Exercise 3

Below are some exercises to consider for different existing GAP packages. Choose an exercise according to your preference or search for other packages that interest you and learn more about them. Please note that there are, of course, many other interesting packages available and that this is only a random selection.

1. Grape (Graphs and Groups)
2. SimplicialSurfaces (Triangulated Surfaces)
3. FinInG (Finite Incidence Geometry)
4. Automata
5. Guava (Codes)
6. Digraphs (Graphs)

Grape

→ Contains methods for computations with graphs and groups.

Exercise 1

a) Construct the Petersen graph in two different ways and test if the two are isomorphic. Note that the Petersen graph is a Kneser graph, which could help to construct the graph.

b) Compute the undirected edges of the Petersen graph. Test if the Petersen graph is 3-vertex colourable and contains a K_4 as subgraph.

c) How can you check whether a given graph is 3-regular?

Exercise 2*

Edge-3-colourings can sometimes be interpreted as the orbits of a subgroup $H \leq \mathrm{Aut}(\Gamma)$ acting on the edge set $E(\Gamma)$ of a graph Γ. Write a function that takes as input a 3-regular graph Γ and returns all edge-3-colourings of Γ such that the colour classes correspond to the orbits of some subgroup $H \leq \mathrm{Aut}(\Gamma)$ acting on $E(\Gamma)$.

Simplicial Surface

$\rightarrow$ Contains functionality for working with the combinatorics of triangulated and polygonal complexes.

Exercise 1

a) Construct an octahedron in two different ways (using e.g. *SimplicialSurfaceByVerticesInFaces, SimplicialSurfaceByUmbrellaDescribtor, SimplicialSurfaceByDownwardIncidence*) and test if your result is correct by using the build-in function *Octahedron*.

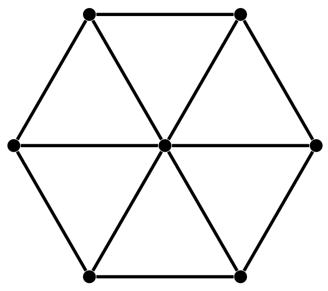
b) Start with the octahedron and constructed the stellated octahedron by using the method *TetrahedralExtension*. Show that the stellated octahedron is a sphere.

c)* Compute an embedding of the stellated octahedron in $\mathbb{R}^3$ and visualise it by using the package GAPic and the method *DrawComplexToJavaScript*.

Exercise 2

A triangulated n-gon consists of n triangles which are all incident to a common vertex as shown in the following figure. A triangulated double n-gon is constructed by joining the boundaries of two triangulated n-gons.

a) Write a function that constructs for a given n the triangulated n-gon and test your method by using the build-in function *SimplicialUmbrella*.



b) Write a function that constructs for a given n the triangulated double n-gon. You can use the function for the triangulated n-gon and use the build-in method *JoinBoundaries*.

FinInG

→ Contains methods for computation in Finite Incidence Geometry.

These exercises are inspired by Michel Lavrauw's exercise sheet from the Spring GAP Days 2025. More exercises for the FinInG can be found here and here.

Exercise 1

a) Construct the Fano plane, i.e. the projective plane $PG(2, 2)$. Show that the Fano plane has 7 points and 7 lines.

b) Choose a point on the Fano plane and compute all lines that pass through it.

c) Verify that each line in the Fano plane contains exactly 3 points and that each point lies on exactly 3 lines.

Exercise 2*

An arc of a projective plane is a set of points, no three of which on a line.

a) Construct the projective plane $PG(2, 3)$ and construct an arc of size 4.

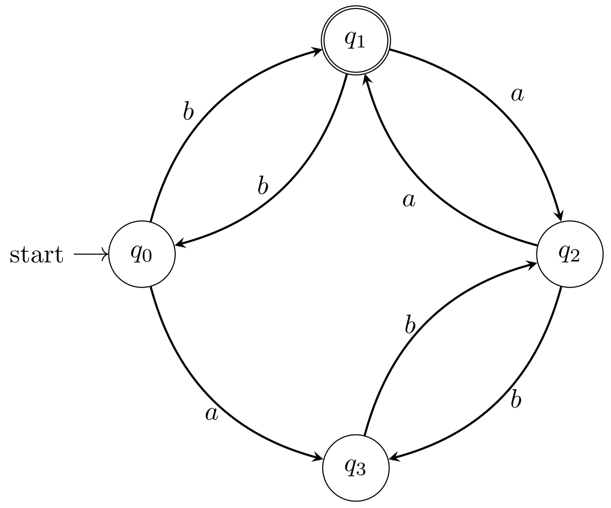
b) Write a function which determines whether a given set of points of a projective plane is an arc.

Automata

→ Contains functionalities for dealing with automata.

Exercise 1

a) Construct and display the automaton given in the following image.



b) Compute the corresponding rational expression and use build-in functions to obtain the automaton from the rational expression. Consider that the two automata are not the same but indeed defining the same rational expression.

c) Compute the transition semigroup of the given automaton and compute its generators.

Guava

→ Contains methods for computations with codes.

These exercises are inspired by Michel Lavrauw's exercise sheet from the Spring GAP Days 2025. More exercises for the Guava can be found here.

Exercise 1

a) Define a linear code over $GF(3)$ using the generator matrix:
$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}$

b) Compute its length, dimension and minimum distance.

c) Encode the message $(1, 2)$ and decode the received word $(1, 2, 0, 2)$

d) Introduce one error in the codeword and attempt to decode it.

Digraphs

$\rightarrow$ Contains methods for graphs, digraphs, and multidigraphs.

Exercise 1

a) Construct the Petersen graph in two different ways (use *DigraphSymmetricClosure* to obtain a symmetric digraph) and test if your result is correct by using the build-in function *PetersenGraph*.

b) Test if the Petersen graph is planar, hamiltonian, 3-vertex colourable and contains a K_4 as subgraph. Compute the automorphism group of the Petersen graph and its structure description.

c) How can you check whether a given graph is 3-regular?

d) Visualise the Petersen graph by using the methods *Splash* and *DotDigraph*.

Exercise 2*

Let $G = (V, E)$ be a simple graph. The line graph $L(G) = (V', E')$ of G is a graph whose vertices correspond to the edges of G, with two vertices in $L(G)$ being adjacent if and only if their corresponding edges in G share a common vertex. Hence, this construction transforms edge adjacency in G into vertex adjacency in $L(G)$. Implement a function that takes a graph G as input and returns its line graph $L(G)$. You can confirm that your implementation is correct by using the built-in function *LineUndirectedDigraph*.