Numerical Geometry: Assignment 2

In this homework, you are going to build and work with a *doubly-connected edge list*, one of the most versatile data structures for representing geometric subdivisions. This data structure will be the foundation for the project coming up, so it is important that you have a good understanding of it. If you are not completely familiar with doubly-connected edge lists, we recommend you read Section 2.2 of the book before starting the homework.

There are 3 tasks to complete:

1. Given a mesh described in a JSON file as in homework 1, build a `mesh` object with the following structure:
   - `mesh.nodes`: an array of nodes; a `node` has structure
     - `node.id`: a unique integer identifying the node (use the indices given in the JSON file),
     - `node.pos`: a pair of coordinates representing the position of the node.
   - `mesh.faces`: an array of faces; a `face` has structure
     - `face.id`: a unique integer identifying the face (use the indices given in the JSON file),
     - `face.incidentEdge`: *one* (!) edge that is incident to `face`, i.e., has `face` on its left side.
   - `mesh.edges`: an array of (half-)edges; an `edge` has structure
     - `edge.orig`: the origin node of the half-edge,
     - `edge.dest`: the destination node of the half-edge,
     - `edge.incidentFace`: the face to the left of the half-edge,
     - `edge.next`: the next half-edge on the boundary of the incident face,
     - `edge.oppo`: the opposite half-edge.
2. Given a target point described by a pair of coordinates, locate the triangle containing the point using the *marching triangle* algorithm. One way of doing this is to start from an arbitrary triangle, take an arbitrary vertex of this triangle, from which we draw a line segment to the target vertex; then, we can go on with the marching algorithm.
3. Given a line segment described by two pairs of coordinates, compute the intersection (consisting of a sequence of points) of this segment with the triangulation, using the marching triangle algorithm.

Completing the three tasks *perfectly* will grant you the maximal grade. Here are suggestions of bonus tasks (but we encourage you to be creative!):

- Make the marching algorithm more efficient by reducing the number of calls to `orient2d`. How many FLOPs does your implementation require per triangle?
- Visualize the half-edges in a pretty way.

Task	Points
Half-edge data structure	7
Point location	7
Intersection segment-mesh	6
Bonus: efficiency	3
Bonus: visualization	2
—	—
Total	20

Practical information

- **Groups:** *The groups must be the same for the 2 homeworks and the project*
- **Collaboration:** *You are allowed, and even encouraged, to exchange ideas on how to address this assignment with students from other groups. However, you must do all the writing (report and codes) only with your own group; it is strictly forbidden to share the production of your group. Plagiarism will be checked.*
- **Writing:** *You write a small Read Me file, preferably in Markdown, explaining what you have done, especially if you went further than asked.*
- **Language:** *All reports and communications are equally accepted in French and English.*
- **Deliverables:** *Each group is asked to submit their project on the course website:*
  - *The readme,*
  - *The .html, .js and .json files.*
- **Deadline:** *The homework is due for Friday, October 27 at 18:15.*
- **Questions:** *You can adress questions by sending an email to the teaching assistants (matteo.couplet@uclouvain.be, antoine.quiriny@uclouvain.be, illy.perl@uclouvain.be). Direct messages on Teams will not be considered*