > 3$ . THE TRIANGULATION CONSISTS OF  $n - 2$  TRIANGLES. SINCE THE POLYGON HAS  $n$  EDGES BUT THERE ARE ONLY  $n - 2$  TRIANGLES, BY THE PIGEONHOLE PRINCIPLE, THERE ARE AT LEAST TWO TRIANGLES WITH TWO POLYGON'S EDGES. THESE ARE THE EARS.
- ANOTHER PROOF: IT IS KNOWN THAT A SIMPLE POLYGON CAN ALWAYS BE TRIANGULATED. LEAVES IN THE DUAL-TREE OF THE TRIANGULATED POLYGON CORRESPOND TO EARS AND EVERY TREE OF TWO OR MORE NODES MUST HAVE AT LEAST TWO LEAVES.



# TRIANGULATION: IMPLEMENTATION

## **Algorithm: TRIANGULATION**

Initialize the ear tip status of each vertex.

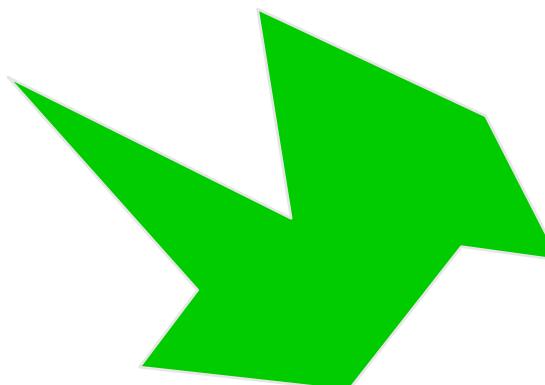
while  $n > 3$  do

    Locate an ear tip  $v_2$ .

    Output diagonal  $v_1 v_3$ .

    Delete  $v_2$ .

    Update the ear tip status of  $v_1$  and  $v_3$ .



# TRIANGULATION: IMPLEMENTATION

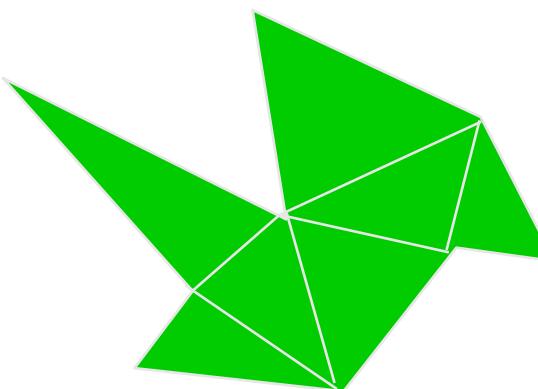
$O(n^2)$   
+  
 $O(n)$   
iterations

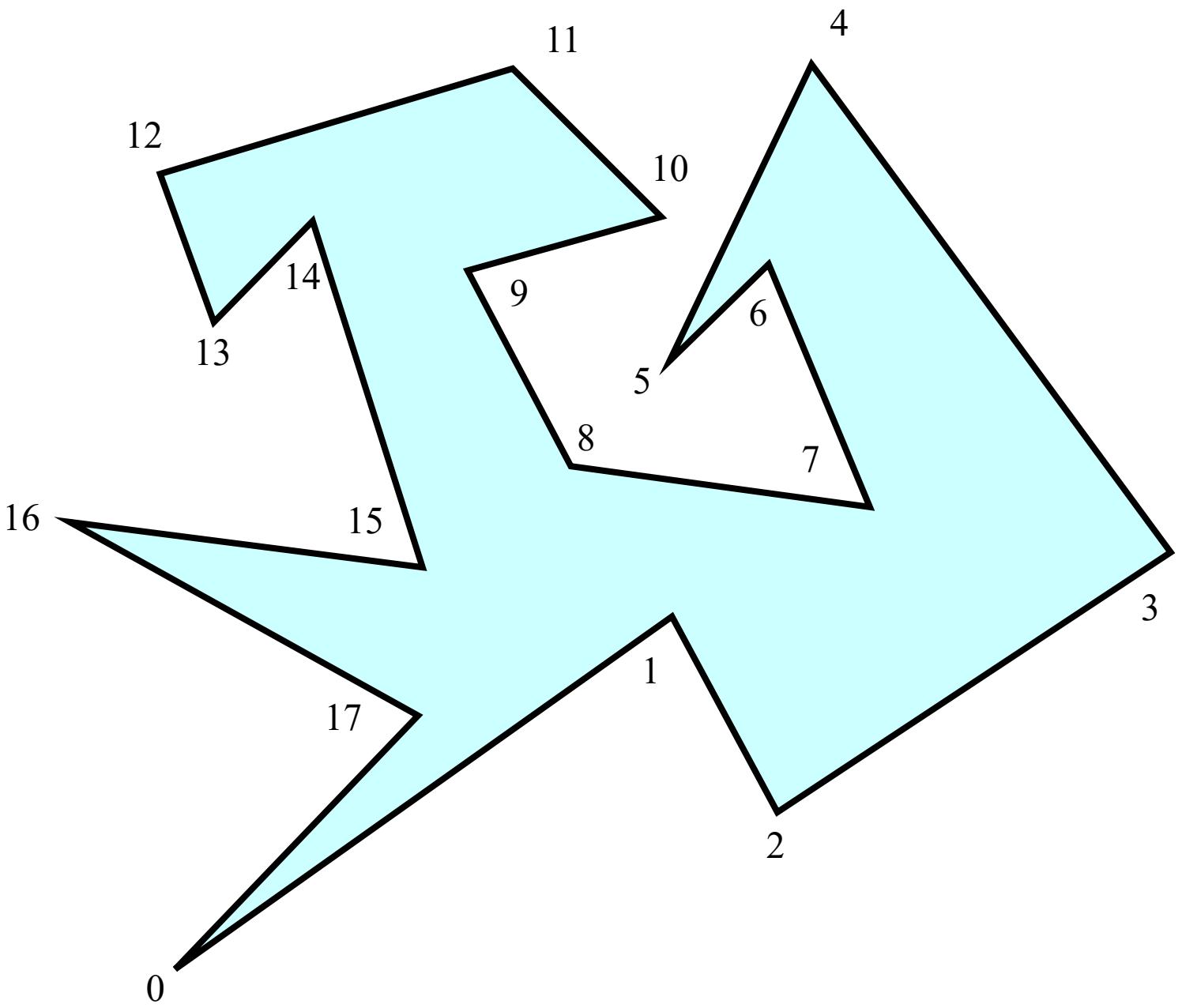
## Algorithm: TRIANGULATION

```
Initialize the ear tip status of each vertex.  
while n > 3 do  
    Locate an ear tip  $v_2$ .  
    Output diagonal  $v_1 v_3$ .  
    Delete  $v_2$ .  
 $O(n)$  —> Update the ear tip status of  $v_1$  and  $v_3$ .
```

Total:  $O(n^3)$

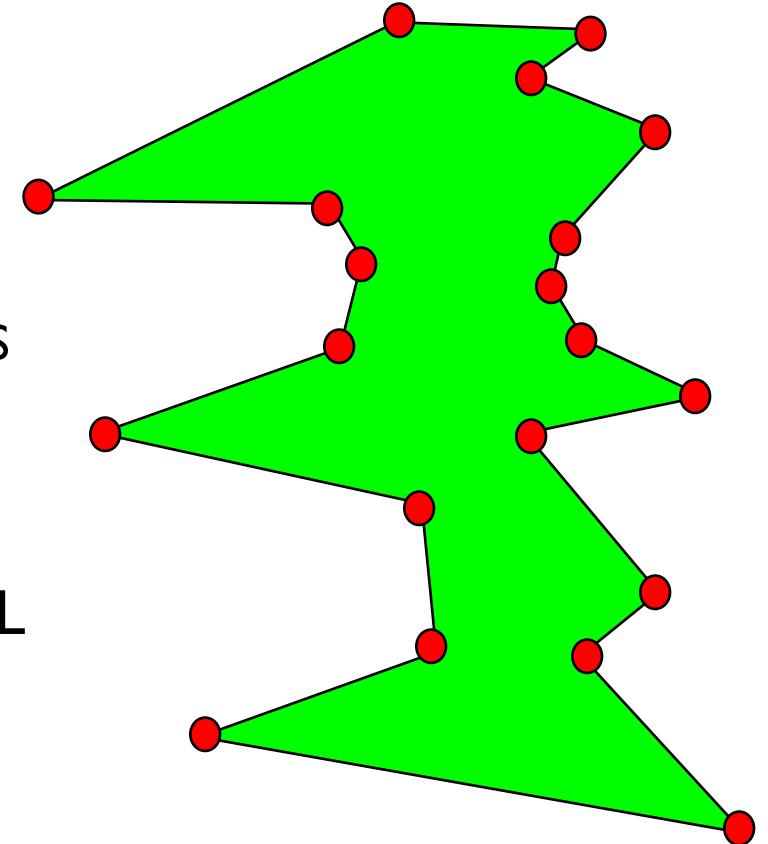
Can be made:  $O(n^2)$





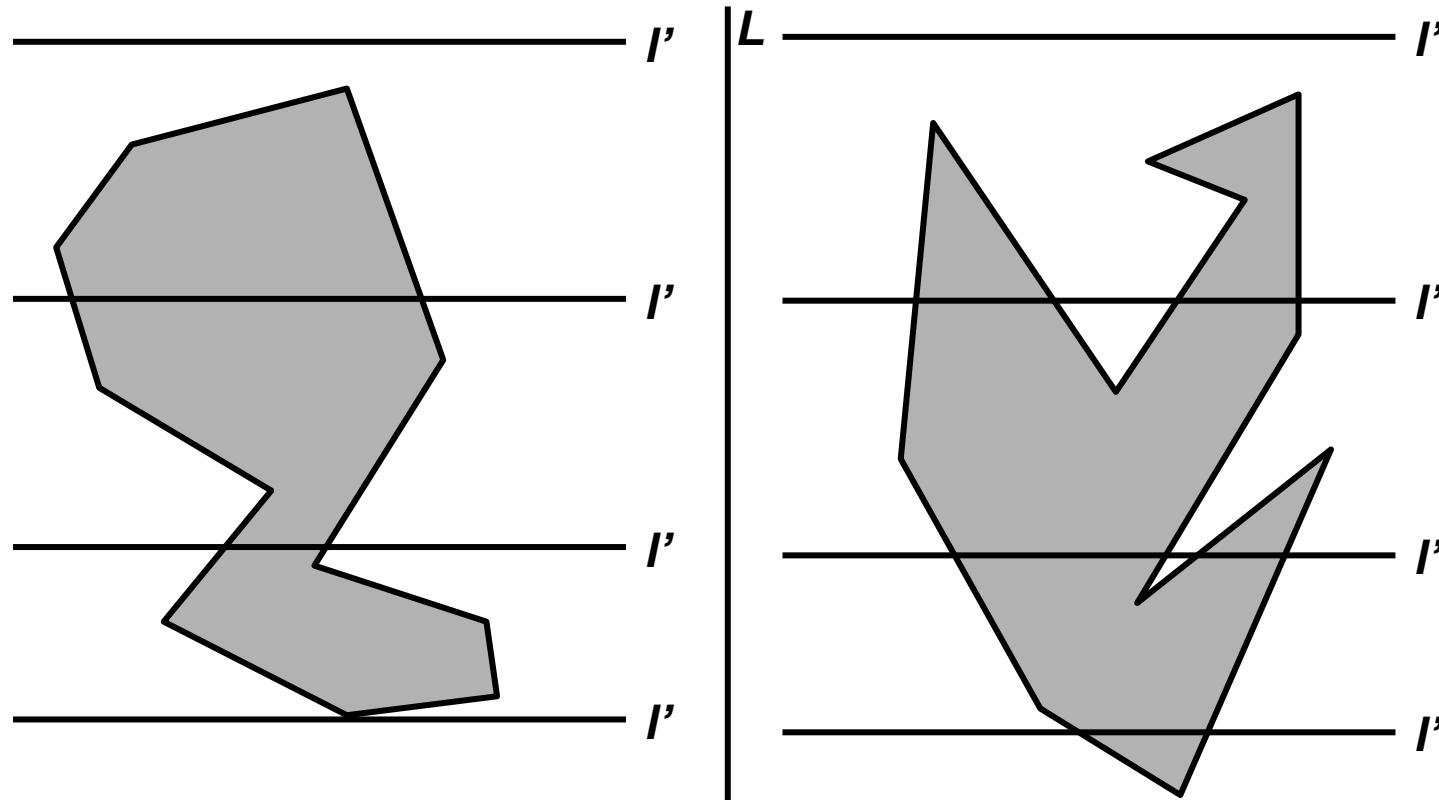
# MONOTONE PARTITIONING

- A **CHAIN** IS **MONOTONE** WITH RESPECT TO A LINE  $L$  IF EVERY LINE ORTHOGONAL TO  $L$  INTERSECTS THE CHAIN IN AT MOST 1 POINT
- **POLYGON** IS **MONOTONE** WITH RESPECT TO A LINE  $L$  IF BOUNDARY OF  $P$  CAN BE SPLIT INTO 2 POLYGONAL CHAINS  $A$  AND  $B$  SUCH THAT EACH CHAIN IS MONOTONE WITH RESPECT TO  $L$
- MONOTONICITY IMPLIES SORTED ORDER WITH RESPECT TO  $L$
- MONOTONE POLYGON CAN BE (GREEDILY) TRIANGULATED IN  $O(N)$  TIME!



# MONOTONE POLYGON

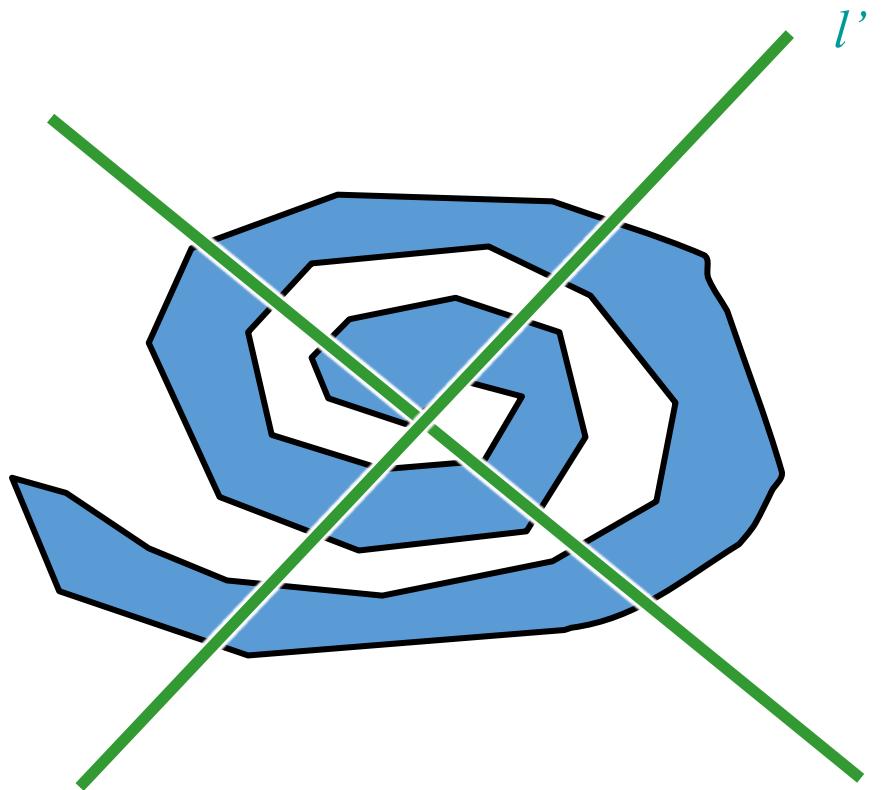
Monotone  
w.r.t. line L



not monotone  
w.r.t. L



# MONOTONE POLYGON



not monotone  
w.r.t. any line



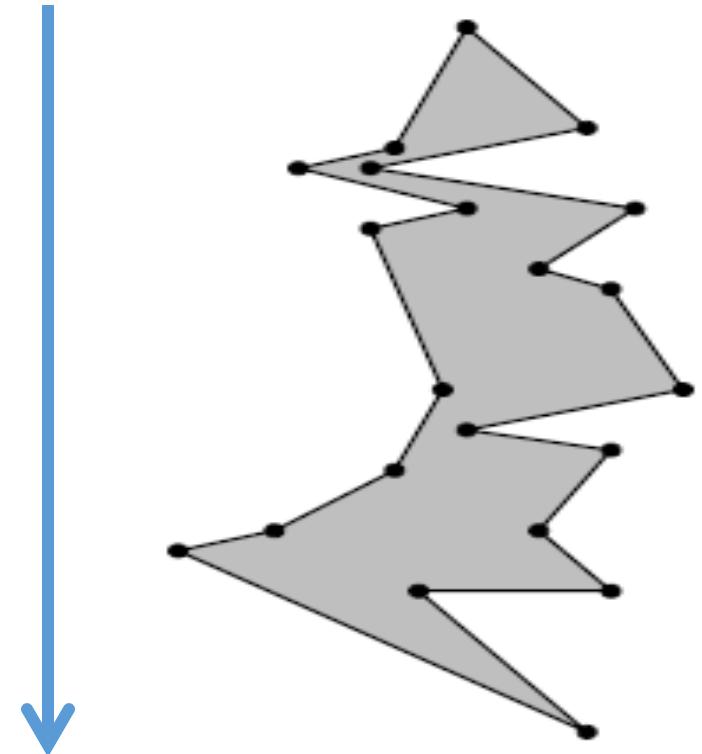
# POLYGON TRIANGULATION

- **ALGORITHM: POLYGON TRIANGULATION: MONOTONE POLYGON WITH RESPECT TO Y-LINE**
  - Sort vertices by y coordinates
  - Partition into monotone polygons
  - Triangulate each monotone polygon



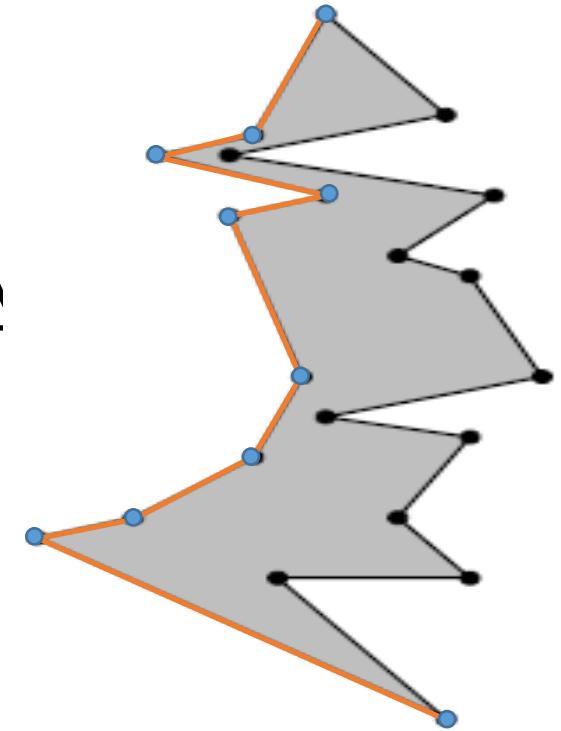
# MONOTONE POLYGON

- A Y-MONOTONE POLYGON HAS A TOP VERTEX, A BOTTOM VERTEX, AND TWO Y-MONOTONE CHAINS BETWEEN TOP AND BOTTOM AS ITS BOUNDARY



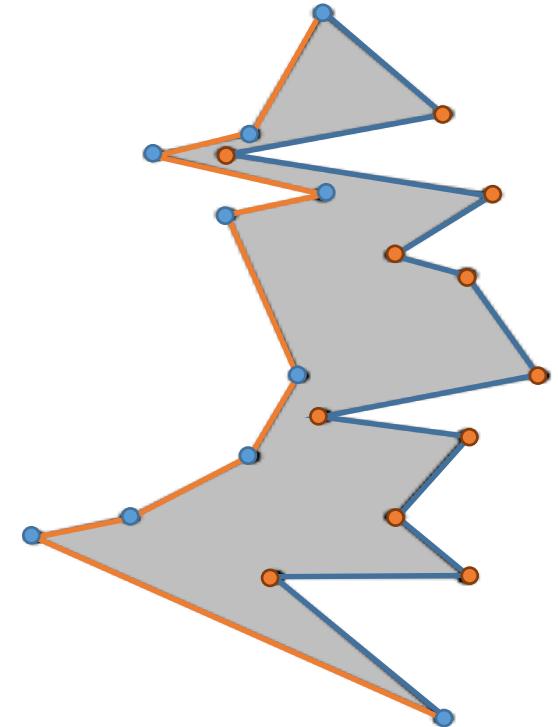
# MONOTONE POLYGON

- A POLYGONAL CHAIN  $C$  IS STRICTLY MONOTONE W.R.T.  $L'$  IF EVERY  $L$  ORTHOGONAL TO  $L'$  MEETS  $C$  IN AT MOST ONE POINT.
  - Simply monotone if  $L \cap C$  has at most one connected line segment.



# MONOTONE POLYGON

- A POLYGON P IS SAID TO BE MONOTONE W.R.T. A LINE L IF  $\partial P$  CAN BE SPLIT INTO TWO MONOTONE CHAINS W.R.T. L.



# MONOTONE POLYGON

- THE FOLLOWING PROPERTY IS CHARACTERISTIC FOR Y-MONOTONE POLYGONS:
  - If we walk from a topmost to a bottommost vertex along the left (or the right) boundary chain, then we always move downwards or horizontally, never upwards.

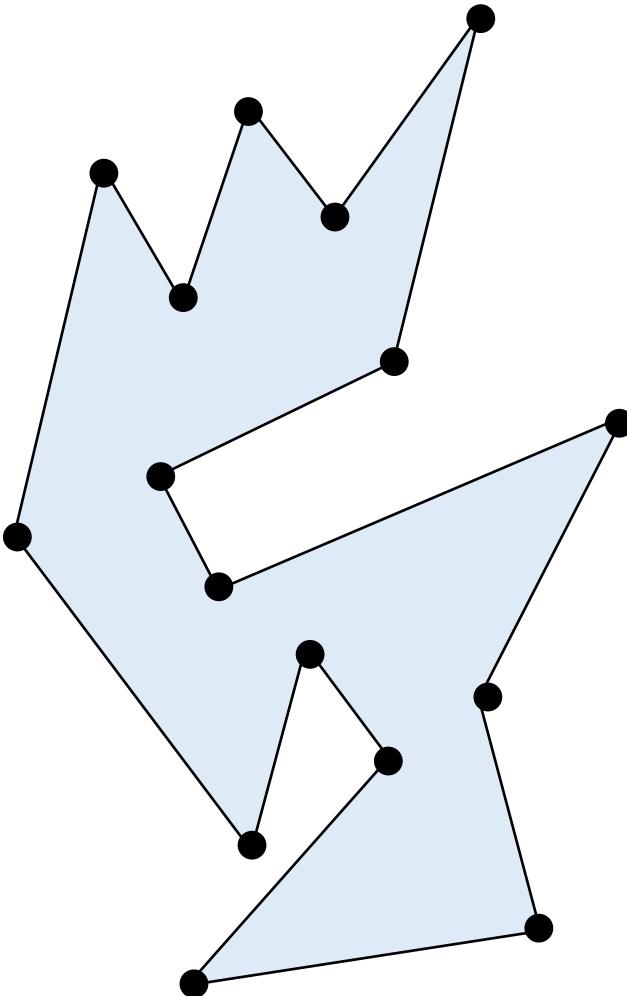


# TRIANGULATION OF MONOTONE POLYGON

- THE ALGORITHM TO TRIANGULATE A MONOTONE POLYGON DEPENDS ON ITS MONOTONICITY.
- DEVELOPED IN 1978 BY GAREY, JOHNSON, PREPARATA, AND TARJAN
- DESCRIBED IN BOTH
  - Preparata pp. 239-241 (1985)
  - Laszlo pp. 128-135 (1996)
  - The former uses y-monotone polygons, the latter uses x-monotone.

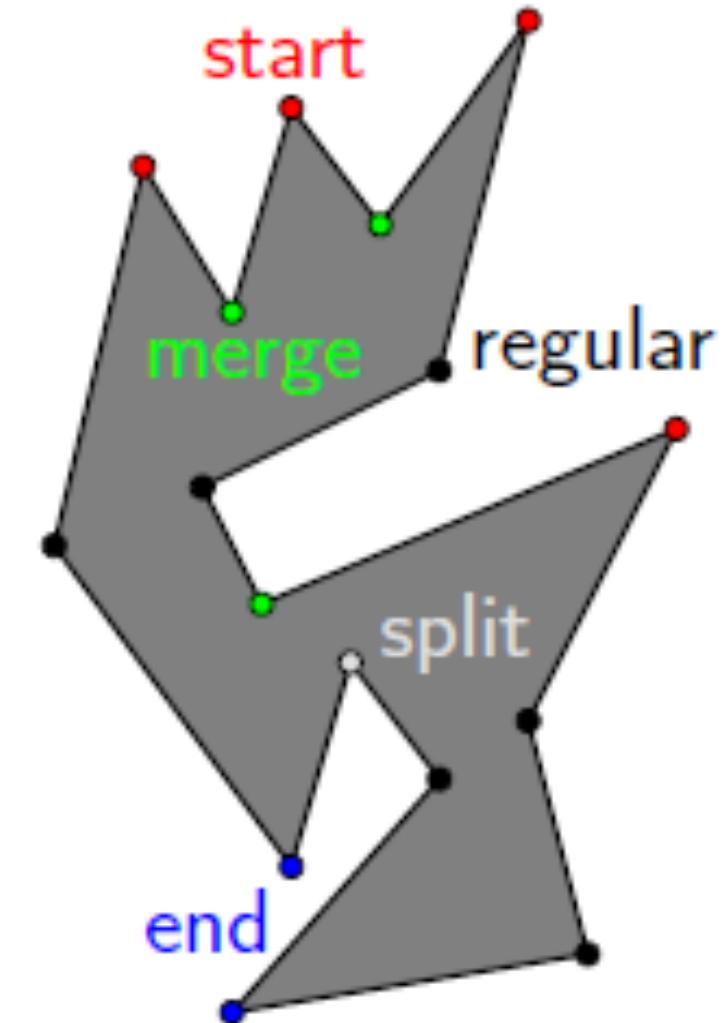


# WHAT KIND OF VERTICES DOES A NON-Y-MONOTONE POLYGON HAVE WITH RESPECT TO SWEEP OF Y?



# PROPERTIES OF MONOTONE POLYGON

- IF A POLYGON P HAS NO INTERIOR CUSPS, THEN IT IS MONOTONE
- WHAT TYPES OF VERTICES DOES A SIMPLE POLYGON HAVE?
  - start
  - end
  - split
  - merge
  - regular



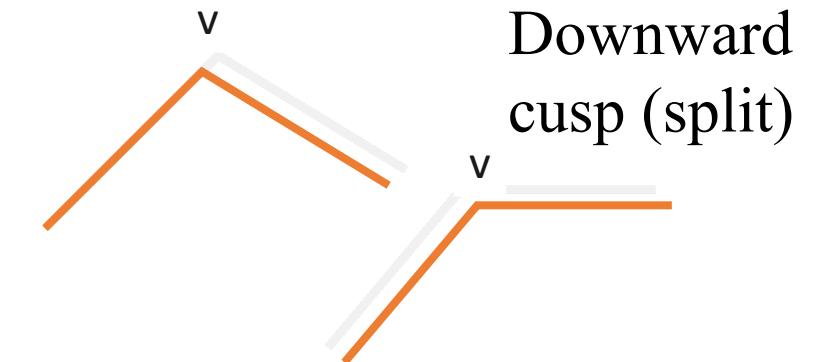
# PROPERTIES OF MONOTONE POLYGON

- REFLEX VERTEX WHOSE ADJACENT VERTICES ARE EITHER BOTH AT OR ABOVE  $v$ , OR BOTH AT OR BELOW IT.
- IF A POLYGON HAS NO INTERIOR CUSPS THEN IT IS MONOTONE WITH RESPECT TO THE VERTICAL LINE

Interior cusps



Upward  
Cusp (merge)

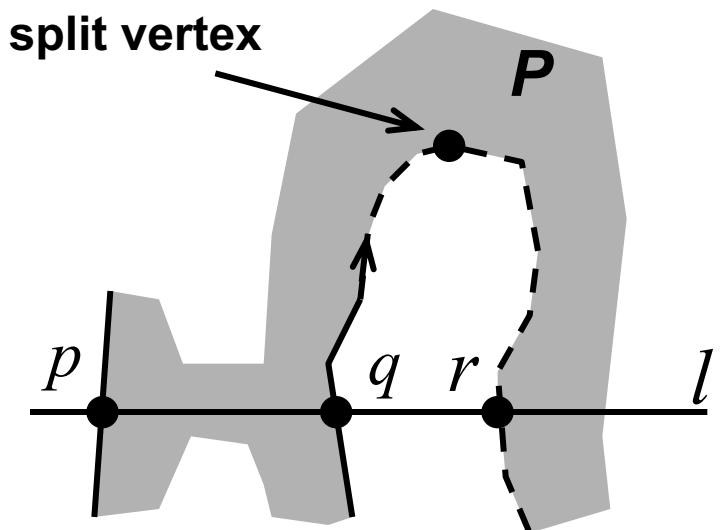


Downward  
cusp (split)



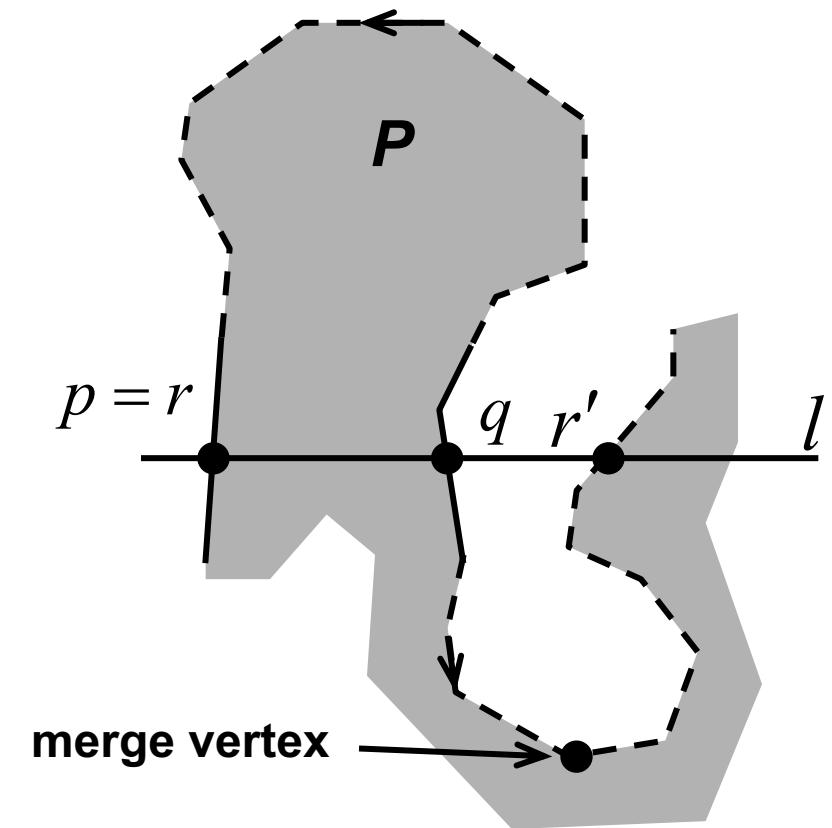
# PROPERTIES OF MONOTONE POLYGON

- **LEMMA:** A POLYGON IS Y-MONOTONE IF IT HAS NEITHER SPLIT VERTICES NOR MERGE VERTICES
- **PROOF:** IF P IS NOT MONOTONE, THERE MUST EXIST A LINE L INTERSECTING P IN MORE THAN A SINGLE SEGMENT.  
LET  $[P,Q]$  BE ITS LEFTMOST SUB SEGMENT.
- FOLLOW THE BOUNDARY OF P STARTING AT Q, WHERE P IS ON THE LEFT. AT SOME POINT R WE MUST CROSS L.
- IF  $R \neq P$  THEN THE HIGHEST VERTEX MUST BE A SPLIT ONE.



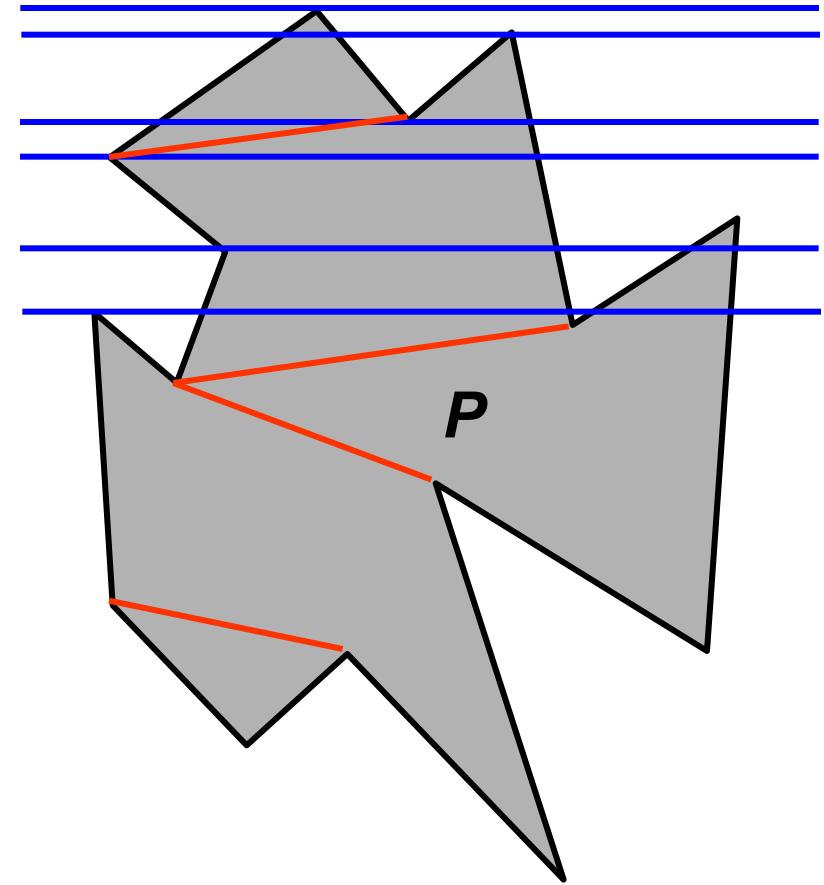
# PROPERTIES OF MONOTONE POLYGON

- IF  $R = P$  WE FOLLOW THE BOUNDARY FROM  $Q$  IN OPPOSITE DIRECTION.
- AT SOME POINT  $r'$  WE MUST CROSS  $L$ .  
 $r' \neq p$  AS OTHERWISE IT CONTRADICTS THAT  $P$  IS NOT MONOTONE.
- THIS IMPLIES THAT THE LOWEST ENCOUNTERED VERTEX MUST BE A MERGE ONE.



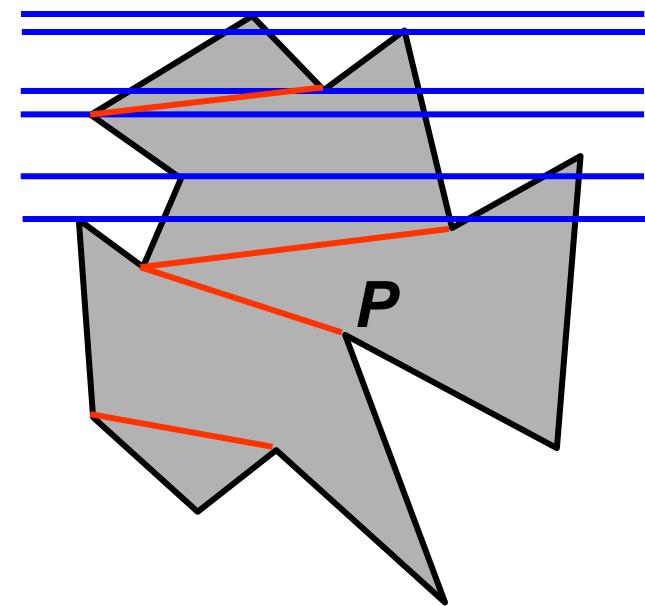
# GETTING RID OF SPLIT AND MERGE VERTICES

- SORT P'S VERTICES FROM TOP TO BOTTOM. IT TAKES  $O(n \log n)$  TIME.
- APPLY SCAN LINE FROM TOP TO BOTTOM TO ENCOUNTER VERTICES.
- DIAGONALS ARE INTRODUCED AT SPLIT AND MERGE VERTICES.



# POLYGON TRIANGULATION

- ALGORITHM: POLYGON TRIANGULATION: MONOTONE PARTITION
  - Sort vertices by y coordinates
    - Perform plane sweep to construct trapezoids
  - Partition into monotone polygons
  - Triangulate each monotone polygon



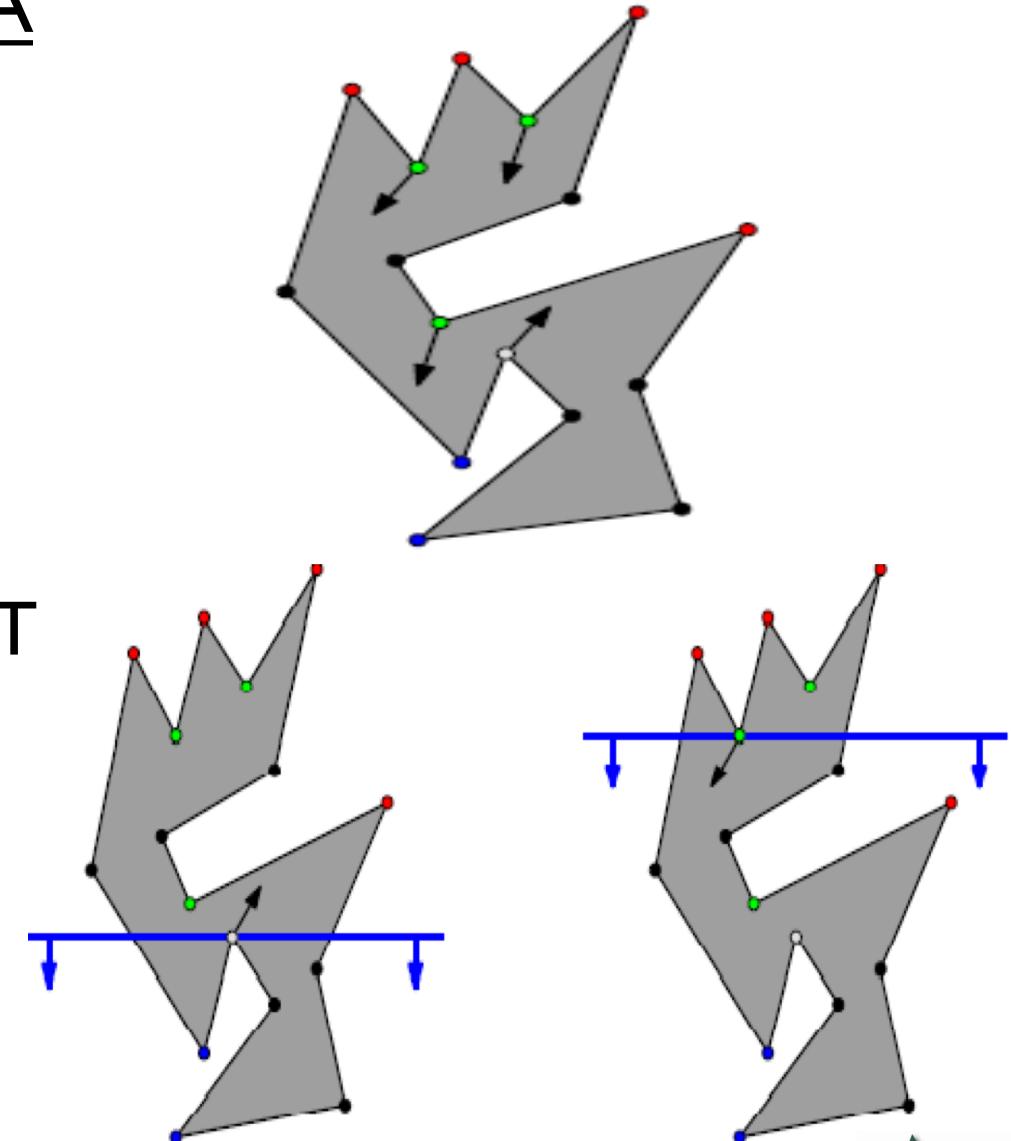
# TRAPEZOIDALIZATION

- “DRAW” HORIZONTAL LINE THROUGH EACH VERTEX
  - Consider only the connected segment inside the polygon containing the vertex
    - Two supporting vertices – top and bottom
  - IF AN “INTERIOR” SUPPORTING VERTEX IS AN INTERIOR CUSP, BREAK IT
    - Connect downward for a upward cusp
    - Connect upward for an downward cusp
    - These connections partitions the polygon into monotone parts.



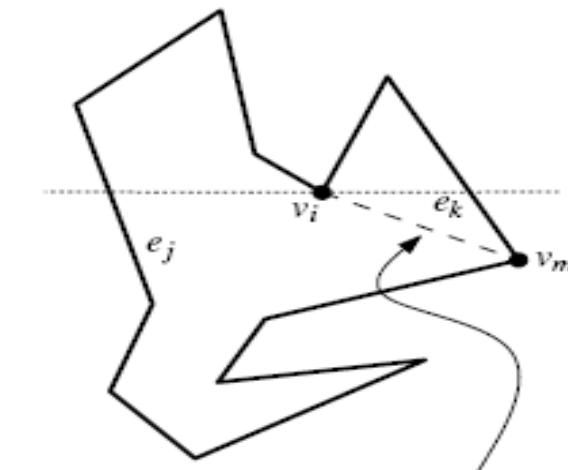
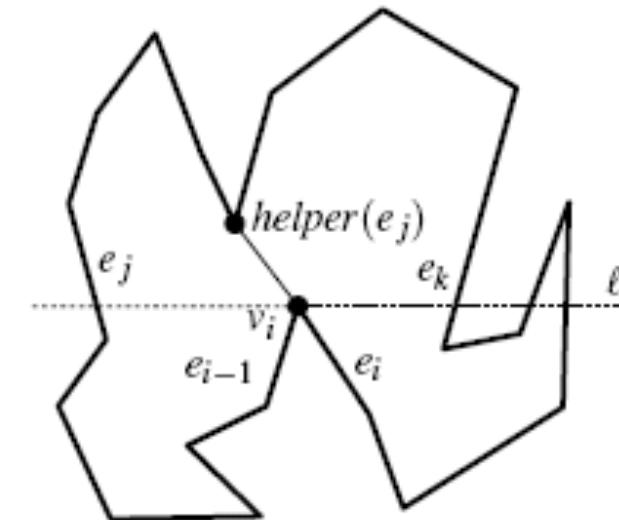
# SWEET IDEA

- FIND DIAGONALS FROM EACH MERGE VERTEX DOWN, AND FROM EACH SPLIT VERTEX UP
- A SIMPLE POLYGON WITH NO SPLIT OR MERGE VERTICES CAN HAVE AT MOST ONE START AND ONE END VERTEX, SO IT IS Y-MONOTONE



# SWEET IDEA

- FOR VERTEX OF INTEREST  $v$ , FIND THE CLOSEST VERTEX (IN THE Y DIRECTION) THAT IS BETWEEN THE EDGES TO THE LEFT AND RIGHT OF  $v$

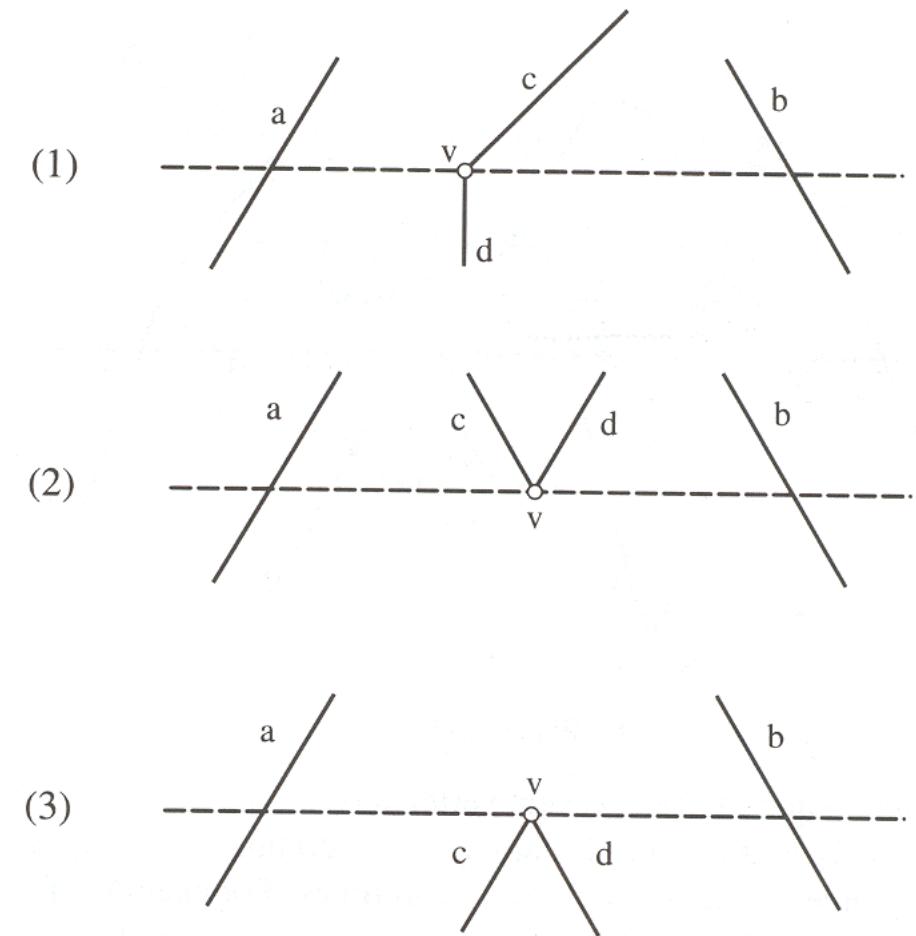


diagonal will be added  
when the sweep line  
reaches  $v_m$



# FORMING TRAPEZOIDS

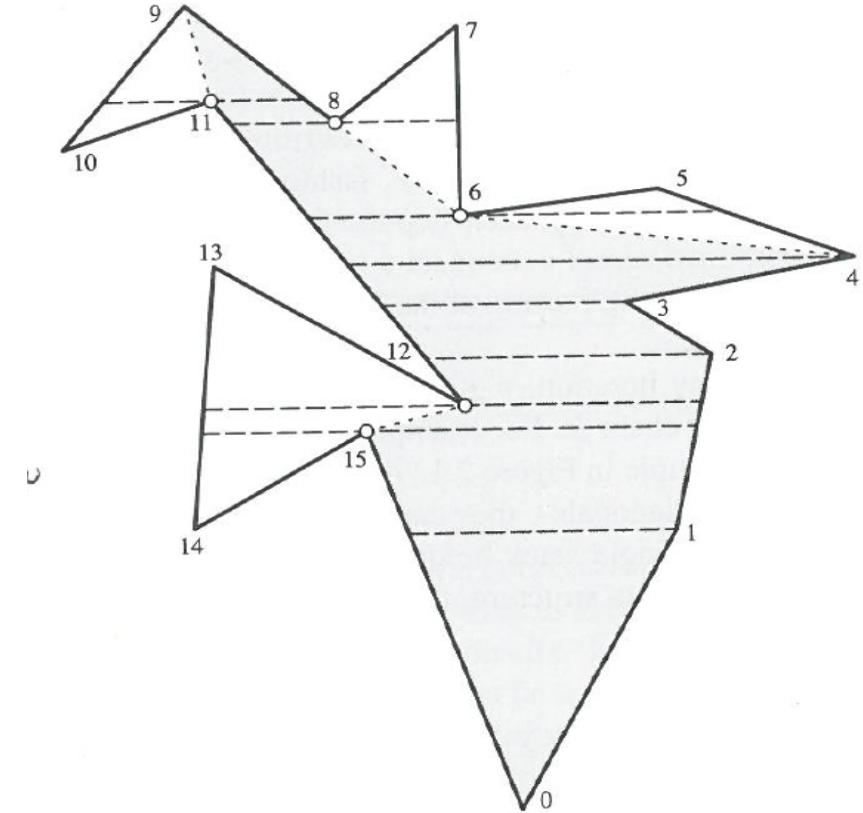
- MAINTAIN A LIST OF SIDES INTERSECTED BY THE SWEEPING LINE, SORTED BY THE X-COORD OF INTERSECTION
- AT EACH EVENT, UPDATE THE LIST
  - Can be done in  $O(\log N)$  if the list is maintained as a balanced binary tree
- OVERALL:  $O(N \log N)$



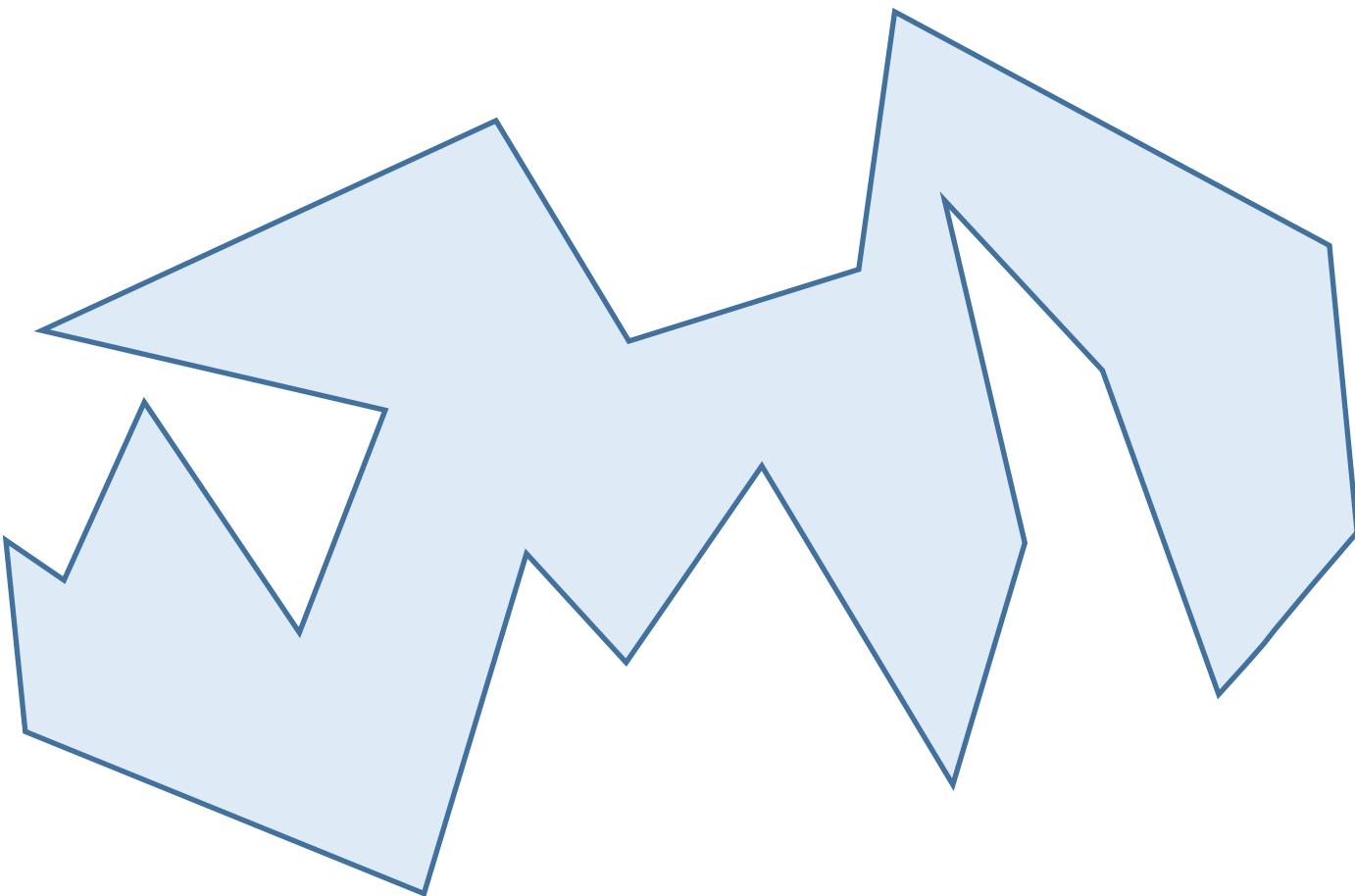
# FORMING TRAPEZOIDS

- **BREAKING INTO MONOTONE PARTS**

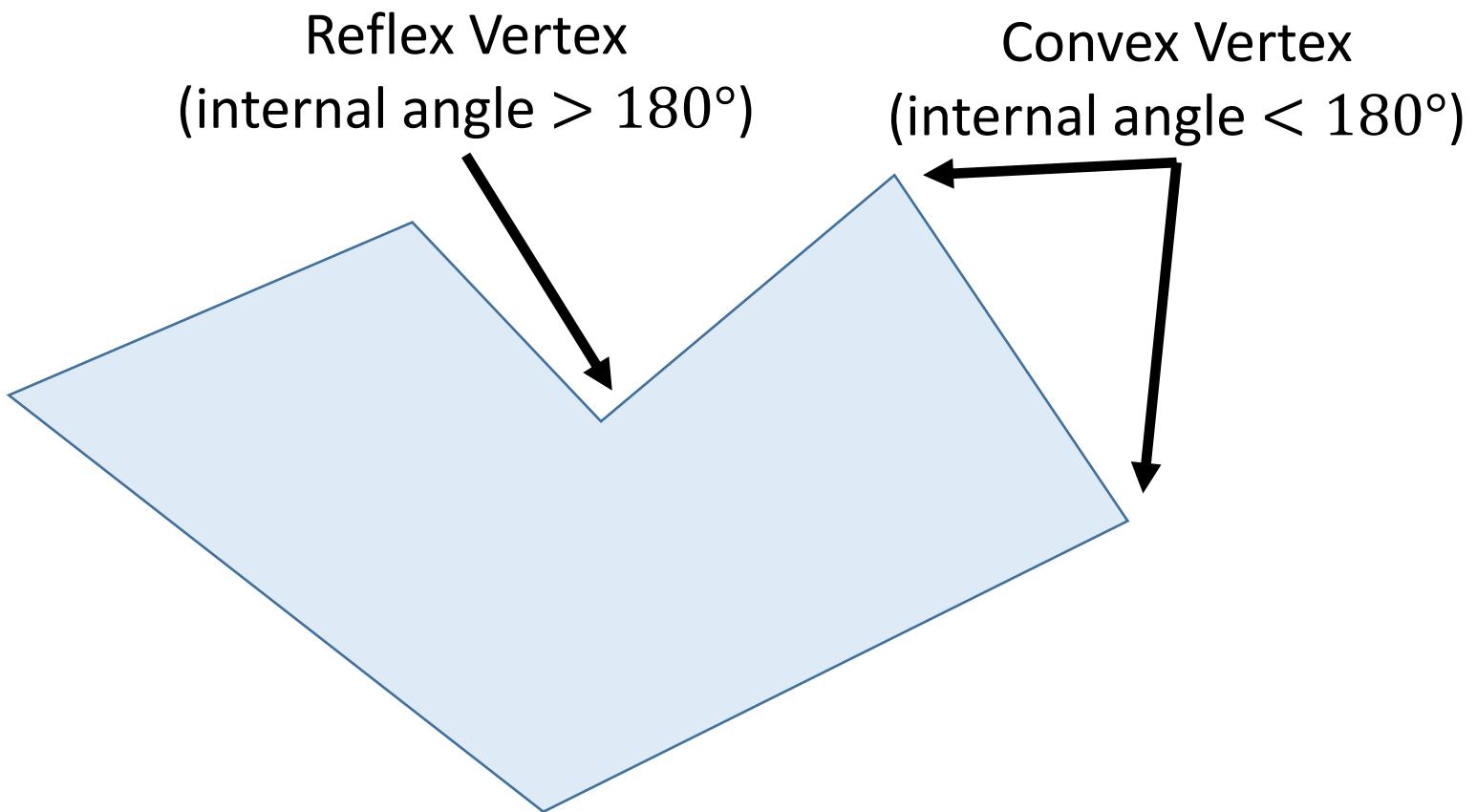
(full algorithm explained in book)



# LINE SWEEP EXAMPLE

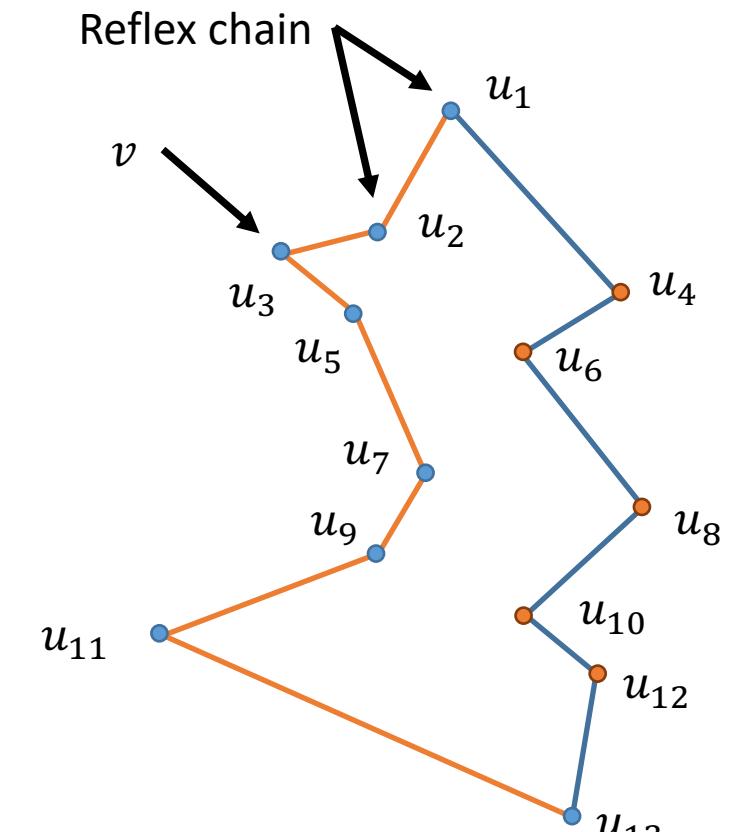


# REMINDER



# TRIANGULATION OF MONOTONE POLYGON

- SORT VERTICES BY Y-COORDINATE BY A MERGE OF THE TWO CHAINS.
- LET  $u_1, u_2, \dots, u_N$  BE THE SORTED SEQUENCE OF VERTICES, SO  $y(u_1) > y(u_2) > \dots > y(u_N)$ .
- THE ALGORITHM PREFORMS OPERATIONS ON A REFLEX CHAIN, WHICH IS STORED AS A STACK
- INITIALIZATION
  - Reflex chain pushes two top vertices
  - Let  $v$  be the third highest vertex



## DESCRIPTION OF THE PROCESSING TRIANGULATION

- THE ALGORITHM PROCESSES ONE VERTEX AT A TIME IN ORDER OF DECREASING Y COORDINATE, CREATING DIAGONALS OF POLYGON P
  - At each step process 1 of 3 cases
- EACH DIAGONAL BOUNDS A TRIANGLE, AND LEAVES A POLYGON WITH ONE LESS SIDE STILL TO BE TRIANGULATED



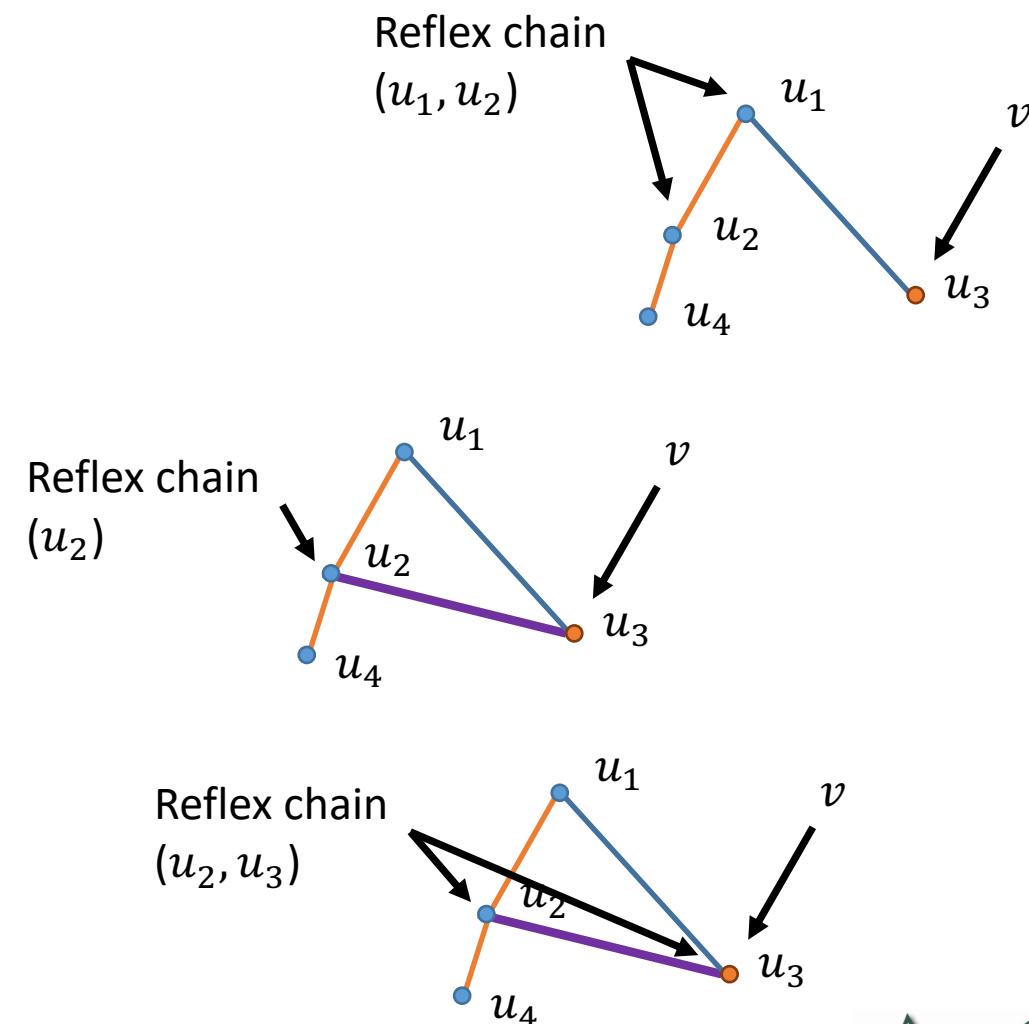
# TRIANGULATION OF MONOTONE POLYGON

- WHILE  $v \neq \text{LOWEST VERTEX}$  DO:
  - Case 1:  $v$  is on chain opposite reflex chain
  - Case 2:  $v$  is adjacent to bottom of reflex chain and  $v+$  is strictly convex
  - Case 3:  $v$  is adjacent to bottom of reflex chain and  $v+$  is reflex or flat



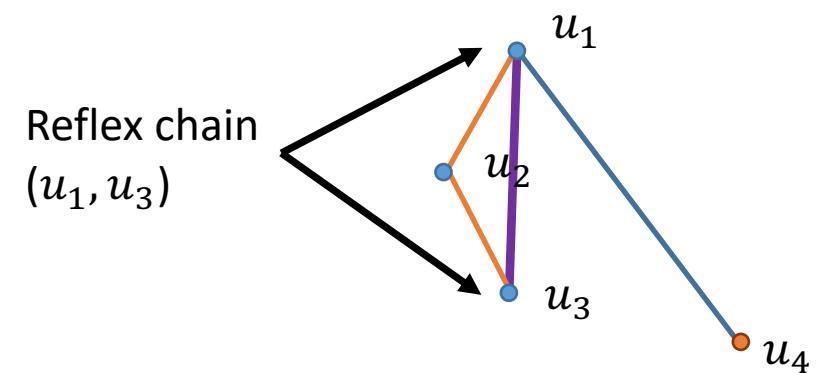
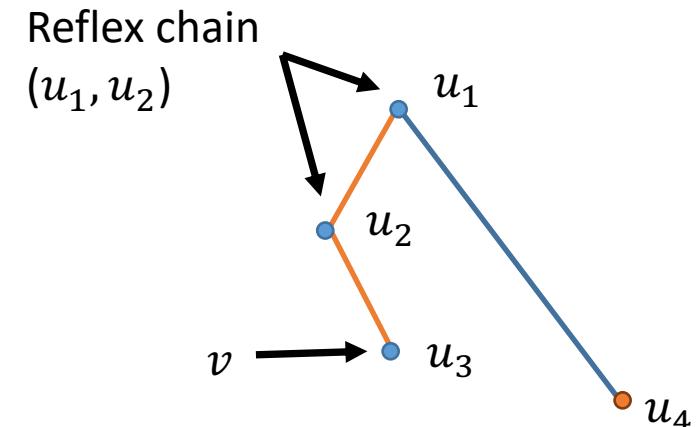
# TRIANGULATION OF MONOTONE POLYGON

- CASE I: V IS ON CHAIN  
OPPOSITE REFLEX CHAIN
  - Draw diagonal from  $v$  to second vertex from top of chain
  - Remove top of chain
  - If chain has one element then add  $v$ , advance  $v$



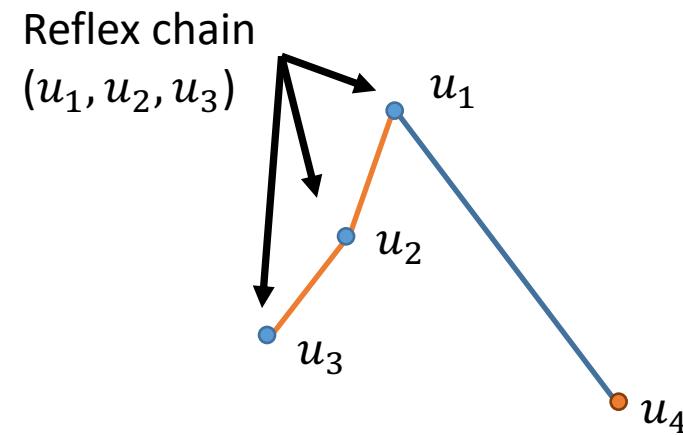
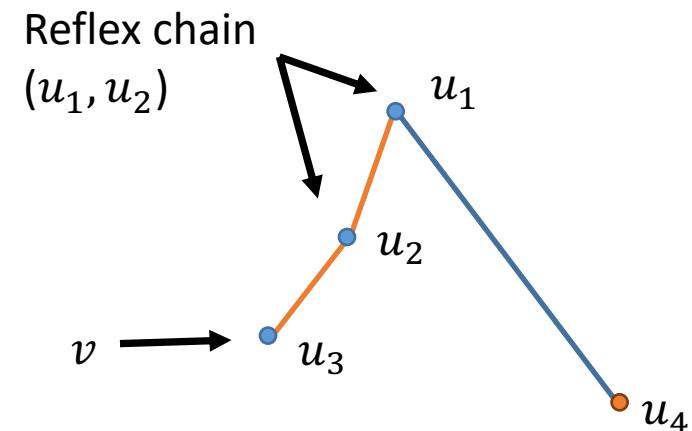
# TRIANGULATION OF MONOTONE POLYGON

- CASE 2:  $v$  IS ADJACENT TO BOTTOM OF REFLEX CHAIN AND  $v+$  IS STRICTLY CONVEX
  - Draw diagonal from  $v$  to second vertex from bottom of chain
  - Remove bottom of chain
  - Add  $v$  to chain, advance  $v$

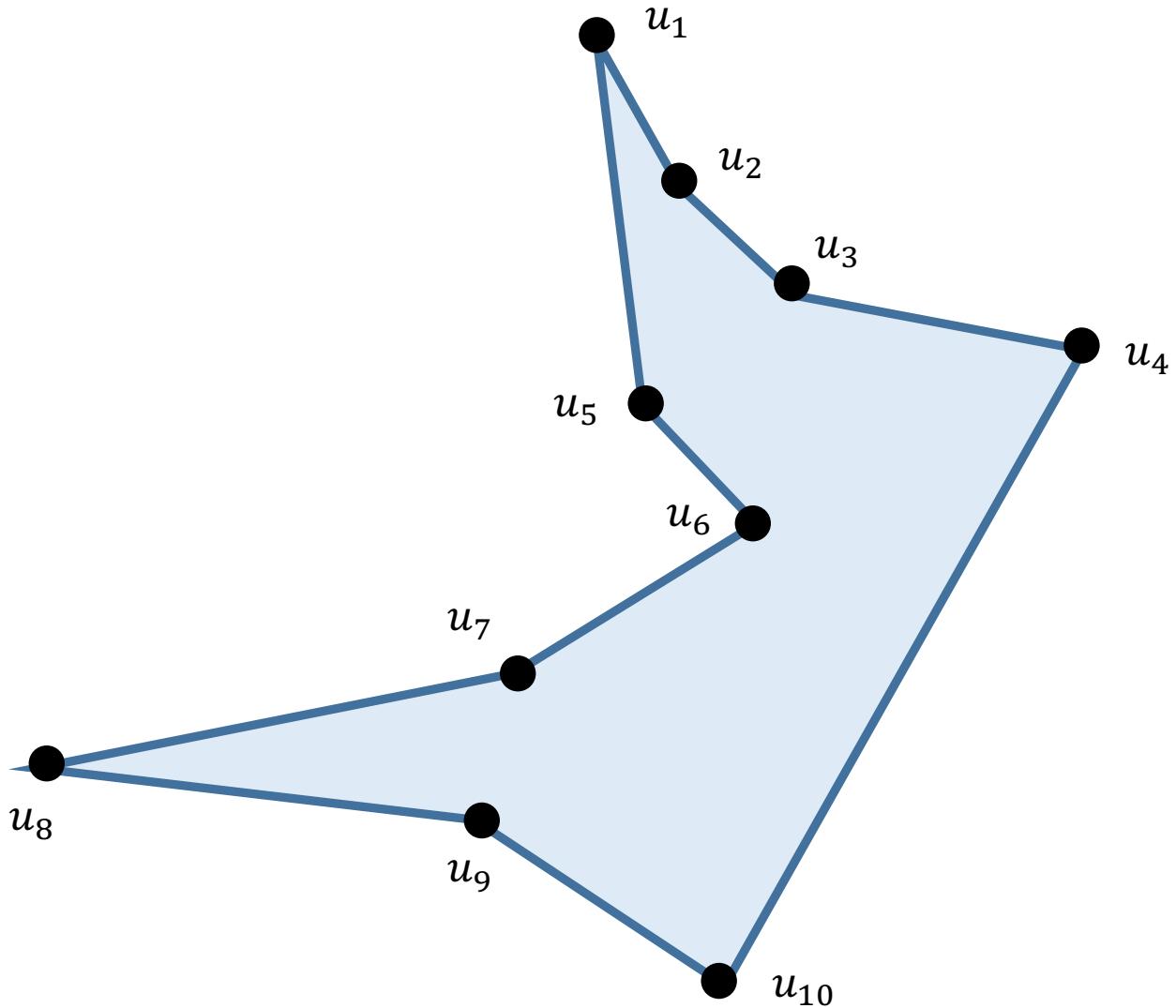


# TRIANGULATION OF MONOTONE POLYGON

- CASE 3:  $v$  IS ADJACENT TO BOTTOM OF REFLEX CHAIN AND  $v+$  IS REFLEX OR FLAT
  - Add  $v$  to bottom of reflex chain, advance  $v$



## TRIANGULATION OF MONOTONE POLYGON EXAMPLE



# TRIANGULATION OF MONOTONE POLYGON EXAMPLE

## Case 1: $v$ is on chain opposite reflex chain

- Draw diagonal from  $v$  to second vertex from top of chain
- Remove top of chain
- If chain has one element then add  $v$ , advance  $v$

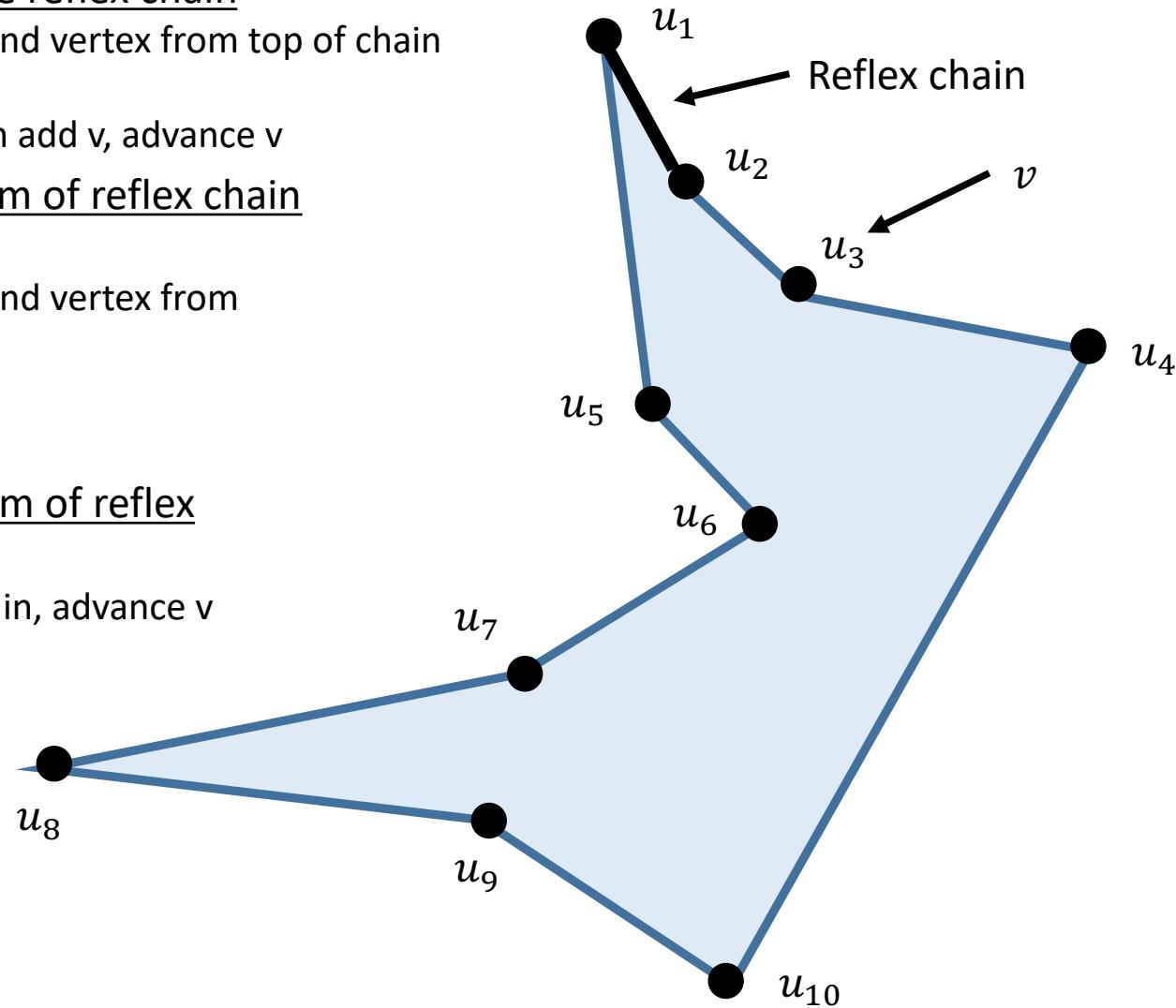
## Case 2: $v$ is adjacent to bottom of reflex chain

AND  $v+$  is strictly convex

- Draw diagonal from  $v$  to second vertex from bottom of chain
- Remove bottom of chain
- Add  $v$  to chain, advance  $v$

## Case 3: $v$ is adjacent to bottom of reflex chain AND $v+$ is reflex or flat

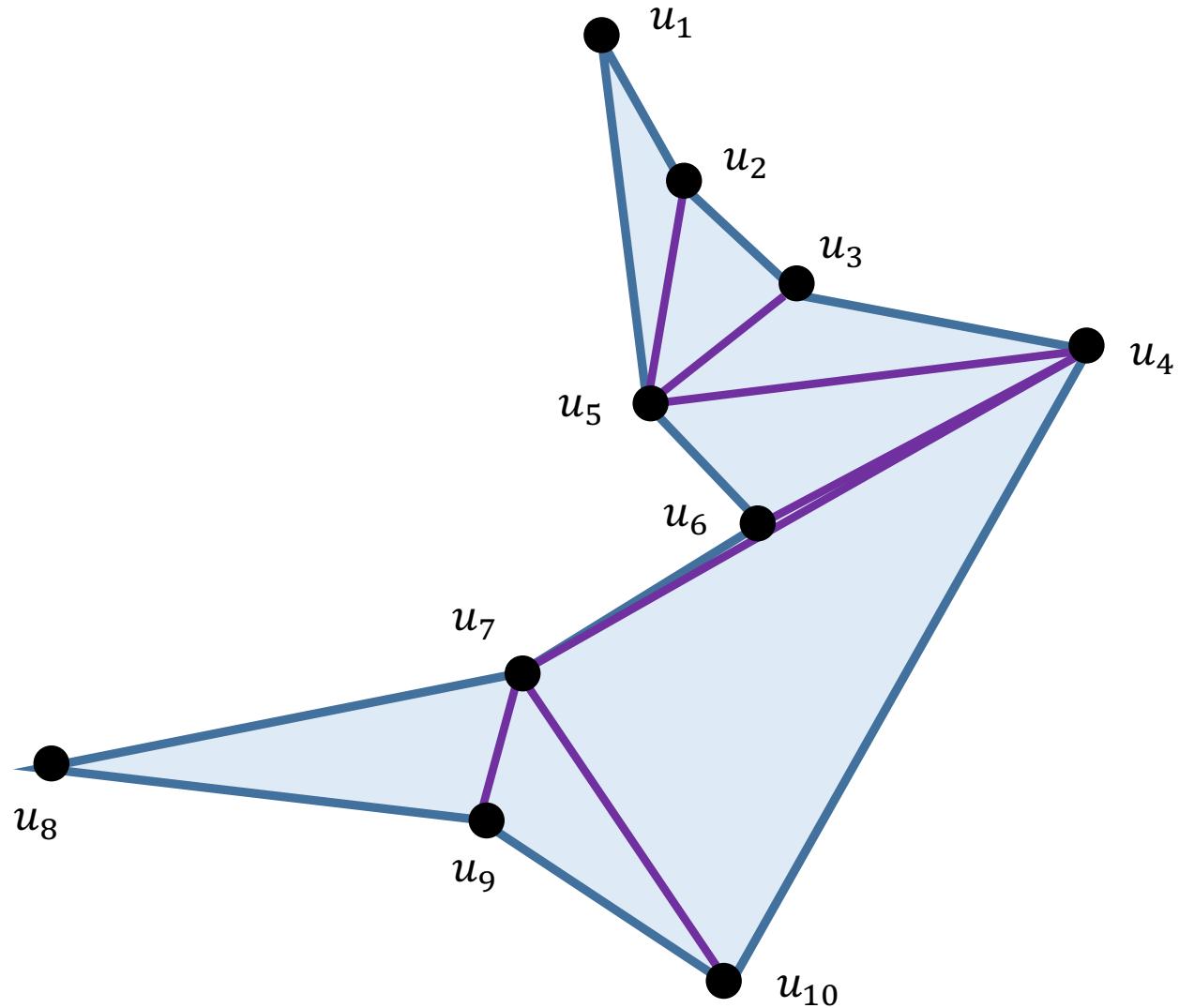
- Add  $v$  to bottom of reflex chain, advance  $v$



Case 3



## TRIANGULATION OF MONOTONE POLYGON EXAMPLE



## ANALYSIS OF TRIANGULATING A MONOTONE POLYGON

- THE INITIAL SORT (MERGE) REQUIRES  $O(N)$  TIME.
- TRIANGULATION VISITS EACH OF THE  $N$  VERTICES AND PLACES ON THE STACK EXACTLY ONCE, EXCEPT WHEN THE WHILE FAILS IN CASE (II).
  - This happens at most once per vertex, so that time can be charged to the current vertex.
- $\Rightarrow$  THE ALGORITHM REQUIRES  $O(N)$  TIME TO TRIANGULATE A MONOTONE POLYGON, WHERE  $N$  IS THE NUMBER OF VERTICES OF THE POLYGON.



## PUTTING IT ALL TOGETHER

- ANALYSIS OF TRIANGULATION OF SIMPLE POLYGON
  - Sort vertices by y coordinates:  $O(N \log N)$
  - Perform plane sweep to construct trapezoids:  $O(N \log N)$
  - Partition into monotone polygons:  $O(N)$
  - Triangulate each monotone polygon:  $O(N)$
- OVERALL:  $O(N \log N)$ 
  - $N$  is the number of vertices of the polygon



## SUMMARY

- TRIANGULATION BY EAR REMOVAL IMPROVES THE DIAGONAL-BASED TRIANGULATION FROM  $O(n^4)$  TO  $O(n^2)$
- PARTITION A SIMPLE POLYGON INTO MONOTONE PARTS AND TRIANGULATION OF MONOTONE POLYGONS REQUIRES  $O(n \log n)$  TIME COMPLEXITY



## DESCRIPTION OF THE PROCESSING TRIANGULATION OF Y-MONOTONE POLYGONS

- THE ALGORITHM PROCESSES ONE VERTEX AT A TIME IN ORDER OF DECREASING Y COORDINATE, CREATING DIAGONALS OF POLYGON P.
  - The sweep line moves from top to down and stops at each vertex of polygon P
- EACH DIAGONAL BOUNDS A TRIANGLE, AND LEAVES A POLYGON WITH ONE LESS SIDE STILL TO BE TRIANGULATED



## DESCRIPTION OF THE PROCESSING TRIANGULATION OF Y-MONOTONE POLYGONS

- Sort the vertices top-to-down by a merge of the two chains
- Initialize a stack. Push the first two vertices in the stack
- Take the next vertex  $u$ , and triangulate as much as possible, top-down, while popping the stack
- Push  $u$  onto the stack



# TOWARDS LINEAR-TIME TRIANGULATION

Year	Complexity	Authors
1911	$O(n^2)$	Lennes
1978	$O(n \log n)$	Garey et al.
1983	$O(n \log r)$ , $r$ reflex	Hertel & Melhorn
1984	$O(n \log s)$ , $s$ sinuosity	Chazelle & Incerpi
1986	$O(n \log \log n)$	Tarjan & Van Wyk
1988	$O(n + nt_0)$ , $t_0$ int. triangles	Toussaint
1989	$O(n \log^* n)$ , randomized	Clarkson, Tarjan & Van Wyk
1990	$O(n \log^* n)$ bounded integer coordinates	Kirkpatrick, Klawe, Tarjan
1990	$O(n)$	Chazelle
1991	$O(n \log^* n)$ , randomized	Seidel



# LINEAR-TIME TRIANGULATION

- **CHAZELLE'S ALGORITHM (HIGH-LEVEL SKETCH)**
  - Computes visibility map
  - Algorithm is like MergeSort (divide-and-conquer)
    - Partition polygon of  $n$  vertices into  $n/2$  vertex chains
    - Merge visibility maps of subchains to get one for chain
  - Improve this by dividing process into 2 phases:
    1. Coarse approximations of visibility maps for linear-time merge
    2. Refine coarse map into detailed map in linear time





