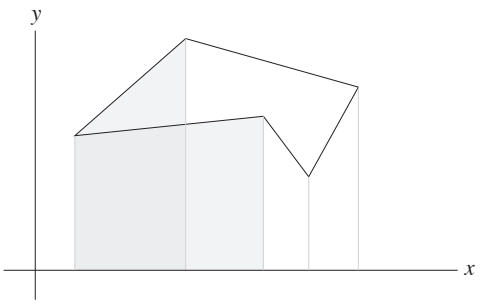
CS 763 Fall 2020 due Thursday September 17
A. Lubiw

ASSIGNMENT 1

ACKNOWLEDGE YOUR SOURCES.

1. [10 marks] The area of a simple polygon can be found by triangulating it and then adding up the areas of the triangles. This gives a linear time algorithm, but only via Chazelle's difficult linear time triangulation algorithm.

   (a) [4 marks] Use the idea shown in following figure (which shows the areas below some of the polygon edges) to give a straight-forward linear time algorithm to compute the area of a simple polygon. Assume the polygon vertices $p_1, \ldots, p_n$ are given in clockwise order, and that $p_i = (x_i, y_i)$. Make sure you justify correctness and runtime.

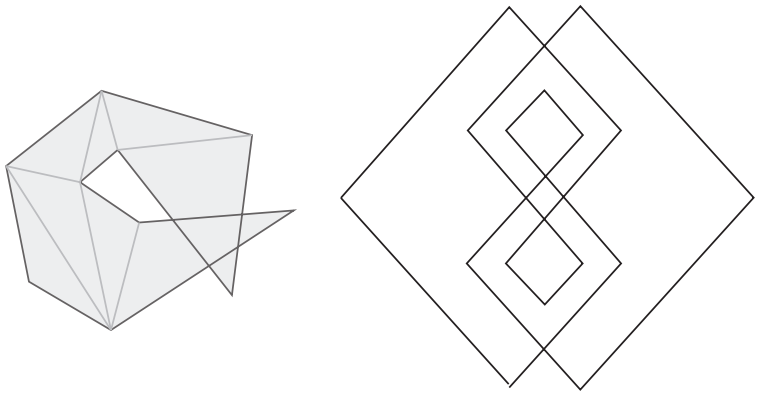


The same approach is sensible for a larger class than just simple polygons. An *overlapping polygon* is a union of triangles joined edge-to-edge, see part (a) of the figure below. More formally, we define a *triangulated overlapping polygon* recursively as follows: (1) a triangle is a triangulated overlapping polygon; (2) if P is a triangulated overlapping polygon of t triangles and $e = (u, v)$ is a boundary edge, then an overlapping polygon of $t + 1$ triangles is formed by adding a triangle to the outside of edge e so long as the sum of the angles of the triangles at each of u and v is less than 360°. Define an *overlapping polygon* to be the boundary of a triangulated overlapping polygon. Overlapping polygons arise, for example, when unfolding a convex polyhedron, see `http://jeffe.cs.illinois.edu/open/unfold.html`.

Define the *area* of a triangulated overlapping polygon to be the sum of the areas of its triangles—even if the triangles overlap.

   (c) [3 marks] Suppose you are given an overlapping polygon by its boundary (in clockwise order), but you are not given its triangulation. (In fact the best known algorithm to find a triangulation takes $O(n^3)$ time.) Prove that your area algorithm from part (a) still works. Hint: Use the recursive definition of a triangulated overlapping polygon.

   (d) [3 marks] Give an example of a non-simple polygon where your area algorithm does not give a sensible answer. Explain why.



(a) (b)

(e) [for fun, not for marks, no need to hand it in] Show that the overlapping polygon in figure (b) above has two quite different triangulations. By your result above, the area is the same either way.

2. [10 marks] One way of evaluating algorithms is to see how they behave on "random" inputs. This may be difficult to analyze theoretically, but can be done experimentally by generating random inputs and running the algorithm on them. To apply this idea to algorithms whose input is a simple polygon we would like to generate random simple polygons efficiently. One definition of a random simple polygon on n vertices is to choose uniformly from the finite set of simple polygons whose vertices are points in the $n \times n$ grid, i.e., such that each coordinate is a natural number in $[1..n]$.

   Consider the following procedure:

   1. Choose n points at random in the $n \times n$ grid.
   2. Generate a random simple polygon whose vertices are the chosen points.

   One issue is that different point sets may have different numbers of simple polygons defined on them.

   (a) [2 marks] Show that there are point sets with only one simple polygon. You must find an infinite class, i.e., describe such a point set for each n, or at least for an infinite set of n's (e.g., n even, or n a power of 2).

   (b) [2 marks] Show that there are point sets with an exponential number of simple polygons (Again, describe an infinite class.)

   Another issue with the above plan is how to efficiently generate a random simple polygon on a given point set. In fact, this is an open problem. What is wrong with the following ideas:

   (c) [3 marks] Choose a random order of the points. If joining them in this order gives a simple polygon, output it. Otherwise, start over by choosing a new random order.

   (d) [3 marks] Let p_1 be the point of minimum x-coordinate and—in case of ties—maximum y-coordinate. Sort the points radially around p_1 and join them in that order.