GraphTheory Updates in Maple 2025

Description

A substantial effort was put into Graph Theory for Maple 2025, including new commands for graph computation and generation.

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

New commands

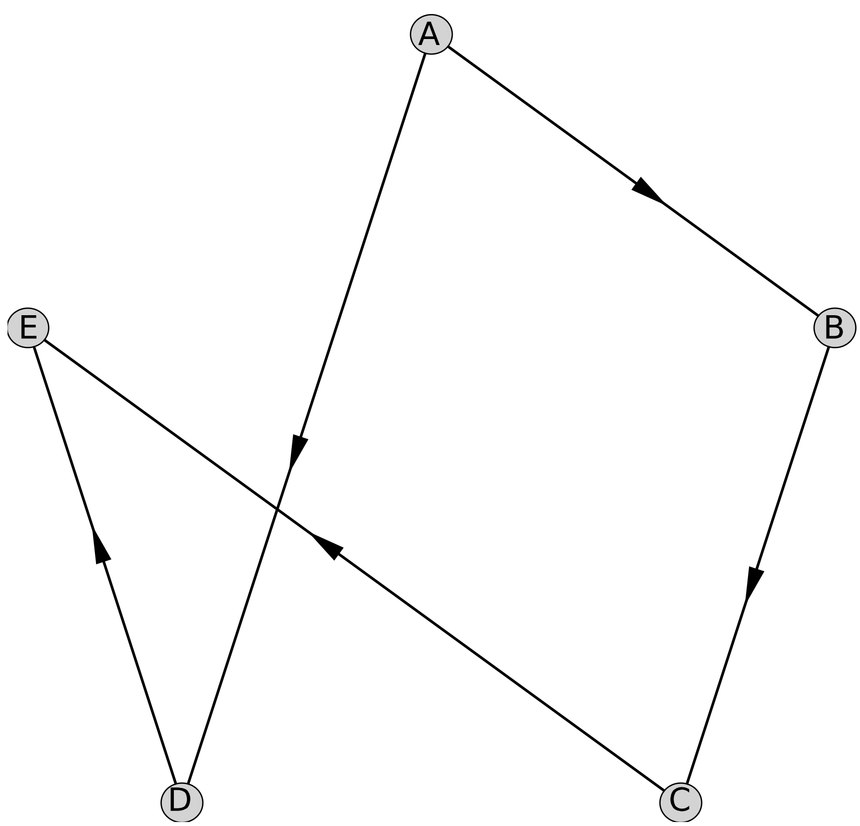
AllSimplePaths

The new AllSimplePaths command returns an iterator with which one can step through all paths from a given vertex to another vertex in a directed graph.

```
> G1 := Graph({["A", "B"], ["A", "D"], ["B", "C"], ["C", "E"], ["D", "E"]});
```

G1 := Graph 1: a directed graph with 5 vertices and 5 arcs

```
> DrawGraph(G1);
```



```
> iterator := AllSimplePaths( G1, "A", "E" );
```

$$iterator := \left[\quad Path\ Iterator \quad \right]$$

```
> iterator:-getNext();
```

["A", "D", "E"]

```
> iterator:-getNext();
```

["A", "B", "C", "E"]

```
> iterator:-hasNext();
```

false

Ancestors and Descendants

The new Ancestors command returns a list of ancestors of the given vertex in the given directed graph. The related new command Descendants returns a list of descendants of the given vertex.

```
> Ancestors( G1, "A" );
```

[ ]

```
> Ancestors( G1, "E" );
```

["A", "B", "C", "D"]

```
> Descendants( G1, "A" );
```

["B", "C", "D", "E"]

FindCycle

The new FindCycle command finds a cycle, if one exists in the given graph.

```
> FindCycle(G1);
```

[ ]

```
> FindCycle( Graph( {["A", "B"], ["B", "C"], ["C", "A"]} ) );
```

["C", "A", "B", "C"]

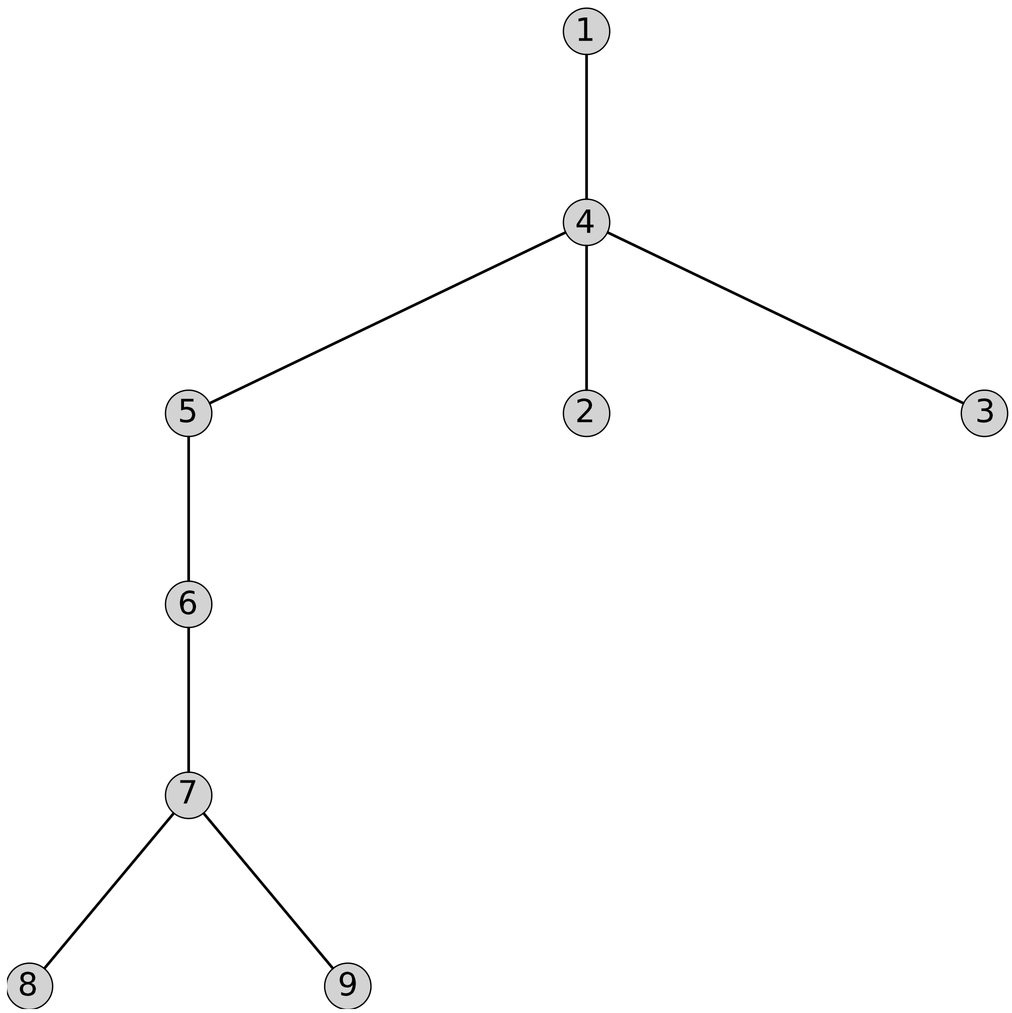
IsCaterpillarTree and IsLobsterTree

The new IsCaterpillarTree command tests whether the graph is a caterpillar tree, a tree for which there is some path such that every vertex is either on the path or the neighbor of a vertex on the path.

```
> CT := Graph({{1,4},{2,4},{3,4},{4,5},{5,6},{6,7},{7,8},{7,9}});
```

CT ≔ *Graph 2: an undirected graph with 9 vertices and 8 edges*

```
> DrawGraph(CT);
```




```
> IsCaterpillarTree(CT);
```

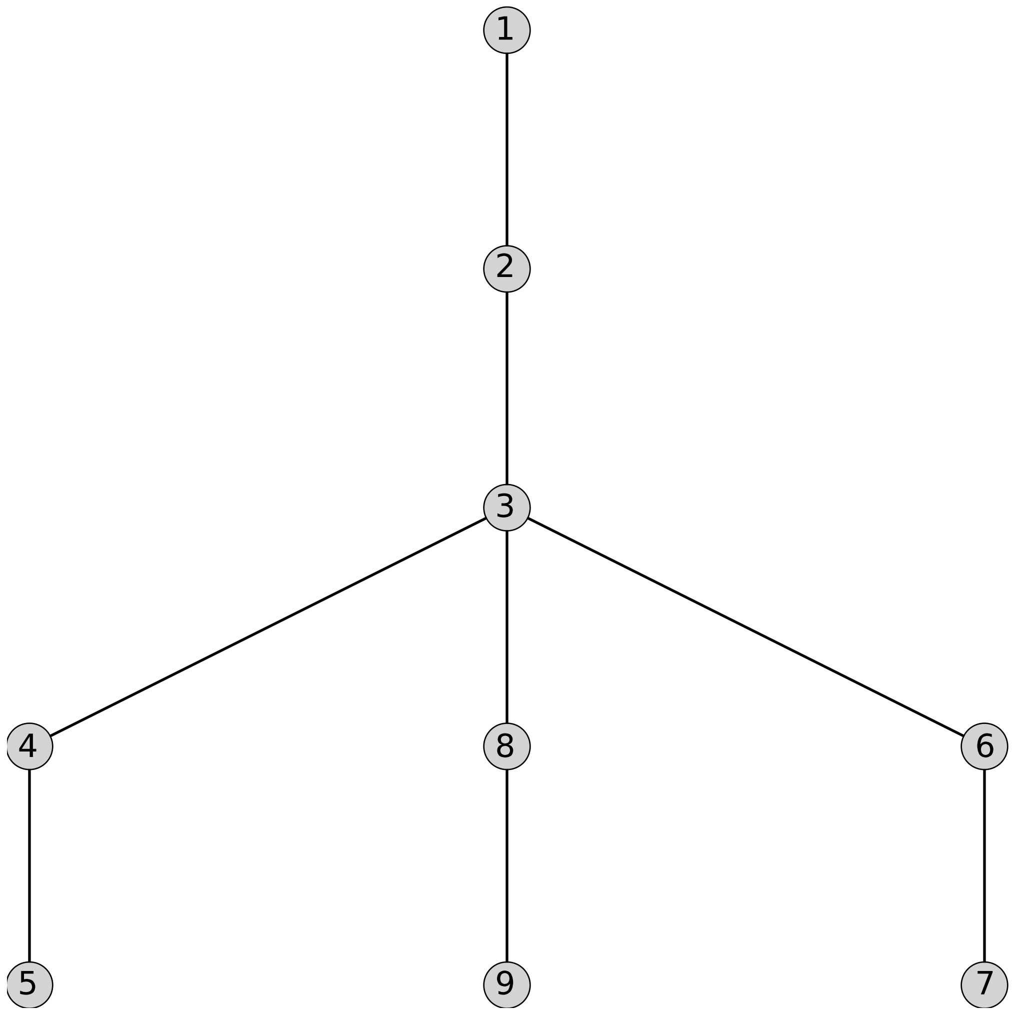
true

The new IsLobsterTree command tests whether the graph is lobster tree, a tree such that the result of removing all leaf vertices is a caterpillar tree.

```
> LT := Graph({{1,2},{2,3},{3,4},{4,5},{3,6},{6,7},{3,8},{8,9}});
```

LT ≔ *Graph 3: an undirected graph with 9 vertices and 8 edges*

```
> DrawGraph(LT);
```




```
> IsLobsterTree(LT);
```

true

```
> IsCaterpillarTree(LT);
```

false

IsPlatonicGraph

The new IsPlatonicGraph command tests whether the graph is Platonic. The Platonic graphs consist of those graphs whose skeletons are the Platonic solids (polyhedra whose faces are identical).

```
> IsPlatonicGraph( SpecialGraphs:-CubeGraph() );
```

true

LongestPath

The new LongestPath command computes the longest path within a given (directed) graph.

```
> LongestPath(G1);
```

$$["A", "B", "C", "E"]$$

LowestCommonAncestors

The new LowestCommonAncestors command computes the set of lowest common ancestors in a given directed graph.

```
> LowestCommonAncestors( G1, "C", "D" );
```

$$\{"A"\}$$

ModularityMatrix

The new ModularityMatrix command computes the modularity matrix of the graph G.

```
> ModularityMatrix(G1);
```

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ -\frac{2}{5} & -\frac{1}{5} & \frac{4}{5} & -\frac{1}{5} & 0 \\ -\frac{2}{5} & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} & 1 \\ -\frac{2}{5} & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} & 1 \\ -\frac{4}{5} & -\frac{2}{5} & -\frac{2}{5} & -\frac{2}{5} & 0 \end{bmatrix}$$

ResistanceDistance

The new ResistanceDistance command computes the resistance distance of the graph G.

```
> ResistanceDistance(SpecialGraphs:-CubeGraph());
```

$$\begin{bmatrix} 0 & \frac{7}{12} & \frac{7}{12} & \frac{3}{4} & \frac{7}{12} & \frac{3}{4} & \frac{3}{4} & \frac{5}{6} \\ \frac{7}{12} & 0 & \frac{3}{4} & \frac{7}{12} & \frac{3}{4} & \frac{7}{12} & \frac{5}{6} & \frac{3}{4} \\ \frac{7}{12} & \frac{3}{4} & 0 & \frac{7}{12} & \frac{3}{4} & \frac{5}{6} & \frac{7}{12} & \frac{3}{4} \\ \frac{3}{4} & \frac{7}{12} & \frac{7}{12} & 0 & \frac{5}{6} & \frac{3}{4} & \frac{3}{4} & \frac{7}{12} \\ \frac{7}{12} & \frac{3}{4} & \frac{3}{4} & \frac{5}{6} & 0 & \frac{7}{12} & \frac{7}{12} & \frac{3}{4} \\ \frac{3}{4} & \frac{7}{12} & \frac{5}{6} & \frac{3}{4} & \frac{7}{12} & 0 & \frac{3}{4} & \frac{7}{12} \\ \frac{3}{4} & \frac{5}{6} & \frac{7}{12} & \frac{3}{4} & \frac{7}{12} & \frac{3}{4} & 0 & \frac{7}{12} \\ \frac{5}{6} & \frac{3}{4} & \frac{3}{4} & \frac{7}{12} & \frac{3}{4} & \frac{7}{12} & \frac{7}{12} & 0 \end{bmatrix}$$

ShortestAncestralPath and ShortestDescendantPath

The new ShortestAncestralPath constructs the shortest ancestral path between two nodes in the given directed graph.

```
> ShortestAncestralPath( G1, "C", "D") ;
```

[["A", "B", "C"], ["A", "D"]]

You can similarly find the shortest descendent path.

New functionality for existing commands

IsReachable and Reachable

The IsReachable and Reachable commands now have a new option distance to constrain the distance within a given vertex.

```
> IsReachable( G1, "A", "E", distance = 1 );
```

false

```
> Reachable( G1, "A", distance = 1 );
```

["A", "B", "D"]

ShortestPath

The ShortestPath command accepts an option avoidvertices to constrain the search space for a shortest path to avoid some specified set of vertices.

```
> ShortestPath( G1, "A", "E" );
```

["A", "D", "E"]

```
> ShortestPath( G1, "A", "E", avoidvertices = {"D"} );
```

["A", "B", "C", "E"]

Additions to SpecialGraphs

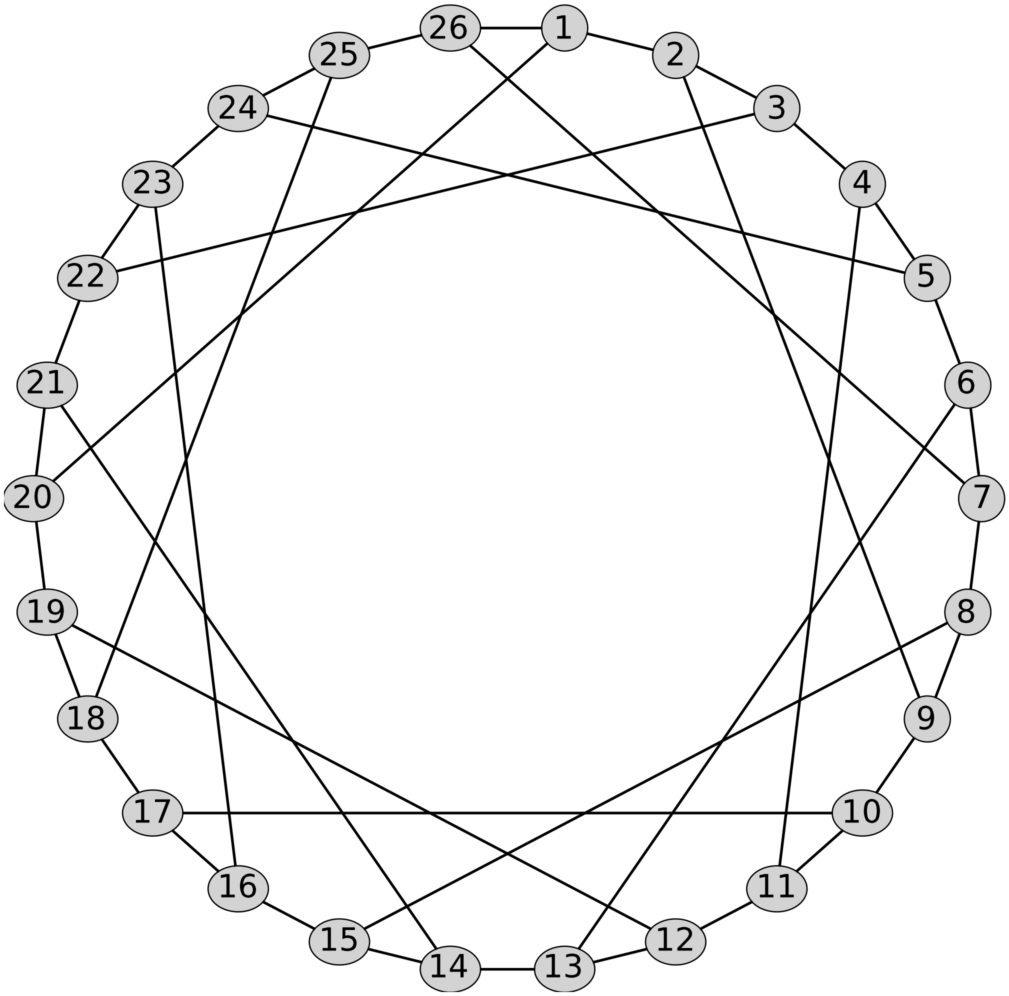
The SpecialGraphs subpackage now includes commands for the F26a graph and Hanoi graph.

- The F26a graph may be understood visually

```
> FG := SpecialGraphs:-F26AGraph();
```

FG := *Graph 4: an undirected graph with 26 vertices and 39 edges*

```
> DrawGraph(FG);
```



- The Hanoi graph is a graph whose edges correspond to allowed moves of the tower of Hanoi problem.

```
> HG4 := SpecialGraphs:-HanoiGraph(4);
```

HG4 := Graph 5: an undirected graph with 81 vertices and 120 edges