SOLUTION KEY -- Math 480 HW 5 -- Graph Theory

Homework 5

Due May 6, 2009

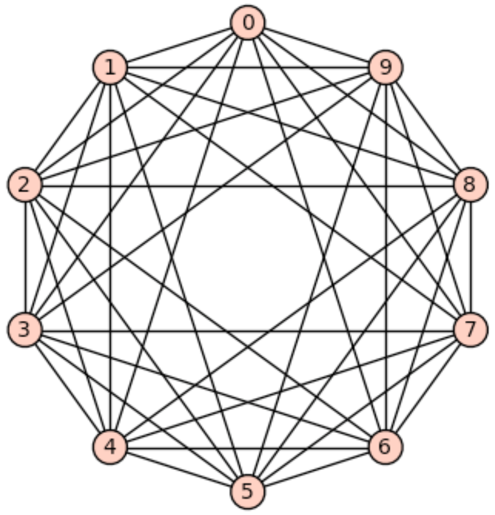
There are 5 problems.

Email solution worksheet with your *name* in the title to `sagehw@gmail.com`.

Problem 1. Ten people are seated around a circular table. Each person knows everybody else at the table *except* the peson sitting directly across the table from them. Create a graph that models this situation and plot it. [Hint: This is purposely ambigious, since you have to chose what an edge means.]

```
# The following line uses that dict([(a,b), (c,d)]) is the dictionary {a:b, c:d}.
# It also uses that one could get the number of the person directly across by
# adding 5 mod 10.
v = dict([(i,[j for j in range(10) if j != (i+5)%10 and j != i]) for i in range(10)])
g = Graph(v)
```

```
show(g,layout='circular')
```



Problem 2. Suppose there are 10000 people and each person knows roughly 20 people personally. Do a computational *experiment* to get a rough sense for how many degrees of separation (length of path) there are between two random people. [Hint: Use **graphs.RandomGNP** to make a graph with 10000 vertices where the probability two vertices are connected is 20/10000, then find lots of shortests paths.]

```
G = graphs.RandomGNP(10000,20/10000)
```

```
def test():
   i = randint(0,9999)
   j = randint(0,9999)
   return G.shortest_path_length(i,j)
```

```
v = [test() for _ in range(1000)]
import scipy.stats
scipy.stats.mean(v)
```

```
    3.4209999999999998
```

```
# the degree of sep. tends to be between 3 and 4
```

```
max(v)
```

```
    4
```

```
min(v)
```

```
    1
```

Problem 3. Here is a description of the Birthday Paradox from Wikipedia:

> "In probability theory, the birthday problem, or birthday paradox pertains to the probability that in a set of randomly chosen people some pair of them will have the same birthday. In a group of at least 23 randomly chosen people, there is more than 50% probability that some pair of them will both have been born on the same day. For 57 or more people, the probability is more than 99%, and it reaches 100% when the number of people reaches 366 (there are a maximum of 365 possible birthdays, ignoring leap years)."

In this problem you will get a computational "sense" that the above is true, by using Sage to simulate it. Let n be the number of people (e.g., 23 or 57).

1. Using the **graphs.RandomGNP** function, create a function that returns a graph with n vertices, where the probability that any two vertices are connected is 1/365. Note -- if you plot these graphs **g**, use **g.plot(layout='circular'**).
2. Using the edges method on a graph, compute the number of edges (e.g., just do **len(g.edges())**) for a couple of random graphs.
3. Compute the average number of edges for 100 randomly chosen graphs with $n = 23$.
4. Compute the average number of edges for 100 randomly chosen graphs with $n = 57$.

```
def f(n):
   return graphs.RandomGNP(n,1/365)
```

```
len(f(23).edges())
```

```
    2
```

```
v = [len(f(23).edges()) for i in range(100)]
scipy.stats.mean(v)
```

```
    0.65000000000000002
```

```
v = [len(f(57).edges()) for i in range(100)]
scipy.stats.mean(v)
```

```
4.6500000000000004
```

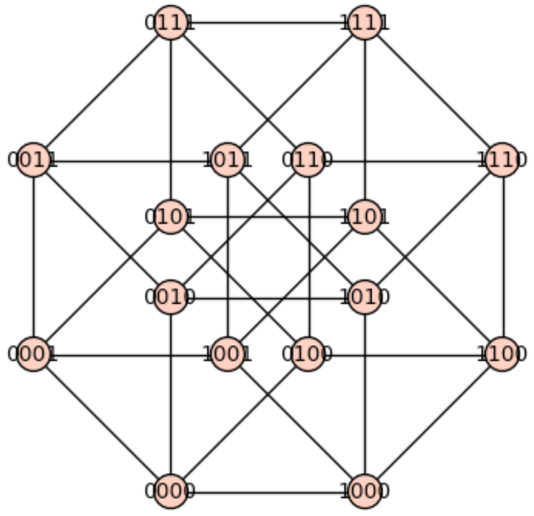
Problem 4. Recall that the chromatic number of a graph is the minimal number of colors needed to color the vertices of the graph so no two adjacent vertices are the same color.

1. Compute the *chromatic number* of the n-cube graphs for $n = 2, 3, 4, 5, 6, 7$. Make a conjecture (and maybe prove it). [Hint: Use the command **graphs.CubeGraph**.]
2. Compute the chromatic number of the complete graphs K_n for $n = 2, 3, 4, 5, 6, 7$. Make a conjecture (and maybe prove it). [Hint: Use the command **graphs.CompleteGraph**.]

```
for n in [2,3,..,7]:
   print n, graphs.CubeGraph(n).chromatic_number()
```

```
2 2
3 2
4 2
5 2
6 2
7 2
```

```
graphs.CubeGraph(4).show()
```



CONJECTURE: One can always two color the n-cube graph.

Proof. Embed the n-cube graph in $\mathbf{R}^n$ by putting the vertices at the points whose coordinates all 0 and 1. Then two vertices are adjacent if exactly one component of the vector changes from a 0 to a 1 or vice versa. Color a vertex red if the sum of the coordinates of that vertex is even, and color it blue if the sum is odd. Then two adjacent vertices will get different colors, so this is a 2-coloring. At least two colors are needed because there is always a vertex with degree at least 1.

```
for n in [2,3,..,7]:
   print n, graphs.CompleteGraph(n).chromatic_number()
```

```
2 2
3 3
4 4
5 5
6 6
7 7
```

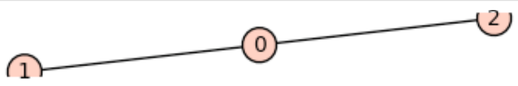
CONJECTURE: The chromatic number of the complete graph K_n is n.

Proof: n colors suffice by making all vertices a different color. No fewer colors work, since all vertices must be a different color since they are adjacent.

Problem 5. Create a Sage interact that takes as input a nxn matrix with entries 0 or 1, draws a plot of the corresponding graph (with that matrix as adjacency matrix), and also computes and displays at least 5 different things about that graph.

```
@interact
def f(n=input_box([[0,1,1],[1,0,0],[1,0,0]])):
   g = Graph(matrix(n))
   show(g)
   print "chromatic_number = ", g.chromatic_number()
   print "girth = ", g.girth()
   print "is_planar = ", g.is_planar()
   print "is_bipartite = ", g.is_bipartite()
   print "is_eulerian = ", g.is_eulerian()
```

n [[0, 1, 1], [1, 0, 0], [1, 0, 0]]



```
chromatic_number =  2
girth =  +Infinity
is_planar =  True
is_bipartite =  True
is_eulerian =  False
```