Computational Geometry—236719

(Fall 2025–2026, Gill Barequet and Tom Agami)

Assignment no. **4** (wet)

Given: 18/Nov/2025
Due: **18/Dec/2025**
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 should occur when using 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 Python. For other operating systems or programming languages, consult fisrt with *Tom*.

You may use system packages and libraries for implementing **basic data structures** such as trees, queues, and, etc. The geometric core of the algorithm must be implemented by you. In particular, do **not** use the Python packages `sweepline` and `pysweepline`.

We supply a Python utility module for segment intersection and for a priority queue. It is not mandatory to use this module, but it can reduce your workload.

6 Warning

We also know how to use ChatGPT and similar AI tools. Moreover, we know how to identify code produced by them, mainly by their *behavior* when they face specific inputs. Therefore, we strongly suggest not to try to hide the use of AI code by changing variable and function names, indentation, coding conventions, etc., because none of this will hide the traces of AI code.

7 Questions and a FAQ File

Questions should be directed to *Tom* (`tom.agami@campus.technion.ac.il`). Tom will create a FAQ file and publish its address.

8 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 *Tom* (`tom.agami@campus.technion.ac.il`) by the deadline.

9 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."

10 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!