

## COMPUTATIONAL GEOMETRY

### Homework Set # 2

Due at the beginning of class on Wednesday, February 20, 2008.

Recommended Reading: O'Rourke, Chapter 1 (sections 1.3–1.6) and Chapter 2 (sections 2.1–2.2).

**Reminder:** In all of the exercises, be sure to give at least a brief explanation or justification for each claim that you make.

(1). [20 points] Suppose that you are given a simple polygon  $P$ , specified (as usual) as a list of  $n$  vertices,  $v_0, v_1, \dots, v_{n-1}$ , given in counterclockwise order about the polygon. Your goal is to find a diagonal of  $P$ , *any* diagonal,  $v_i v_j$ . Describe an efficient algorithm to compute a diagonal, ideally in worst-case linear time, *without* resorting to computing a triangulation of  $P$ . (Since Chazelle's (highly complex) algorithm computes a triangulation in linear time, we know that it is possible; but much simpler methods are also possible!) Make your algorithm as efficient as possible (ideally  $O(n)$ ) in the worst case (you need not worry about constants; "efficient" is measured in the usual Big-Oh notation). What is the running time? (While your algorithm need not be written in code, try to write it very clearly and succinctly, step by step.)

(2). [40 points] Write an intersection predicate, `RayIntersectProp`, in C which takes four points,  $a, b, c, d$ , which define two (infinite) rays,  $\vec{ab}$  and  $\vec{cd}$ , and determines if they intersect *properly* or not. (The rays are considered to be *closed* – they include their endpoint ( $a$  on  $\vec{ab}$   $c$  on  $\vec{cd}$ )). Here, intersection will mean that they have a *proper* intersection: there is *exactly one* point of the interior of  $\vec{ab}$  that also lies in the interior of  $\vec{cd}$ . (We do not consider them to "intersect" if they share only a common endpoint ( $a = c$ ) or if the endpoint of one ray lies on the interior of the other ray. We also do not consider them to intersect if they are colinear.) You are allowed to assume that  $a \neq b$  and that  $c \neq d$ .

Your predicate should start:

```
bool RayIntersectProp( tPointi a, tPointi b, tPointi c, tPointi d )
{
```

[FILL IN HERE]

```
}
```

You may use the predicates `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 within your ability; try to read and understand the code fragments of section 1.5.

**Submitting:** If you type up the code fragment (which is optional) and have access to the internet, then please email it to me (jsbm@ams.sunysb.edu) so that I can have examples for future students

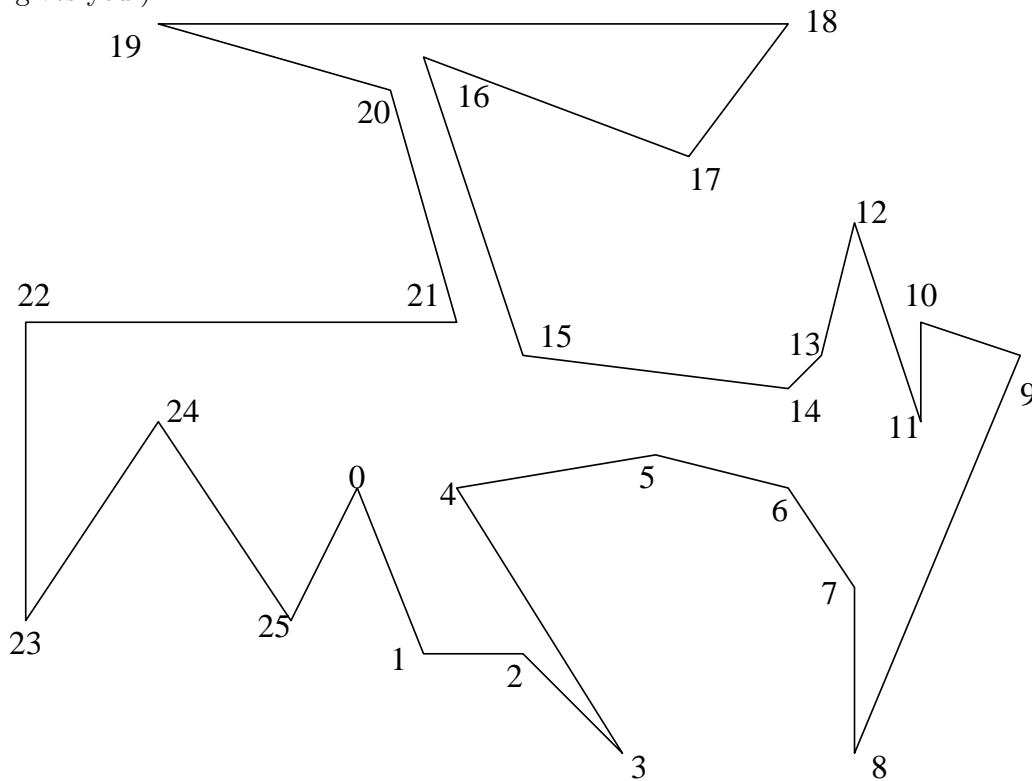
to see. Also, if you are comfortable with C programming, you may find it convenient to test it by compiling it and executing the code (within a simple main program).

**Optional** (for extra credit): Compile, execute, and time your code. (You will have to run it on a very large number of examples (e.g., randomly generated), measure the total time, and divide by the number of calls in order to get a meaningful estimate of the time per call.) It may be useful also to have a graphical user interface that allows the user to mouse-click input and draws the rays in a graphics window, etc. (This feature is most useful for debugging too!)

(3). [10 points] O'Rourke, problem 2, section 1.6.4, page 36.

(4). [30 points] For the simple polygon below, give the order in which diagonals are output when **Triangulate** is executed on it (exactly as is done in Table 1.2 for the polygon shown in Figure 1.27). The coordinates of the points (in order 0–25) are given by: (12,12), (14,7), (17,7), (20,4), (15,12), (21,13), (25,12), (27,9), (27,4), (32,16), (29,17), (29,14), (27,20), (26,16), (25,15), (17,16), (14,25), (22,22), (25,26), (6,26), (13,24), (15,17), (2,17), (2,8), (6,14), (10,8).

(Optionally, you may download, compile and execute the code from the textbook web site, and submit a printout of the result of the execution. You should also be able, however, to perform the execution “by hand” on the midterm. Please confirm that your hand calculation matches what the program gives you.)



Also, *draw* the resulting triangulation by plotting the diagonals you obtain.