Introduction to Sage Graphs

Al Doerr
Department of Mathematical Sciences
University of Massachusetts LowellLowell, MA, USA
alan_doerr@uml.edu

Ken Levasseur
Department of Mathematical Sciences
University of Massachusetts LowellLowell, MA, USA
kenneth_levasseur@uml.edu

November 16, 2015

Abstract

This document illustrates some of the basic ways to create a graph in Sage and some of the fundamenal functions on them.

1 Graphs

Graphs play an important role in discrete mathematics. They can be created and plotted using Sage. Here we are using the Sage Cell Server. Each cell contains Sage expressions that can be evaluated by clicking on the "Evaluate" button below it. You can edit the contents of the cell to experiment - the orginal expressions will return when the page reloads.

The simplest way to create a graph is to build a dictionary(or hash function). If you wrap Graph() around a dictionary you get a graph and the plot method can be used to view an embedding of the graph on the plane.

```
t={}
for i in range(16)[2:]:
    t[i]=[(i/2).floor()]
Graph(t).plot()
```

There are many families of graphs included in the graphs package. One standard family is the set of complete bipartite graphs, $K_{m,n}$, where m and n are positive.

```
graphs.CompleteBipartiteGraph(2,4).plot()
```

Another family is of random graphs. Here is how to generate and display a random graph with a certain number of vertices and each edge determined by a fixed probability. The default is 25 vertices with a 0.15 probability of any edge being present.

```
graphs.RandomGNP(25,0.15).plot()
```

The graphs package also contains a variety of functions that can be applied to a graphs. For example here we generate a graph and ask for a hamiltonian cycle in the graph.

```
g=graphs.DodecahedralGraph()
g.hamiltonian_cycle().plot()
```