CS 157: Assignment 6

Douglas R. Lanman
8 May 2006

Problem 1: Evaluating Convex Polygons

This write-up presents several simple algorithms for determining whether a given set of two-dimensional points defines a convex polygon (i.e., a convex hull). In Section 1.1, we introduce the notion of regular polygons and provide examples of both convex and non-convex point sets. Section 1.2 presents an algorithm for ordering a set of points such that a *counterclockwise traversal* defines the interior of the convex hull. In Section 1.3, we apply this method to determine whether a given set of points are the vertices of a convex hull. Finally, in Section 1.4, we derive a lower bound for the worst-case running time of all such vetex-ordering algorithms.

1.1: Examples of Convex and Non-convex Polygons

Give an example of a convex polygon and a non-convex polygon with 4, 5, and 6 vertices.

A general n-sided polygon can be described by an ordered sequence of vertices $P = \{p_0, p_1, \ldots, p_{n-1}\}$. We define the interior of the polygon to be to the left of the line from $p_{i \bmod n}$ to $p_{i+1 \bmod n}$ for $0 \leq i \leq n-1$. As a result, the order of vertices within P matters. For this problem, however, we will concentrate on a simple case for P consisting of the set of *regular polygons*. A regular n-sided polygon has vertices evenly-distributed around a circle of radius r as follows.

$$p_i = (x_i,\ y_i) = \left(r\cos\left(\frac{2\pi i}{n}\right),\ r\sin\left(\frac{2\pi i}{n}\right)\right),\ \text{for } 0 \leq i \leq n-1,\ r > 0 \tag{1}$$

Every regular polygon is also a *convex polygon*. In general, a convex polygon is one whose vertex list P satisfies the following condition: the angle inside P formed by $(p_{i-1 \bmod n}, p_i, p_{i+1 \bmod n})$ is strictly less than 180 degrees, $\forall i \in [0, n-1]$. In other words, when traveling along the sequence $(p_{i-1 \bmod n}, p_i, p_{i+1 \bmod n})$ one always takes a left turn at p_i. Examples of regular, convex polygons for $n = \{4, 5, 6\}$ are shown in Figure 1.

A *non-convex polygon* can be defined as one which contains right turns or internal angles greater than or equal to 180 degrees. In general, if we append the vertex $(0, 0)$ to any regular polygon P

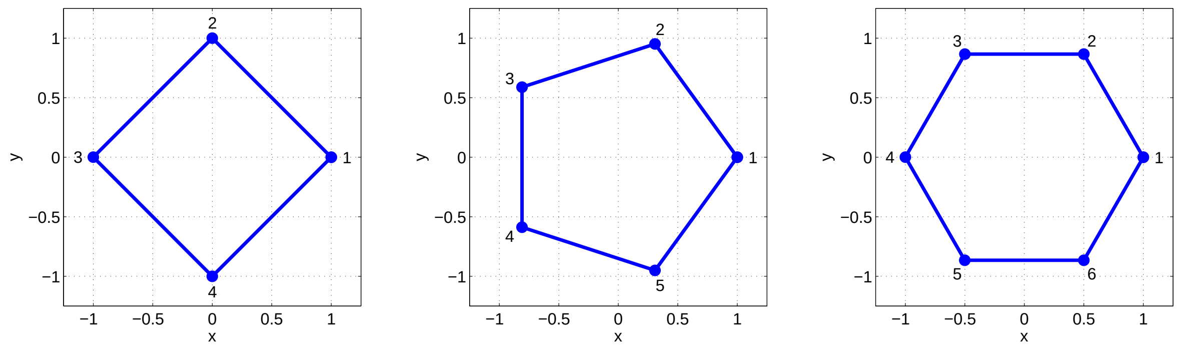
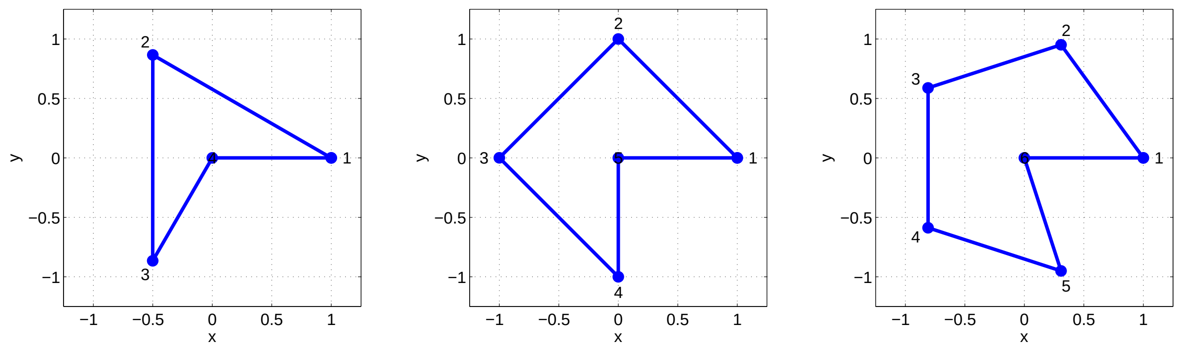

(a) square (b) pentagon (c) hexagon

Figure 1: Examples of convex polygons



(a) $n = 4$ (b) $n = 5$ (c) $n = 6$

Figure 2: Examples of non-convex polygons

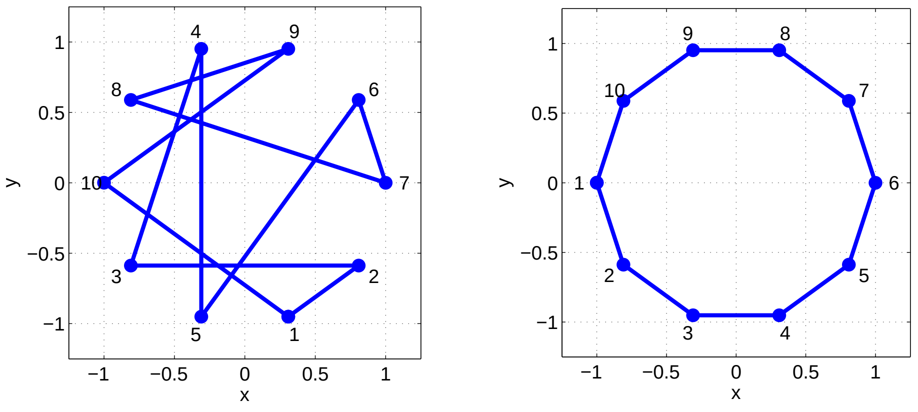
defined by Equation 1, then we will create a non-convex polygon P'. Examples of the resulting non-convex polygons are shown in Figure 2.

1.2: Finding a Counterclockwise Ordering

Given a set of points $\{p_0, p_1, \ldots, p_{n-1}\}$ which are the vertices of a convex polygon in arbitrary order, find an ordering σ of the points such that $P = \{p_{\sigma(0)}, p_{\sigma(1)}, \ldots, p_{\sigma(n-1)}\}$ is a counterclockwise traversal. Your algorithm should have a running time of $O(n \log n)$.

High-level Description: While the problem statement originally called for a divide-and-conquer solution, such an approach is needlessly complicated – as we can assume for this problem that the input consists of a set of convex points in a randomly-shuffled order. As a result (and with permission from the TA), I propose the following simple solution derived from Graham's Scan [1, 2]. The inputs to Order-Vertices are the vertex coordinates (x, y) and the output is the counterclockwise ordering σ. The pseudocode description is provided below.

```
Order-Vertices($x, y$)
1   ▷ Find bottom-left vertex.
2   $n \leftarrow length[x]$
3   $i \leftarrow 1$
4   for $j \leftarrow 2$ to $n$
5       do if $x[j] < x[i]$
6              then $i \leftarrow j$
7          if $x[j] = x[i]$ and $y[j] < y[i]$
8              then $i \leftarrow j$
9   ▷ Determine polar angles.
10  for $j \leftarrow 1$ to $n$
11      do if $x[j] = x[i]$
12             then if $y[j] > y[i]$
13                      then $\theta[j] = \pi/2$
14                      else $\theta[j] = -\pi/2$
15             else $\theta[j] = \arctan\left(\frac{y[j]-y[i]}{x[j]-x[i]}\right)$
16  sort $\theta$ into monotonically-increasing order to find $\sigma$
17  return $\sigma$
```



(a) input point set (b) ordered point set

Figure 3: Example of output produced by Order-Vertices for $n = 10$

Proof of Correctness: Before we prove the overall correctness of Order-Vertices, let's verify the outcome of each section independently. In the first section (on lines 1-8), we determine the left-most vertex (breaking ties in favor of lower points). The correctness of this section is immediate – a linear search through the coordinates returns the desired point.

In the second section (on lines 9-15), we determine the *polar angles* each vertex makes with respect the the reference vertex in the lower-left corner. In general, we can use the following simple trigonometric relationship to determine the angle θ (from the x-axis) between points p_i and p_j.

$$\theta[j] = \arctan\left(\frac{y[j] - y[i]}{x[j] - x[i]}\right) \tag{2}$$

The correctness of this solution is also immediate. Similarly, we assume that the sorting routine (on line 16) correctly arranges the polar angles into monotonically-ascending order. Note that we handle the special case of the x coordinates of the vertices being equal (e.g., Figure 1(b)) explicitly on lines 11-14.

In conclusion, the individual sections of Order-Vertices are correct. To prove the overall correctness, consider the example in Figure 3(a). If we draw a line from vertex 10 (the left-most vertex) along the $-y$-axis, and if we rotate this line to the right, then it will touch each point in counterclockwise order. This is due to the fact that the input set is convex. As a result, the polar angles of each point uniquely determine their correct order in the output σ.

Analysis of Running Time: The asymptotic worst-case running time of Order-Vertices is $O(n \log n)$. We can prove this directly by analyzing the pseudocode. On lines 1-8 we search for the left-most vertex (breaking ties in favor of lower points). Since this section only involves constant time comparisons and assignments, then the running time is $O(n)$ corresponding to the order of iterations of the **for** loop on line 4.

Similarly, the section of code on lines 9-15 is used to determine the set of polar angles (with respect to the lower-left vertex). Once again, this section involves constant time operations and has a running time of $O(n)$ due to the **for** loop on line 10.

In order to produce the desired counterclockwise traversal σ, we must sort the points by their polar angles. This can be accomplished using a variety of sorting algorithms presented in [1] with a running time of $O(n \log n)$. As a result, the asymptotic worst-case running time of Order-Vertices is $O(n \log n)$. (QED)

1.3: Determining if a Point Set is Convex

Extend your algorithm to solve the decision problem: given a set of points $\{p_0, p_1, \ldots, p_{n-1}\}$, decide whether they are the vertices of a convex polygon.

High-level Description: Initially, one might try solving this problem using the approach outlined in Problem 33.1-5 in [1]. That is, after ordering the points using Order-Vertices, check that each consecutive pair of edges has an internal angle less than 180 degrees (i.e., that we always make a left turn when traversing the ordered vertex list). While this would work for *simple polygons* with no self-intersections, one could imagine a *complex polygon* which intersects itself – resulting in a non-convex polygon with all "left" turns. As a result, we must improve upon this naive approach.

Recall the following classic result from planar geometry: the sum of the internal angles of a simple n-sided polygon is equal to $(n-2)\pi$ radians for $n \geq 3$ [4]. Given this constraint, we can easily verify that a polygon is simple (i.e., is not self-intersecting) and, as a result, we can check that the internal angles are all less than 180 degrees (and that this constraint is satisfied). As a result, we propose Test-Convex below to determine if a set of points represents a convex polygon. The inputs to Test-Convex are the vertex coordinates (x, y) and the output is a boolean indicating whether or not the points define a convex polygon. (Note that subscript addition is performed modulo n in this pseudocode description.)

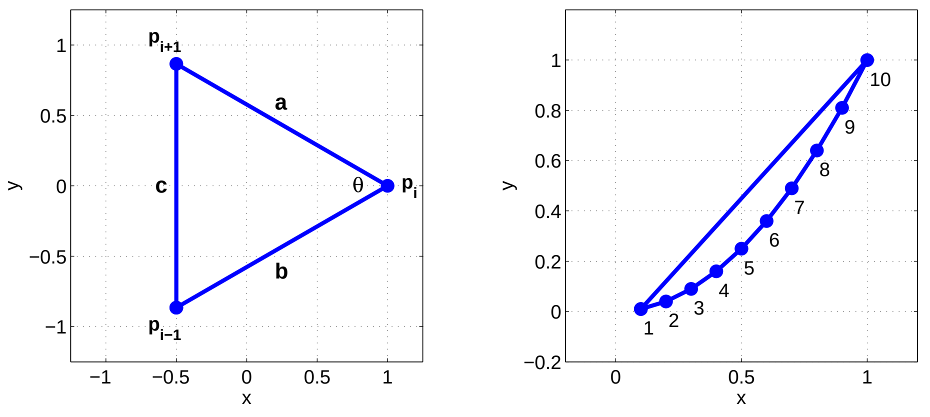
```
Test-Convex($x, y$)
 1  $\sigma \leftarrow$ Order-Vertices($x, y$)
 2  reorder $(x, y)$ using $\sigma$
 3  $n \leftarrow length[x]$
 4  $A \leftarrow 0$
 5  for $i \leftarrow 1$ to $n$
 6      do $C \leftarrow \det\begin{pmatrix} (x[i+1]-x[i]) & (y[i+1]-y[i]) \\ (x[i+2]-x[i]) & (y[i+2]-y[i]) \end{pmatrix}$
 7         if $C \leq 0$
 8            then return false
 9         $a \leftarrow \sqrt{(x[i+1]-x[i])^2 + (y[i+1]-y[i])^2}$
10         $b \leftarrow \sqrt{(x[i-1]-x[i])^2 + (y[i-1]-y[i])^2}$
11         $c \leftarrow \sqrt{(x[i+1]-x[i-1])^2 + (y[i+1]-y[i-1])^2}$
12         $A \leftarrow A + \arccos\left(\frac{a^2+b^2-c^2}{2ab}\right)$
13  if $A = (n-2)\pi$
14     then return true
15     else return false
```

Proof of Correctness: The correctness of Test-Convex follows from the definition of a convex polygon. That is, a convex polygon must have internal angles strictly less than 180 degrees and the sum of its internal angles must equal $(n-2)\pi$ radians. As a result, we must prove that our method for determining the internal angles and their sum is correct.

Recall that the cross product can be used to determine how consecutive line segments turn at a point p_i, as shown in [1] on pages 935 and 936. In general, the sign of the cross product C, where

$$C = \det\begin{pmatrix} (x[i+1]-x[i]) & (y[i+1]-y[i]) \\ (x[i+2]-x[i]) & (y[i+2]-y[i]) \end{pmatrix} \tag{3}$$

is positive if and only if the edge turns to the left at p_i [1]. As a result, we check this condition for each vertex on lines 6-8 (returning false if this test fails).



(a) Law of Cosines illustration (b) Example used to prove $\Omega(n \log n)$

Figure 4: Law of Cosines used in Test-Convex and the example from Section 1.4.

Assuming that every pair of consecutive edges makes a left turn, we still must verify that the polygon is simple. We can do this by storing the running sum of the internal angles. In order to compute the internal angles, we simply apply the Law of Cosines. For the example shown in Figure 4(a), the Law of Cosines requires

$$c^2 = a^2 + b^2 - 2ab\cos\theta \tag{4}$$

As a result, the internal angle θ is given by

$$\theta = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right) \tag{5}$$

This operation is performed on lines 9-12 and the constraint that the sum of internal angles equals $(n-2)\pi$ is verified on line 13. As a result, Test-Convex correctly determines if the input point set represents a convex polygon.

Analysis of Running Time: The running time of Test-Convex is clearly $O(n \log n)$. We can prove this by direct inspection. Lines 3, 4, 13-15 all involve constant time operations. Similarly, lines 6-12 also involve constant time operations (assuming arccos is implemented by a constant-time lookup table). As a result, the asymptotic worst-case running time for lines 3-15 is determined by the **for** loop on line 5 and is $O(n)$. Since the reordering on line 2 can be accomplished in $O(n)$ given σ, the overall asymptotic worst case running time is due to the call to Order-Vertices on line 1 and is $O(n \log n)$. (QED)

1.4: A Lower Bound on all Vertex-Ordering Algorithms

Prove that every vertex-ordering algorithm whose comparison operations are binary (i.e., any test performed has only two outcomes) must have complexity $\Omega(n \log n)$ in the worst case.

A simple example illustrates that any algorithm which determines the order of vertices in a convex polygon (e.g., the approach presented in Section 1.2) must have complexity $\Omega(n \log n)$ in the worst case. Consider the point set shown in Figure 4(b). In this example, we have ten points on the parabola given by $y = x^2$. The desired counterclockwise ordering is given by sorting the vertices by their x coordinate. Recall Theorem 8.1 from [1]: any comparison sort requires $\Omega(n \log n)$ in the worst case. As a result, the vertex-ordering task must also have a worst-case complexity of $\Omega(n \log n)$.

As presented in Chaper 8 of [1], this lower bound can be understood by considering the *decision tree* for the vertex-ordering task. Unlike one-dimensional (i.e., array) sorting algorithms, the solution of the vertex-ordering task is not unique; that is, the sorted vertices can appear in any cyclic permutation. Regardless, we can always select a vertex to be first in the list. Afterwards, there remaining $n-1$ vertices which must be ordered. As with array sorting, the decision tree must contain all $(n-1)!$ permutations as a leaf. Following the derivation of [1], the height h of the decision tree must satisfy the following relationship.

$$2^h \geq (n-1)! \Rightarrow h \geq \log\left((n-1)!\right) = \Omega(n \log n) \tag{6}$$

In conclusion, we have demonstrated that all vertex-ordering algorithms must have worst-case complexity $\Omega(n \log n)$ by the example in Figure 4(b). It is important to note that this example was first proposed by Shamos and Preparata [3]. (QED)

Conclusion

This write-up reviewed the notion of convex polygons and presented several algorithms for determining whether a given point set represents one. The algorithm presented in Section 1.3 (and its analysis in Section 1.4) is of general utility, however this document has not explored the larger issue of determining a convex hull for an arbitrary point set. This more general problem is of central importance to the field of computation geometry. Interested readers are referred to Shamos and Preparata [3] for a detailed presentation of convex hull algorithms. In addition, see [2] for a divide-and-conquer solution to Problem 1.2.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw-Hill, 2001.

[2] Jeff Erickson. Convex hulls. `http://compgeom.cs.uiuc.edu/~jeffe/teaching/373/notes/x05-convexhull.pdf`, 2003.

[3] Michael I. Shamos and Franco P. Preparata. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[4] Wikipedia. Polygon. `http://en.wikipedia.org/wiki/Polygon`, 2003.