

# COT 4521: INTRODUCTION TO COMPUTATIONAL GEOMETRY

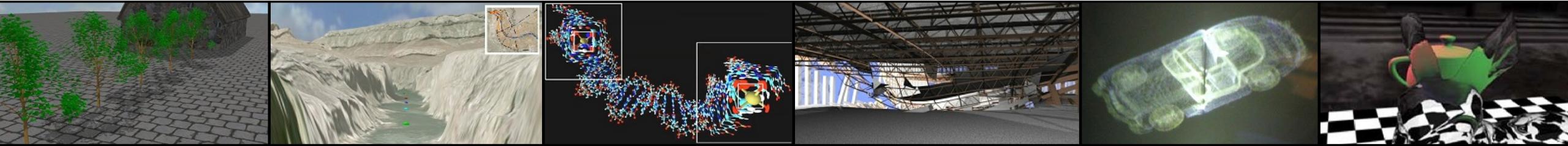
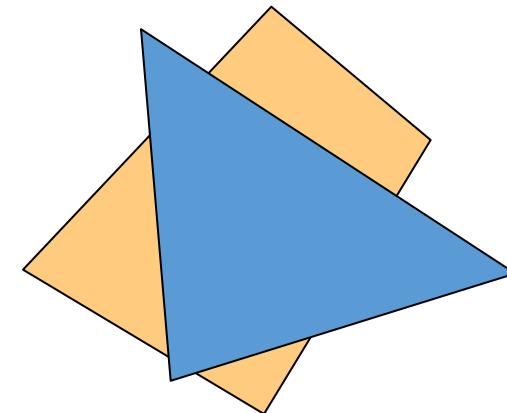
---



## Convex Polygon Intersection

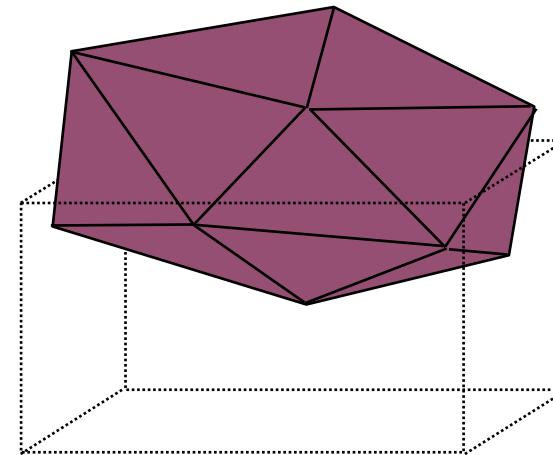
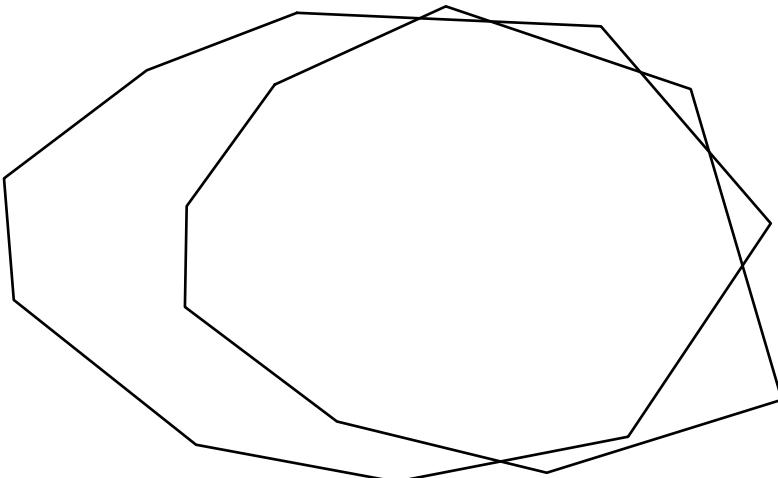
Paul Rosen  
Assistant Professor  
University of South Florida

Some slides from Valentina Korzhova



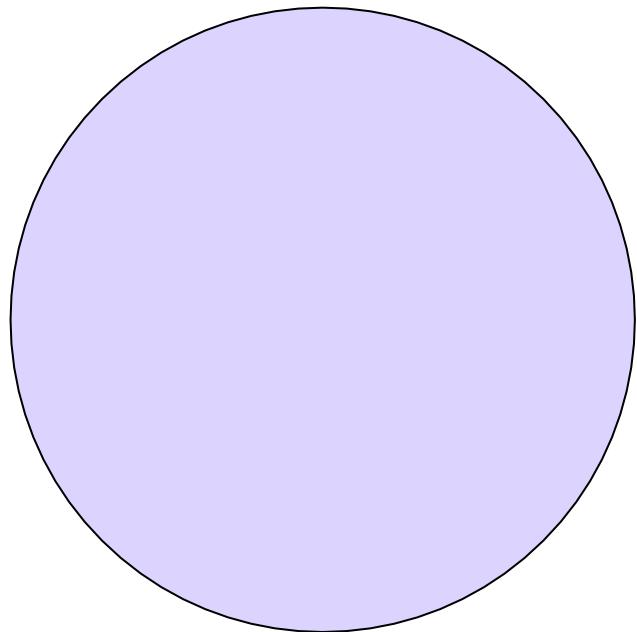
# PROBLEM DESCRIPTION

- HERE WE CONSIDER A CATEGORY OF RELATED INTERSECTION CONSTRUCTION PROBLEMS: GIVEN TWO (OR MORE) GEOMETRIC OBJECTS, CONSTRUCT A NEW OBJECT WHICH IS THEIR INTERSECTION.
  - The objects dealt with will be polygons, polyhedra, half-planes, half-spaces, and related types.
  - The objects will generally be specified by their vertices and orientations to identify the interior of the object. (The latter may be given by convention.)

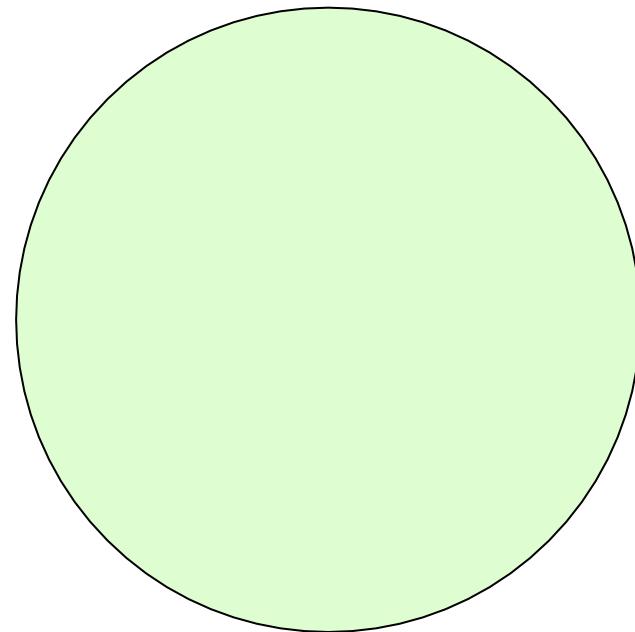


# BRIEF SET THEORY REVIEW

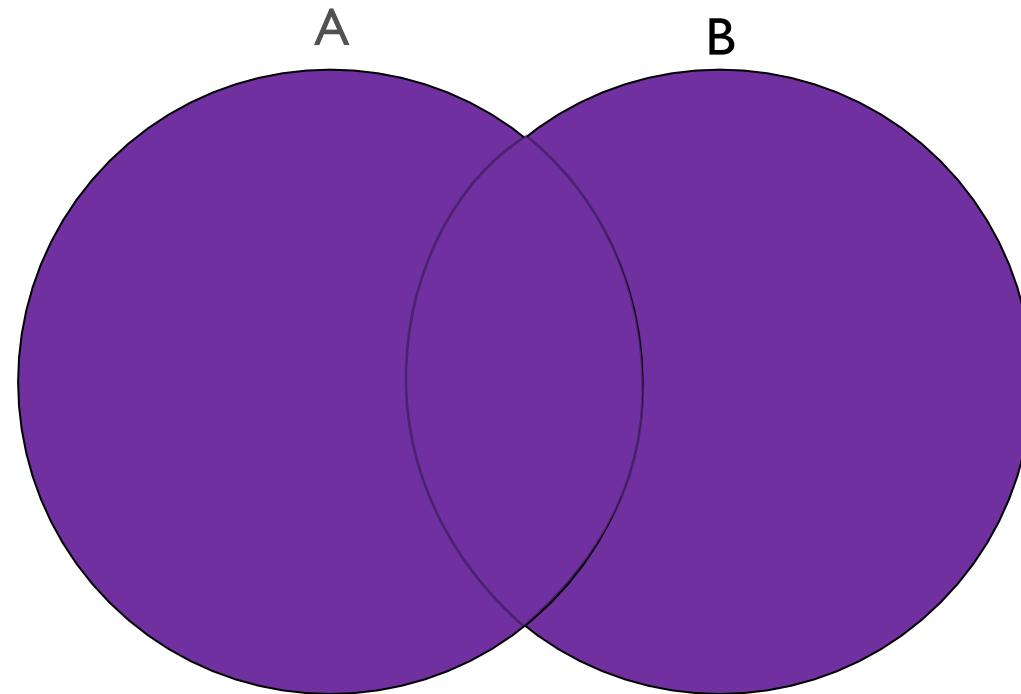
A



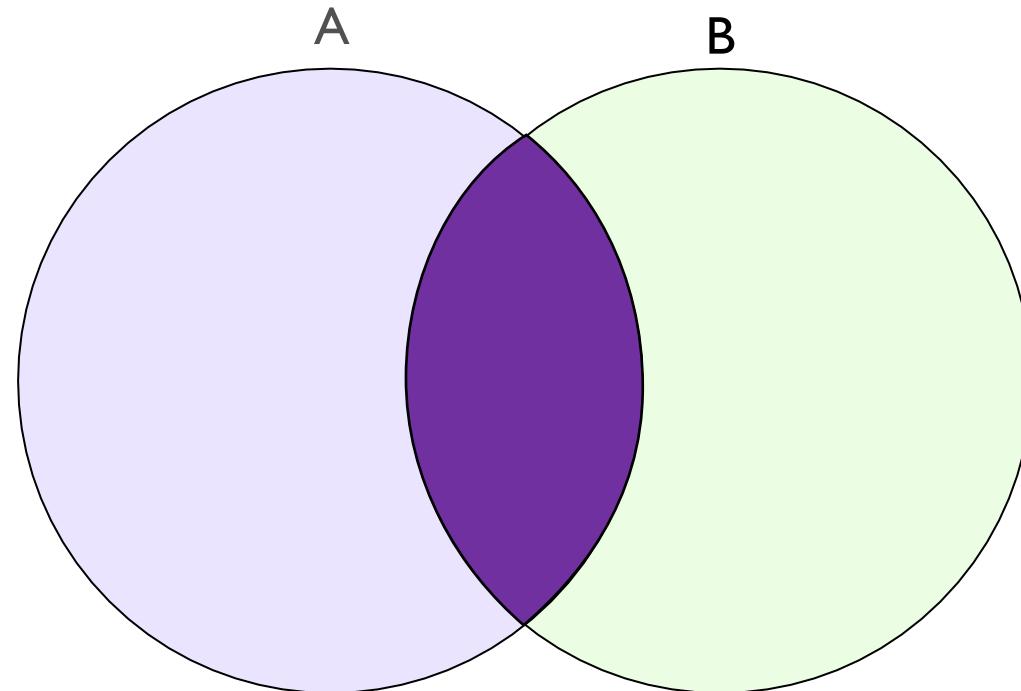
B



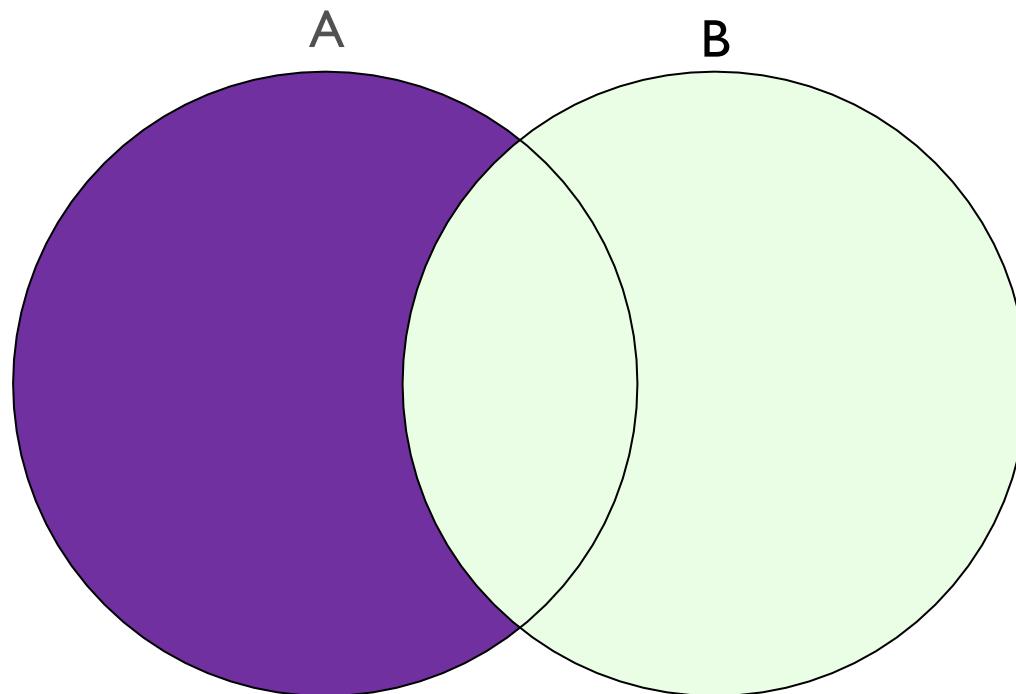
# SET THEORY—UNION: $A \cup B$



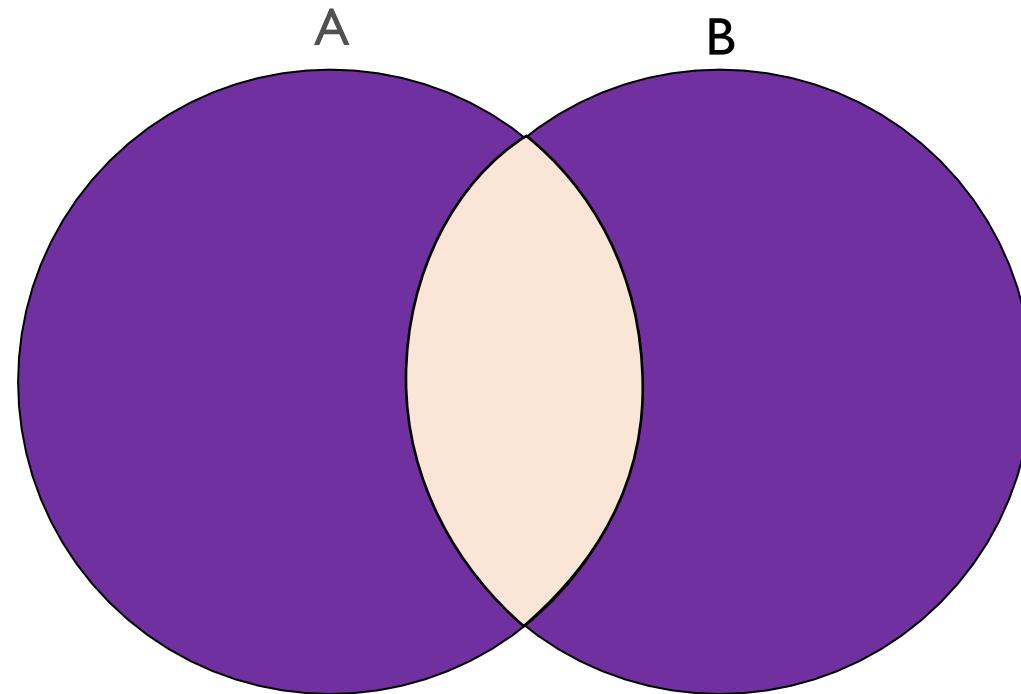
# SET THEORY—INTERSECTION: $A \cap B$



# SET THEORY—SET DIFFERENCE: $A \setminus B$

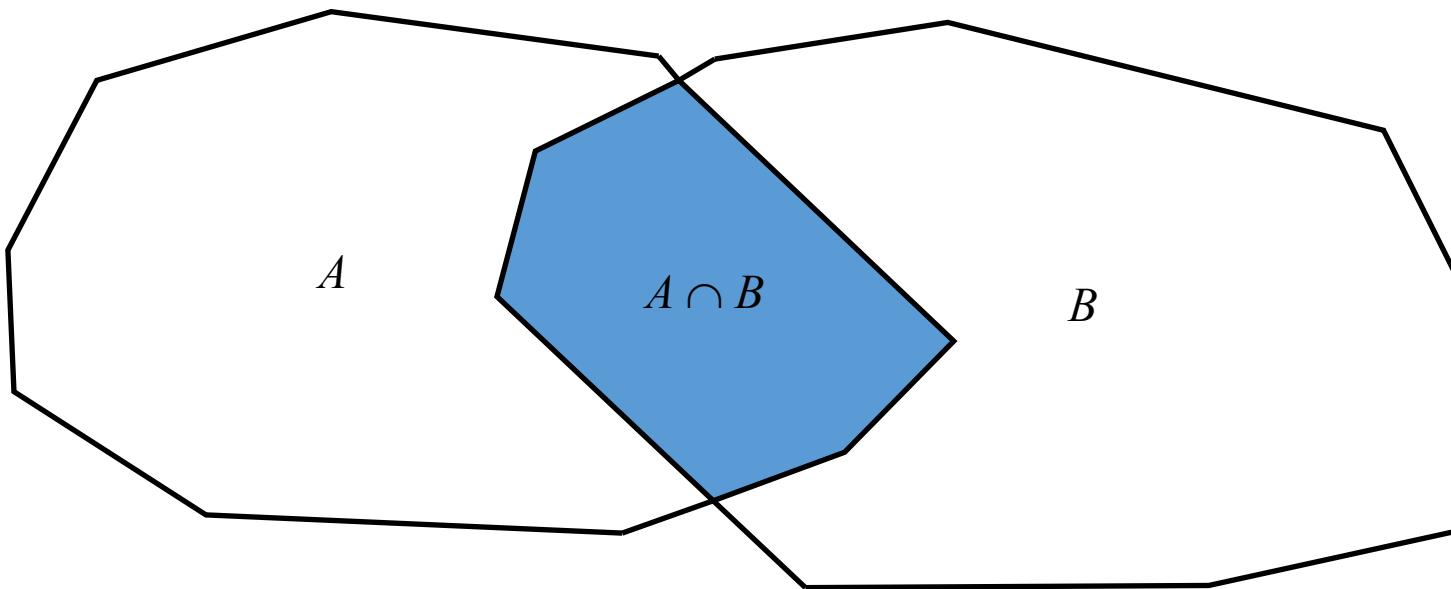


# SET THEORY—SYMMETRIC DIFFERENCE: $A \ominus B$



## INTERSECTION EXAMPLE

- GIVEN TWO CONVEX POLYGONS, CONSTRUCT THEIR INTERSECTION.
  - (Polygon  $\equiv$  boundary and interior, intersection  $\equiv$  all points that are members of both polygons.)



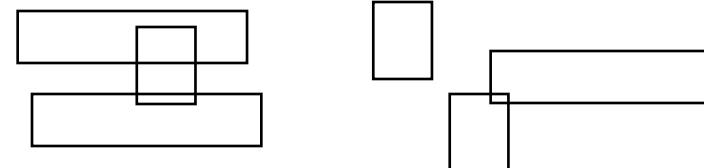
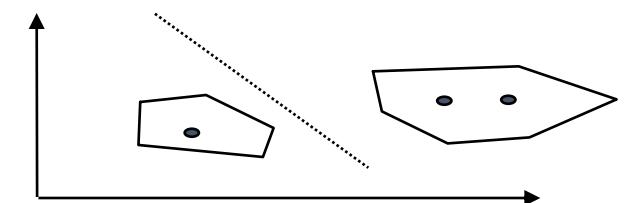
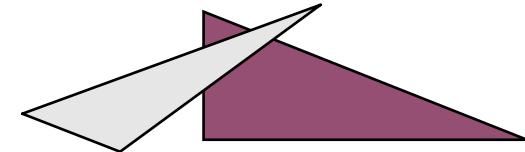
# GENERAL INTERSECTION PROBLEMS

- TEST OR DECISION PROBLEM
  - Given two geometric objects, determine if they intersect.
- PAIRWISE COUNTING OR REPORTING PROBLEM
  - Given a data set of  $N$  geometric objects, count or report their intersections.
- CONSTRUCTION PROBLEM
  - Given a data set of  $N$  geometric objects, construct a new object which is their intersection.

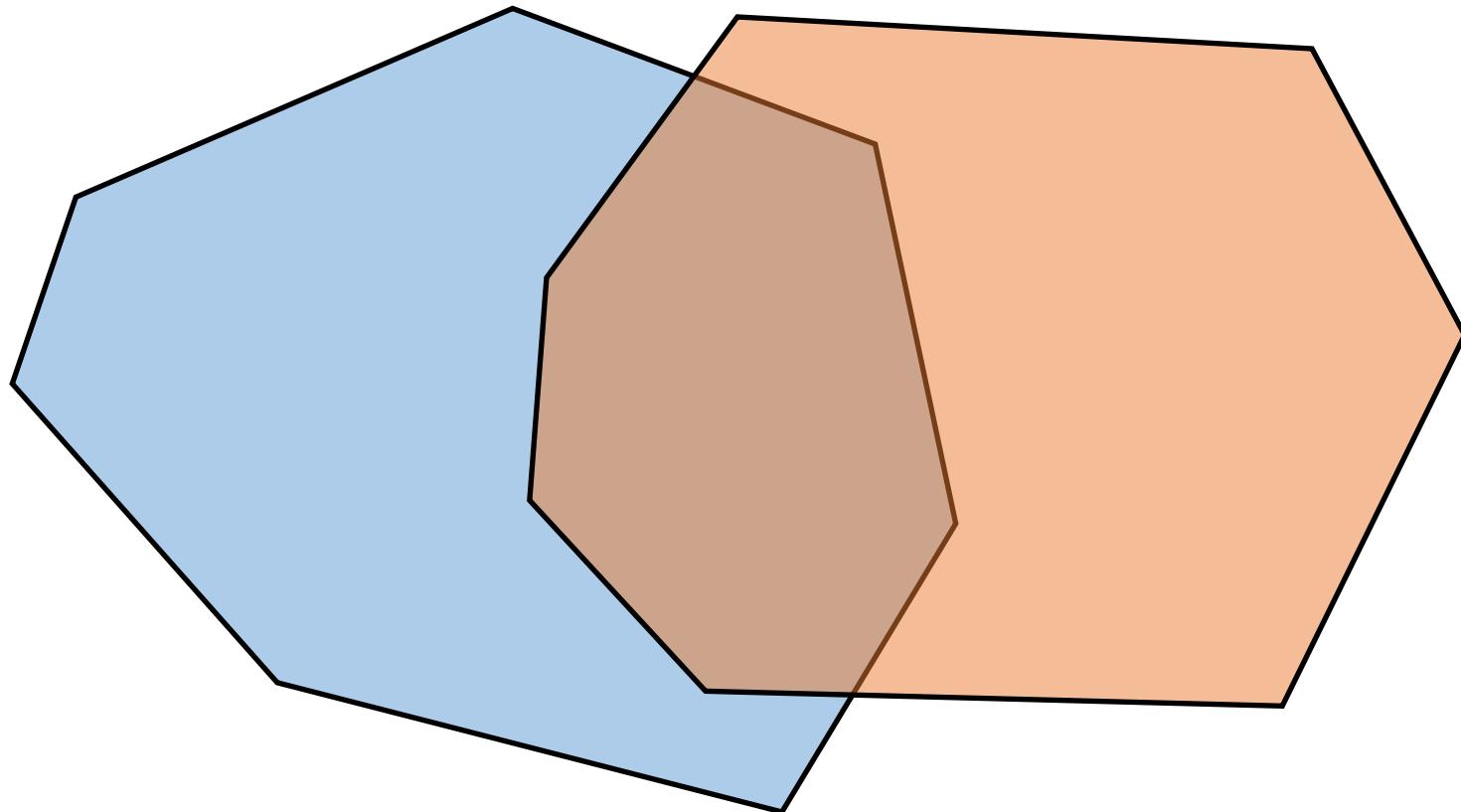


# APPLICATIONS

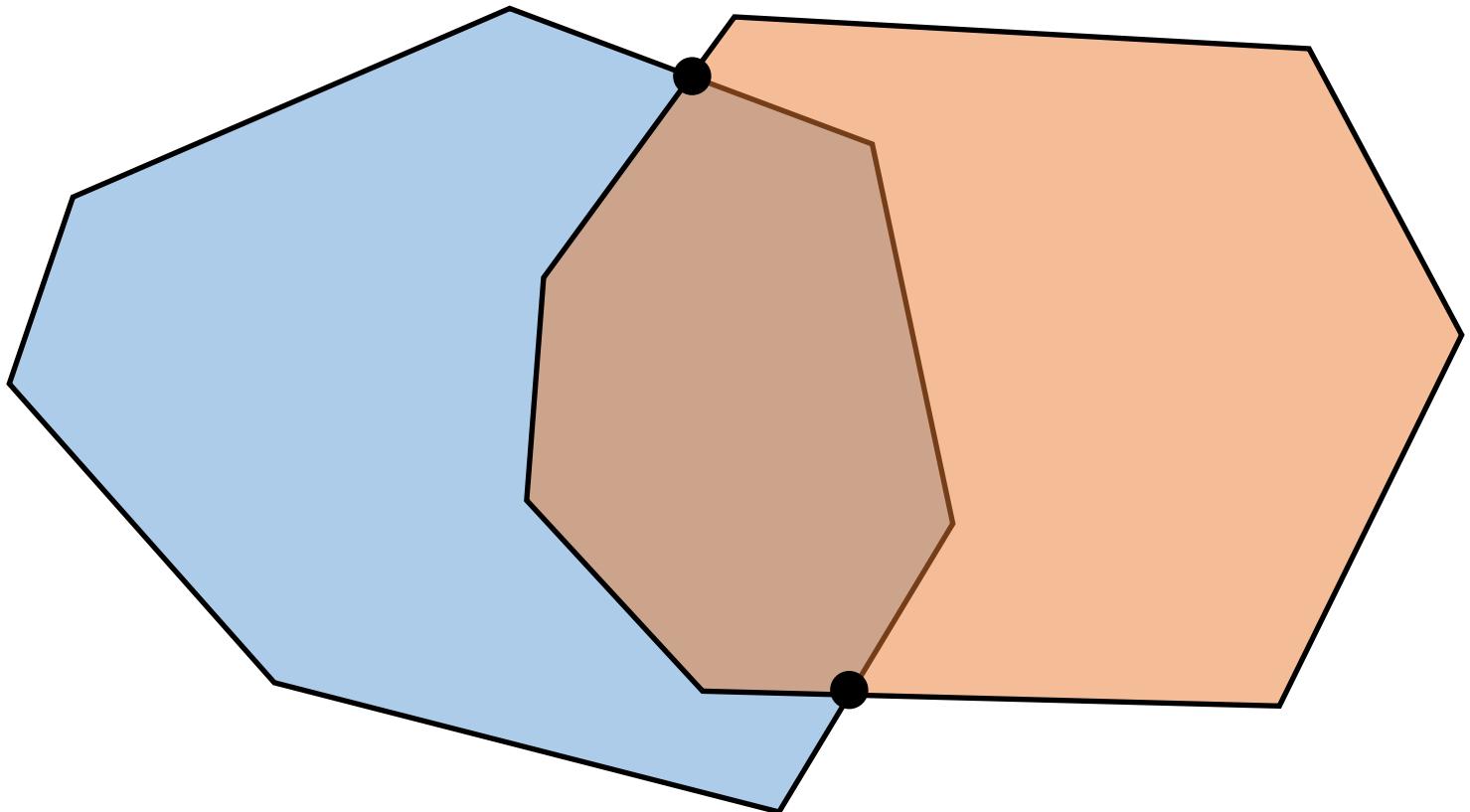
- DOMAIN: GRAPHICS
  - Problem: Hidden-line and hidden surface removal
  - Approach: Intersection of two polygons
  - Type: Construction
- DOMAIN: PATTERN RECOGNITION
  - Problem: Finding a linear classifier between two sets of points
  - Approach: Intersection of convex hulls
  - Type: Test
- DOMAIN: VLSI DESIGN
  - Problem: Component overlap detection
  - Approach: Intersection of rectangles
  - Type: Pairwise



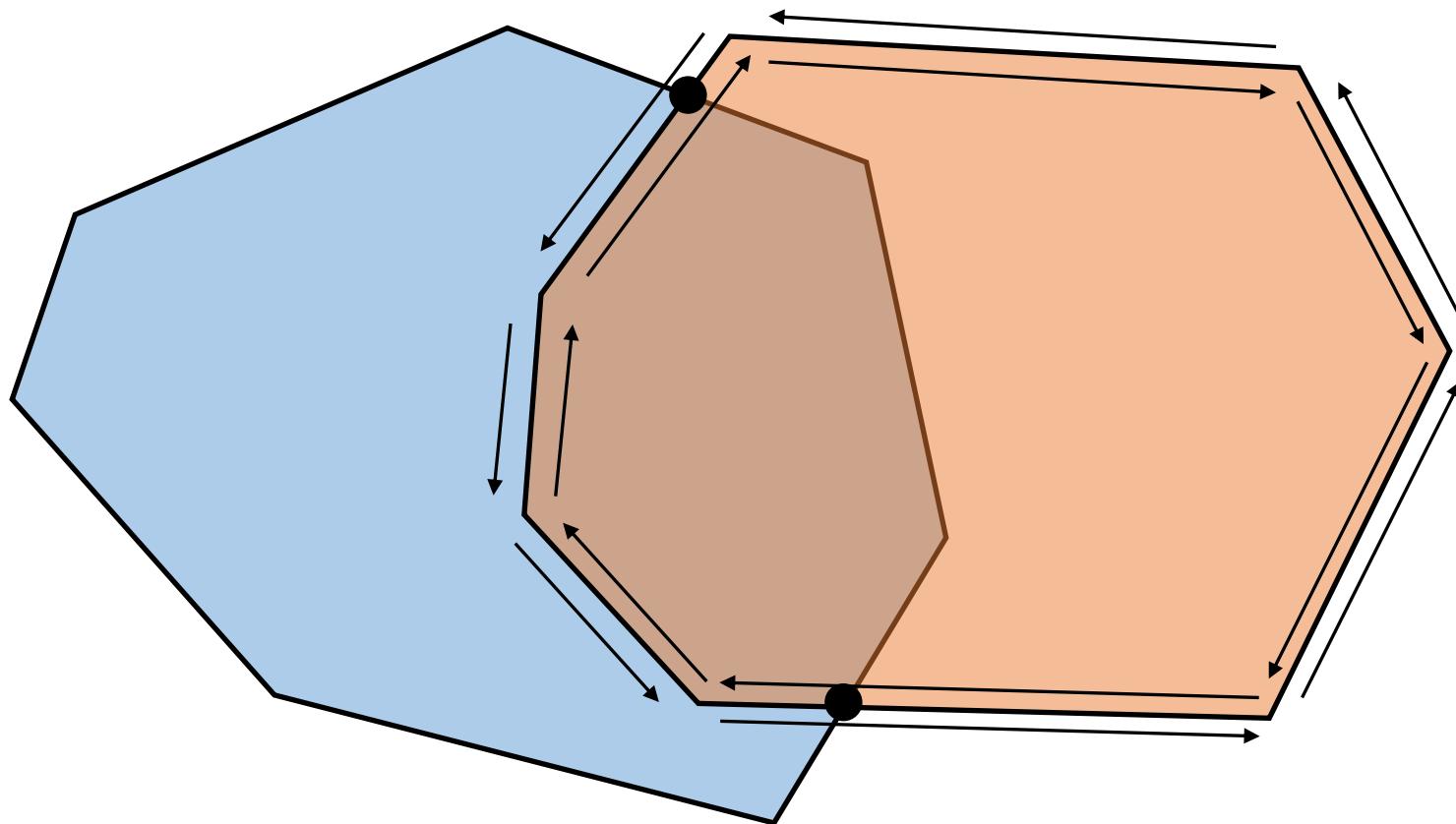
# CAN WE USE THE LINE INTERSECTION ALGORITHM TO FIND THE INTERSECTION OF 2 REGIONS?



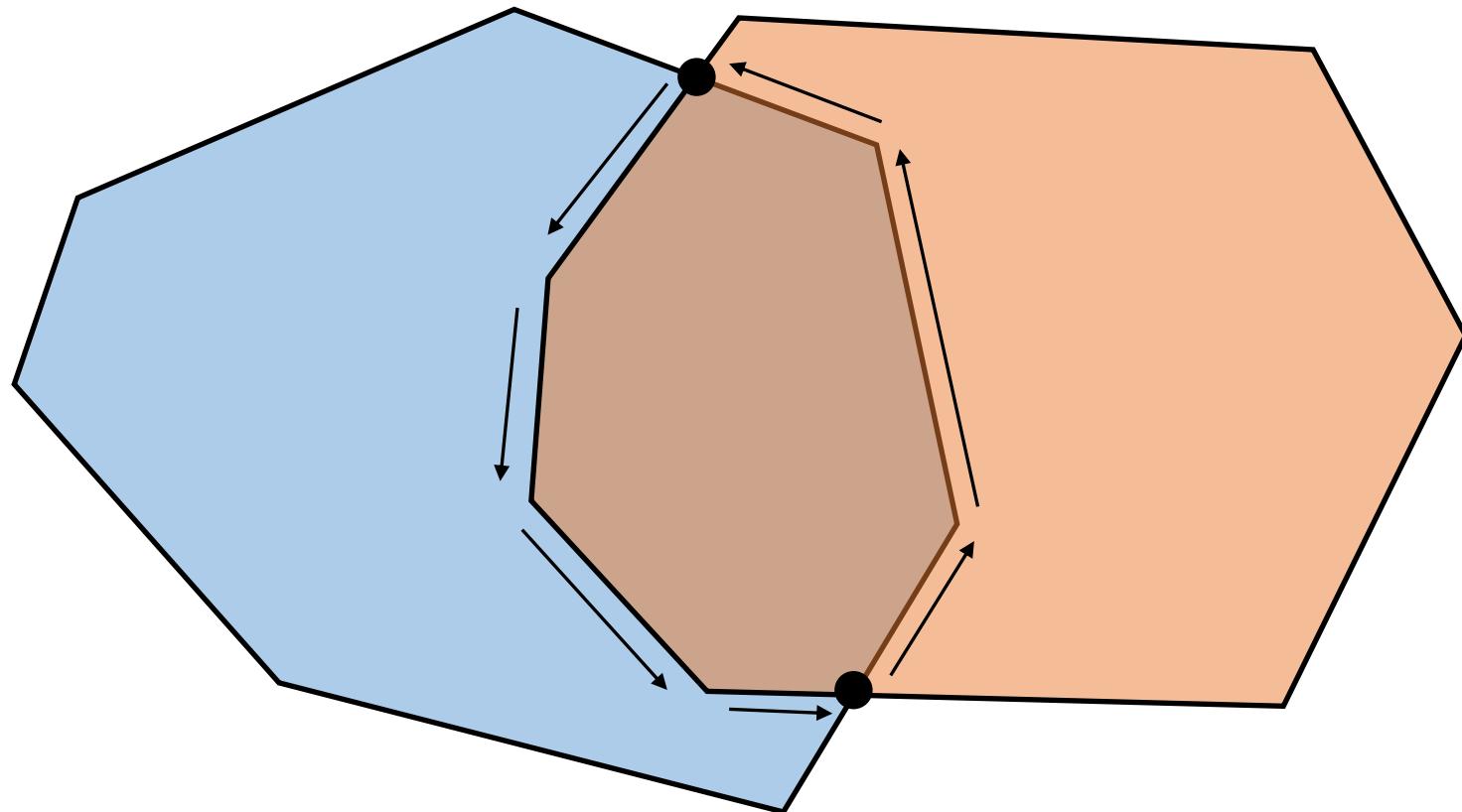
# RUN SWEEP ALGORITHM



## CONNECTIVITY STORED AS DOUBLY LINKED LIST



# WALK THE GRAPH

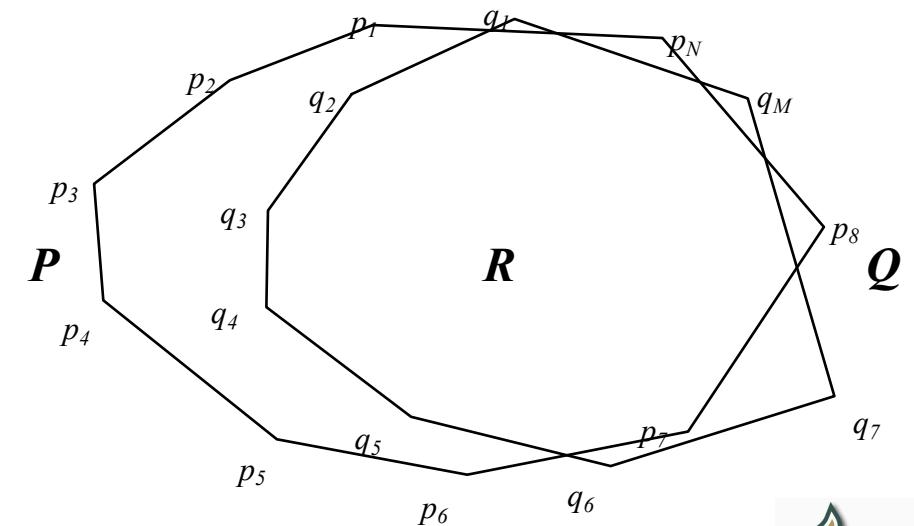


But we can do better!



# CONVEX POLYGON INTERSECTION

- **DEFINITION:** For convex (rather than arbitrary) polygons, problem is linear.
  - **INSTANCE:** Convex polygons  $P$  and  $Q$ , with vertex sets  $P = \{p_1, p_2, \dots, p_N\}$  and  $Q = \{q_1, q_2, \dots, q_M\}$  respectively.
  - **QUESTION:** Construct polygon  $R$  which is their intersection.



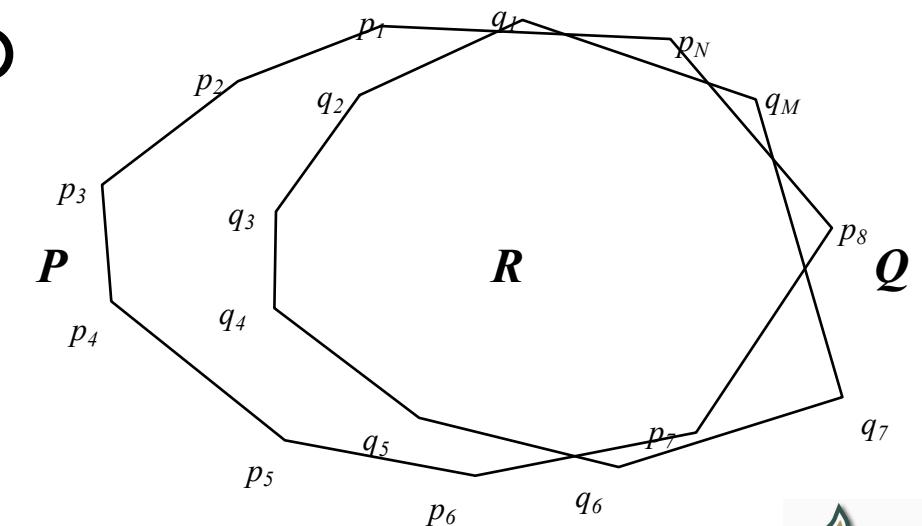
# INTERSECTION OF LINE SEGMENT

- ANALYSIS
  - Preprocessing: Intersection finding  $O((N + K) \log N)$
  - Query:  $O(N + M)$
  - Storage:  $O(N + M)$  edge links
- CAN WE DO ANY BETTER?



# INTERSECTION OF CONVEX POLYGONS

- THE INTERSECTION OF TWO CONVEX POLYGONS WILL HAVE LINEAR  $O(N + M)$  COMPLEXITY (BOTH THE OBJECT AND THE CONSTRUCTION PROCESS).
- BY CONVENTION,  $P$ ,  $Q$ , AND  $R$  WILL BE ORIENTED COUNTERCLOCKWISE, SO THAT THE INTERIORS OF THE POLYGONS LIE TO THE LEFT OF THEIR EDGES



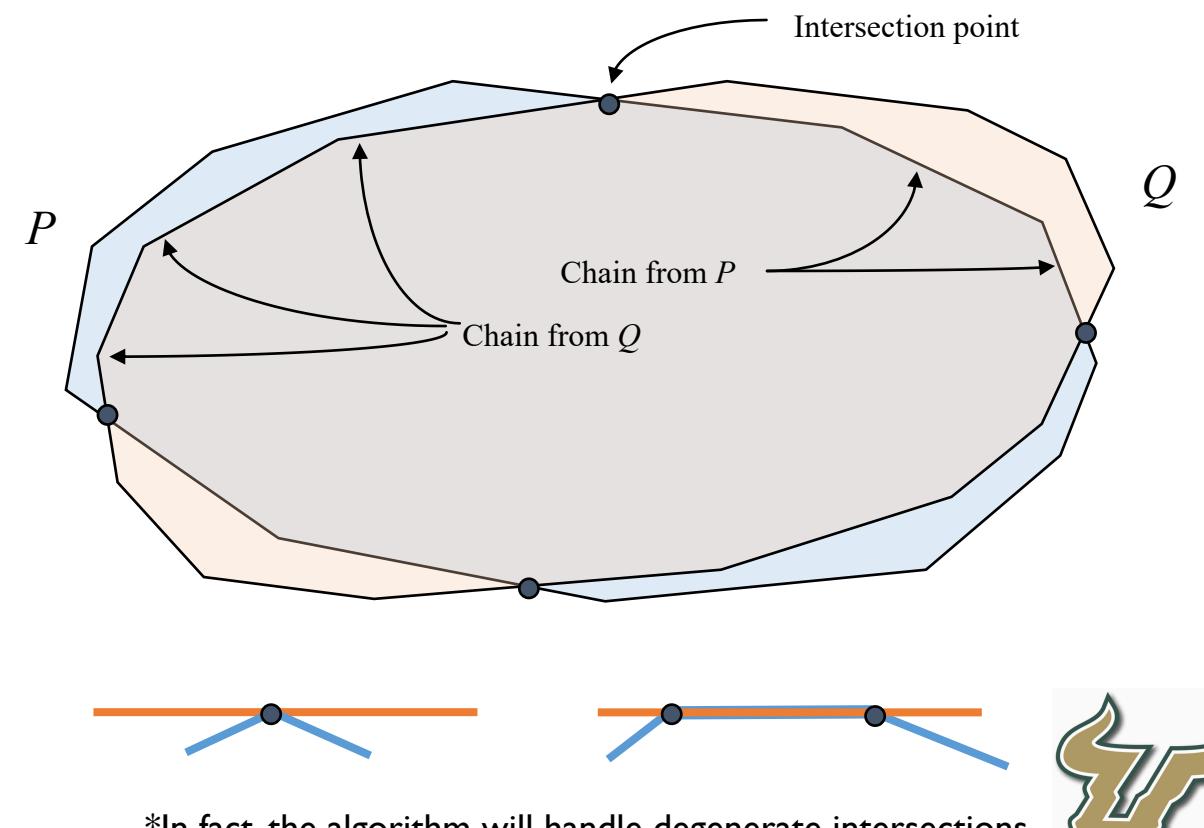
## O'ROURKE-CHIEN-OLSON-NADDOR ALGORITHM

- THIS ALGORITHM DESCRIBED IN THREE SOURCES:
  1. O'Rourke with C code.
  2. Preparata
  3. Laszlo with C++ code.
    - Presentation is a synthesis of all three with focus on Laszlo.
- THE ALGORITHM IS BASED ON THREE UNDERGRADUATES' HOMEWORK ASSIGNMENT SOLUTIONS IN 1982.
  - Simpler than previously existing linear  $O(N + M)$  algorithm, e.g., Shamos' 1978 algorithm.



# INTERSECTION OF CONVEX POLYGONS

- INTERSECTION STRUCTURE
  - Form the intersection polygon  $R$  of two convex polygons  $P$  and  $Q$
  - For the moment, assume that the edges of  $P$  and  $Q$  intersect non-degenerately\* (when two edges intersect, they do so at a single point)

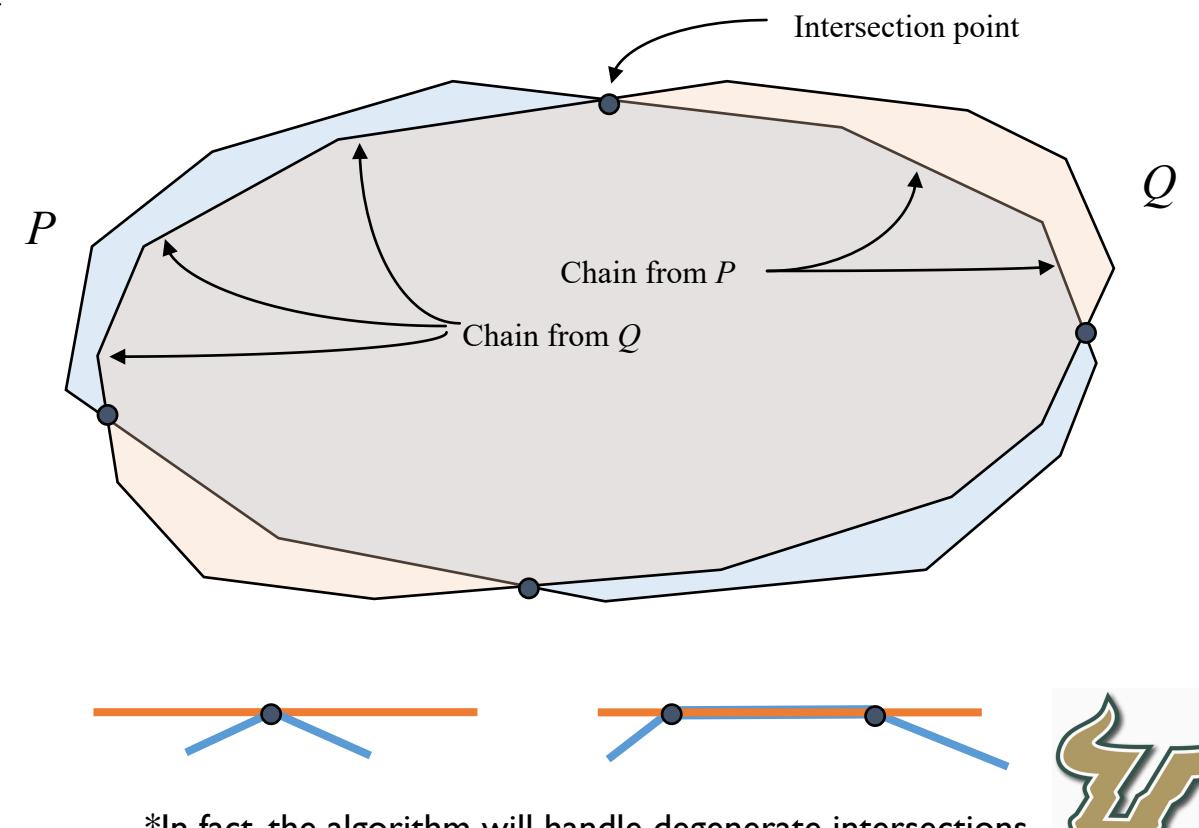


\*In fact, the algorithm will handle degenerate intersections.



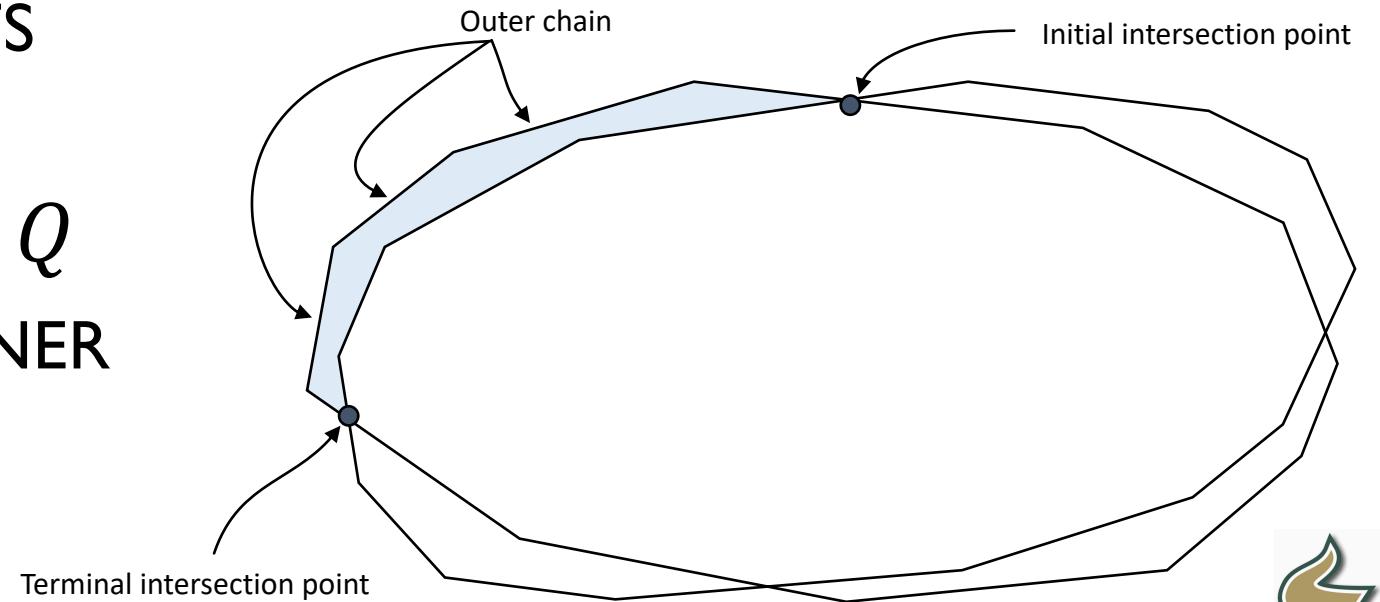
# INTERSECTION OF CONVEX POLYGONS

- **GIVEN THIS ASSUMPTION**
  - The boundary of  $R = P \cap Q$  consists of alternating chains of vertices from  $P$  and  $Q$
  - Consecutive chains are joined at intersection points, where an edge of  $P$  intersects an edge of  $Q$



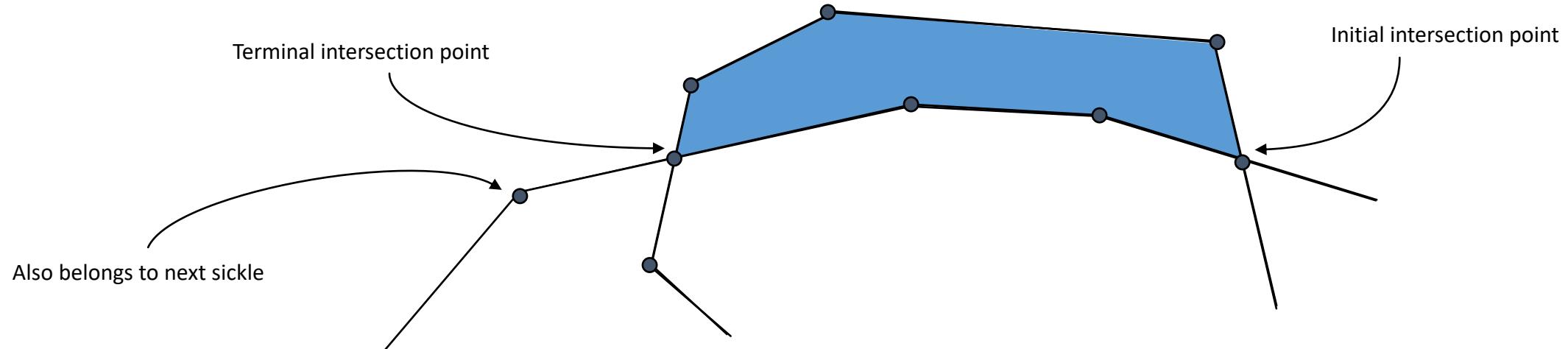
# INTERSECTION OF CONVEX POLYGONS

- REGIONS THAT ARE WITHIN ONE POLYGON, BUT NOT BOTH, ARE CALLED **SICKLES**.
- SICKLES ARE BOUNDED BY TWO CHAINS (INNER AND OUTER) OF VERTICES, ONE FROM EACH POLYGON, AND TERMINATED BY INTERSECTION POINTS AT EITHER END
- THE BOUNDARY OF  $P \cap Q$  IS FORMED FROM THE INNER CHAINS OF THE SICKLES



# SICKLES

- A VERTEX (OF P OR Q) BELONGS TO A SICKLE EITHER IF:
  1. It lies between the initial and terminal intersection points on either chain.
  2. It is the destination of the edge of the internal chain containing the terminal intersection point of the sickle.
    - Note that some vertices may belong to  $> 1$  sickles.



# ALGORITHM OVERVIEW

- INITIALLY RANDOM STARTING EDGES ARE SELECTED FROM EACH POLYGON
- THE ALGORITHM THEN HAS TWO PHASES, BOTH HANDLED BY SAME ADVANCE RULES:
  1. Get current edges  $\text{edge}(p_{i-1}p_i)$  and  $\text{edge}(q_{j-1}q_j)$  into same sickle
  2. Advance current edges  $\text{edge}(p_{i-1}p_i)$  and  $\text{edge}(q_{j-1}q_j)$  together through each sickle
    - Before either leaves the current sickle for the next sickle the terminal intersection point will be found
    - Add inner chain vertices to R



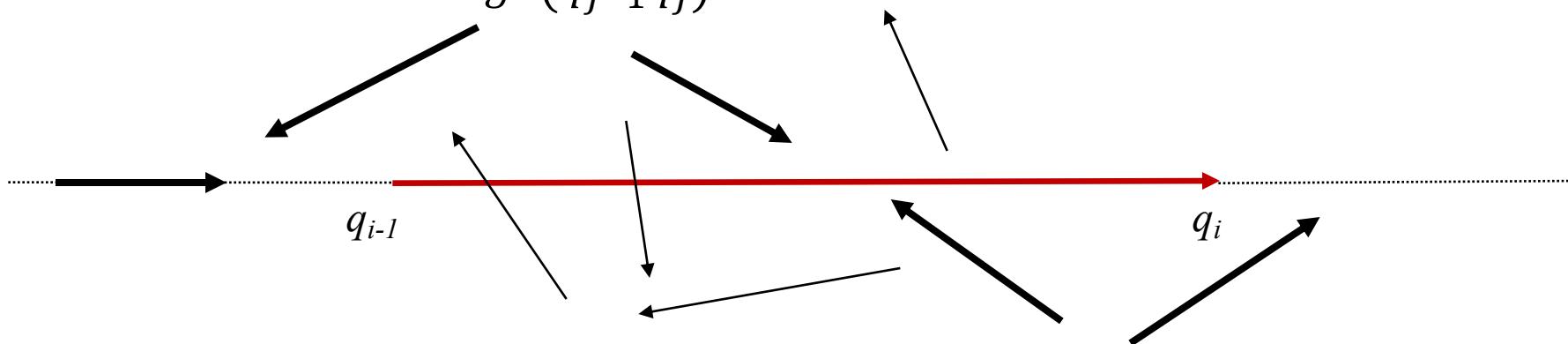
# PSEUDOCODE

```
1. Define current edges of P and Q:  
2.   edge( $p_{i-1}p_i$ ) and edge( $q_{j-1}q_j$ ), with destinations  $p_i$  and  $q_j$ ,  
     respectively.  
3.    $p_0 = p_N$  and  $q_0 = q_M$ , by definition.  
4. IntersectConvexPolygons(P, Q) /* Informal */  
5.   i = j = 1 /* Arbitrarily start with vertex 1 of each polygon. */  
6.   repeat  
7.     if ((w = edge( $p_{i-1}p_i$ )  $\cap$  edge( $q_{j-1}q_j$ ))  $\neq \emptyset$ ) then  
8.       add w to R  
9.       Select current edge to advance based on configuration.  
10.      if (selected edge is on inner chain) then  
11.        add selected edge destination to R  
12.        increment index of selected edge  
13.      end if  
14.      until iterations >  $3(N + M)$   
15.      if ( $R = \emptyset$ ) then /* Boundaries of P and Q don't intersect */  
16.        resolve special cases  $P \subseteq Q, Q \subseteq P, P \cap Q = \emptyset$   
17.    end function
```



# AIMING AT

- ADVANCE RULES ARE USED TO SELECT THE CURRENT EDGE TO ADVANCE.
  - The advance rules depend on the notion of “aiming at”.
  - The bold arrows aim at  $\text{edge}(q_{j-1}q_j)$ .

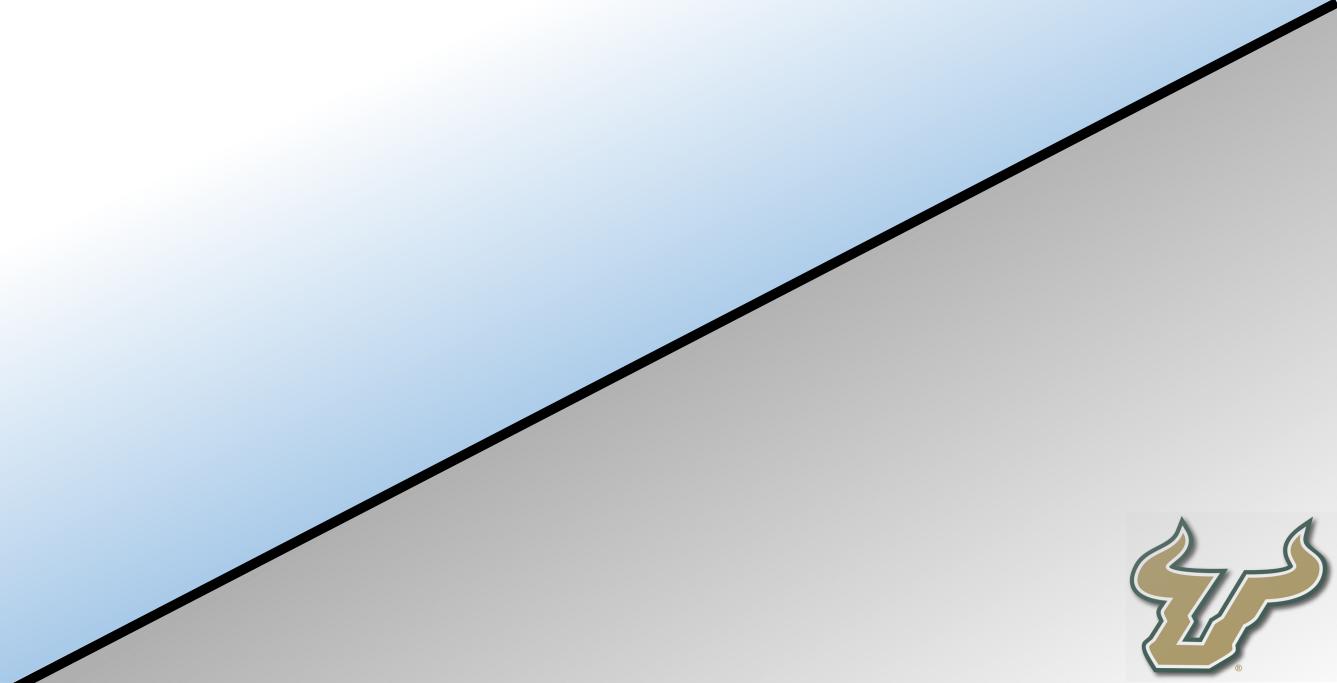


- IF THEY ARE NOT COLLINEAR,  
 $\text{edge}(p_{i-1}p_i)$  AIMS AT  $\text{edge}(q_{j-1}q_j)$  IF EITHER OF THESE CONDITIONS HOLD:
  1.  $\text{edge}(p_{i-1}p_i) \times \text{edge}(q_{j-1}q_j) > 0$  and  $p_i$  is in the left half plane of  $\text{edge}(q_{j-1}q_j)$ .
  2.  $\text{edge}(p_{i-1}p_i) \times \text{edge}(q_{j-1}q_j) < 0$  and  $p_i$  is in the right half plane of  $\text{edge}(q_{j-1}q_j)$ .
- IF THEY ARE COLLINEAR,  
 $\text{edge}(p_{i-1}p_i)$  AIMS AT  $\text{edge}(q_{j-1}q_j)$  IF  $p_i$  DOES NOT LIE BEYOND  $q_{j-1}$ .



# HALF-PLANE

- A HALF-SPACE IS EITHER OF THE TWO PARTS INTO WHICH A HYPERPLANE DIVIDES AN AFFINE SPACE
- A HALF-SPACE CAN BE EITHER OPEN OR CLOSED
- IN 2D, A HALF-SPACE IS CALLED A HALF-PLANE



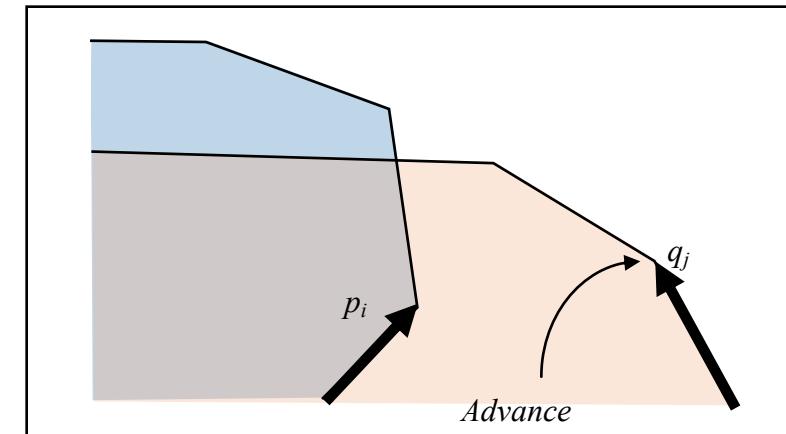
# ADVANCE RULES, OVERALL INTENT

- ADVANCE RULES ARE USED TO SELECT THE CURRENT EDGE TO ADVANCE
- O'ROURKE:
  - ‘If  $\text{edge}(p_{i-1}p_i)$  aims at the line containing  $\text{edge}(q_{j-1}q_j)$  but does not cross it, we want to advance  $\text{edge}(p_{i-1}p_i)$  to close in on a possible intersection point with  $\text{edge}(q_{j-1}q_j)$ .’
- LASZLO:
  - ‘The advance rules are designed so that the intersection point which should be found next is not skipped over. They distinguish between the current edge which may contain the next intersection point and the current edge which cannot possibly contain the next intersection point; the latter edge is advanced.’



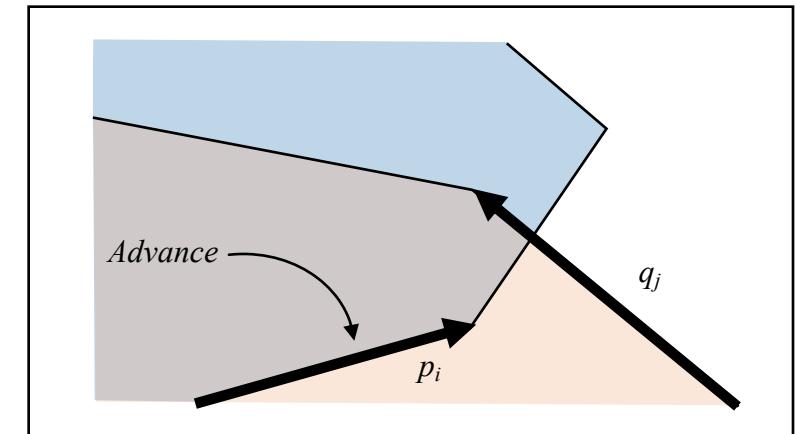
## ADVANCE RULES (BY LASZLO)

- CASE I –  $\text{edge}(p_{i-1}p_i)$  AND  $\text{edge}(q_{j-1}q_j)$  AIM AT EACH OTHER:
  - Advance whichever edge is to the right of the other
- ASSUME  $\text{edge}(q_{j-1}q_j)$  IS TO THE RIGHT; THEN THE NEXT INTERSECTION POINT CANNOT LIE ON  $\text{edge}(q_{j-1}q_j)$  BECAUSE  $q_j$  IS OUTSIDE THE INTERSECTION POLYGON  $R$ .



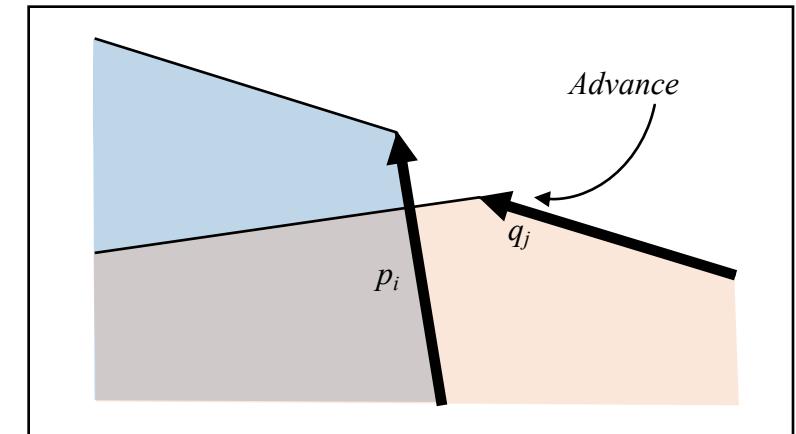
## ADVANCE RULES (BY LASZLO)

- CASE 2 –  $\text{edge}(p_{i-1}p_i)$  AIMS AT  $\text{edge}(q_{j-1}q_j)$  BUT NOT VICE VERSA
  - Add  $p_i$  to R if  $p_i$  is left of  $\text{edge}(q_{j-1}q_j)$ , then advance  $p_i$
- $\text{edge}(p_{i-1}p_i)$  CANNOT CONTAIN THE NEXT INTERSECTION POINT. IN THE FIGURE,  $p_i$  **IS NOT** RIGHT OF  $\text{edge}(q_{j-1}q_j)$ , AND **IS ADDED TO R**



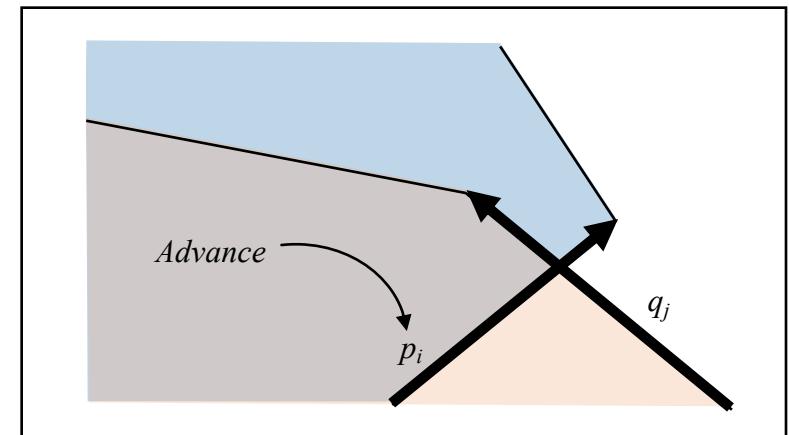
## ADVANCE RULES (BY LASZLO)

- CASE 3 –  $\text{edge}(q_{j-1}q_j)$  AIMS AT  $\text{edge}(p_{i-1}p_i)$  BUT NOT VICE VERSA:
  - Add  $q_j$  to  $R$  if  $q_j$  is left of  $\text{edge}(p_{i-1}p_i)$ , then advance  $q_j$
- $\text{edge}(q_{j-1}q_j)$  CANNOT CONTAIN THE NEXT INTERSECTION POINT. IN THE FIGURE,  $q_j$  IS RIGHT OF  $\text{edge}(p_{i-1}p_i)$ , AND IS NOT ADDED TO  $R$ .

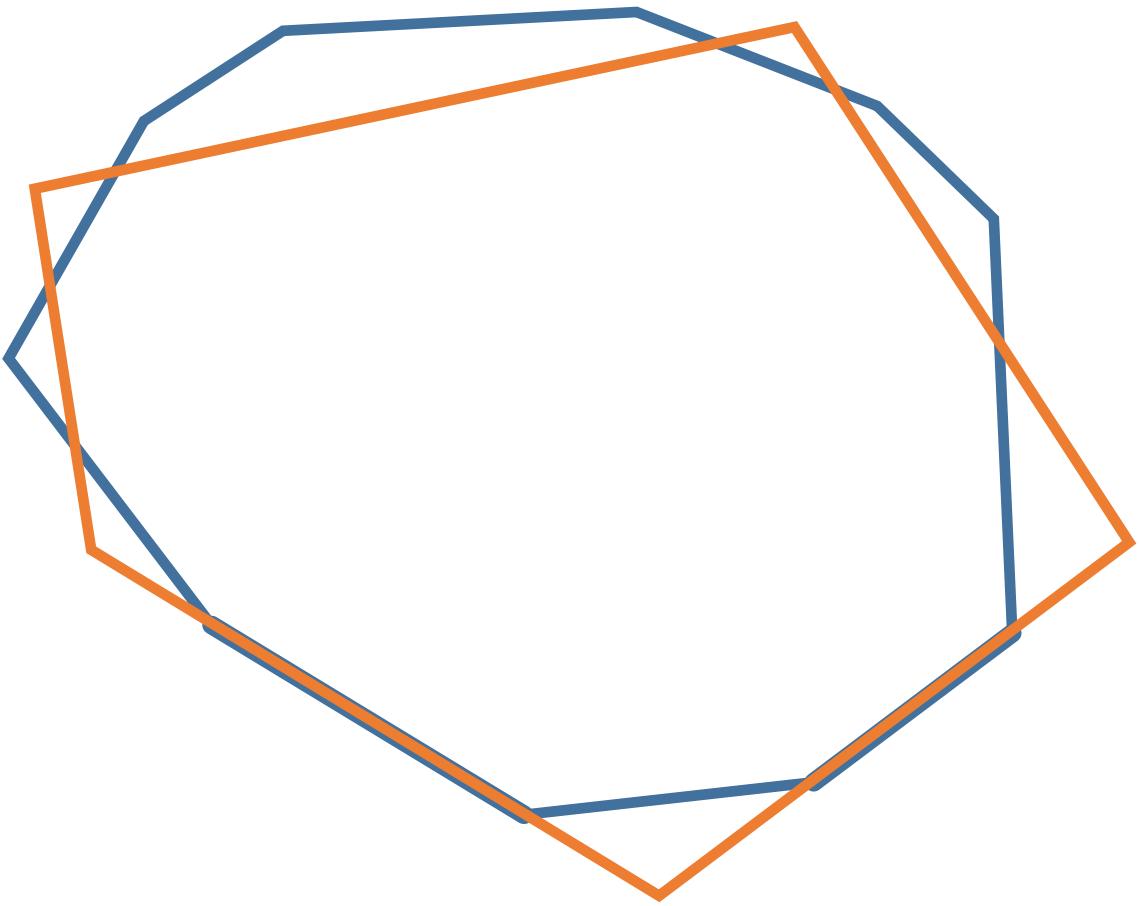


## ADVANCE RULES (BY LASZLO)

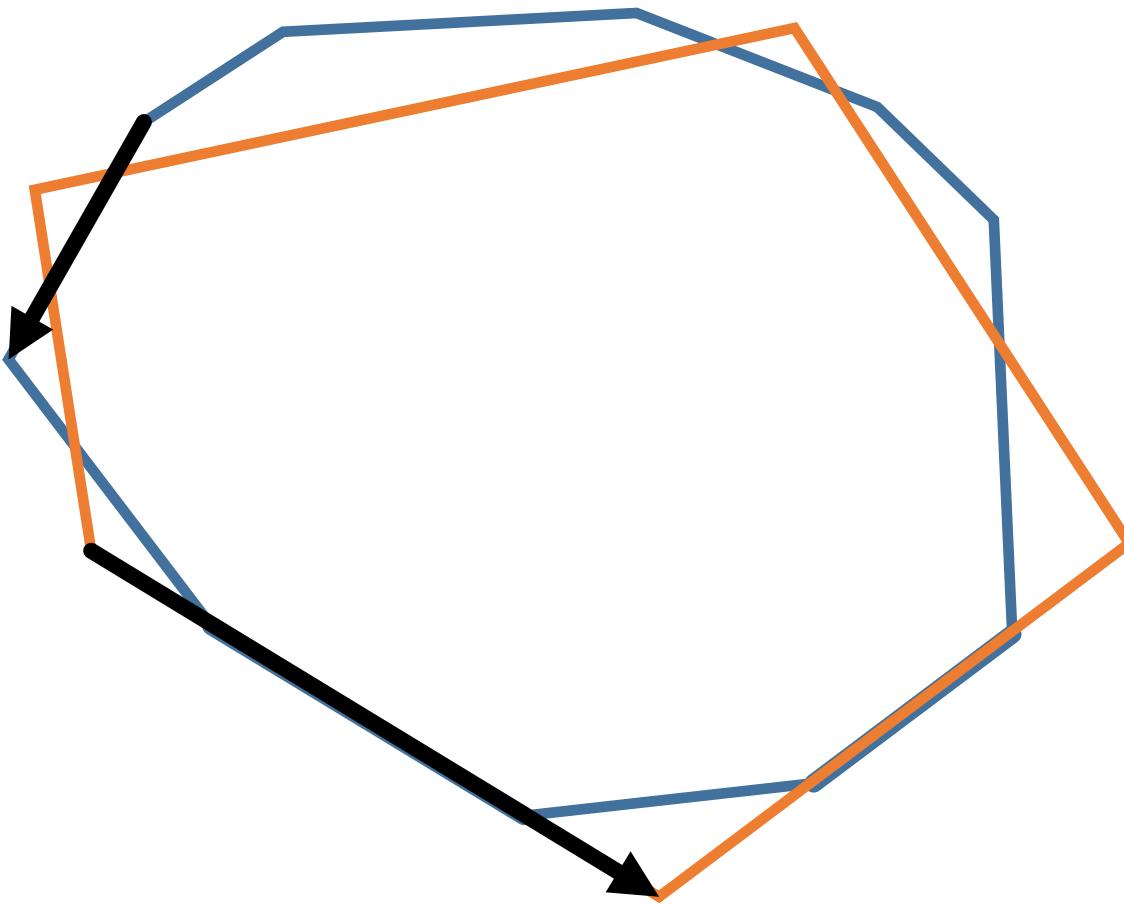
- CASE 4 –  $\text{edge}(p_{i-1}p_i)$  AND  $\text{edge}(q_{j-1}q_j)$  DO NOT AIM AT EACH OTHER:
  - Advance whichever edge ends to the right of the other.



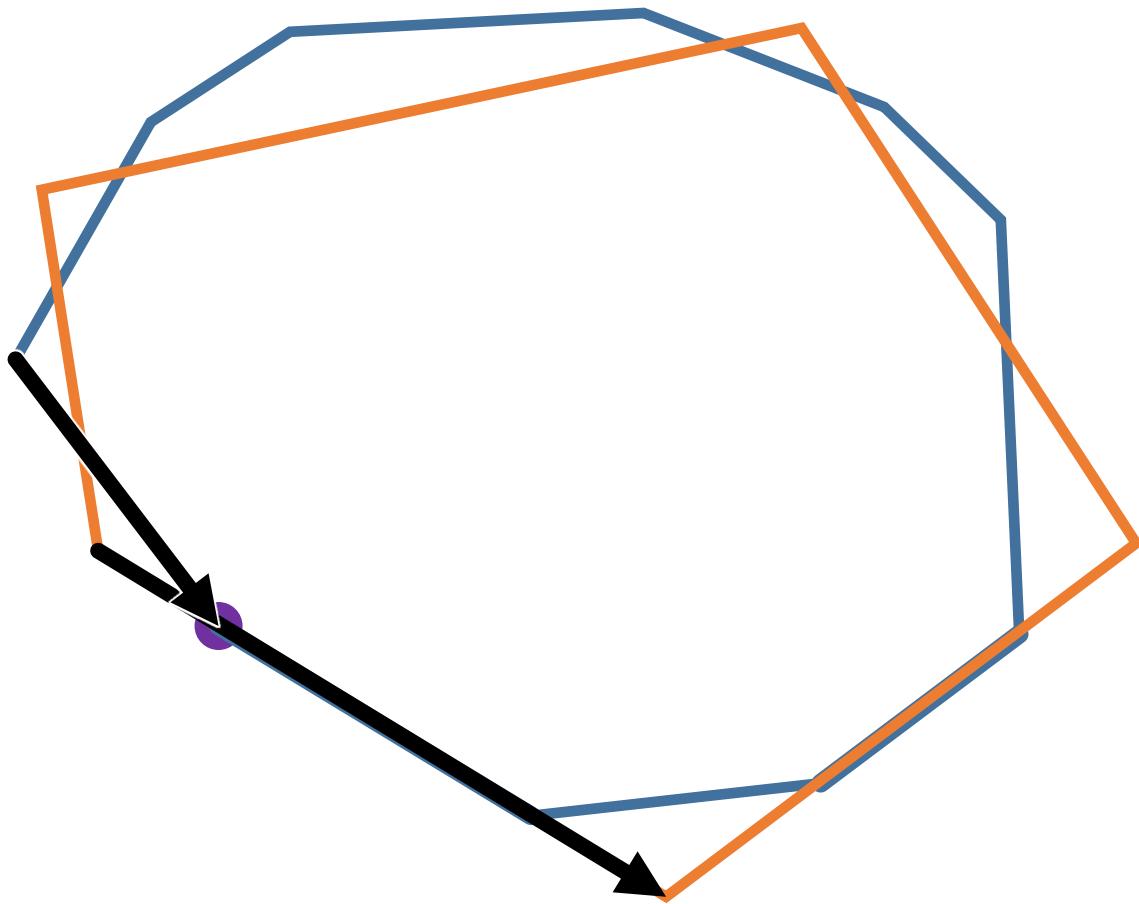
# INTERSECTION EXAMPLE



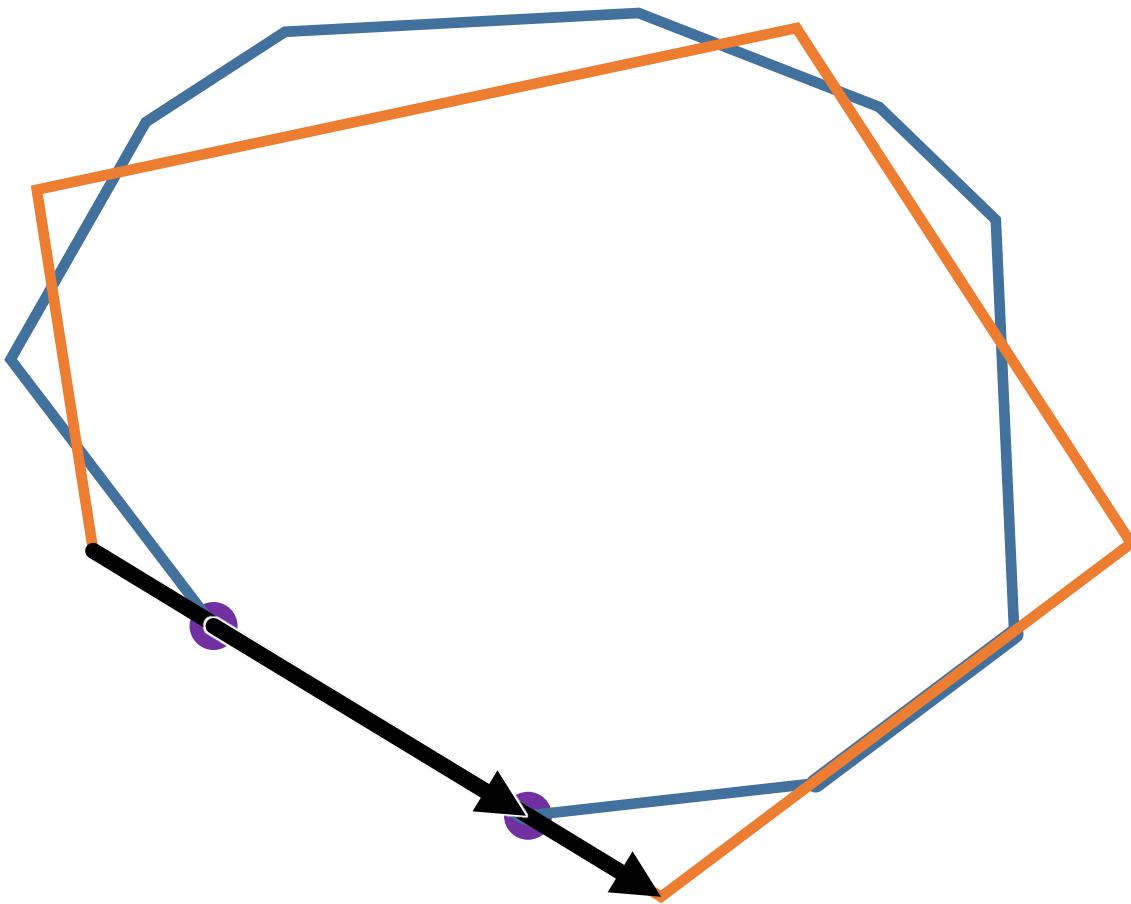
# INTERSECTION EXAMPLE



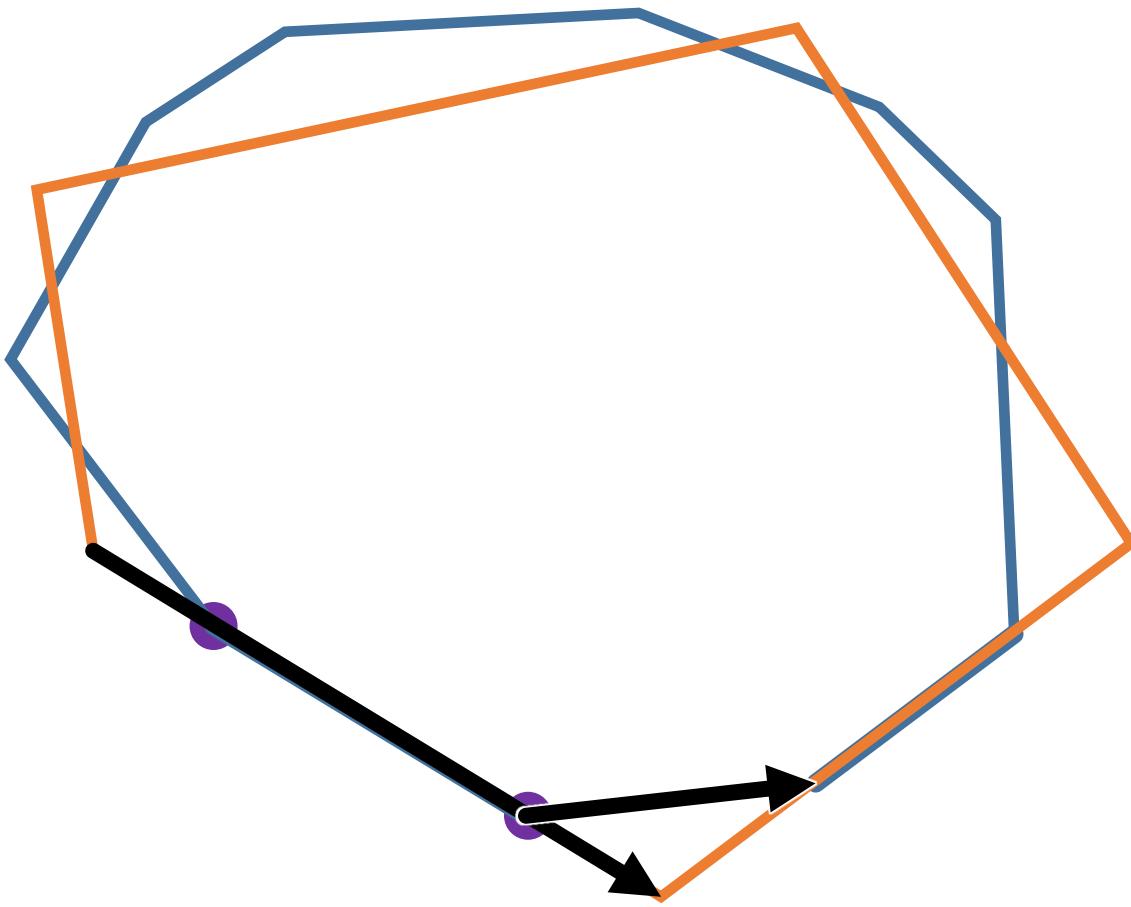
# INTERSECTION EXAMPLE



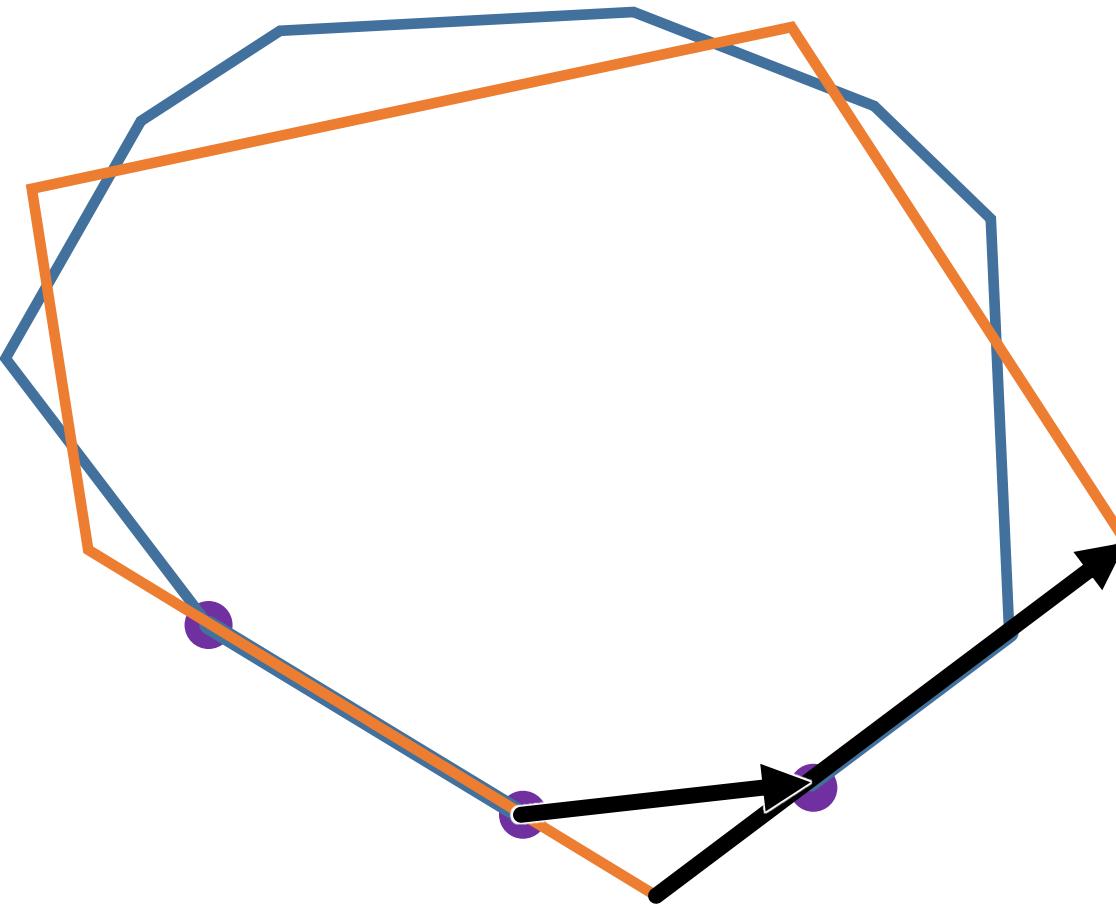
# INTERSECTION EXAMPLE



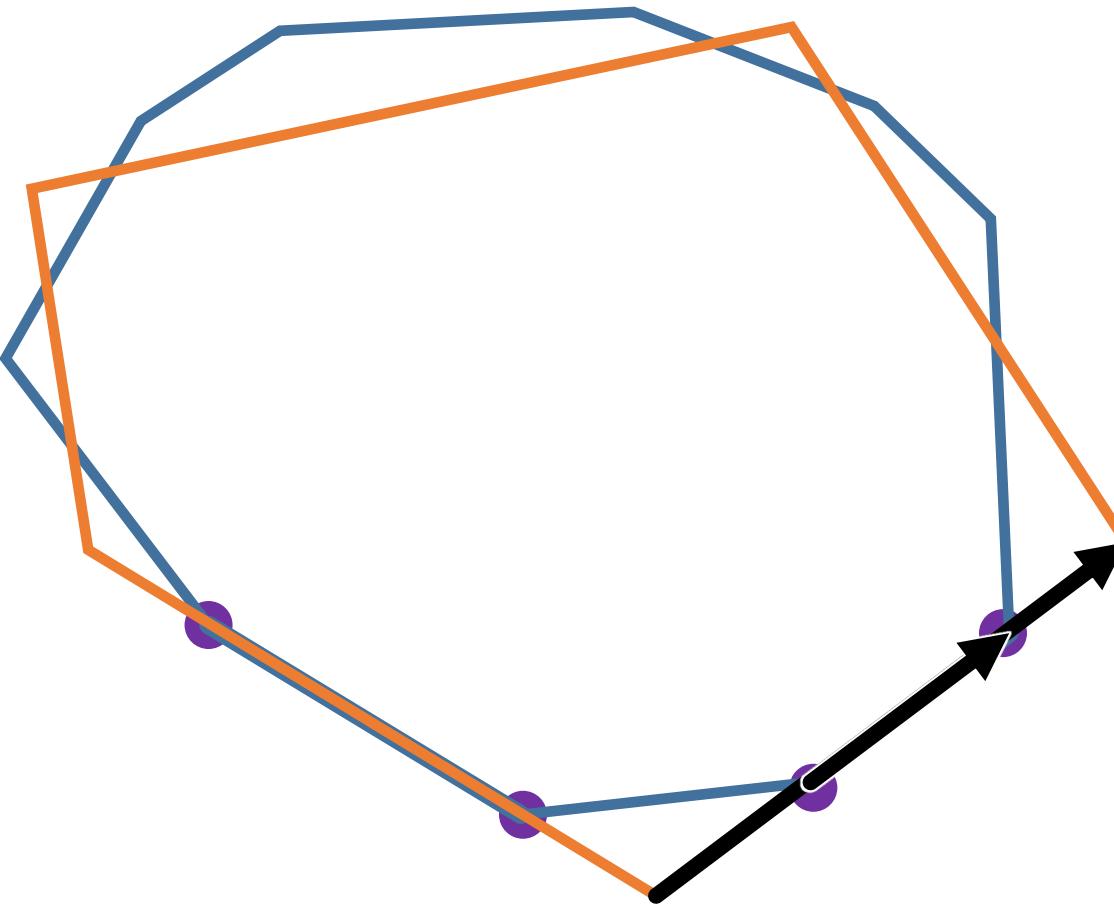
# INTERSECTION EXAMPLE



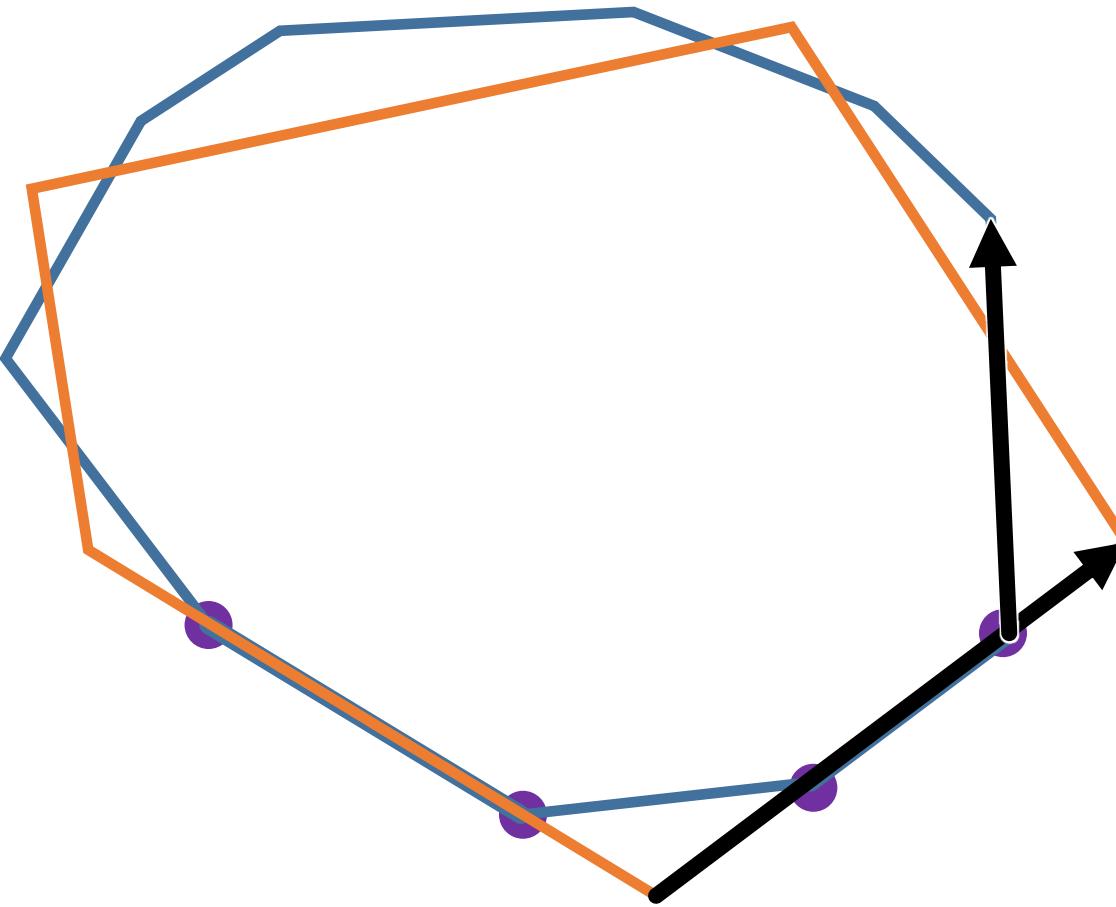
# INTERSECTION EXAMPLE



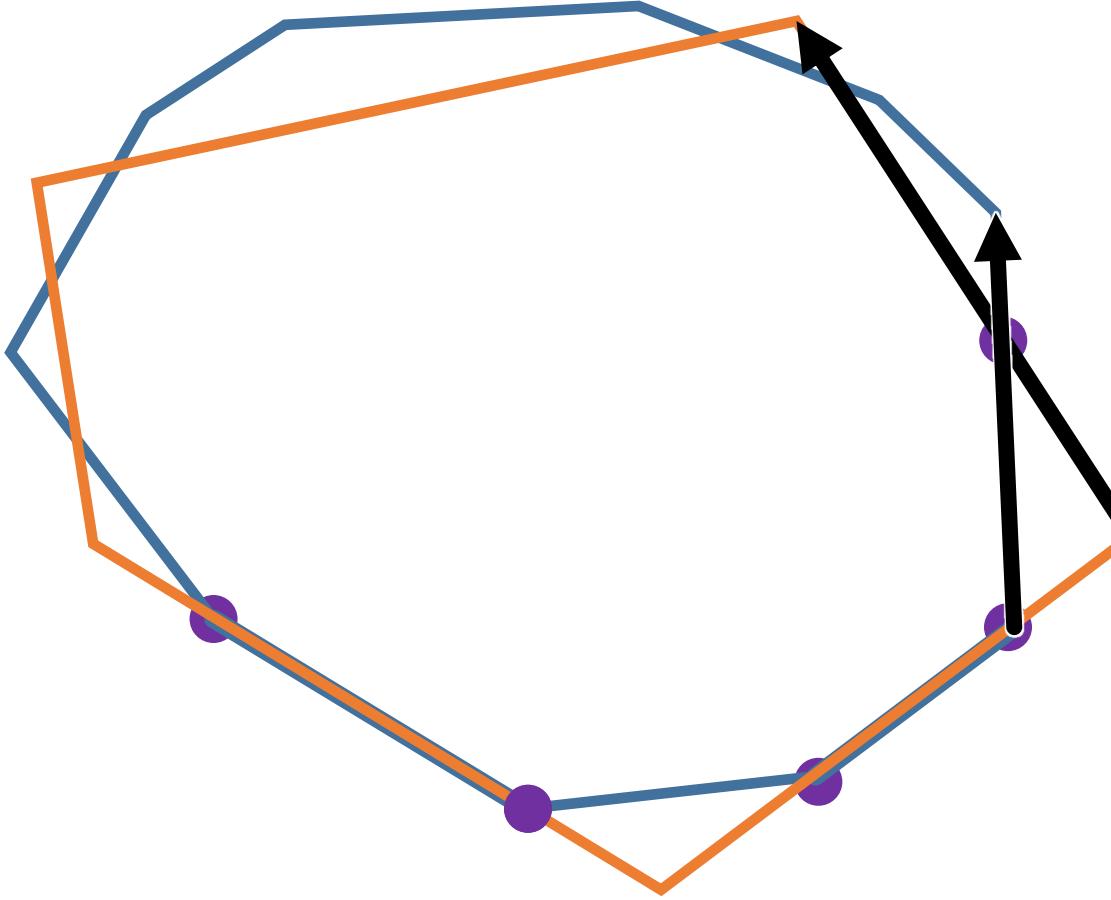
# INTERSECTION EXAMPLE



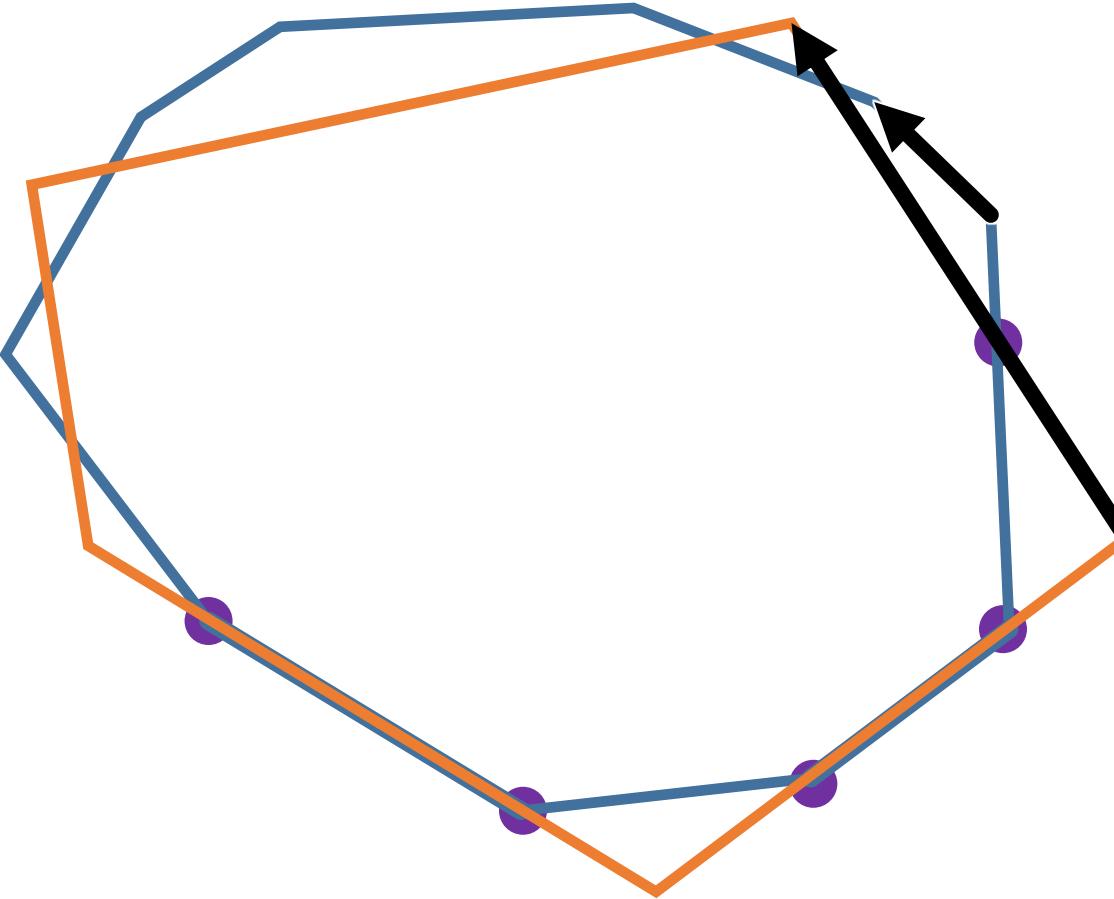
# INTERSECTION EXAMPLE



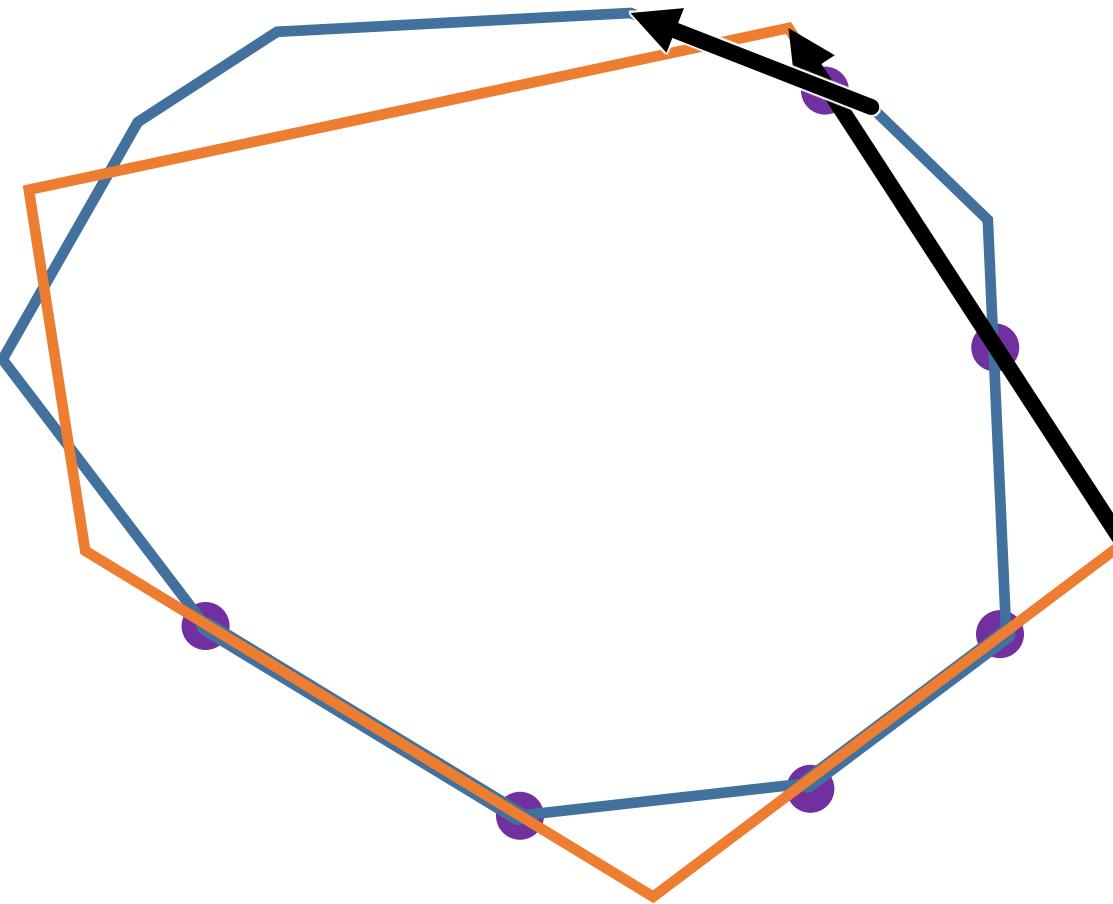
# INTERSECTION EXAMPLE



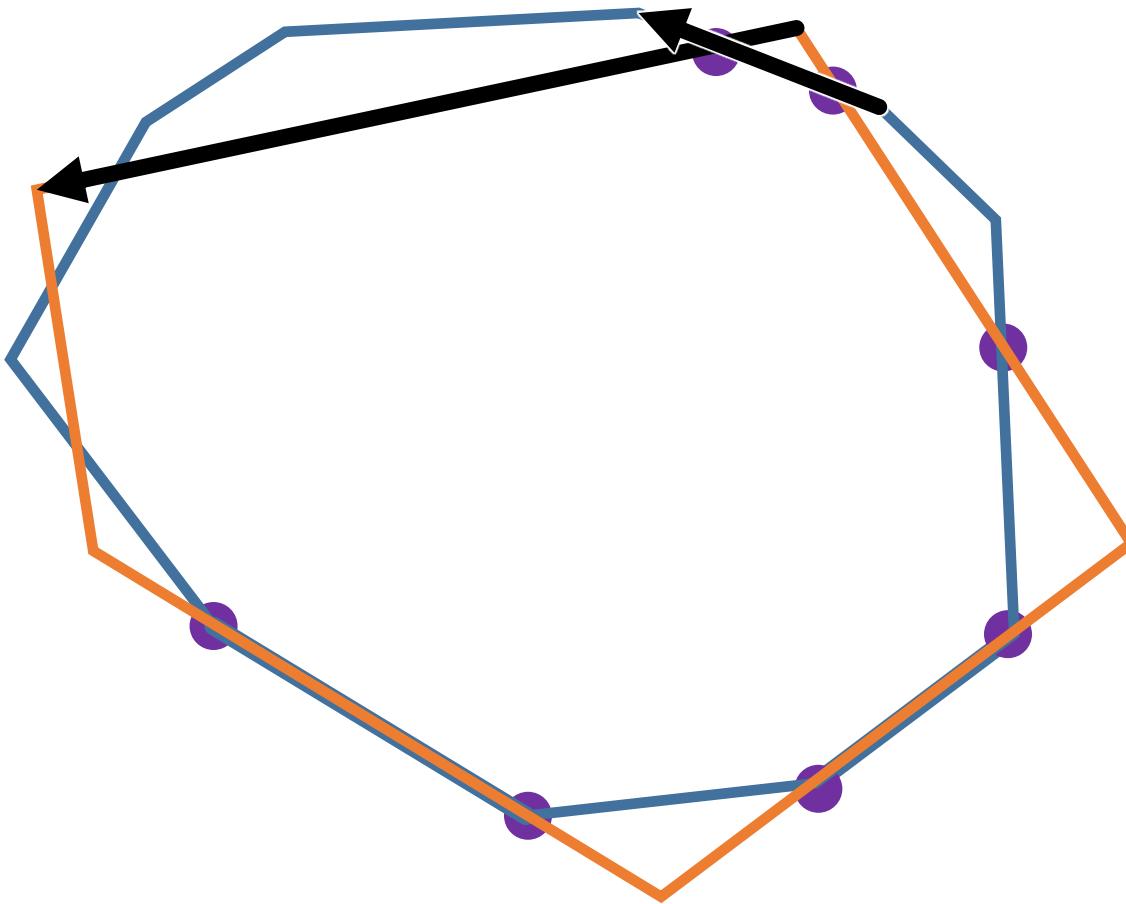
# INTERSECTION EXAMPLE



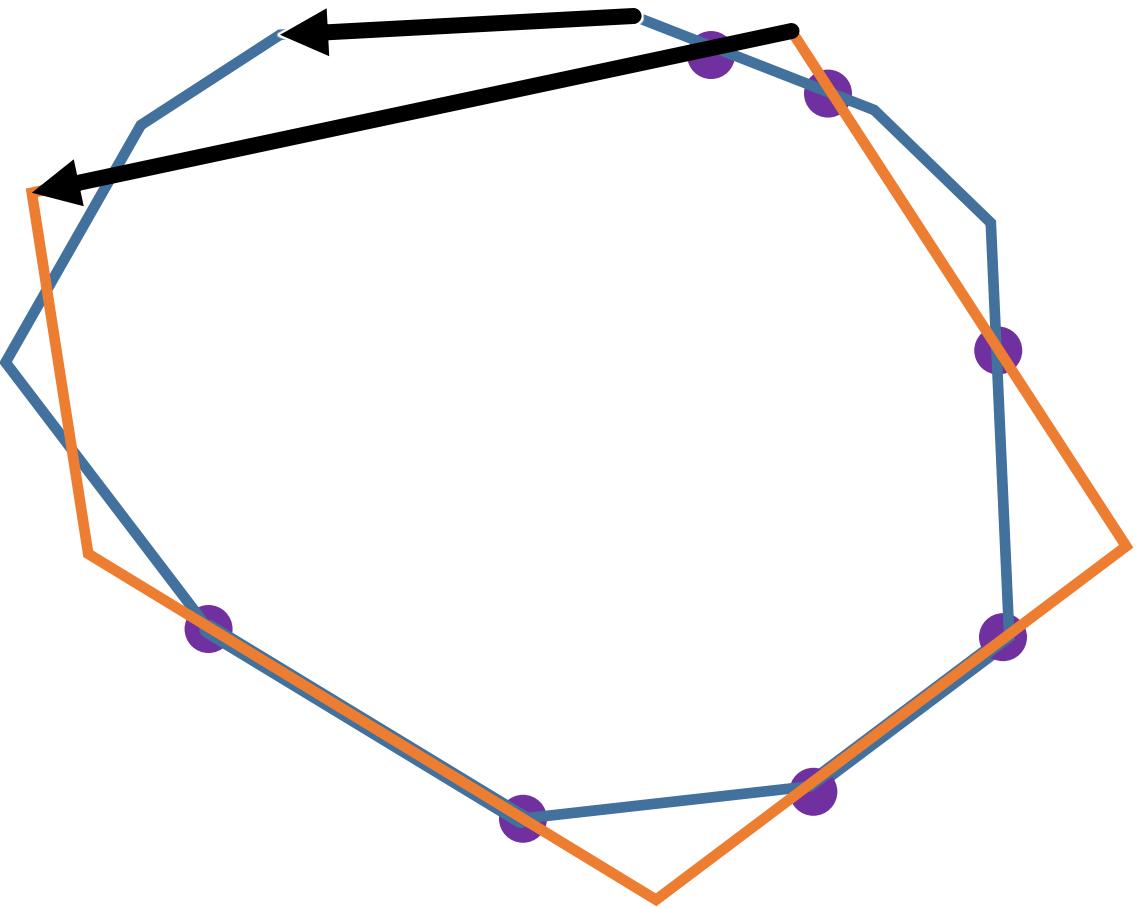
# INTERSECTION EXAMPLE



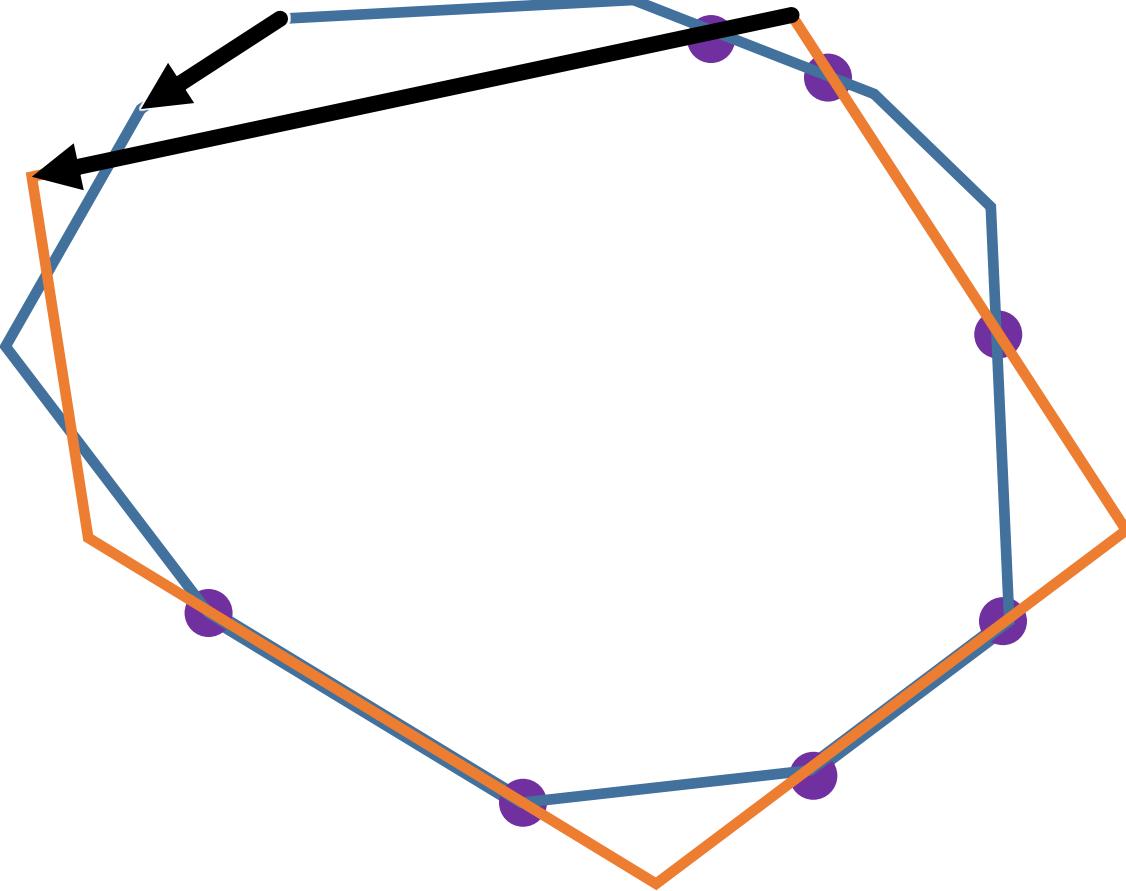
# INTERSECTION EXAMPLE



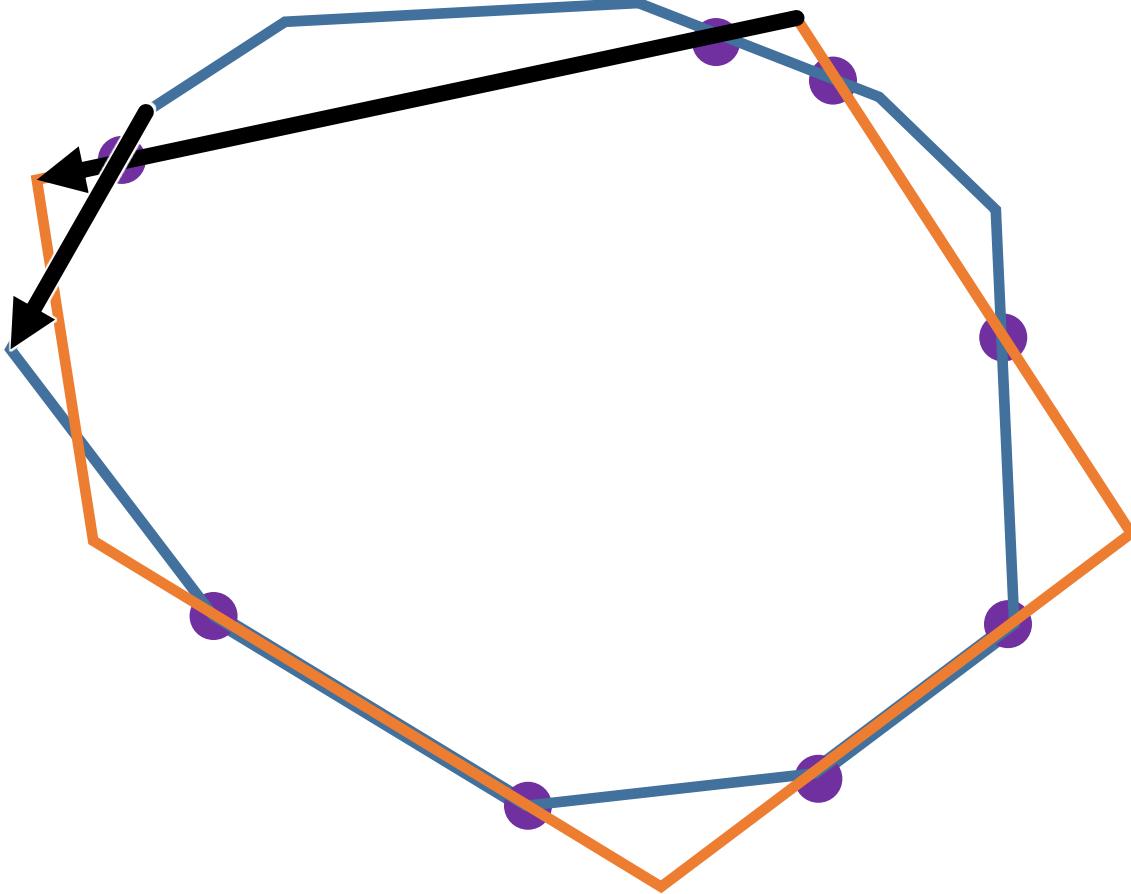
# INTERSECTION EXAMPLE



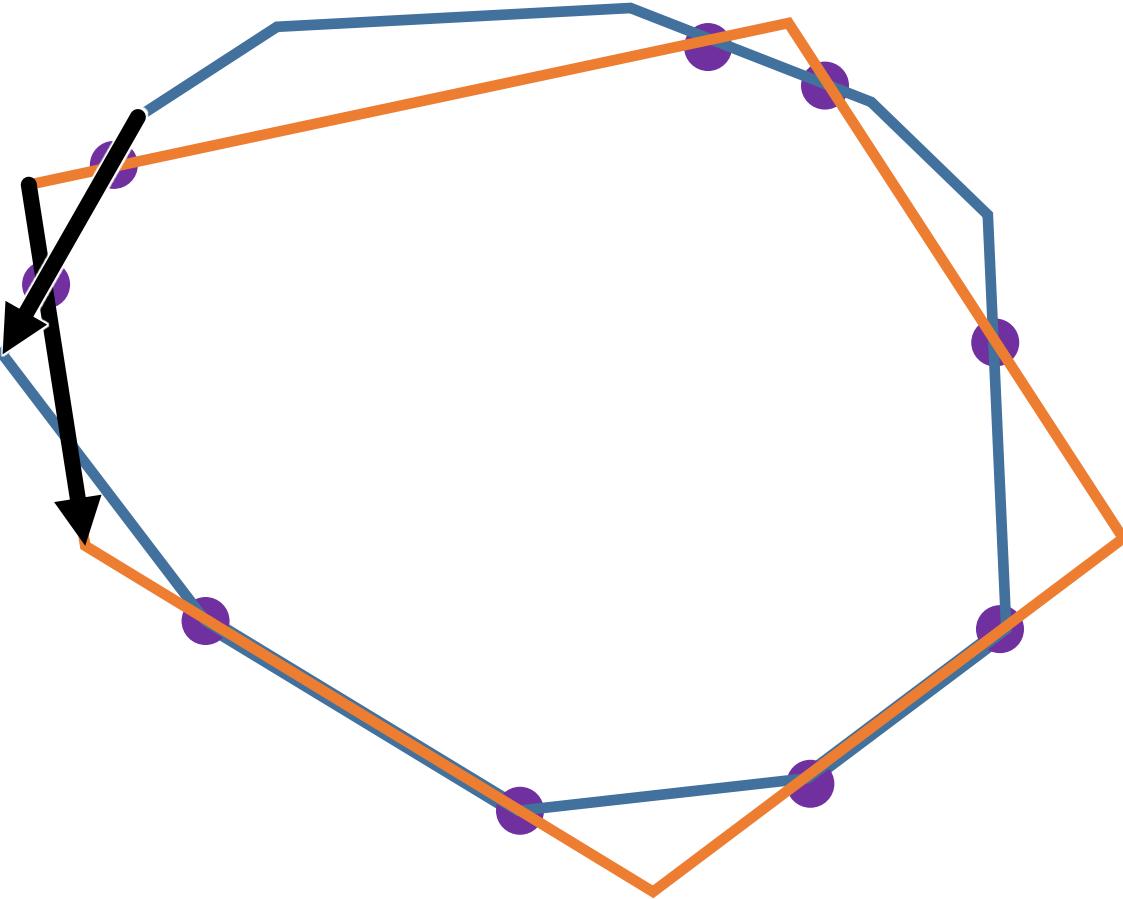
# INTERSECTION EXAMPLE



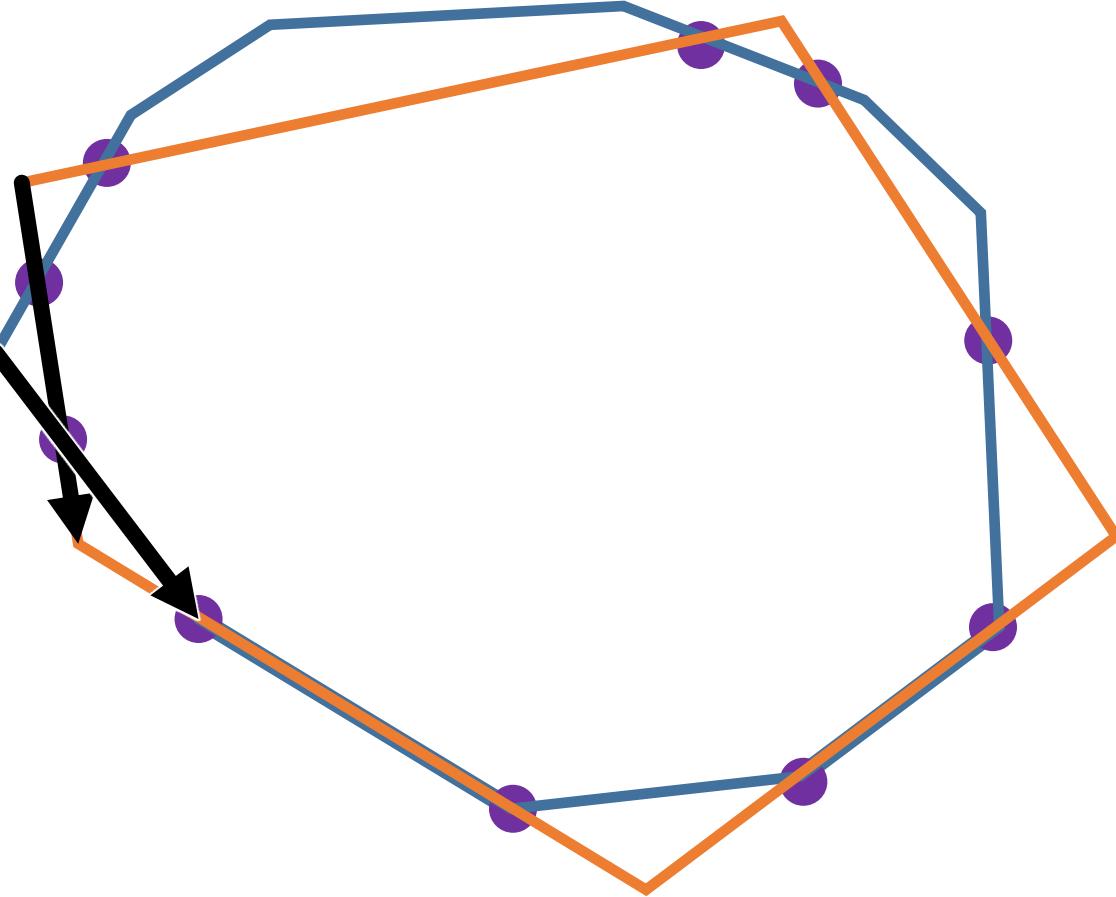
# INTERSECTION EXAMPLE



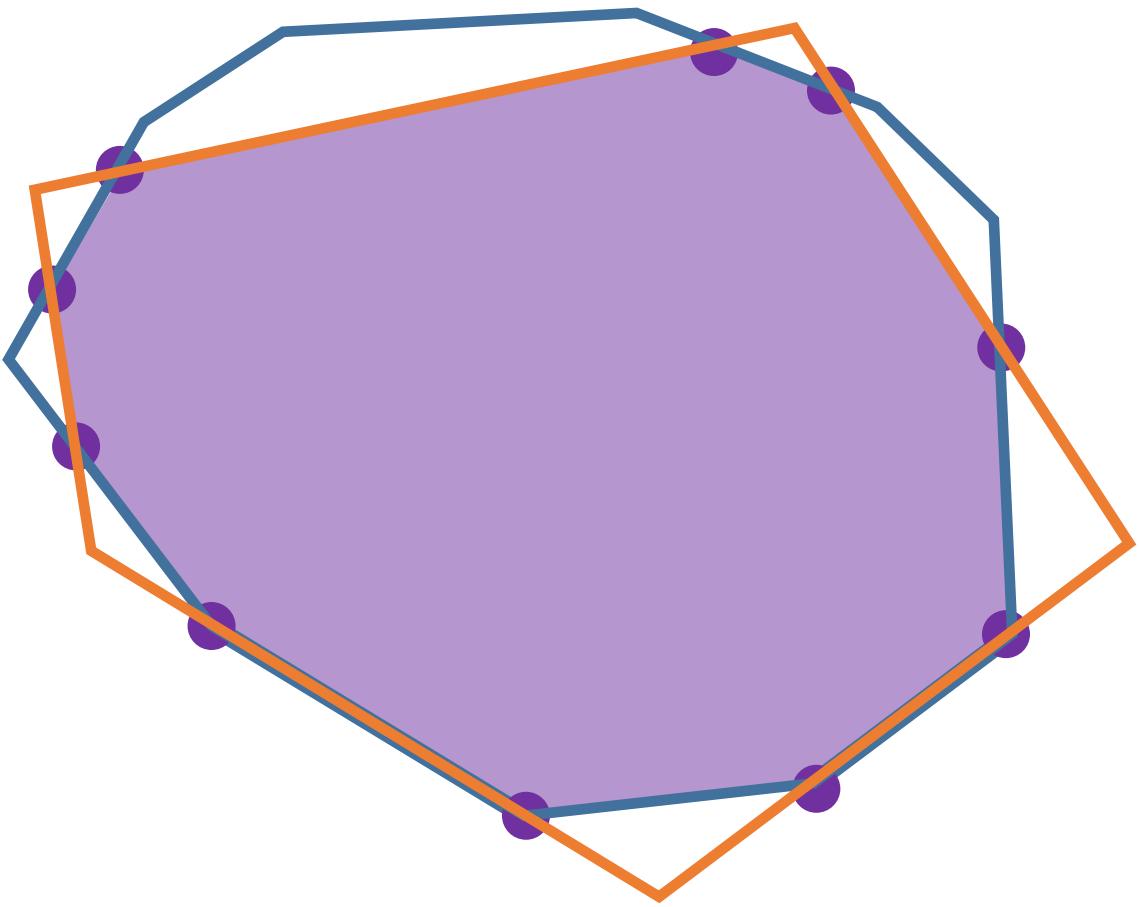
# INTERSECTION EXAMPLE



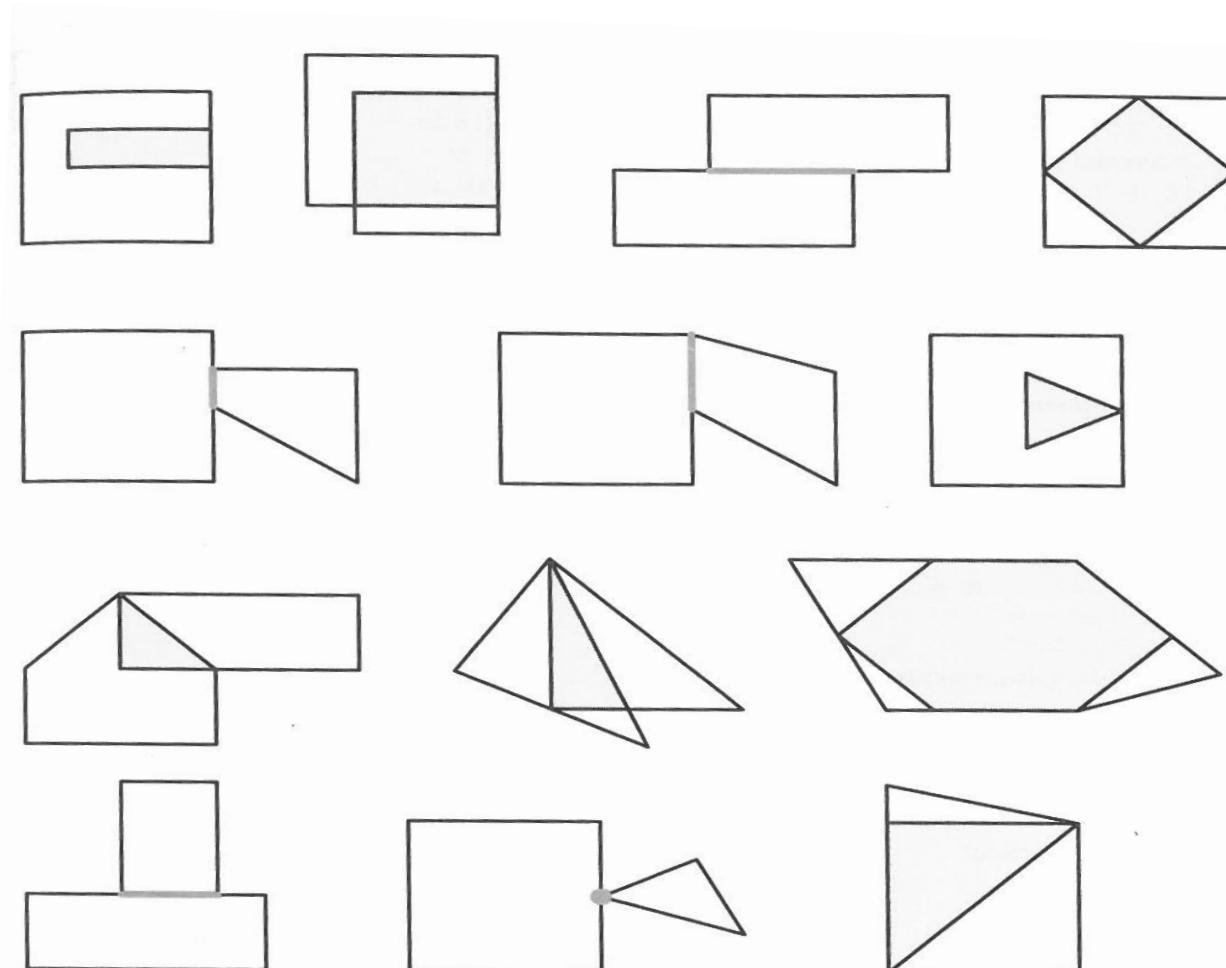
# INTERSECTION EXAMPLE



# INTERSECTION EXAMPLE



# EXAMPLE DEGENERATE INTERSECTIONS



# PROOF OF CORRECTNESS

- CORRECTNESS FOLLOWS FROM TWO ASSERTIONS:
  - I. If current edges p and q belong to the same sickle, then the next intersection point, at which the sickle terminates, will be found next.
  2. If the boundaries of P and Q intersect, current edges p and q will cross at some intersection point after no more than  $2(N + M)$  iterations (advances).



## ANALYSIS

- $\leq 2(N + M)$  ITERATIONS (ADVANCES) SUFFICE TO FIND THE FIRST INTERSECTION POINT AND GET P AND Q INTO THE SAME SICKLE.
- THEREAFTER,  $\leq (N + M)$  ADDITIONAL ITERATIONS WILL PRODUCE  $R$ .

⇒ THE ALGORITHM REQUIRES  $O(N + M)$ , I.E. LINEAR TIME.

