Computational topology
Homework 3 (due: December 27th 2020)

Each problem is worth a certain amount of points. The point assignment should somewhat reflect the difficulty of the problem. 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:

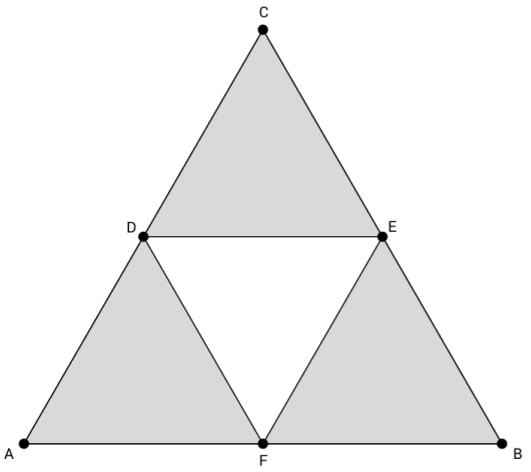
1. **a namesurname.pdf file written in LaTeX** and containing the solutions to the theoretical problems you have chosen as well as solutions and explanations for the programming problems (also make sure you sign your name on the top of the first page),
2. **.py files** containing the code (one for each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

1 Theoretical problems

1. **(4 points) Homology.**

   For the simplicial complex X pictured below:

   (a) write down the chain groups C_n,
   (b) determine the boundary homomorphisms $\partial_n \colon C_n \to C_{n-1}$,
   (c) find the cycles $Z_n = \ker \partial_n$,
   (d) find the boundaries $B_n = \mathrm{im} \partial_n$,
   (e) determine the homology groups with $\mathbb{Z}$ coefficients, $H_n(X; \mathbb{Z})$,
   (f) determine the homology groups with $\mathbb{Z}_2$ coefficients, $H_n(X; \mathbb{Z}_2)$,
   (g) determine the Betti numbers of X and
   (h) compute the Euler characteristic of X.



2. **(5 points) Homology.**

   For the simplicial complex X pictured below:

(a) write down the chain groups C_n,

(b) determine the boundary homomorphisms $\partial_n \colon C_n \to C_{n-1}$,

(c) find the cycles $Z_n = \ker \partial_n$,

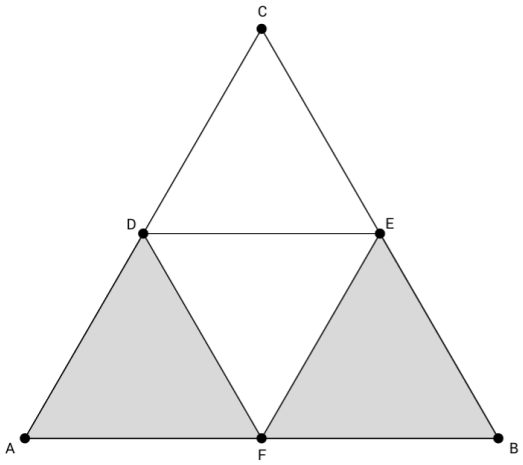
(d) find the boundaries $B_n = \mathrm{im} \partial_n$,

(e) determine the homology groups with $\mathbb{Z}$ coefficients, $H_n(X; \mathbb{Z})$,

(f) determine the homology groups with $\mathbb{Z}_2$ coefficients, $H_n(X; \mathbb{Z}_2)$,

(g) determine the Betti numbers of X and

(h) compute the Euler characteristic of X.



2 Programming problems

3. **(3 points) Vietoris-Rips complex**

The file `rips.py` should contain a function `cliques(VG, EG)` that finds all cliques in a graph, which is given as a list of vertices and a list of edges (you can transform these two into a more efficient data structure). It should also contain the function `VR(S, epsilon)` that returns a dictionary where keys are the dimensions of simplices in the Vietoris-Rips complex $\mathrm{VR}_\epsilon(S)$ and values are lists of all simplices of corresponding dimension.

Sample input for `VR(S, epsilon)`:

```
S = [(0, 0), (1, 1), (2, 3), (-1, 2), (3, -1), (4, 2)]
epsilon = 3
```

Sample output for `VR(S, epsilon)`:

```
{0:  [(0,), (1,), (2,), (3,), (4,), (5,)],
 1:  [(0, 1), (0, 3), (1, 2), (1, 3), (1, 4), (2, 5)],
 2:  [(0, 1, 3)]}
```

Make sure that `cliques` is at least somewhat efficient, ie. do not go through all $2^{|VG|}$ possible subsets of edges but try to limit your search as much as possible. Explain your method in your report. Which test case will give you the worst possible run time? How many vertices can your algorithm handle in 1 second or less in this worse-case scenario? How many vertices can it handle in 10 seconds or less? What about 100 seconds?

Your report should include a few test cases for `VR` as well as a few separate test cases for `cliques`. It is forbidden to use external library for Vietoris-Rips complex generation.

4. **(3 points) Čech complex**

   The file `cech.py` should contain a function `cech(S, epsilon)` that returns a dictionary where keys are the dimensions of simplices in the Čech complex $\check{C}_\epsilon(S)$ and values are lists of all simplices of corresponding dimension.

   Sample input for `cech(S, epsilon)`:

```
S = [(−2, 1), (−2, −2), (1, −1), (1.5, 2.5)]
epsilon = 1
```

   Sample output for `cech(S, epsilon)`:

```
{0:  [(0,), (1,), (2,), (3,)],
 1:  [(0, 1), (0, 2), (0, 3), (1, 2), (2, 3)],
 2:  [(0, 1, 2)]}
```

   You can use the code from the lab work to generate Vietoris-Rips complex and then use the mini-ball algorithm to add simplices of higher dimensions.

   Then download one of the point clouds available on `http://graphics.stanford.edu/data/3Dscanrep`. They are stored in ply format that already contains surface reconstruction which we do not need. Extract only the points from the downloaded file and create a Čech complex on the subsample (choose it appropriately). Use the script on ucilnica to export the obtained complex back into ply format and open it with a 3D editor (such as Blender). Does it look similar to the image on the Stanford site?

   Experiment with sample size and the radius used in the construction of the Čech complex to obtain the best possible reconstruction (including an image of the complex would be nice). Your report should include information on the best radius and subsample size (and the method you used in order to obtain them) and the picture of your best reconstruction.

5. **(3 points) Collapsibility**

   Write an algorithm that takes a simplicial complex given as a list of maximal simplices (which are not necessarily all of the same dimension) and simplifies it by collapsing any free faces.

   Your file `collapse.py` should contain a function `collapse(X, progress = True)` that returns the list of all simplices that are left after all possible collapses have been made. If the optional parameter is `True`, it prints the progress report to the console. Here is the first few lines of output for the cylinder:

```
Initial simplicial complex:
[(1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 6), (1, 5, 6), (1, 2, 6)]

Free faces:
[((1, 2, 3), (1, 3)), ((4, 5, 6), (4, 6)), ((1, 5, 6), (1, 5)),...]
Choose a simplex sigma with a free face tau:
sigma = (1, 2, 3)
tau = (1, 3)
Remaining simplices after the elementary collapse:
[(2, 3, 4), (3, 4, 5), (4, 5, 6), (1, 5, 6), (1, 2, 6)]
...
```

   Run it for a 2-sphere, a cylinder, a Moebius strip and the Dunce hat given below. Did you get the expected results?

```
S2 = [(1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4)]
C = [(1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 6), (1, 5, 6), (1, 2, 6)]
M = [(1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 6), (2, 5, 6), (1, 2, 6)]
D = [(1, 2, 5), (1, 5, 6), (1, 6, 7), (1, 2, 7), (1, 4, 9),
     (1, 9, 10), (1, 10, 11), (1, 4, 11), (1, 2, 13), (1, 13, 14),
     (1, 14, 15), (1, 4, 15), (2, 3, 5), (2, 3, 7), (3, 4, 9),
     (3, 4, 11), (2, 3, 13), (3, 4, 15), (3, 7, 8), (3, 8, 9),
     (3, 11, 12), (3, 12, 13), (3, 15, 16), (3, 5, 16),
     (5, 6, 17), (5, 16, 17), (6, 7, 17), (7, 8, 17),
     (8, 9, 17), (9, 10, 17), (10, 11, 17), (11, 12, 17),
     (12, 13, 17), (13, 14, 17), (14, 15, 17), (15, 16, 17)]
```

Finally, try an example where maximal simplices have different dimensions.

```
X = [(1, 2, 3), (2, 3, 5), (3, 4), (5, 6)]
```

Include the sequence of collapses for X in your report and come up with a few more test cases.