CSN-523: Computational Geometry
(Spring 2016-2017)

Tutorial-6 Date: Mar 29, 2017

Name: Batch (B1/B2): Enrolment Number:

---

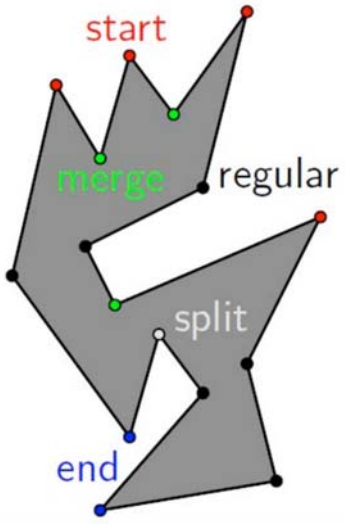
1. What are turn vertices in a polygon? Elaborate different types of turn vertices. What is general way to handle them in monotone partitioning?
   Solution:
   Turn Vertex: A vertex where the direction in which we walk switches from downward to upward or vice versa.

   Different types of turn vertex:

   • Start Vertex - its two neighbors lie below it and the interior angle < 180°
   • End Vertex - its two neighbors lie above it and the interior angle < 180°
   • Split Vertex - its two neighbors lie below it and the interior angle > 180°
   • Merge Vertex - its two neighbors lie above it and the interior angle > 180°

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 $\mathcal{T}$.



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

HANDLEMERGEVERTEX(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 $\mathcal{T}$.
4. Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
5. **if** $helper(e_j)$ is a merge vertex
6. **then** Insert the diagonal connecting v_i to $helper(e_j)$ in $\mathcal{D}$.
7. $helper(e_j) \leftarrow v_i$

2. What are the data structures used in monotone partitioning?
Solution:

A doubly connected edge list DCEL is used to store the monotone polygons. A DCEL has the following properties:
1. Any vertex has access to one edge that it is on.
2. Any face has access to one edge that surrounds it.
3. Any edge has access to 2 vertices and the face to it's right and the face to it's left.

A priority queue Q on the vertices of P, using their y-coordinates as priority. If two points have the same y-coordinates, the one with smaller x has higher priority.

Status structure T, is a Balanced Binary Search Tree that stores the edges. It represents the current portion of the polygon the sweep line is currently crossing is needed. T is a BBST, sorted based on relative x-position of edges.

3. Explain the steps of MakeMonotone(P) routine for partitioning a polygon P into monotone pieces.
Solution:

Input: A simple polygon P stored in a doubly-connected edge list D
Output: A partitioning of P into monotone subpolygons stored in D

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

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 $\mathcal{T}$.

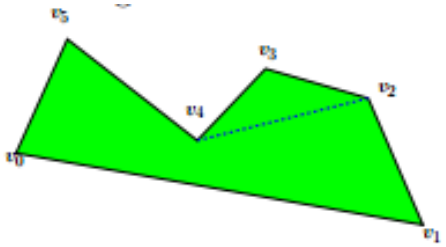
HANDLEREGULARVERTEX(v_i)
1. **if** the interior of $\mathcal{P}$ lies to the right of v_i
2. **then if** $helper(e_{i-1})$ is a merge vertex
3. **then** Insert the diagonal connecting v_i to $helper(e_{i-1})$ in $\mathcal{D}$.
4. Delete e_{i-1} from $\mathcal{T}$.
5. Insert e_i in $\mathcal{T}$ and set $helper(e_i)$ 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 $\mathcal{D}$.

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

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

HANDLEMERGEVERTEX(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 $\mathcal{T}$.
4. Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
5. **if** $helper(e_j)$ is a merge vertex
6. **then** Insert the diagonal connecting v_i to $helper(e_j)$ in $\mathcal{D}$.
7. $helper(e_j) \leftarrow v_i$

4. Prove that every strictly convex vertex of Monotone Mountain that is not on the base is ear tip.
Solution:
Refer to Book and slides.
If v is a convex vertex than its' two neighbors must see each other inside the polygon. As such the diagonal they define is legal, and v is an ear, as the other chain is a single segment that cannot intersect this diagonal.



5. Let A and B be two convex polygons, represented by their vertices. Write an algorithm to determine whether A lies completely inside B.
   Solution:
   1. Find convex hull of B
   2. Find convex hull of points A & B
   3. If both set have same points then A lies completely inside B
      else
      A does not lies completely inside B

6. Suppose that you run Graham's algorithm to compute the convex hull of a set of points S. Now, you want to add one point to S and compute the new convex hull. Describe how to do this in linear time.
   Solution:
   Use incremental convex hull algorithm as described in slides.

   Let p_i be a point to be added to the convex hull P.

   If p lies in the exterior of the polygon defined by P:

   - Compute the points p_l (top most) and pr (bottom) defining the supporting lines from pi to the polygon

   - Replace the chain p_l , . . . , p_r in P with the chain p_l , p_i , p_r

7. For a polygon with r reflex vertices, what is the lower bound and upper bound on the fewest number of convex pieces it can be partitioned?
   Solution:

   Theorem (Chazelle): Let Φ be the fewest number of convex pieces into which a polygon may be partitioned. For a polygon of 'r' reflex vertices:

$$\left\lceil \frac{r}{2} \right\rceil + 1 \leq \Phi \leq r + 1$$

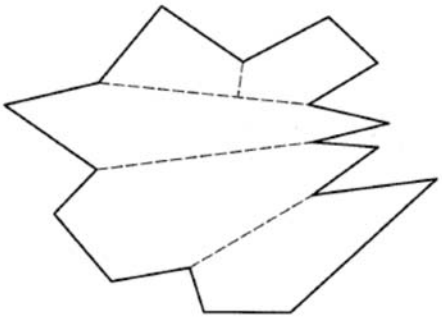


FIGURE 2.11 $\lceil r/2 \rceil + 1$ convex pieces: $r = 7$; 5 pieces.

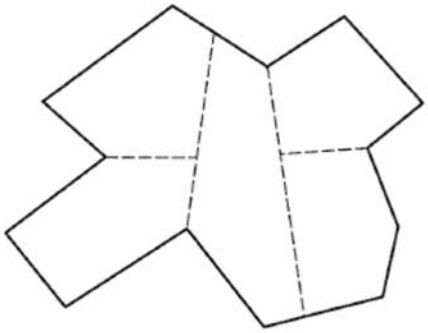


FIGURE 2.10 $r + 1$ convex pieces: $r = 4$; 5 pieces.

Lower bound: Must eliminate all reflex vertices. Single segment resolves at most 2 reflex angles.

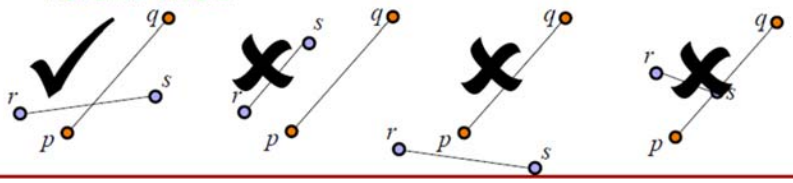
Upper bound: Bisect each reflex angle.

8. Explain the difference between intersection and proper intersection of two given line segments pq and rs.
Solution:

Proper Intersection:

Line segments $\overline{pq}$ and $\overline{rs}$, *intersect properly* if they intersect in their interior:

- ProperIntersect(p , q , r , s):
  - If(Collinear(p , q , r) OR Collinear(p , q , s))
    return FALSE
  - If(Collinear(r , s , p) OR Collinear(r , s , q))
    return FALSE
  - If(Left(p , q , r)==Left(p , q , s)) return FALSE
  - If(Left(r , s , p)==Left(r , s , q)) return FALSE
  - return TRUE



INTERSECTION:

Line segments $\overline{pq}$ and $\overline{rs}$, *intersect* if either they abut or they intersect properly:

- Intersect(p , q , r , s):
  - If(ProperIntersect(p , q , r , s)) return TRUE
  - If(Between(p , q , r)) return TRUE
  - If(Between(p , q , s)) return TRUE
  - If (Between(r , s , p)) return TRUE
  - If(Between(r , s , q)) return TRUE
  - return FALSE