Computational topology
Homework 2 (due: November 24th 2024)

Each problem is worth a certain amount of points. Some problems are theoretical, others require you also submit the code (that conforms to the requirements given in the problem description). You may choose which problems to solve, **15 points is equal to 100%**.

You have to submit your solutions before the deadline as **one .zip file** to the appropriate mailbox at `https://ucilnica.fri.uni-lj.si/course/view.php?id=111` (near the top of the page).

This .zip file should contain:

1. **a NameSurname.pdf file written in LaTeX** and containing the solutions to the theoretical problems you have chosen as well as solutions and explanations for the programming problems (also make sure you sign your name on the top of the first page),
2. **.jl files** containing the code (one for each of the programming problems you have chosen).

1 Theoretical problems

1. **(3 points) Triangulations.**

   Let S be a set of n points in the plane, $n \in \mathbb{N}$.

   (a) Show that for all $n \in \mathbb{N}$ there are at most $2^{\frac{n(n-1)}{2}}$ triangulations of S. Show that for all $n \in \mathbb{N}$ there are at most $2^{\frac{n(n-1)(n-2)}{6}}$ triangulations of S. These two upper bounds are very imprecise. For which n is the first bound better than the second?

   (b) The degree of a point in a triangulation T is the number of edges in T, incident to that point. For each $n \geq 3$ construct a set S such that all possible triangulations of S have a point of degree $n-1$.

   (c) If not all points in S are collinear, then any triangulation T of S has at most $3n-3$ edges. Use this fact to prove that any triangulation T of S has a point of degree 5 or less.

2. **(1 point) Vietoris-Rips Complex and Čech Complex.**

   Let $S = \{(0,0), (2,0), (1,0.5), (1,1.5)\} \subset \mathbb{R}^2$.

   (a) Build the Vietoris-Rips complex $\mathrm{VR}_{2\epsilon}(S)$ and the Čech complex $\check{\mathrm{C}}_{\epsilon}(S)$ for $\epsilon = 0.8$.

   (b) Build the Vietoris-Rips complex $\mathrm{VR}_{2\epsilon}(S)$ and the Čech complex $\check{\mathrm{C}}_{\epsilon}(S)$ for $\epsilon = 1$.

   In each case list all the simplices, determine its dimension and find the Euler characteristic.

3. **(1 point) Voronoi diagram.** For all $n \in \mathbb{N}$, $n > 3$, find a configuration of n points in the plane such that their Voronoi diagram will have a cell with $n-1$ vertices.

4. **(3 points) Chessboard Complex.**

   The *chessboard complex* of a $m \times n$ chessboard is a simplicial complex $\Delta_{m\times n}$. The vertices of $\Delta_{m\times n}$ correspond to the squares of the chessboard. Simplices of $\Delta_{m\times n}$ correspond to non-taking placements of rooks (ie. placements where no two rooks are in the same column or in the same row).

(a) Show that the chessboard complex $\Delta_{3\times 2}$ is homeomorphic to the circle S^1. (Hint: Show that the simplicial complex you obtain is a 1-dimensional connected manifold with no boundary.)

(b) Show that the chessboard complex $\Delta_{4\times 3}$ is homeomorphic to a torus $S^1 \times S^1$. (Show that the complex you obtain is an orientable connected 2-dimensional manifold without boundary with Euler characteristic 0. Alternatively, you can try to construct an explicit homeomorphism.)

(c) Is the chessboard complex $\Delta_{n\times(n-1)}$ is a manifold for all n? List some properties that support your hypothesis (you do not need to prove it).

2 Programming problems

5. **(2 points) Line sweep triangulation in arbitrary direction**

Given a point cloud $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, write a function

```
line_sweep(S, sweep_direction = [1; 0])
```

that returns a list of abstract simplices as tuples of indices of points of the given point cloud. By default the sweep direction should be along the x-axis. In case that the optional parameter (a two component column vector) is passed to the function that vector should be used as sweep direction instead.

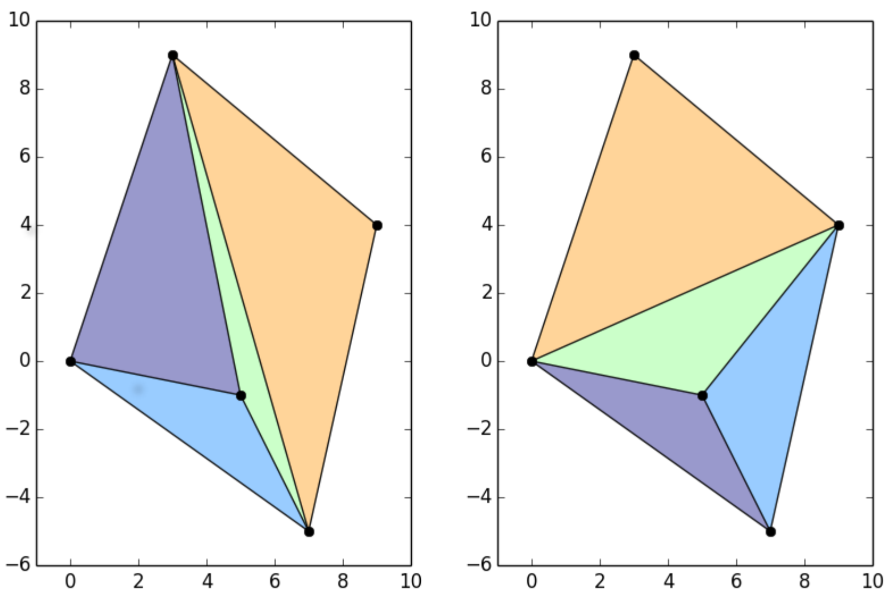
Submit a file named `line_sweep.jl`.

Plot the points, edges and triangles in the plane.

Sample inputs:

```
S = [(0, 0), (3, 9), (5, -1), (9, 4), (7, -5)], sweep_direction = [1; 0]
S = [(0, 0), (3, 9), (5, -1), (9, 4), (7, -5)], sweep_direction = [0; 1]
```

Sample plots:



Make up two more test cases consisting of at least 100 points and include the resulting triangulations (with two sweep directions of your choice) in your report.

6. **(2 points) Delaunay triangulation**

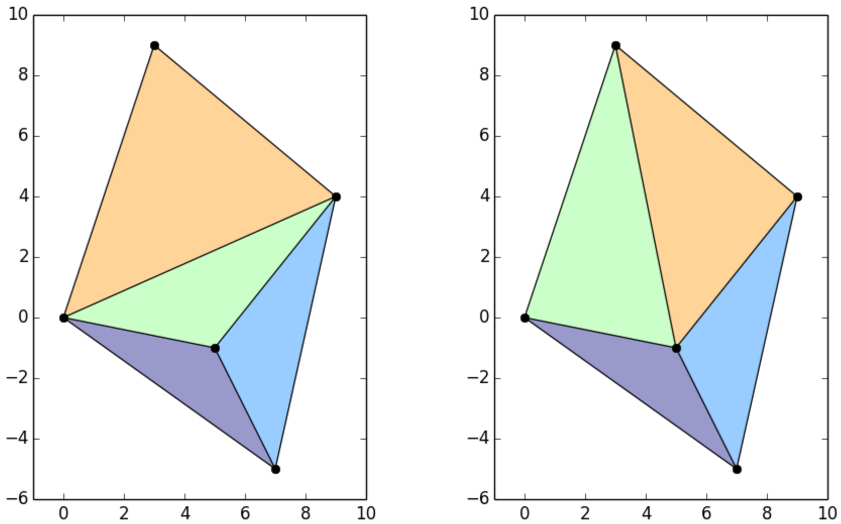
   Your file `delaunay.jl` should contain a function `delaunay(S, T)`, which takes as input any abstract triangulation `T` on the planar set of points listed as `S` and optimizes it to produce the Delaunay triangulation of the underlying set of points. The function should return a new abstract triangulation on the same abstract vertices. Do this by implementing the edge-flip algorithm. Plot both the initial triangulation and the resulting Delaunay triangulation.

   Plot the Delaunay triangulation in the plane.

   Sample inputs:

```
S = [(0, 0), (3, 9), (5, -1), (9, 4), (7, -5)]
T = [(5,), (3,), (5, 3), (1,), (5, 1), (3, 1), (5, 3, 1), (4,), (5, 4),
     (5, 3, 4), (3, 4), (3, 1, 4), (1, 4), (2,), (4, 2), (4, 1, 2), (1, 2)]
```

   Sample plots:



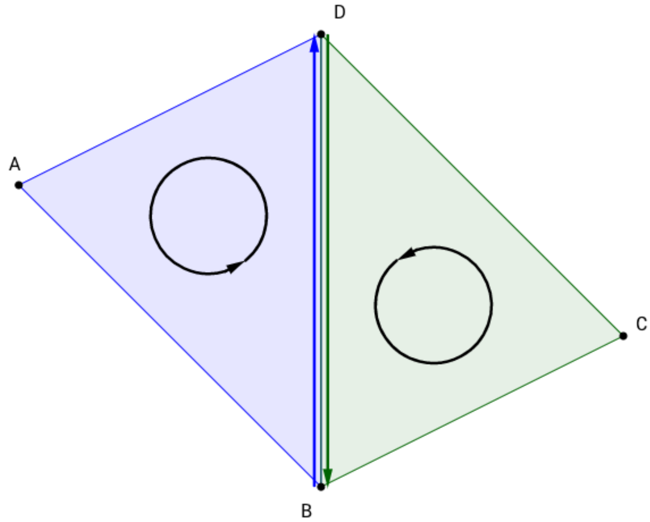
   Make up at least two more test cases consisting of at least 100 points and include plots of original and resulting Delaunay triangulations in your report.

7. **(3 points) Orientation of surfaces**

   Your file `is_orientable.jl` should contain a function `is_orientable(T)`, which returns `true` if a 2-manifold given by its triangulation T (via maximal simplices) is orientable and `false` otherwise. Also include a function `oriented(T)`, which returns a list of oriented triangles if the 2-manifold is orientable and `nothing` otherwise.

   Test your function on triangulations of a torus, a Klein bottle, a sphere, a cylinder and a Möbius band (you can construct these by yourself or search the web).

   A 2-manifold is *orientable* if you can choose the orientations of all of its triangles consistently. Two triangles that share a common edge are *consistently oriented* if they induce opposite orientations on the common edge (see figure).



Sample input:
`M = [(1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 6), (2, 5, 6), (1, 2, 6)]`

Sample output:
```
false
Oriented triangles:
nothing
```

Sample input:
`S2 = [(1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4)]`

Sample output:
```
true
Oriented triangles:
[(1, 2, 3), (1, 4, 2), (1, 3, 4), (2, 4, 3)]
```