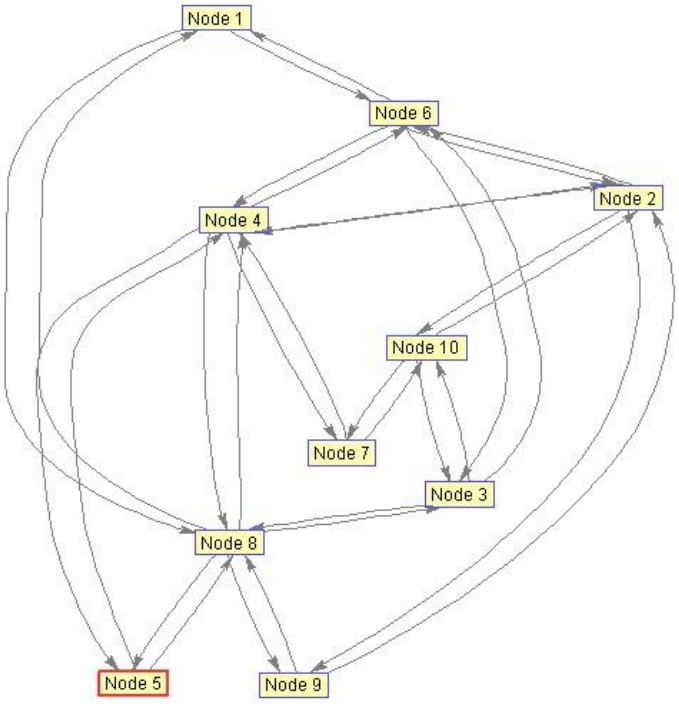
Course Example MATLAB SOLUTION

Homework - 7

Assignment: The Adjacency Matrix **A** is the *NxN* matrix of [0,1] that represents which nodes are connected to which other nodes. The matrix is zero, unless there is a connect between node *i* and node *j*, then there is a 1 in the **A***(i,j)* cell.

1) Build the Adjacency Matrix **A** for this small System of Systems of 10 nodes shown in Figure 1




```
A=[   0   0   0   0   0   1   0   1   0   0;
0   0   0   1   0   1   0   0   1   1;
0   0   0   0   0   1   0   1   0   1;
0   1   0   0   1   1   1   1   0   0;
0   0   0   1   0   0   0   1   0   0;
1   1   1   1   0   0   0   0   0   0;
0   0   0   1   0   0   0   0   0   1;
1   0   1   1   1   0   0   0   1   0;
0   1   0   0   0   0   0   1   0   0;
0   1   1   0   0   0   1   0   0   0]
```

To check your Matrix. you can use the MATLAB biograph function, which is used above to graph an Adjacency Matrix. Your exact locations may be different.

```
>> bg=biograph(A);
>> view(bg);
```

2. Use my provided Dykstra's shortest path algorithm., grShortPath(E) to calculate shortest path lengths. However, this function takes an Edge Matrix (E) instead of an Adjacency (A). I have provided 2 helper functions MakeEfromADJ(A) and grValidation. MakeEfromADJ converts your Adjacency to and Edge

representation (series of rows with source node # and target node # for each edge. grValidation is used within grShortPath to check if grShortPath is being passed a well-formed edge matrix.

```
E=makeEfromADJ(A);
dsp=grShortPath(E);
```

dsp=

```
    2    2    2    2    2    1    3    1    2    3
    2    2    2    1    2    1    2    2    1    1
    2    2    2    2    2    1    2    1    2    1
    2    1    2    2    1    1    1    1    2    2
    2    2    2    1    2    2    2    1    2    3
    1    1    1    1    2    2    2    2    2    2
    3    2    2    1    2    2    2    2    3    1
    1    2    1    1    1    2    2    2    1    2
    2    1    2    2    2    2    3    1    2    2
    3    1    1    2    3    2    1    2    2    2
```

The function grShortPath returns a Matrix of shortest path lengths between every node.
What is the characteristic path length (mean of all shortest paths) for this Graph. Don't include self-loops (length of node to itself).

```
%remove diagonal
>>dsp2=dsp-diag(diag(dsp));
dsp2 =
```

```
    0    2    2    2    2    1    3    1    2    3
    2    0    2    1    2    1    2    2    1    1
    2    2    0    2    2    1    2    1    2    1
    2    1    2    0    1    1    1    1    2    2
    2    2    2    1    0    2    2    1    2    3
    1    1    1    1    2    0    2    2    2    2
    3    2    2    1    2    2    0    2    3    1
    1    2    1    1    1    2    2    0    1    2
    2    1    2    2    2    2    3    1    0    2
    3    1    1    2    3    2    1    2    2    0
>> sum(sum(dsp2))/(10^2 -10)
sum(sum(dsp2))/(10^2-10)

ans =

    1.7333
```

3. Small worlds are a phenomenon when nodes in a graph have neighbor that are highly connected. This forms cliques or clusters which tends to shorten the characteristic path length. A measure of this neighborhood effect is the cluster coefficient (described in Watts article). For each node, determine its neighbors. What fraction of the possible connections between all neighbors is present in the graph for that node? Then, take the mean of all cluster coefficients across all nodes. Each coefficient is between 0 and 1 so the mean across all nodes will also be between 0 and 1.

Cluster Coefficients for 10 nodes

cc =

```
    0
 0.1667
    0
 0.2000
 1.0000
 0.1667
    0
 0.1000
    0
    0
```

For example, Node 2 has 4 neighbors (nodes 4,6,9 and 10). The combinatorial 4 choose 2 (MATLAB command nchoosek(n,k) makes 6 possible edges. But only 1 exists (from node 6 to node 4)
So the coefficient for Node 2 is 1/6 or 0.1667. The average is thus

mean(cc) = 0.1633

4. What is the average degree (in this case, both out degree and in degree are the same) for this Graph?

Take the sum of the rows... then the mean of these sums. The sum of each row or column is the degree out or in from each node.

```
>> sum(A)

ans =

   2    4    3    5    2    4    2    5    2    3

>> mean(sum(A))

ans =

   3.2000
```

5. What is the graph's density, or percentage of links used.

```
Vertices=length(A)
Edges=length(E)
Density = Edges/(Vertices^2-Vertices)
```

6. Which edge (edge pair - we assume bidirectional), if removed, would increase the characteristic path length CPL the most? If removed, what is the new CPL for this Graph

Here is a little script I wrote:

```
AllEdges=find(triu(A)==1); % each edge in the upper triangular of Adj
CPL=zeros(1,length(AllEdges));  %hold the mean path length for each removal

j=AllEdges; % j is the list of edges (an index into Adjacency)
```

```
for k=1:length(AllEdges)
    Acopy=A;
    Acopy(j(k))=0;
    NewA=triu(Acopy)+triu(Acopy)'; %build a new Adjacency by adding
upper/lower
    dsp=grShortPath(makeEfromADJ(NewA));
    dsp2=dsp-diag(diag(dsp)); %remove the diagonal… set = 0 in one line
    CPL(k)= sum(sum(dsp2))/(10^2 -10);

end
CPL %list of CPL for all edges removed

[M,I]=max(CPL) %M is the max CPL, corresponds to 7th index of AllEdges vector
Acopy=A;
Acopy(AllEdges(I))=9  %look for "9", that is edge (4 <--> 7) with highest CPL
```

```
>>  CPL=
  1.7778
  1.8222
  1.8000
  1.8000
  1.7778
  1.8000
  1.8889
  1.8222
  1.8444
  1.7778
  1.8222
  1.8222
  1.8444
  1.8000
  1.8222
  1.8000

>> [M,I]=max(CPL)

M =

  1.8889

I =

   7

Acopy =

   0   0   0   0   0   1   0   1   0   0
   0   0   0   1   0   1   0   0   1   1
   0   0   0   0   0   1   0   1   0   1
   0   1   0   0   1   1   9   1   0   0
   0   0   0   1   0   0   0   1   0   0
   1   1   1   1   0   0   0   0   0   0
   0   0   0   1   0   0   0   0   0   1
```

```
1   0   1   1   1   0   0   0   1   0
0   1   0   0   0   0   0   1   0   0
0   1   1   0   0   0   1   0   0   0
```

So the 7th cell in the AllEdges is 64. This is the index into the Adjacency Matrix, which is located at cell (4,7). … in other word edge 4 <->7. Removal of this edge (in both directions) leads to the maximum CPL of 1.8889

7. Centrality has been defined multiple ways (see Linton, page 218-219). One way for point centrality of a node is which node is closest to all other nodes. This can be done as the smallest summation of distances to all other nodes, or the largest inverse of the summation of distances to all other nodes. See page 225. Which node(s) is most central using this definition?

Again using the grShortPath which returns the dsp... simply the reciprocal 1./sum(dsp)

…or more precisely you can remove the diagonal elements (make them 0)

```
dsp2=dsp-diag(diag(dsp));
central=1./sum(dsp2)

  >>
ans =

   0.0556
   0.0714
   0.0625
   0.0769
   0.0588
   0.0714
   0.0476
   0.0769
   0.0588
   0.0526

>> [X,I]=max(central)

X =

   0. 0769

I =

    4
```

So Node 4 is the most central.

```
>>find(central==X)
```

… likewise node 8 is also tied for max centrality.

8 a/b. What is the Graph's diameter (max across all largest geodesic paths) and radius (min of all largest geodesic paths)

```
% Eccentricity
ecc = max(dsp2');

% Diameter of graph
Gdiameter = max(ecc)

% Radius of graph
Gradius = min(ecc)
```

9. Discuss the possible utility or lack of utility for using these graph theoretic Measures of Performance / Measures of Merit for System of Systems.

```
function ci = JMCclusteringCC(g)
% clusteringcoef    - clustering coefficient of given adjacency matrix
%
%   coefs = clusteringcoef(g) cluster coefficient is ratio of the number of
%   edges Ei between the first neighbors of the vertex i, and the
%   respective number of edges, Ei(max) = ai(ai-1)/2, in the complete graph
%   that can be formed by the nearest neighbors of this vertex:
%
%   g is a graph or an alternatively adjacency matrix.
%
%          2 Ei
%   ci = -----------
%        ai (ai - 1)
%
%   A note that do no have a link with others has clustering coefficient of
NaN.
%   Modified by jmc,19 Aug

[rows, cols]=size(g);
if (rows==cols)
    % g is adj
    adj=g;
elseif (cols==2)
    % g is edge
    adj=makeADJfromE(g);
elseif (cols==3)
    %n+1 vector
    adj=makeADJfromE(g);
else
   disp('error, size of input is not ADJ or Edge');
   return
end

n = length(adj);
ci = zeros(1,n);
lnk=zeros(1,n);
for k = 1:n
    neighbours = [find(adj(:,k))]';
    neighbours(neighbours==k) = []; % self link deleted
    a = length(neighbours); lnk(k)=a;
    if a == 0; ci(k) = NaN; continue; end
    if a < 2, continue; end
    E = 0;
    for ks = 1:a
        k1 = neighbours(ks);
        for k2 = neighbours(ks+1:a)
            if adj(k1,k2) || adj(k2,k1)
                E = E + 1;
            end
        end
    end
    ci(k) = 2 * E / (a * (a-1));
end
```