

# Cheatsheet for YAGS

## Graph definitions

### Adjacency list

```
g:=GraphByAdjacencies([[[]],[4],[1,2],[[]]])
```



### Adjacency matrix

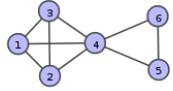
```
M:=[[false, true, false], [true, false, true], [false, true, false]]; g:=GraphByAdjMatrix(M);
```

### List of edges

```
g:=GraphByEdges([[1,2],[2,3],[3,4]]);
```

### Complete cover

```
g:=GraphByCompleteCover([[1,2,3,4],[4,5,6]]);
```



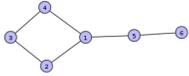
### By relation

```
f:=function(x,y) return Intersection(x,y)<>[]; end;;
```

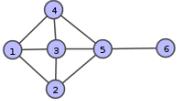
```
g:=GraphByRelation([[1,2,3],[3,4,5],[5,6,7]],f);
```

### By walks

```
g:=GraphByWalks([[1,2,3,4,1],[1,5,6]]);
```



```
g:=GraphByWalks([[1,[2,3,4],5],[5,6]]);
```



### As intersection graph

```
g:=IntersectionGraph([[1,2,3],[3,4,5],[5,6,7]]);
```

### As a copy

```
h:=CopyGraph(g)
```

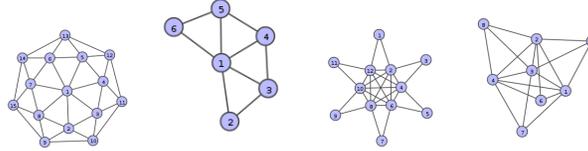
### As an induced subgraph

```
h:=InducedSubgraph(g,[3,4,6]);
```

## Graph families (with parameters)

- `g:=DiscreteGraph(n)`
- `g:=CompleteGraph(n)`
- `g:=PathGraph(n)`  $n$  vertices.
- `g:=CycleGraph(n)`
- `g:=CubeGraph(n)`
- `g:=OctahedralGraph(n)`
- `g:=JohnsonGraph(n,r)` Vertices are subsets of  $\{1, 2, \dots, n\}$  with  $r$  elements, edges between subsets with intersection of  $r - 1$  elements.

- `g:=Circulant(n,J)` Second parameter is a list of jumps
- `g:=CompleteBipartiteGraph(n,m)`
- `g:=CompleteMultipartiteGraph(n1,n2[, n3 ...])`
- `g:=TorusGraph(n,m)`
- `g:=TreeGraph(L)`  $L$  is a list. Vertices at depth  $k$  have  $L[k]$  children.
- `g:=TreeGraph(n,k)` Same as `TreeGraph([n,n,...,n])` (the list has length  $k$ )
- `g:=WheelGraph(n)`
- `g:=WheelGraph(7,2)` Second optional parameter is the radius of the wheel.
- `g:=FanGraph(4);`
- `g:=SunGraph(6);`
- `g:=SpikyGraph(4);`
- Examples: Wheel, Fan, Sun, Spiky:



## Named graphs

### Platonic

Tetrahedron, Octahedron, Cube, Dodecahedron, Icosahedron.

### Other

TrivialGraph, DiamondGraph, ClawGraph, HouseGraph, BullGraph, AntennaGraph, KiteGraph, AGraph, ChairGraph, DartGraph, DominoGraph, FishGraph, GemGraph, HouseGraph, ParachuteGraph, ParapluieGraph, PawGraph, PetersenGraph, RGraph, SnubDisphenoid.

## Random graphs

- `g:=RandomGraph(n)`
- `g:=RandomGraph(n,p)` Graph with  $n$  vertices, each edge with probability  $p$  to appear.

## New graphs from old

- `h:=RemoveVertices(g,[1,3]);`
- `h:=AddEdges(g,[1,2]);`
- `h:=RemoveEdges(g,[1,2],[3,4]);`

## Parameters

- `Order(g)`
- `Size(g)`
- `CliqueNumber(g)`
- `VertexDegree(g,v)`
- `MaxDegree(g)`
- `MinDegree(g)`
- `Girth(g)`
- `NumberOfCliques(g)`
- `NumberOfConnectedComponents(g)`

## Boolean tests

- `IsCompleteGraph(g)`
- `IsCliqueHelly(g)`
- `IsDiamondFree(g)`
- `IsEdge(g,x,y)` Or `IsEdge(g,[x,y])`
- `IsIsomorphicGraph(g,h)`
- `IsCompactSurface(g)`
- `IsSurface(g)`
- `IsLocallyConstant(g)`
- `IsLocallyH(g,h)`
- `IsLoopless(g)`

## Products

- `p=BoxProduct(g,h)`
- `p=TimesProduct(g,h)`
- `p=BoxTimesProduct(g,h)`
- `p=DisjointUnion(g,h)`
- `p=Join(g,h)`
- `p=GraphSum(g,l)`  $l$  is a list of graphs. Suppose that  $g$  has  $n$  vertices. In the disjoint union of the first  $n$  graphs of  $l$  (using `TrivialGraphs` if needed to fill  $n$  slots), add all edges between graphs corresponding to adjacent vertices in  $g$ .
- `p=Composition(g,h)` is the same as `GraphSum(g,l)`, where  $l$  is a list of length the order of  $g$ , with all components equal to  $h$ .

## Operators

- `h:=CliqueGraph(g)`
- `h:=CliqueGraph(g,m)` Stops when a maximum of  $m$  cliques have been found.
- `h:=LineGraph(g)`
- `h:=ComplementGraph(g)`
- `h:=Cone(g)`
- `h:=Suspension(g)`
- `h:=ParedGraph(g)`
- `h:=CompletelyParedGraph(g)`
- `h:=QuotientGraph(g,p)`  $p$  is a partition of vertices. The vertices of  $h$  are the parts of  $p$ , with two vertices adjacent if there are two vertices adjacent in  $g$  in the corresponding parts. Singletons in  $p$  may be omitted.
- `h:=QuotientGraph(g,l1,l2)`  $l1, l2$  are lists of vertices of the same length, with repetitions allowed. In  $h$ , each vertex of the first list is identified with the corresponding vertex in the second list.
- `h:=Link(g,x)` The subgraph of  $g$  induced by the neighbors of  $x$ .
- `h:=SpanningForest(g)`

## Lists

- `VertexNames(g)`
- `Cliques(g)`
- `Cliques(g,m)` Stops if a maximum of  $m$  cliques have been found.
- `Basement(kng,kmg,x)`  $n \leq m$
- `AdjMatrix(g)`
- `Adjacency(g,v)`

- 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  $\{1, \dots, n\}$ .

## 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 ] ].
```