Gaining information about graph structure

`vcount(g)`	count the number of vertices
`ecount(g)`	count the number of edges
`neighb ors (g,v)`	list the neighbors of a vertex
`incide nt(g,v)`	list the incident edges of a vertex
`is.dir ect ed(g)`	determine whether the graph is directed
`are.co nne cte d(g ,v1 ,v2)`	determine whether there is an edge between two vertices
`get.ed ge( g,id)`	get endpoints of an edge
`get.ed ges (g,es)`	get endpoints of a list of edges

Graph, vertex and edge attributes

In the following functions, `?` can be `graph`, `vertex` or `edge`.

```
get.?.a tt rib ute (g,n)
set.?.a tt rib ute (g,n,v)
list.?.at tri but es(g)
?.attr ibu tes(g)
remove.?.a tt rib ute (g,n)
```

Various methods for creating graphs

```
graph.e mpty()
graph.s tar(n)
graph.l at tic e(c (n,m))
graph.r ing(n)
graph.t ree(n)
graph.f ull(n)
graph.f ul l.c ita tion(n)
graph.a tl as( 0-1252)
graph( c(1 ,2, 2,3 ,3, 4,...))
graph.e dg eli st( edg e.m atrix)
graph.f or mul a(1 -2, 3,4-+5)
```

Vertex and edge sequences and iterators

`V(g)`	list vertices in a `igraph.vs` object
`V(g)$n umber`	get or set vertices properties
`V(g)[n umb er<50]`	get a subset of vertices
`E(g)`	list edges in a `igraph.es` object

Method for structural manipulation of graphs

`g[]`	get and set adjacency matrix
`g[[]]`	get adjacency list

Degree and degree distribution of the vertices

```
degree(g)
degree.di str ibu tion(g)
```

Graph Algebra

```
graph.u ni on( g1,g2)
graph.d if fer enc e(g 1,g2)
```

Print Graph

`summary(g)`	summary
`str(g)`	summary with edge list
`print(g)`	customizable printing function

Neighborhood of graph vertices

`neighb orh ood.si ze(g,o)`	gives a list of neighborhood size for each vertex
`neighb orh ood (g,o)`	gives a list of neighborhood vertices for each vertex
`graph.n ei ghb orh ood (g ,o)`	gives the neighborhood graphs
`connec t.n eig hbo rho od (g,o)`	creates a new graph by connect each vertex with its neighbor vertices