A substantial effort was put into Graph Theory for Maple 2024, including new commands for graph computation and generation.

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

### New commands

These commands are new for Maple 2024:

- `AllGraphs`
- `Condensation`
- `FindAsteroidalTriple`
- `IsArchimedeanGraph`
- `IsAsteroidalTripleFree`
- `IsDominatingSet`
- `IsIndependentSet`
- `IsStronglyConnected`
- `IsHedgehog`
- `IsPancyclic`
- `IsTree`
- `MinCut`
- `MoralGraph`
- `RelationGraph`
- `WienerIndex`

### AllGraphs

The new `AllGraphs` command returns an iterator with which one can step through all graphs matching a particular set of criteria, such as vertex count or edge count. It is also possible to cause the Iterator to return only connected graphs or only graphs which are not isomorphic to a graph previously returned by this Iterator.

```maple
> iterator := AllGraphs(vertices = 3, nonisomorphic = true);
```

### Condensation

The new command `Condensation` computes the condensation of a given (directed) graph.

The condensation of a graph $G$ is a graph $C$ whose vertices correspond to the strongly connected components of $G$. The condensation $C$ will have an arc from $u$ to $v$ whenever there is an arc from the strongly connected component of $G$ corresponding to $u$ to the strongly connected component of $G$ corresponding to $v$.

```maple
> T := Digraph({[1, 2], [1, 3], [2, 3], [3, 2], [4, 3]});
> C := Condensation(T);
```

### FindAsteroidalTriple and IsAsteroidalTripleFree

The new command `FindAsteroidalTriple` searches for an asteroidal triple in the given graph. The related new command `IsAsteroidalTripleFree` returns true when the given graph does not contain an asteroidal triple.

An asteroidal triple for a graph $G$ is a triple $(u, v, w)$ of non-adjacent vertices of $G$ such that for each pair from the triple, there exists a path between them that does not intersect the third vertex or any of its neighbors.

```maple
> C6 := CycleGraph(6);
```
IsArchimedeanGraph

The new command `IsArchimedeanGraph` tests whether a given graph is an Archimedean graph. The Archimedean graphs are those graphs which form the skeletons of the Archimedean solids. The Archimedean solids comprise 13 convex polyhedra whose faces are regular polygons and whose vertices are all symmetric to each other, but which are not Platonic solids (polyhedra whose faces are identical).

IsArchimedeanGraph(SpecialGraphs::SnubCubeGraph());

true

IsDominatingSet

The new command `IsDominatingSet` tests whether a set is a dominating set for a given graph. A dominating set of a graph $G$ is a subset $S$ of the vertices of $G$, with the condition that every vertex in $G$ must either be an element of $S$ or the neighbor of an element of $S$.

P5 := PathGraph(5)

S3 := {1, 3, 5}

IsDominatingSet(P5, S3);

true

MinCut

The new `MinCut` command works per the Max-Flow Min-Cut Theorem and uses the flow output to compute a cut-set. The `EdgeConnectivity` and `VertexConnectivity` commands have been updated to use MinCut so that they can now also return cut-sets.

MG := Graph(\{(1, 2), (1, 3), (2, 3), (2, 4), (3, 4), (3, 5), (3, 6), (3, 7), (4, 7)\})

MG := Graph 8: an undirected graph with 7 vertices and 9 edge(s)

(ms.flowM) := MaxFlow(MG, 2, 7); i

The MinCut command will call MaxFlow by default but can reuse the flow matrix if MaxFlow has been called already.

MinCut(MG, 2, 7, flows = flowM);

\{(3, 7), (4, 7)\}

EdgeConnectivity(MG, output = cutset);

\{(3, 5)\}

VertexConnectivity(MG, output = cutset);

\{3\}
MoralGraph

The *moral graph* of a directed graph $G$ is a graph $M$ with the same vertices as $G$ but with all directed edges made undirected, and with the property that whenever two directed edges in $G$ share a destination vertex, then their source vertices are connected in $M$.

The new command `MoralGraph` constructs the moral graph given a directed graph.

\[
DG := \text{Graph}( \{[1,3],[2,3],[2,4],[3,5],[3,6],[3,7],[4,7]\}, \text{vertexpositions} = \{[1,2], [2,2], [1,1], [2,1], [0,0], [1,0], [2,0]\})
\]

\[
DG := \text{Graph 9: a directed graph with 7 vertices and 7 arc(s)}
\]

\[
MG := \text{MoralGraph}(DG)
\]

\[
MG := \text{Graph 10: an undirected graph with 7 vertices and 9 edge(s)}
\]

\[
\text{NumberOfEdges}(MG) = 9
\]

\[
\text{SetVertexPositions}(MG, \text{GetVertexPositions}(DG))
\]

\[
\text{plots:display}(\text{DrawGraph}-(DG|MG))
\]

RelationGraph

The *RelationGraph* command constructs a graph on a given vertex list in which an edge is present in the graph whenever a particular Boolean predicate on its two vertices is true.

\[
RG := \text{RelationGraph}(\text{seq} 1 \ldots 16, \text{NumberTheory:-AreCoprime})
\]

\[
RG := \text{Graph 11: an undirected graph with 16 vertices and 79 edge(s)}
\]
In this example, the neighborhood of the element 5 will be all those vertices whose values are coprime with 5:

\[
\text{Neighborhood}(RG, 5) = [1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16]
\]

\[\text{WienerIndex}\]

The \text{WienerIndex} command computes the Wiener index on a given graph. The Wiener index of a graph is the sum of the lengths of the shortest paths between all pairs of vertices in the graph.

\[G := \text{CycleGraph}(20)\]

\[G := \text{Graph 12: an undirected graph with 20 vertices and 20 edge(s)}\]

\[\text{WienerIndex}(G)\]

\[40\]

\[\text{New functionality for existing commands}\]

\[\text{Distance and ShortestPath}\]

The \text{Distance} and \text{ShortestPath} commands now use the edge weights of a weighted matrix. They both have a new option \text{ignoreweights} to compute the distance or shortest path in the underlying unweighted graph.

\[W6 := \text{Graph( \{[[1,2],10],[[1,6],6],[[2,3],10],[[3,4],10],[[4,5],10],[[5,6],5]]\);}\]

\[W6 := \text{Graph 13: a directed weighted graph with 6 vertices and 6 arc(s)}\]
\begin{itemize}
  \item \texttt{DrawGraph(W6, layout = circle)};
  \begin{figure}
    \centering
    \includegraphics[width=0.5\textwidth]{archimedean_graph.png}
  \end{figure}
  \end{itemize}

\begin{itemize}
  \item \texttt{ShortestPath(W6, 1, 6)};
    \texttt{[1, 2, 3, 4, 5, 6]}
  \end{itemize}

\begin{itemize}
  \item \texttt{Distance(W6, 1, 6)};
    \texttt{5}
  \end{itemize}

\begin{itemize}
  \item \texttt{ShortestPath(W6, 1, 6, 'ignoreweights')};
    \texttt{[1, 6]}
  \item \texttt{Distance(W6, 1, 6, 'ignoreweights')};
    \texttt{1}
  \end{itemize}

\section*{MaxFlow}

The \texttt{MaxFlow} command now works on all graphs. It now treats an unweighted graph as a graph with all edge weights equal to 1, or to the edge multiplicity for multigraphs.

\section*{Additions to SpecialGraphs}

The \texttt{SpecialGraphs} subpackage now includes commands for all the Archimedean graphs, as well as the Möbius ladder graph and the Wagner graph.

\section*{Archimedean Graphs}

The \texttt{Archimedean graphs} are those graphs which form the skeletons of the Archimedean solids. The Archimedean solids comprise 13 convex polyhedra whose faces are regular polygons and whose vertices are all symmetric to each other, but which are not Platonic solids (polyhedra whose faces are identical). They were first enumerated by Archimedes.

The new command \texttt{IsArchimedeanGraph} tests whether a given graph is an Archimedean graph.

The following new commands generate each of the Archimedean graphs.
<table>
<thead>
<tr>
<th>Archimedean solid</th>
<th>Degree</th>
<th>Vertices</th>
<th>Edges</th>
<th>Command</th>
<th>2-D</th>
<th>3-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>truncated tetrahedron</td>
<td>3</td>
<td>12</td>
<td>18</td>
<td>&gt; G := SpecialGraphs:-TruncatedTetrahedronGraph(</td>
<td>➤ DrawPlanar(G, showlabels = false)</td>
<td>➤ DrawGraph(G, dimension = 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>);</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G :=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Graph 14: an undirected graph with 12 vertices and 18 edge(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuboctahedron</td>
<td>4</td>
<td>12</td>
<td>24</td>
<td>&gt; letters := [seq(&quot;ABCDEFGHIJKLMNOPQRSTUVWXYZ&quot;) ]</td>
<td>➤ DrawPlanar(G, showlabels = false)</td>
<td>➤ DrawGraph(G, dimension = 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&gt; G := SpecialGraphs:-CuboctahedronGraph(</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>letters)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G :=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Graph 15: an undirected graph with 12 vertices and 24 edge(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncated cube</td>
<td>3</td>
<td>24</td>
<td>36</td>
<td>&gt; G := SpecialGraphs:-TruncatedCubeGraph( )</td>
<td>➤ DrawPlanar(G, showlabels = false)</td>
<td>➤ DrawGraph(G, dimension = 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G :=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Graph 16: an undirected graph with 24 vertices and 36 edge(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncated octahedron</td>
<td>3</td>
<td>24</td>
<td>36</td>
<td>&gt; G := SpecialGraphs:-TruncatedOctahedronGraph( )</td>
<td>➤ DrawPlanar(G, showlabels = false)</td>
<td>➤ DrawGraph(G, dimension = 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G :=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Graph 17: an undirected graph with 24 vertices and 36 edge(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>small rhombicuboctahedron</td>
<td>4</td>
<td>24</td>
<td>48</td>
<td>&gt; G := SpecialGraphs:-SmallRhombicuboctahedron.Graph( )</td>
<td>➤ DrawPlanar(G, showlabels = false)</td>
<td>➤ DrawGraph(G, dimension = 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G :=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Graph 18: an undirected graph with 24 vertices and 48 edge(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name of Polyhedron</td>
<td>Vertices</td>
<td>Edges</td>
<td>Dimension</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------</td>
<td>-------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Great Rhombicuboctahedron (also called truncated cuboctahedron)</td>
<td>3</td>
<td>48</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Snub Cube</td>
<td>5</td>
<td>24</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Icosidodecahedron</td>
<td>4</td>
<td>30</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Truncated Dodecahedron</td>
<td>3</td>
<td>60</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Truncated Icosahedron</td>
<td>3</td>
<td>60</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Möbius ladder graph and Wagner graph

The **Möbius ladder graph** on $2n$ vertices may be understood visually as the **ladder graph** on the same vertices, with the addition of two edges: one connecting the left-hand side of the top rung to the right-hand side of the bottom rung, and one connecting the right-hand side of the top rung to the left-hand side of the bottom rung.

\[ \text{Möbius ladder graph on } 2n \text{ vertices} \]

\[ \text{ladder graph on the same vertices, with the addition of two edges: one connecting the left-hand side of the top rung to the right-hand side of the bottom rung, and one connecting the right-hand side of the top rung to the left-hand side of the bottom rung.} \]

#### Small rhombicosidodecahedron

<table>
<thead>
<tr>
<th>Small rhombicosidodecahedron</th>
<th>4</th>
<th>60</th>
<th>120</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G := \text{SpecialGraphs::SmallRhombicosidodecahedronGraph}()$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$G := \text{Graph 24: an undirected graph with 60 vertices and 120 edge(s)}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Small Rhombicosidodecahedron Graph](image)

#### Great rhombicosidodecahedron (also called truncated icosidodecahedron)

<table>
<thead>
<tr>
<th>Great rhombicosidodecahedron</th>
<th>3</th>
<th>120</th>
<th>180</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G := \text{SpecialGraphs::GreatRhombicosidodecahedronGraph}()$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$G := \text{Graph 25: an undirected graph with 120 vertices and 180 edge(s)}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Great Rhombicosidodecahedron Graph](image)

#### Snub dodecahedron

<table>
<thead>
<tr>
<th>Snub dodecahedron</th>
<th>5</th>
<th>60</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G := \text{SpecialGraphs::SnubDodecahedronGraph}()$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$G := \text{Graph 26: an undirected graph with 60 vertices and 150 edge(s)}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Snub Dodecahedron Graph](image)

\[ \text{Möbius ladder graph on } 6 \text{ vertices} \]

\[ \text{ladder graph on the same vertices, with the addition of two edges: one connecting the left-hand side of the top rung to the right-hand side of the bottom rung, and one connecting the right-hand side of the top rung to the left-hand side of the bottom rung.} \]

\[ \text{WG : } ML3 := \text{SpecialGraphs::MoebiusLadderGraph}(3) \]

\[ ML3 := \text{Graph 27: an undirected graph with 6 vertices and 9 edge(s)} \]
The Wagner graph is a 3-regular graph which is a particular instance of an Möbius ladder graph on 8 vertices.

\[ \text{DrawGraph(ML3, stylesheet = \{verexpadding = 8\})} \]

\[ \text{DrawGraph(WG, stylesheet = \{verexpadding = 8\})} \]

\[ \text{WG := SpecialGraphs\_WagnerGraph()} \]

\[ \text{WG := Graph 28: an undirected graph with 8 vertices and 12 edge(s)} \]