Computational Geometry—236719

(Fall 2022–23, Gill Barequet and Tomer Adar)

Assignment no. **4** (wet)

Given: November 15, 2022
Due: **December 31, 2022**
Submission in **singletons**

1 General

In this exercise you will implement the plane-sweep algorithm, taught in class, to count the number of intersections of a set of line segments. Note that we are interested only in the **number** of intersections, and not in the actual intersection points.

2 Input

The input is found in a single Ascii file which contains the following data:

1. Number of test cases (a positive integer number n).
2. n sets of segments, each one containing:
   - Number of segments (a positive number m_i, $1 \leq i \leq n$).
   - m_i segments, each one specified by four (4) point coordinates $x_{i_1}, y_{i_1}, x_{i_2}, y_{i_2}$.
3. The number `-1`.

3 Output

A list of n numbers, each one left-justified in a separate line, each line ending with the `newline` character (including the last line!).

4 Assumptions

- The input file contains at most 25,000 sets of segments.
- Each set of segments contains at most 1,000 segments.
- Spacing in the input file is insignificant.
- There are no vertical segments.
- No two segments intersect in more than one point.
- No three segments intersect in one point.
- No numerical errors occur when using C's double-precision floating point numbers. That is, segment endpoints are well separated, as well as intersections of segments, and events of the algorithm are separated enough along the x axis.
- You may **not** assume that the first endpoint of a segment lies to the left of the second endpoint, but need to check and handle both cases.

5 Implementation

Implementation should be done in a Unix or a Windows environment, preferably using C or C++, but other operating systems or programming languages will be allowed. Please inform *Tomer* **in advance** the environment you intend to use.

You may use external code, packages, libraries, C++ STL, etc., for implementing **basic data structures** such as trees, queues, and so on. The geometric core of the algorithm must be implemented by you.

6 Questions and a FAQ File

Questions should be directed to *Tomer* (`tomer-adar@cs.technion.ac.il`). Tomer will create a FAQ file and publish its address.

7 Submission

The submission bundle should include code files (e.g., `*.c` and `*.h` files), a `makefile` file, and a short documentation of your submission. The submission should be sent by E-mail to *Tomer* (`tomer-adar@cs.technion.ac.il`) by the deadline.

8 Checking and Grading

The course staff will compile and run your submission with sample test case(s). Grading will be based on visual inspection of the code (to verify that it is implementing the sweep algorithm taught in class) and on the correctness of the results. We recommend that you implement a simple $O(n^2)$-time algorithm in order to verify the correctness of your results. A failure to compile the code will be considered as "no submission."

9 Example

Input file:

```
3
2
11.3 5.1 3.2 6.9
4.2 7.1 2.8 4.9
6
21.2 49.9 9.6 59.6
10.1 20.1 60.2 49.8
69.9 41.2 60.4 19.7
9.8 40.1 60.2 70.2
20.9 72.1 40.5 20.1
49.7 20.3 40.6 70.2
3
0.1 4.9 6.1 11.2
5.5 8.1 1.1 6.9
5.1 6.2 1.9 9.1
-1
```

Output file:

```
1
4
3
```

GOOD LUCK!