

COMPUTATIONAL GEOMETRY

Homework Set # 2

Due at the beginning of class on Tuesday, October 14, 2008. *Reminder: Show your reasoning!*

Recommended Reading: BKOS: Chapter 2; O'Rourke sections 7.7, 7.8.

DO ANY 4 OF THE FOLLOWING 5 PROBLEMS.

- (1). Problem 2.10, page 43, BKOS.
- (2). Problem 2.14, page 43, BKOS.
Optional: Would your algorithm apply also to line segments that may intersect? (explain)
- (3). Problem 2.9, page 43, BKOS.
- (4). Write an intersection predicate, `TriTriIntersectProp`, in C which takes six points, a, b, c, d, e, f , which define two triangles, Δabc and Δdef , and determines if they *properly* intersect or not. (In order to intersect properly, there must exist a point p *interior* to both triangles (both triangles must be nondegenerate for this to happen; e.g., if a triangle is degenerate (a line segment or a point), then it has no points in its interior); a triangle is considered to be the (closed) convex hull of its three vertices.)
Your predicate should start:

```
bool  TriTriIntersectProp( tPointi a, tPointi b, tPointi c,
                          tPointi d, tPointi e, tPointi f )
{
[FILL IN HERE]
}
```

You may use the predicates from O'Rourke's book, `Area2`, `Left`, `LeftOn`, `Collinear`, `IntersectProp`, `Between`, and `Intersect` from the text, but try to think carefully about making your code complete (handling all cases) and efficient (avoiding duplication of tests).

Note: Even if you have never programmed in C before, this exercise should be fairly straightforward; try to read and understand the code fragments of section 1.5 of O'Rourke.

- (5). Let S be a set of n line segments in general position in the plane (no three endpoints are collinear). Assume that there is at least one (proper) crossing point where two segments properly intersect. Let V denote the set of all (proper) crossing points. We wish to find the axis-aligned bounding box of V . Describe an efficient ($O(n \log n)$, ideally) method to do this.

(CHALLENGE PROBLEM). We can think of the convex hull of a point set S as the region bounded by a shortest "rubber band" that surrounds *all* of the points S . Suppose now that we want the shortest rubber band that surrounds *all but* k of the n points of S . (Think of the k omitted points as "outliers".) How efficiently can you solve this? Consider both "small" values of k and also situations in which k may be "large" (close to n).

Practical question: How would you *really* solve this question in practice? (You may be able to devise a project based on a thorough investigation/implementation.)