

# GraphTheory

A substantial effort was put into Graph Theory for Maple 2021, including new commands for graph computation and advances in visualization.

> *with(GraphTheory) :*

---

[New commands](#)

[New functionality for existing commands](#)

[Performance improvements](#)

[Additions to SpecialGraphs](#)

---

## New commands

These commands are new for Maple 2021:

[EgoGraph](#)

[GraphDensity](#)

[IdentifyGraph](#)

[IsSubgraphIsomorphic](#)

[LeafPower](#)

[Newick](#)

[PrueferCode](#)

[SpanningForest](#)

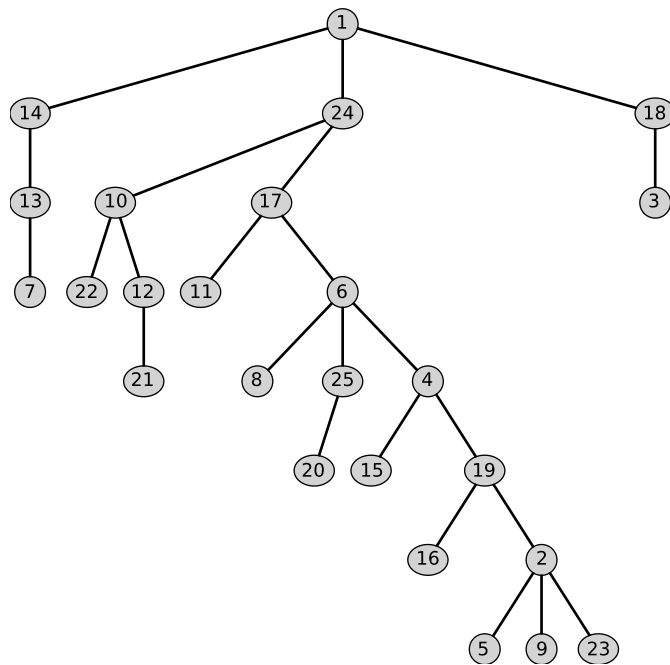
## Tree encodings: Newick and PrueferCode

The new [Newick](#) and [PrueferCode](#) offer alternate ways to encode a tree as a string or list of integers.

> *T := RandomGraphs:-RandomTree(25, seed = 1024)*

*T := Graph 1: an undirected unweighted graph with 25 vertices and 24 edge(s)* **(1.1.1)**

> *DrawGraph(T, style = tree)*



> *Newick*(*T*)  
 "(((7)13)14,(3)18,(((21)12,22)10,(((15,((5,9,23)2,16)19)4,8,(20)25)6,11)17)24)1;" (1.1.2)

> *PrueferCode*(*T*)  
 [18, 2, 13, 6, 2, 17, 14, 1, 4, 19, 1, 24, 25, 12, 10, 10, 24, 2, 19, 4, 6, 17, 6] (1.1.3)

## IdentifyGraph: find isomorphisms among named special graphs

[IdentifyGraph](#) tests a graph for isomorphism against many of the named special graphs known to GraphTheory.

In this example we begin by picking any edge  $\{a, b\}$  from the [Hoffman-Singleton graph](#) and deleting all vertices incident to  $a$  or  $b$ .

> *HS* := *SpecialGraphs*:-*HoffmanSingletonGraph*( )  
*HS* := *Graph 2: an undirected unweighted graph with 50 vertices and 175 edge(s)* (1.2.1)

> *edge* := *Edges*(*HS*)[100]  
*edge* := {20, 25} (1.2.2)

> *G* := *DeleteVertex*(*HS*, *convert*(*Neighborhood*(*HS*, *edge*[1]), *set*))  
**union** *convert*(*Neighborhood*(*HS*, *edge*[2]), *set*)  
*G* := *Graph 3: an undirected unweighted graph with 36 vertices and 90 edge(s)* (1.2.3)

With *IdentifyGraph*, we discover that this subgraph has a name: it is isomorphic to the [Sylvester graph](#).

> *IdentifyGraph*(*G*)  
 (1.2.4)

## IsSubgraphIsomorphic: test for isomorphism against subgraphs of a graph

[IsSubgraphIsomorphic](#) tests whether a given graph is isomorphic to a subgraph of another given graph. This problem is strictly harder than the graph isomorphism problem.

Returning to the above example, here we show again that the Sylvester graph is isomorphic to a subgraph of the Hoffman-Singleton graph. Note however that we did not need to explicitly construct the subgraph beforehand.

```
> SG := SpecialGraphs:-SylvesterGraph( )
      SG := Graph 4: an undirected unweighted graph with 36 vertices and 90 edge(s) (1.3.1)
```

```
> IsSubgraphIsomorphic(SG, HS)
      true (1.3.2)
```

## New functionality for existing commands

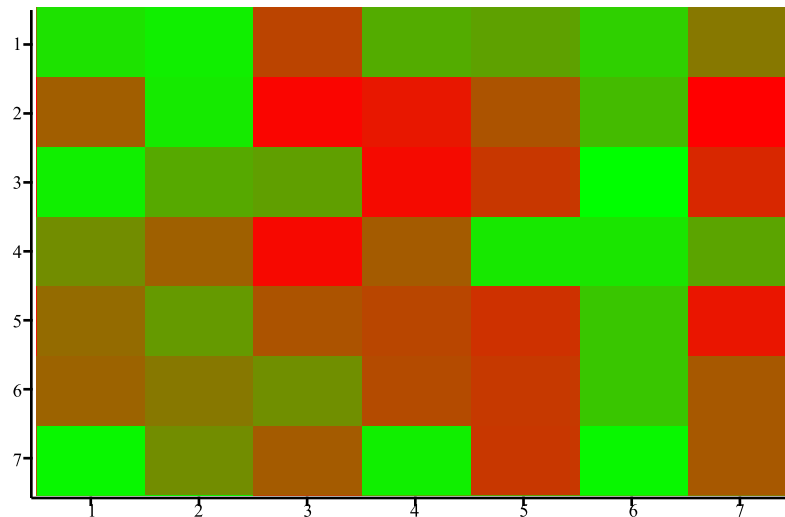
The [BipartiteMatching](#) command has been extended to support weighted graphs, on which it computes a minimum weight maximum matching using the Kuhn-Munkres algorithm, also known as the Hungarian algorithm.

We begin with a  $7 \times 7$  matrix whose rows represent seven workers and whose columns represent seven tasks, in which entry  $(i, j)$  represents the cost for worker  $i$  to complete task  $j$ . Our goal is to find an assignment of tasks to workers which minimizes the total cost.

```
> M := 
$$\begin{bmatrix} 12 & 7 & 73 & 33 & 37 & 19 & 53 \\ 63 & 9 & 97 & 90 & 67 & 27 & 99 \\ 7 & 34 & 38 & 95 & 78 & 1 & 84 \\ 45 & 62 & 96 & 64 & 10 & 11 & 36 \\ 58 & 40 & 67 & 72 & 80 & 23 & 91 \\ 61 & 53 & 44 & 70 & 77 & 23 & 65 \\ 4 & 45 & 64 & 7 & 78 & 4 & 65 \end{bmatrix} :$$

```

```
> dataplot(M, heatmap, color = green ..red)
```



We now transform  $M$  into a 14x14 block matrix which will be the weight matrix for a bipartite graph:

$$> WM := Matrix( \langle \langle Matrix(7); M \rangle \langle M^{oT}; Matrix(7) \rangle \rangle )$$

$$WM := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 63 & 7 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 9 & 34 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 73 & 97 & 38 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 33 & 90 & 95 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 37 & 67 & 78 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 19 & 27 & 1 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 53 & 99 & 84 & \dots \\ 12 & 7 & 73 & 33 & 37 & 19 & 53 & 0 & 0 & 0 & \dots \\ 63 & 9 & 97 & 90 & 67 & 27 & 99 & 0 & 0 & 0 & \dots \\ 7 & 34 & 38 & 95 & 78 & 1 & 84 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{2.1}$$

14 × 14 Matrix

The 14 vertices of this graph will be the seven workers and seven tasks.

$$\begin{aligned} > V := [seq(sprintf("Worker %d", i), i = 1 ..7), seq(sprintf("Task %d", i), i = 1 ..7)]; \\ V := ["Worker 1", "Worker 2", "Worker 3", "Worker 4", "Worker 5", "Worker 6", "Worker 7", \\ "Task 1", "Task 2", "Task 3", "Task 4", "Task 5", "Task 6", "Task 7"] \end{aligned} \tag{2.2}$$

$$\begin{aligned} > G := GraphTheory:-Graph(V, WM) \\ G := Graph 5: an undirected weighted graph with 14 vertices and 49 edge(s) \end{aligned} \tag{2.3}$$

Here we see an optimal solution for this problem, assigning Task 1 to Person 7, Task 2 to Person 2, etc.

> *GraphTheory:-BipartiteMatching*(*G*)  
 153, {{"Task 1", "Worker 7"}, {"Task 2", "Worker 2"}, {"Task 3", "Worker 1"}, {"Task 4",  
 "Worker 5"}, {"Task 5", "Worker 6"}, {"Task 6", "Worker 3"}, {"Task 7", "Worker 4"}}} (2.4)

Here is constructed the bipartite graph explicitly, but we can optionally also simply provide the original 7x7 matrix *M* directly to *BipartiteMatching* to produce a similar result:

> *GraphTheory:-BipartiteMatching*(*M*)  
 153, {[1, 7], [2, 2], [3, 1], [4, 5], [5, 6], [6, 3], [7, 4]} (2.5)

## Performance improvements

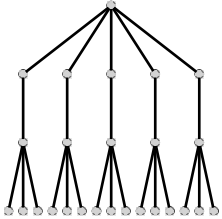
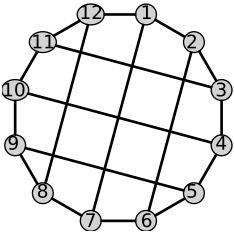
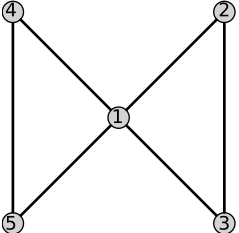
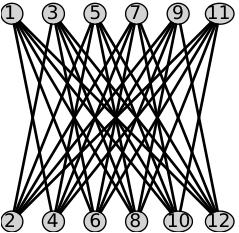
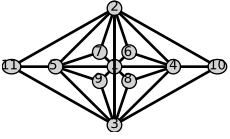
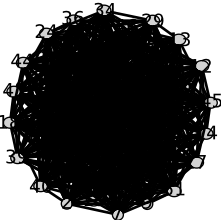
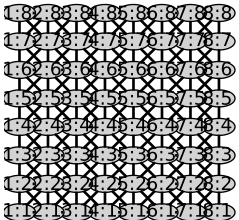
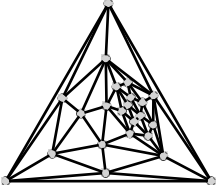
The performance of the following *GraphTheory* commands has been substantially improved:

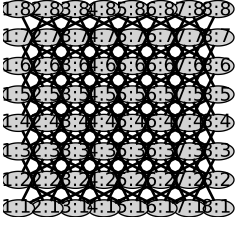
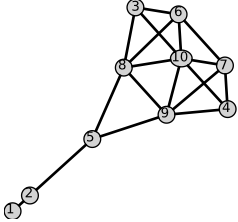
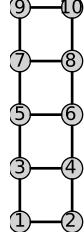
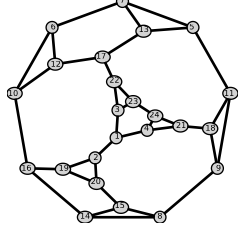
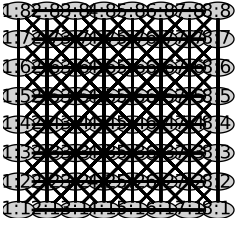
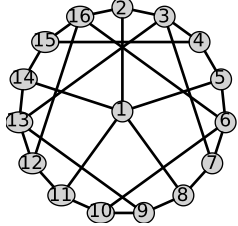
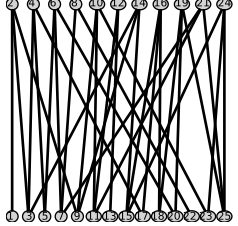
Command	Approximate Speedup Factor (compared with Maple 2020)
<a href="#">AllPairsDistance</a> (weighted)	3.1
<a href="#">Digraph</a>	4.8
<a href="#">Graph</a>	8.8
<a href="#">GraphPower</a>	4.6
<a href="#">IsBipartite</a>	200
<a href="#">RandomBipartiteGraph</a>	18
<a href="#">RandomDigraph</a>	18
<a href="#">RandomGraph</a>	22
<a href="#">RandomTournament</a>	35
<a href="#">ReverseGraph</a>	18
<a href="#">TransitiveClosure</a> (unweighted)	2.5
<a href="#">TransitiveClosure</a> (weighted)	3.4
<a href="#">TransitiveReduction</a> (unweighted)	7.6
<a href="#">TransitiveReduction</a> (weighted)	3.4
<a href="#">UnderlyingGraph</a>	4.0

# Additions to SpecialGraphs

Maple 2021 provides support for 16 additional Special Graphs, bringing the total to 113.

> with(SpecialGraphs) :

<b>Banana Tree</b>	<b>Bidiakis Cube</b>	<b>Butterfly Graph</b>	<b>Crown Graph</b>
<p>&gt; DrawGraph( BananaTree(5 , 4), size = [250, 250])</p> 	<p>&gt; DrawGraph( BidiakisCube( , size = [250, 250])</p> 	<p>&gt; DrawGraph( ButterflyGraph( , size = [250, 250])</p> 	<p>&gt; DrawGraph( CrownGraph( 6), size = [250, 250])</p> 
<b>Goldner-Harary Graph</b>	<b>Gosset Graph</b>	<b>King's Graph</b>	<b>Kittell Graph</b>
<p>&gt; DrawGraph( GoldnerHararyGraph( , size = [250, 250])</p> 	<p>&gt; DrawGraph( GossetGraph( , style = spring)</p> 	<p>&gt; DrawGraph( KingsGraph(8 , 8), size = [300, 300])</p> 	<p>&gt; DrawGraph( KittellGraph( , style = planar, size = [250, 250])</p> 
<b>Knight's Graph</b>	<b>Krackhardt Kite Graph</b>	<b>Ladder Graph</b>	<b>Markström Graph</b>
<p>&gt; DrawGraph( KnightsGraph( 8, 8), size = [300, 300])</p>	<p>&gt; DrawGraph( KrackhardtKiteGraph( , size = [250, 250],</p>	<p>&gt; DrawGraph( LadderGraph( 5), size = [250, 250])</p>	<p>&gt; DrawGraph( MarkstroemGraph( , style = planar, size</p>

	<p style="text-align: center;"><i>style = spring</i>)</p> 		<p style="text-align: right;"><i>style = spring</i>)</p> <p style="text-align: right;">= [250, 250])</p> 
<p style="text-align: center;"><b>Queen's Graph</b></p>	<p style="text-align: center;"><b>Sousselier Graph</b></p>	<p style="text-align: center;"><b>Walther Graph</b></p>	<p style="text-align: center;"><b>Watkins Snark</b></p>
<p>&gt; DrawGraph(  <i>QueensGraph</i>(  8, 8), size  = [300, 300])</p> 	<p>&gt; DrawGraph(  <i>SousselierGraph</i>(  <i>h</i>( ), size  = [250, 250])</p> 	<p>&gt; DrawGraph(  <i>WaltherGraph</i>(  ), size  = [300, 300])</p> 	<p>&gt; DrawGraph(  <i>WatkinsSnark</i>(  ), style  = <i>spring</i>, size  = [300, 300])</p> 