- `Adjacencies(g)`
- `VertexDegrees(g)`
- `Edges(g)`
- `CompletesOfGivenOrder(g,o)`
- `ConnectedComponents(g)`
- `GraphAttributeStatistics(n,p,F)` Returns information about the parameter `F` for 100 random graphs of order n and edge probability p.
- `BoundaryVertices(g)` For g a triangulation of a compact surface, returns the list of vertices whose link is isomorphic to a path.
- `InteriorVertices(g)`
- `SpanningForestEdges(g)`

Distances

- `Distance(g,x,y)`
- `DistanceMatrix(g)`
- `Diameter(g)`
- `Eccentricity(g,x)`
- `Radius(g)`
- `Distances(g,a,b)` a, b are lists of vertices. Returns a list.
- `DistanceSet(g,a,b)` As before, but returns a set.
- `DistanceGraph(g,d)` The graph with vertex set the vertices of g, two vertices adjacent if their distance is in d.
- `PowerGraph(g,n)` Same as the distance graph with set of distance $\{1, \ldots, n\}$.

Graph morphisms

- `IsoMorphisms(g,h)`
- `AutomorphismGroup(g)`
- `Morphism(g,h)`, `Morphisms(g,h)`, `NextMorphism(g,h,f)`
- `MonoMorphism(g,h)`, `MonoMorphisms(g,h)`, `NextMonoMorphism(g,h,f)`
- `EpiMorphism(g,h)`, `EpiMorphisms(g,h)`, `NextEpiMorphism(g,h,f)`
- `WeakMorphism(g,h)`, `WeakMorphisms(g,h)`, `NextWeakMorphism(g,h,f)`, and more predefined classes of morphisms and the possibility to define new classes

Small Graphs

- `ConnectedGraphsOfGivenOrder(n)` Up to $n = 9$.
- `Graph6ToGraph(s)` `s` is a string.
- `GraphsOfGivenOrder(n)` Up to $n = 9$.
- `ImportGraph6(f)` `f` is a filename.

Graph categories

- `DefaultGraphCategory` A variable that holds the current graph category. Has to be set with, e.g. `SetDefaultCategory(OrientedGraphs)`

Graph categories:

`Graphs`, `UndirectedGraphs`, `LooplessGraphs`, `SimpleGraphs`, `OrientedGraphs`.

Digraphs

- `InNeigh(g,x)` List of in-neighbors of x in g.
- `IsTournament(g)`
- `IsTransitiveTournament(g)`
- `Orientations(g)` List of all oriented graphs that can be obtained from g

Draw

- `Draw(g)` Shows a window with a drawing of `g`. Commands in the `draw` window: `h`:help, `f`:fit graph, `l`: toggle labels, `d`: toggle dynamics, `r`: toggle repulsion, `s`: save & quit, `q`: quit without saving

Backtrack

Example: coloring with two colors:

```
g:=PathGraph(3);
chk:=function(L,g)
    local x,y;
    if L=[] then return true; fi;
    x:=Length(L);
    for y in [1..x-1] do
        if IsEdge(g,[x,y]) and L[x]=L[y] then
            return false;
        fi;
    od;
    return true;
end;
```

then `BacktrackBag([0,1],chk,Order(g),g);` returns `[ [ 0, 1, 0 ], [ 1, 0, 1 ] ]`.