The Matroids and Hypergraphs Packages in Maple 2024

- Maple 2024 adds a new package for dealing with Matroids and a new package for dealing with Hypergraphs.

Matroids

- A matroid is an abstract mathematical object which encodes the notion of *independence*. It has relevant applications in graph theory, linear algebra, geometry, topology, network theory, and more. Matroid theory is a thriving area of research.
- The simplest way to construct a matroid is via a matrix. Matroids constructed this way are called *linear* or *representable*.

```
> A := Matrix([[1,-1,0,1],[1,1,1,0],[1,1,0,1]]);
```

$$A := \begin{bmatrix} 1 & -1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

```
> with(Matroids);
```

$[AreIsomorphic, Bases, CharacteristicPolynomial, Circuits, Contraction, Deletion, DependentSets, Dual, ExampleMatroids, Flats, GroundSet, Hyperplanes, IndependentSets, IsMinorOf, Matroid, Rank, SetDisplayStyle, TuttePolynomial]$

```
> M := Matroid(A);
```

$$M := \left\langle \begin{array}{c} \text{the linear matroid whose ground set is the set of column vectors of the matrix:} \\ \begin{bmatrix} 1 & -1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array} \right\rangle$$

- This matroid encodes the linear dependencies among the columns of A. The so-called *ground set* of the matroid consists of the numbers 1 through 4, interpreted as column indices into A.
- We can ask for which subsets of columns are:
  - linearly independent,
  - linearly dependent, and
  - bases for the column space of A.

```
> IndependentSets(M);
```

$$[\varnothing, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 4\}, \{2, 4\}, \{3, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{2, 3, 4\}]$$

```
> DependentSets(M);
```

$$[\{1,3,4\},\{1,2,3,4\}]$$

```
> Bases(M);
```

$$[\{1,2,3\},\{1,2,4\},\{2,3,4\}]$$

- These answers change if the column vectors are considered over a finite field, e.g. the field with two elements:

```
> Mmodular := Matroid(A,2);
```

$$Mmodular := \left\langle \begin{array}{c} \text{the linear matroid whose ground set is the set of column vectors of the matrix:} \\ \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \mathbf{mod}\ 2 \end{array} \right\rangle$$

```
> Bases(Mmodular);
```

$$[\{1,3\},\{2,3\},\{1,4\},\{2,4\},\{3,4\}]$$

- Notice that the size of a basis changed from 3 to 2. This number is the *rank* of the matroid, which agrees with the familiar notion of rank (of the column space).

```
> Rank(M);
```

$$3$$

```
> Rank(Mmodular);
```

$$2$$

- Matroids are much more general than this! As an abstraction of independence, matroids also encode graph independence.
- Given a graph G, a subset of its edges are called dependent if they contain a path which forms a closed loop, known as a circuit.

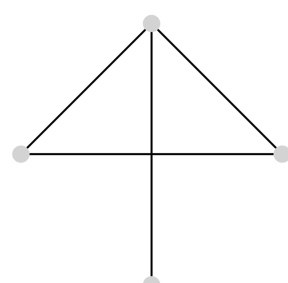
```
> with(GraphTheory):
```

```
> G := Graph({{a,b},{a,c},{b,d},{a,d}});
```

$G :=$ *Graph 1: an undirected graph with 4 vertices and 4 edge(s)*

```
> GraphicMatroid := Matroid(G);
```

$GraphicMatroid :=$ ⟨ the graphic matroid on the graph:



⟩

```
> Circuits(GraphicMatroid);
```

$$[\{"a\_b", "a\_d", "b\_d"\}]$$

- Inspired by linear algebra, one may take the definition of a basis as a maximal independent set. The bases of a graphic matroid are its spanning forests.

```
> Bases(GraphicMatroid);
```

$$[\{"a\_b", "a\_c", "a\_d"\}, \{"a\_b", "a\_c", "b\_d"\}, \{"a\_c", "a\_d", "b\_d"\}]$$

- In fact, every concept about linear independence coming from linear algebra (rank, bases, etc) can be axiomatized and interpreted for a graphic matroid.
- Conversely, the concept of a circuit from graph theory applies to linear matroids.

```
> Rank(GraphicMatroid);
```

$$3$$

```
> Circuits(M);
```

$$[\{1, 3, 4\}]$$

```
> Circuits(Mmodular);
```

$$[\{1, 2\}, \{1, 3, 4\}, \{2, 3, 4\}]$$

- This is the power of the abstraction of matroids. One rigorous definition of a matroid is as follows.
- A matroid is a pair $M = (E, I)$, where
  - E is a finite set called the *ground set* and
  - I is a collection of subsets of E called *independent sets* which satisfy the axioms:
    - (Axiom 1) The empty set is an independent set.
    - (Axiom 2) Every subset of an independent set is independent.
    - (Axiom 3) If $I1$ and $I2$ are independent sets and $I1$ has more elements than $I2$, then there exists an element of $I2$ which when included in $I1$ results in an independent set.
- The matroid package includes functionality for constructing a matroid directly from its independent sets:

```
> AxiomaticMatroid := Matroid([1,2,3], independentsets = [{},{1},{2},
  {3},{1,3},{2,3}]);
```

$$\mathit{AxiomaticMatroid} := \langle \text{a matroid on 3 elements with 5 independent sets} \rangle$$

- In fact, for each of the matroid properties of *independent sets*, *bases*, *dependent sets*, and *circuits* we have seen, one may construct a matroid (provided they satisfy certain axioms, listed on the Matroid help page).
- Each property uniquely determines the rest, and the matroids package supports several other axiomatic constructions (via *flats*, *hyperplanes*, or a *rank function*).
- Algorithms which convert between these representations are called *cryptomorphisms*. The matroids package showcases fast implementations of these algorithms.

```
> Circuits(AxiomaticMatroid);
```

$$[\{1,2\}]$$

```
> Bases(AxiomaticMatroid);
```

$$[\{1,3\},\{2,3\}]$$

- Beyond linear matroids constructed from a matrix, graphic matroids constructed from a graph, and general matroids constructed via axioms, the matroid package also features the construction of *algebraic matroids*, created from polynomial ideals.

```
> with(PolynomialIdeals):
> AlgebraicMatroid := Matroid(<x+y+z^2,z^2+y>);
```

$$AlgebraicMatroid := \left\langle \begin{array}{c} \text{the algebraic matroid on the polynomial ideal:} \\ \langle z^2+y, z^2+x+y\rangle \end{array} \right\rangle$$

```
> DependentSets(AlgebraicMatroid);
```

$$[\{1\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}]$$

- That $\{1\}$ is a dependent set indicates that there exists a polynomial in the ideal which involves only the first variable, x.
- The matroids package features a gallery of well-known matroids, which can be made available by loading the ExampleMatroids subpackage.

```
> with(ExampleMatroids);
```

$[Fano, Hesse, MacLane, NCubeMatroid, NonFano, NonPappus, Pappus, TicTacToe, UniformMatroid, Vamos]$

- Additionally, one may perform several operations on matroids:
- AreIsomorphic: determine if two matroids are the same, under some relabeling of the ground set;
- Deletion and Contraction: generalizations of deletion and contraction of edges of a graph;
- Dual: a generalization of the dual of a planar graph. Unlike for graphs, duals of matroids always exist. For linear matroids, duality corresponds to orthogonal complements of the row space.
- TuttePolynomial and CharacteristicPolynomial: polynomial invariants of matroids which generalize those of a graph;
- IsMinorOf: a test to check if one matroid can be obtained by another via a sequence of deletions and contractions.

```
> ContractionMatroid := Contraction(GraphicMatroid,{4});
```

$$ContractionMatroid := \langle \text{a matroid on 4 elements with 1 circuit}\rangle$$

```
> AreIsomorphic(ContractionMatroid,AxiomaticMatroid);
```

$$false$$

```
> IsMinorOf(ContractionMatroid,GraphicMatroid);
```

$$true, \varnothing, \varnothing$$

```
> Dual(M);
```

$$\langle \text{a matroid on 4 elements with 3 bases of size 1} \rangle$$

```
> Matroids:-TuttePolynomial(GraphicMatroid,x,y);
```

$$x^3 + x^2 + x\, y$$

```
> Matroids:-CharacteristicPolynomial(GraphicMatroid,k);
```

$$k^3 - 4\, k^2 + 5\, k - 2$$

Hypergraphs

- The Hypergraphs package is the computational backbone of the matroids package, and it is much more than that!
- A hypergraph is a pair (V, E) consisting of a finite set V called vertices and a collection E of subsets of V called hyperedges.
- Hypergraphs, as indicated by the name, generalize graphs: a graph can be thought of as a hypergraph where every hyperedge has size two (or size one if self-loops are allowed).
- We create a hypergraph with the Hypergraph command.

```
> with(Hypergraphs);
```

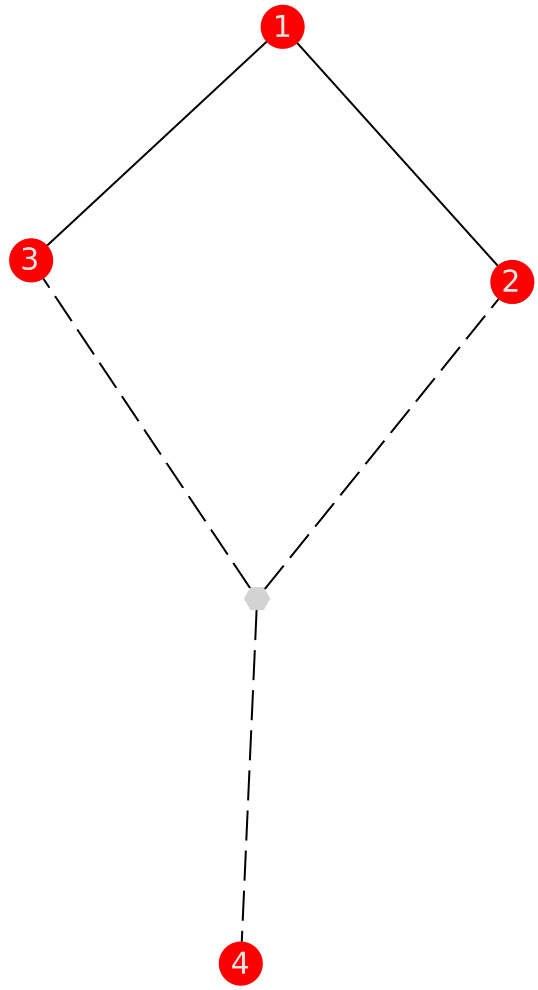
[*AddHyperedges*, *AddVertices*, *AntiRank*, *AreEqual*, *AreIsomorphic*, *ComplementHypergraph*, *DegreeProfile*, *Draw*, *DualHypergraph*, *ExampleHypergraphs*, *Hyperedges*, *Hypergraph*, *IsConnected*, *IsEdge*, *IsLinear*, *IsRegular*, *IsUniform*, *LineGraph*, *Max*, *Min*, *NumberOfHyperedges*, *NumberOfVertices*, *PartialHypergraph*, *Rank*, *SubHypergraph*, *Transversal*, *VertexEdgeIncidenceGraph*, *Vertices*]

```
> H := Hypergraph([1,2,3,4],[{1,2},{1,3},{2,3,4}]);
```

$$H := \; < \textit{a hypergraph on 4 vertices with 3 hyperedges} >$$

- For few vertices and hyperedges, one can visualize a hypergraph as an augmented graph.
- Distinguished nodes of the graph correspond to vertices of the hypergraph. Pairs of nodes are connected, as usual, if they form a (hyper)edge.
- Additional, auxiliary nodes are included for every hyperedge of size greater than two and auxiliary edges connect such nodes with the vertices they include.

```
> Draw(H);
```

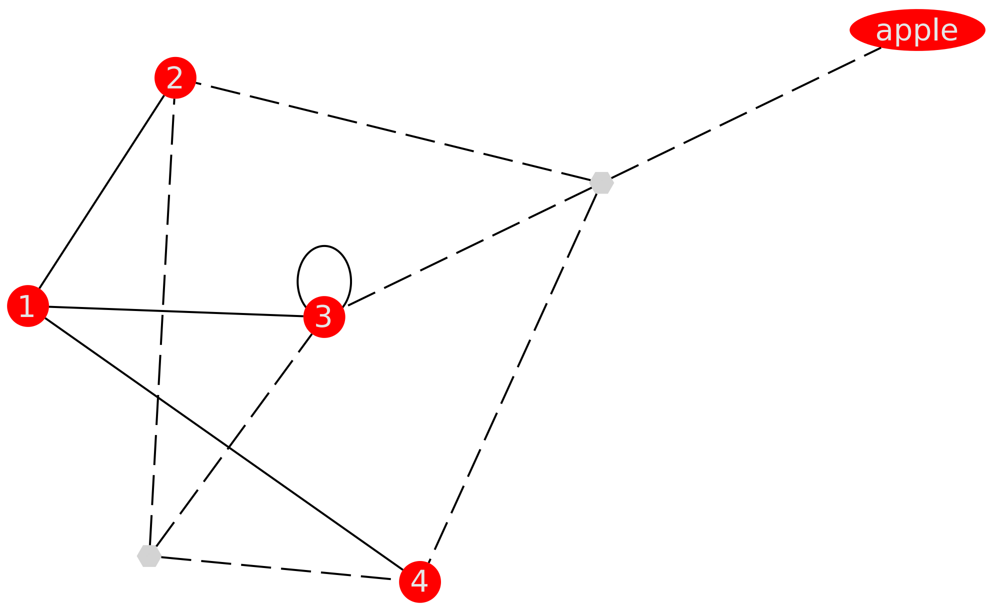


- Procedures for manipulating hypergraphs include AddHyperedges and AddVertices.
- Given a hypergraph, the functions ComplementHypergraph, DualHypergraph, and SubHypergraph create new hypergraphs in the ways the names suggest.
- Basic functionality such as Hyperedges, NumberOfHyperedges, Vertices, and NumberOfVertices are available, as are simple queries including AreEqual, IsConnected, and IsEdge.
- The functions DegreeProfile and VertexEdgeIncidenceGraph directly generalize those notions from graphs to hypergraphs.

```
> H2 := AddHyperedges(AddVertices(H,["apple"]),[{1,4},{2,"apple",3,4},
  {3}]);
```

$$H2 := < a\ hypergraph\ on\ 5\ vertices\ with\ 6\ hyperedges >$$

```
> Draw(H2);
```




```
> [AreEqual(H,H2), IsEdge(H2,{2,1}), NumberOfHyperedges(H2),
  Hypergraphs:-NumberOfVertices(H2), Hypergraphs:-IsConnected(H2),
  DegreeProfile(H)];
```

$$[\mathit{false}, \mathit{true}, 6, 5, \mathit{true}, [2, 2, 2, 1]]$$

- The major advancement in Maple with the hypergraphs package has to do with what goes on behind the scenes.
- Subsets are carefully encoded using bit-vectors to make hefty calculations fast and feasible.

```
> with(ExampleHypergraphs);
```

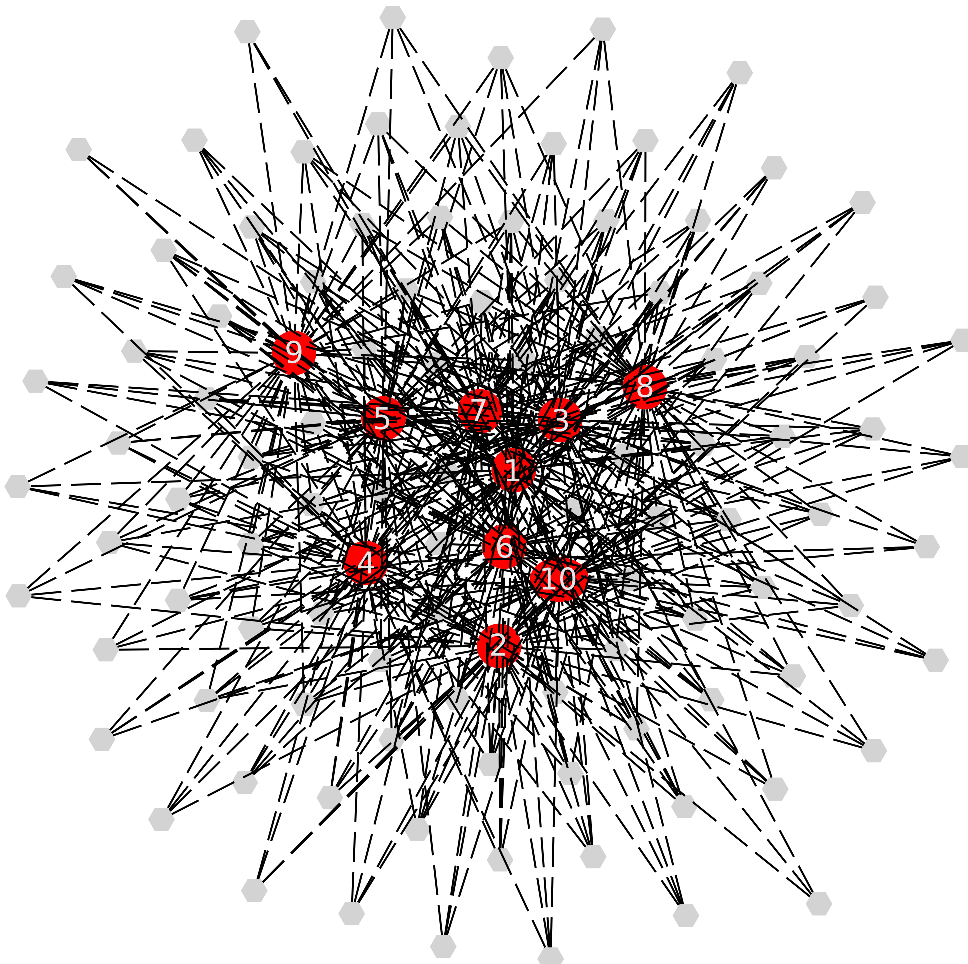
$$[\mathit{Fan}, \mathit{Kuratowski}, \mathit{Lovasz}, \mathit{NonEmptyPowerSet}, \mathit{RandomHypergraph}]$$

- Below, we illustrate the core hypergraph algorithms on a random hypergraph on 10 vertices with 100 hyperedges.

```
> R := RandomHypergraph(10,100);
```

$$R := < \textit{a hypergraph on 10 vertices with 100 hyperedges} >$$

> Draw(R);



- The Min function computes the hyperedges which do not properly contain another hyperedge.
- The Max function computes those which are not properly contained in another hyperedge.
- The Transversal function computes the sets of vertices for which every hyperedge contains some element in that set.

> Hyperedges(Min(R));

$[\{6, 7, 9\}, \{2, 3, 10\}, \{7, 9, 10\}, \{1, 2, 4, 5\}, \{1, 4, 5, 7\}, \{1, 4, 6, 7\}, \{1, 3, 4, 8\}, \{1, 3, 7, 8\}, \{2, 3, 7, 8\}, \{1, 3, 4, 9\}, \{2, 4, 5, 9\}, \{2, 3, 6, 9\}, \{1, 3, 8, 9\}, \{3, 5, 8, 9\}, \{1, 2, 4, 10\}, \{1, 4, 5, 10\}, \{2, 4, 5, 10\}, \{1, 3, 6, 10\}, \{2, 5, 6, 10\}, \{1, 3, 7, 10\}, \{2, 4, 7, 10\}, \{1, 2, 8, 10\}, \{1, 3, 8, 10\}, \{3, 4, 8, 10\}, \{4, 6, 8, 10\}, \{6, 7, 8, 10\}, \{1, 4, 9, 10\}, \{1, 2, 3, 5, 7\}, \{1, 3, 5, 6, 7\}, \{2, 3, 5, 6, 7\}, \{3, 4, 5, 6, 7\}, \{1, 2, 3, 5, 8\}, \{2, 4, 6, 7, 8\}, \{1, 5, 6, 7, 8\}, \{3, 4, 5, 6, 9\}, \{2, 3, 5, 7, 9\}, \{3, 4, 7, 8, 9\}, \{1, 5, 6, 8, 10\}, \{1, 5, 6, 9, 10\}]$

```
> Hyperedges(Max(R));
```

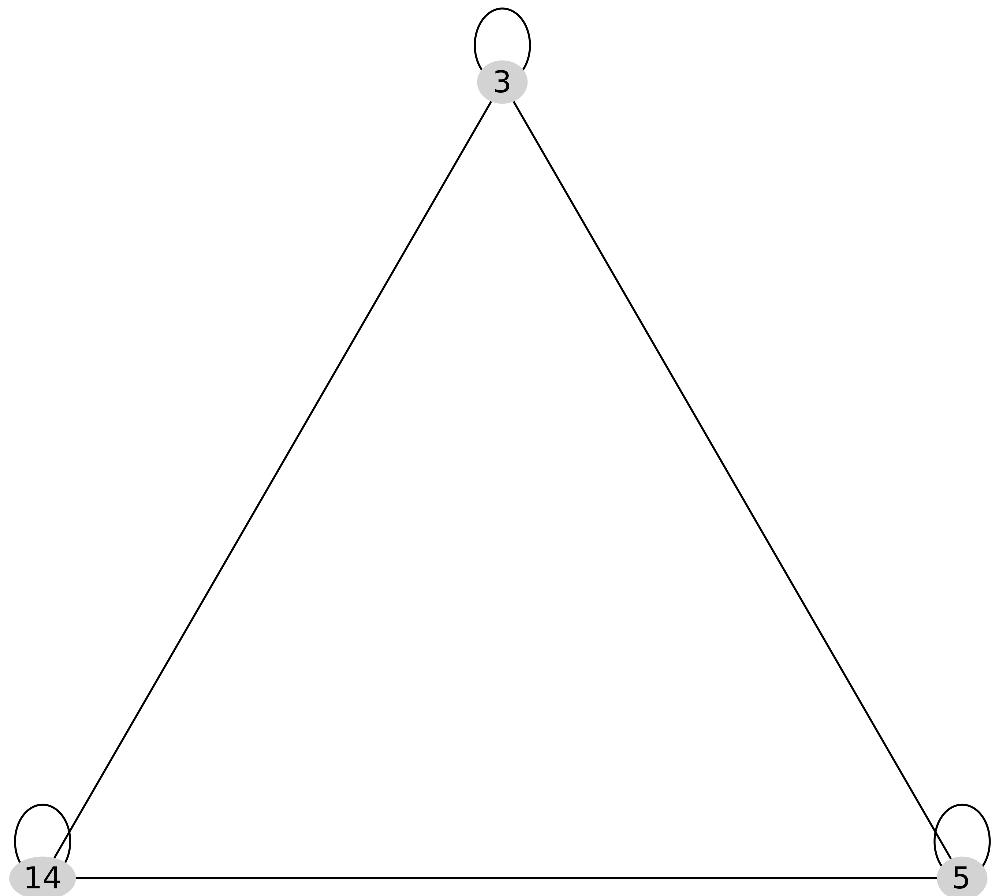
[{2, 4, 5, 10}, {1, 2, 4, 5, 6}, {1, 2, 3, 5, 7}, {2, 4, 5, 7, 9}, {1, 2, 6, 7, 9}, {1, 2, 4, 9, 10}, {1, 2, 5, 6, 7, 8}, {1, 2, 3, 4, 5, 9}, {2, 3, 5, 6, 7, 9}, {2, 3, 4, 5, 8, 9}, {1, 3, 4, 5, 6, 10}, {2, 3, 4, 6, 7, 10}, {1, 2, 5, 6, 7, 10}, {1, 4, 5, 6, 7, 10}, {1, 2, 4, 6, 8, 10}, {2, 3, 4, 7, 8, 10}, {1, 2, 5, 7, 8, 10}, {3, 4, 5, 7, 8, 10}, {2, 4, 6, 7, 8, 10}, {1, 3, 5, 6, 9, 10}, {2, 3, 6, 7, 9, 10}, {2, 3, 6, 8, 9, 10}, {1, 5, 6, 8, 9, 10}, {1, 2, 3, 4, 6, 8, 9}, {1, 3, 5, 6, 7, 8, 9}, {3, 4, 5, 6, 7, 8, 9}, {1, 2, 3, 5, 6, 8, 10}, {1, 2, 3, 6, 7, 8, 10}, {1, 3, 4, 5, 7, 9, 10}, {3, 4, 5, 6, 8, 9, 10}, {1, 4, 5, 7, 8, 9, 10}, {2, 5, 6, 7, 8, 9, 10}, {1, 3, 4, 6, 7, 8, 9, 10}]

```
> Hyperedges(Transversal(R));
```

[{3, 4, 6, 10}, {3, 5, 6, 10}, {2, 3, 7, 10}, {3, 4, 7, 10}, {3, 5, 7, 10}, {1, 7, 9, 10}, {1, 2, 3, 4, 7}, {1, 2, 4, 5, 7}, {1, 3, 4, 5, 7}, {2, 3, 4, 5, 7}, {1, 2, 3, 6, 7}, {1, 3, 4, 6, 7}, {2, 3, 4, 6, 7}, {1, 3, 5, 6, 7}, {1, 2, 3, 7, 8}, {1, 2, 4, 7, 8}, {1, 2, 5, 7, 8}, {1, 3, 5, 7, 8}, {3, 4, 5, 7, 8}, {1, 2, 6, 7, 8}, {2, 4, 6, 7, 8}, {3, 4, 6, 7, 8}, {1, 2, 3, 6, 9}, {1, 2, 4, 6, 9}, {1, 3, 4, 6, 9}, {2, 3, 4, 6, 9}, {2, 3, 5, 6, 9}, {1, 2, 4, 7, 9}, {1, 2, 3, 8, 9}, {1, 2, 4, 8, 9}, {2, 3, 4, 8, 9}, {1, 2, 5, 8, 9}, {3, 4, 5, 8, 9}, {1, 2, 6, 8, 9}, {3, 4, 6, 8, 9}, {1, 2, 7, 8, 9}, {1, 2, 3, 6, 10}, {1, 2, 5, 7, 10}, {1, 5, 6, 7, 10}, {1, 2, 6, 8, 10}, {2, 4, 6, 8, 10}, {1, 5, 6, 8, 10}, {4, 5, 6, 8, 10}, {2, 4, 7, 8, 10}, {4, 6, 7, 8, 10}, {1, 2, 3, 9, 10}, {1, 2, 4, 9, 10}, {1, 3, 4, 9, 10}, {1, 2, 5, 9, 10}, {3, 4, 5, 9, 10}, {1, 2, 6, 9, 10}, {1, 3, 6, 9, 10}, {2, 4, 7, 9, 10}, {4, 5, 7, 9, 10}, {1, 3, 8, 9, 10}, {3, 4, 8, 9, 10}, {1, 5, 8, 9, 10}, {4, 5, 8, 9, 10}, {1, 6, 8, 9, 10}, {5, 6, 8, 9, 10}, {2, 7, 8, 9, 10}, {4, 7, 8, 9, 10}, {5, 7, 8, 9, 10}, {2, 3, 5, 7, 8, 9}, {2, 5, 6, 7, 8, 9}, {1, 2, 4, 5, 6, 10}]

- Put another way, consider the hypergraph $Food$ whose vertices are ingredients in your kitchen, and whose hyperedges are recipes.
- Then $Min(Food)$ are those recipes which require a minimal set of ingredients (i.e. removing any ingredient prevents any recipe from being made).
- $Max(Food)$ are those recipes which maximally use ingredients (i.e. you cannot include an additional ingredient to make a bigger recipe).
- $Transversal(Food)$ are all sets of ingredients an adversary could steal from your fridge which would prevent you from making any recipe.
- In the context of matroids, the sets of subsets that can be used to define a matroid axiomatically are all hypergraphs, and they are stored as such if they are known for a given matroid. Several cryptomorphisms come directly from these hypergraph operations. For example, the Circuits of a matroid M are just $Min(DependentSets(M))$.
- Below, we illustrate the remaining functionality and invite you to check out the details on our help pages!

```
> DrawGraph(Hypergraphs:-LineGraph(H));
```




```
> [Rank(H),AntiRank(H)];
```

$[3, 2]$

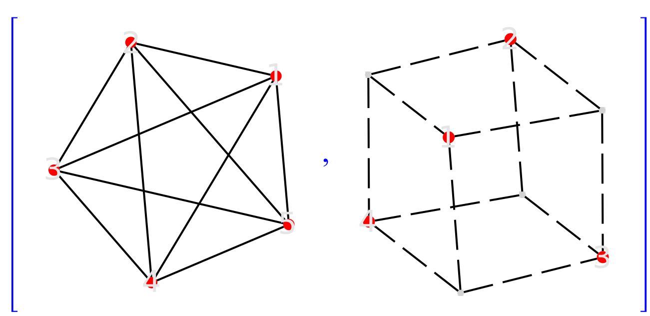
```
> [IsLinear(H),IsRegular(H),IsUniform(H)];
```

[*true*, *false*, *false*]

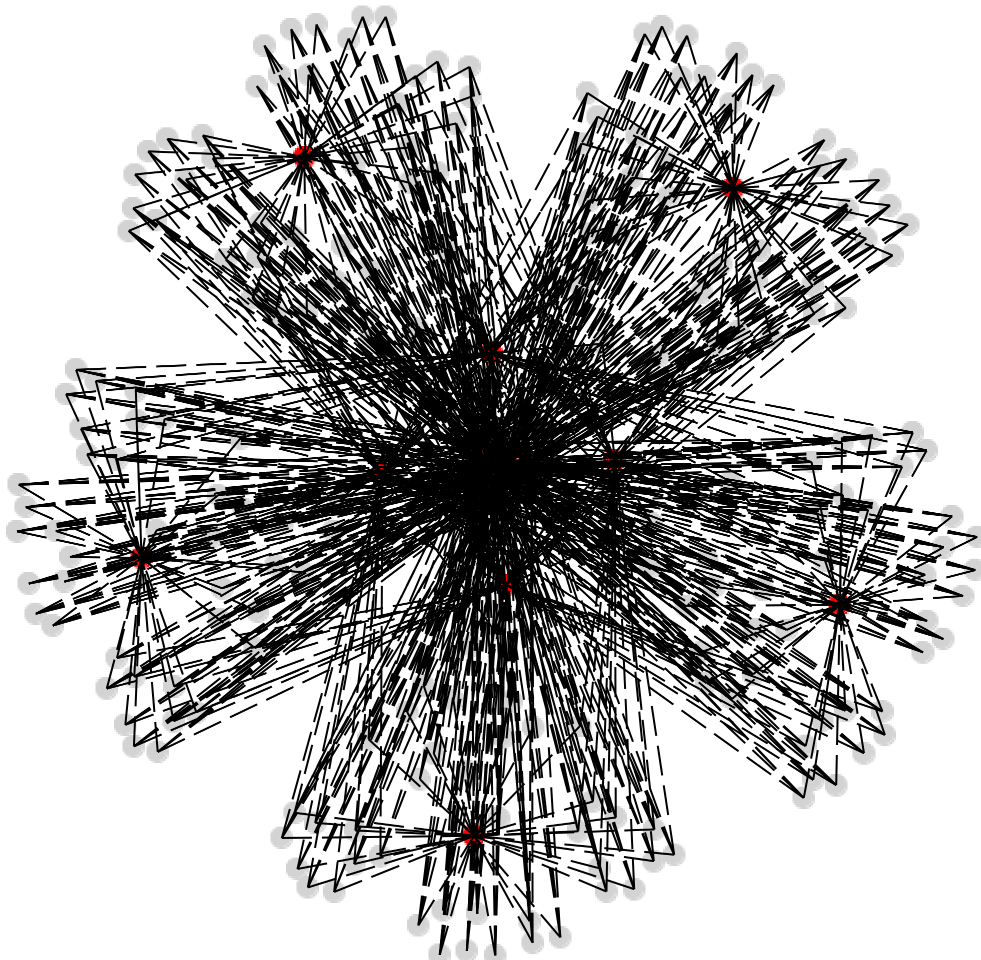
```
> with(ExampleHypergraphs);
```

[*Fan*, *Kuratowski*, *Lovasz*, *NonEmptyPowerSet*, *RandomHypergraph*]

```
> [Draw(Kuratowski({1,2,3,4,5},2)),Draw(Kuratowski({1,2,3,4},3))];
```



```
> Draw(Lovasz(5));
```



```
> NumberOfHyperedges(Lovasz(5));
```

206