

# Yags cheatsheet

## Graph definitions

### Adjacency list

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

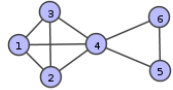


### Adjacency matrix

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

### Complete cover

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



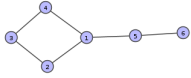
### By relation

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

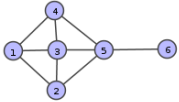
```
g:=GraphByRelation([[1,2,3],[3,4,5],[5,6,7]],f);
```

### 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)
```

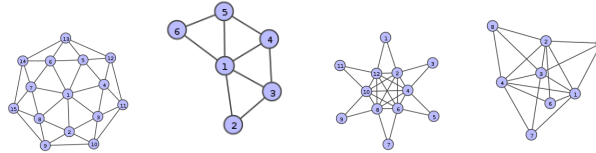
### As an induced subgraph

```
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:=CompleteBipartiteGraph(n,m)`
- `g:=CompleteMultipartiteGraph(n1,n2[, n3 ...])`
- `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, PawGraph, HouseGraph, BullGraph, AntennaGraph, KiteGraph, SnubDisphenoid.

## Random graphs

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

## Modifying graphs

- `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)`

## Boolean tests

- `IsCompleteGraph(g)`
- `IsCliqueHelly(g)`
- `IsDiamondFree(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:=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,l)`  $l$  is a pair of 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.

## Lists

- `VertexNames(g)`
- `Cliques(g)`
- `Cliques(g,m)` Stops if a maximum of  $m$  cliques have been found.
- `AdjMatrix(g)`
- `Adjacency(g,v)`
- `Adjacencies(g)`
- `VertexDegrees(g)`
- `Edges(g)`
- `CompletesOfGivenOrder(g,o)`

## Distances

- `Distance(g,x,y)`
- `DistanceMatrix(g)`
- `Diameter(g)`
- `Eccentricity(g,x)` Should be Eccentricity
- `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  $\{1, \dots, n\}$ .