Sage Quick Reference:
Combinatorics and Graph Theory
Barry Balof
Sage Version 5.9
http://wiki.sagemath.org/quickref

Based on work by Rob Beezer, Steven R. Turner

Lists

`L = [2,17,3,17]` an ordered list

`L[i]` the ith element of L
lists begin with the 0th element

`L.append(x)` adds x to L

`L.remove(x)` removes x from L

`L[i:j]` the i-th through $(j-1)$-th element of L

`range(a)` list of integers from 0 to $a-1$
`range(a,b)` list of integers from a to $b-1$
`[a..b]` list of integers from a to b
`range(a,b,c)`
every c-th integer starting at a and less than b

`len(L)` length of L

`M = [i^2 for i in range(13)]`
list of squares of integers 0 through 12

`N = [i^2 for i in range(13) if is_prime(i)]`
list of squares of prime integers between 0 and 12

`M + N` the concatenation of lists M and N

`sorted(L)` a sorted version of L (L is not changed)
`L.sort()` sorts L (L is changed)

`set(L)` an unordered list of unique elements

Permutations and Combinations

`Permutations(L)` list of permutations of L
`Permutations(L,2)` list of 2-permutations of L
`Combinations(L)` list of all combinations of L (the power set) as lists
`Combinations(L,2)` list of 2-combinations of L as lists
`Partitions(n)` list of unordered partitions of n
`Compositions(n)` list of compositions (ordered partitions) of n
`Subsets(n)` list of subsets of $\{1, 2, \ldots n\}$ as sets.
`Subsets(n,k)` list of k-element subsets of $\{1, 2, \ldots n\}$ as sets.

Poset Examples Operations

`P = posets.BooleanLattice(n)` P is the poset of subsets of a five element set
`P = posets.ChainPoset(6)` P is a 6 element chain (linear) poset
`P = posets.AntichainPoset(6)` P is a 6 element chain (linear) poset
`P = posets.DiamondPoset(8)` P is an antichain of 6 elements, each element of which is greater than a minimal element and less than a maximal element.

`P = Poset({0:[3],1:[2,3],2:[3,4],3:[4],4:[])` Creates a poset where each element is followed by its list of successors (where transitivity is implied).
`P = Poset({0:[3],1:[2,3],2:[3,4],3:[4],4:[])` Creates a poset where each element is followed by its list of successors (where transitivity is implied).

`P.maximal_chains()` List of maximal chains of P
`P.antichains()` List of antichains of P
`P.linear_extensions()` List of linear extensions of P

Binomial and Polynomial Constructions

`binomial(a,b)` $\binom{a}{b}$
`list(binomial(8, i) for i in xrange(9))` list of biniomial coefficients of the form $\binom{8}{i}$ (the 8th row of Pascal's Triangle)
`multinomial(a,b,c,d)` $\binom{a+b+c+d}{a,b,c,d}$
`p=` an expanded polynomial in any number of variables
`p.coefficients()` returns a list of the coefficients of p.
`p.coefficient(x^2)` returns the coefficient of x^2 in p.

Special Number Sequences

`fibonacci(n)` returns the nth Fibonacci Number, with $F_1 = F_2 = 1$
`bell_number(n)` returns the nth Bell Number
`catalan_number(n)` returns the nth Catalan Number
`stirling_number1(n,k)` $\left[{n \atop k}\right]$, the Stirling number of the first kind
`stirling_number2(n,k)` $\left\{{n \atop k}\right\}$, the Stirling number of the second kind
`a=sloane.A000045` sets a as sequence A000045 in Sloane's OEIS. Use `sloane.A <tab>` for a list of SAGE enabled sequences.

Graph Constructions

Sage has many, many (many!) examples of graphs. Type `graphs.` then press ⟨*tab*⟩ for a complete list.

`G.show()` draws a plot of G.
`G.plot()` draws a plot of G.
`G = Graph([(1,3),(3,8),(5,2)])` creates a graph with specified edges (vertices are implied)
`G = Graph({0:[1,2,3], 2:[4]})` creates a graph with listed adjacencies
`G = graphs.RandomGNP(n, p)` creates a random graph on n vertices, where each edge is included with probability p.
`G.add_vertex(v)` adds a vertex v to G.
`G.add_edge((a,b))` adds an edge (a,b) to G.
`G.add_cycle([5,6,7,8])` adds a cycle on vertices G (note: SAGE will use existing vertices if these are included, or add vertices as necessary.)
`G.delete_vertex(v)` deletes the vertex v from G.
etc....

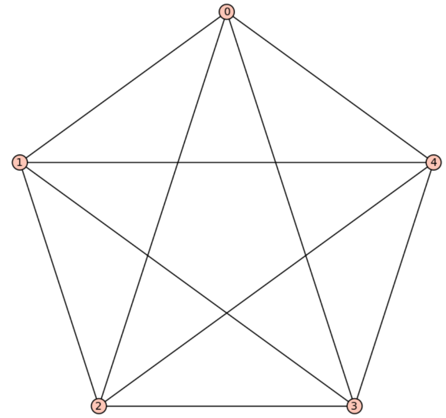
Graph Queries

`G.is_planar()` returns True if G is planar
`G.is_bipartite()` returns True if G is bipartite
`G.is_eulerian()` returns True if G is Eulerian
`G.is_hamiltonian()` returns True if G is Hamiltonian
`G.is_connected()` returns True if G is connected
`G.is_isomorphic(H)` returns True if G and H are isomorphic

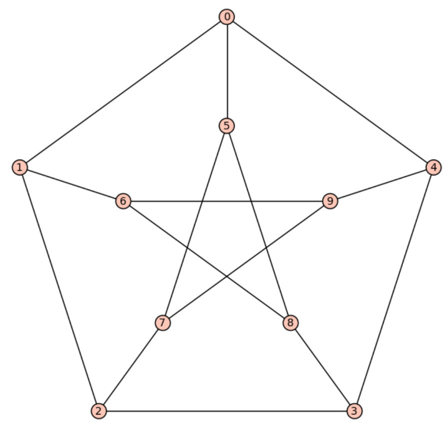
Graph Statistics

`G.size()` number of edges of G
`G.order()` number of vertices of G
`G.girth()` length of the shortest cycle of G
`G.chromatic_polynomial()` returns the chromatic polynomial of G
`G.automorphism_group(G)` returns the autmorphism group of G

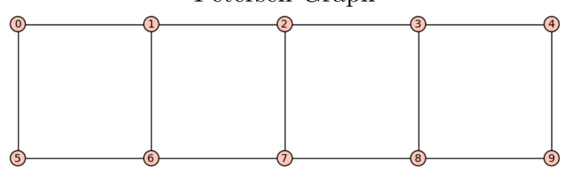
Graph Examples



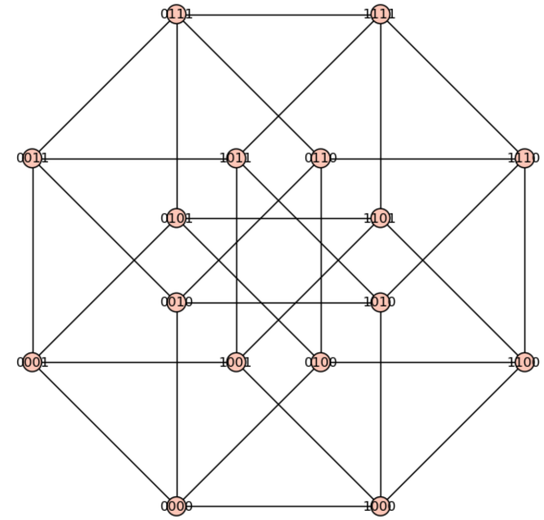
K_5



Petersen Graph



Ladder Graph (5)



Cube Graph (4).
Note that the vertices are labled with 0-1 strings.

More Help

"tab-completion" on partial commands
"tab-completion" on `<object.>` for all relevant methods
`<command>?` for summary and examples