CS319 Week 11: Introduction to graphs in MATLAB

Table of Contents


```
clear;
```

1. Graphs in MATLAB

There is a class called `graph` in MATLAB, which represents a (mathematical) graph - something with vertices (="nodes") and edges between them.

The simplest graph is the empty graph, with no vertices or edges. If the graph constructor is given no arguments, that is what is returned.

```
G0 = graph()
```

```
G0 =
  graph with properties:

    Edges: [0×1 table]
    Nodes: [0×0 table]
```

More typically, however, we want to make a graph with vertices and edges. One way to make such a graph is to use the `addnode()` and `addedge()` methods.

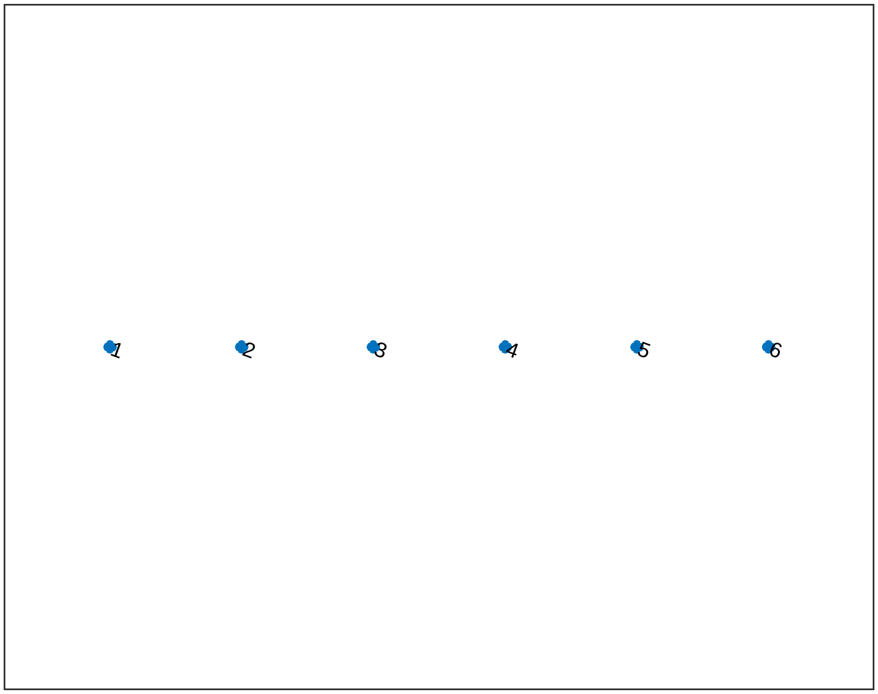
addnode()

Calling `G.addnode(n)` make a graph that has all the same nodes as G, but with n new ones added.

```
G0 = G0.addnode(4); % had zero, now has 4
G0 = G0.addnode(2); % had 4, now has 6
```

G0 is now a graph with 6 vertices and no edges. Let's look at it:

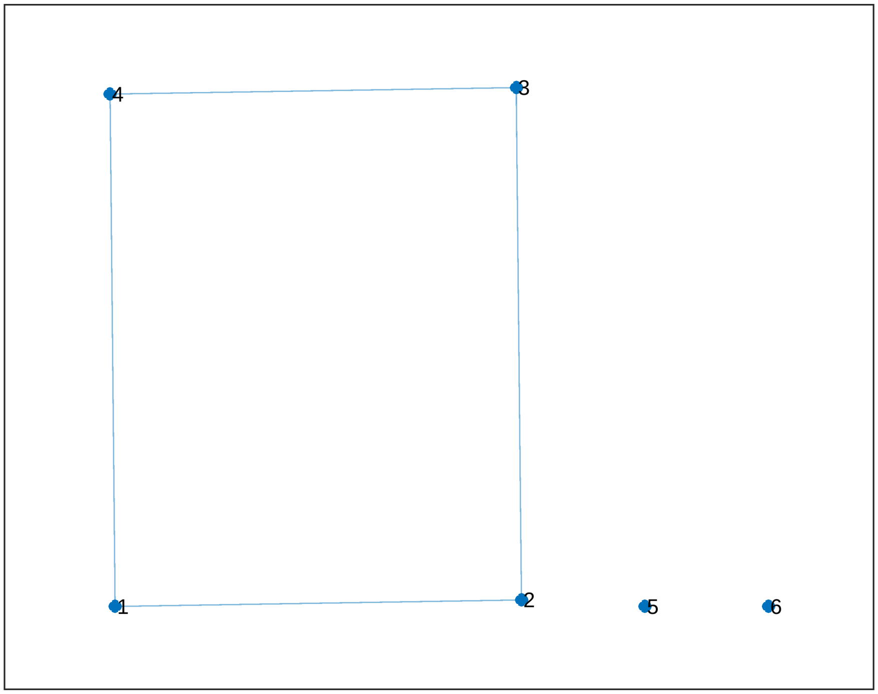
```
plot(G0)
```



addedge()

The addedge() method adds edges. Used as `G.addedge(s,t)`, where s and t are scalar positive intergers, it adds an edge between nodes s and t. If either are not already in G, they are added automatically.

```
G0=G0.addedge(1,2);
G0=G0.addedge(2,3);
G0=G0.addedge(3,4);
G0=G0.addedge(4,1);
plot(G0);
```



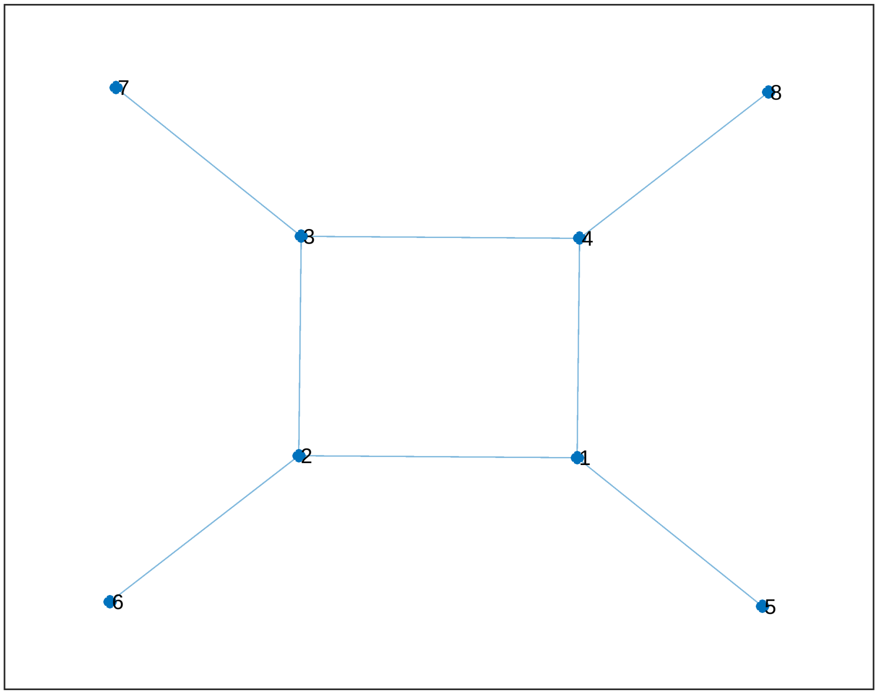
You can also use vectors s and t of the same length. Then there will be new edges added between s(1) and t(1), s(2) and t(2), ...

```
G0=G0.addedge([1 2 3 4],[5 6 7 8])
```

```
G0 = 
  graph with properties:

    Edges: [8×1 table]
    Nodes: [8×0 table]
```

```
plot(G0)
```



The graph constructor

Rather than building an empty graph, and then added nodes and edges, we could just pass the edge vectors to the constructor (remember: that is a function with the same name as the class).

That is, we initialise the graph with two vectors, s and t, that have the same number of entries, n. The resulting graph will have n edges and `max( max(s), max(t) )` nodes.

```
s = [1, 2, 3, 4, 1, 2];
t = [2, 3, 4, 1, 5, 5];
G1 = graph(s,t)
```

```
G1 =
  graph with properties:

    Edges: [6×1 table]
    Nodes: [5×0 table]
```

Notice that the notes and edges of `G1` are stored as tables. (Not something we'll get into right now).

To check some of the properties of the graph, using that class's methods

```
G1.numnodes
```

```
ans = 5
```

```
numedges(G1);
G1.Edges.EndNodes
```

```
ans = 6×2
     1     2
     1     4
     1     5
     2     3
     2     5
     3     4
```

```
G1.findedge(1,2) % should be present
```

```
ans = 1
```

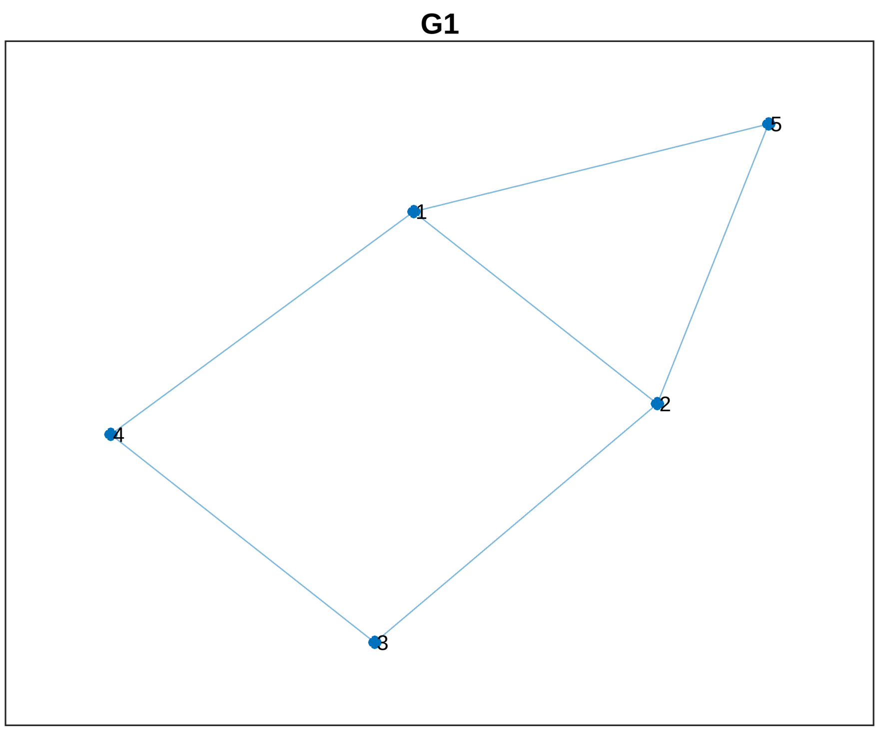
```
G1.findedge(1,3) % should not be present
```

```
ans = 0
```

2. Plotting Graphs

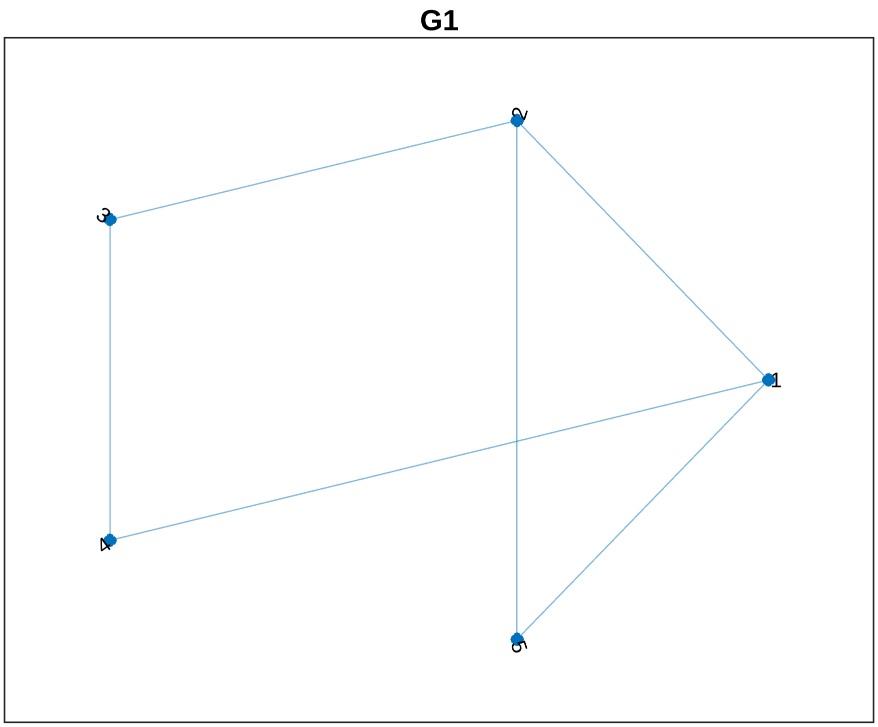
As we've seen, to plot a graph we just use the `plot()` function:

```
plot(G1); title("G1")
```



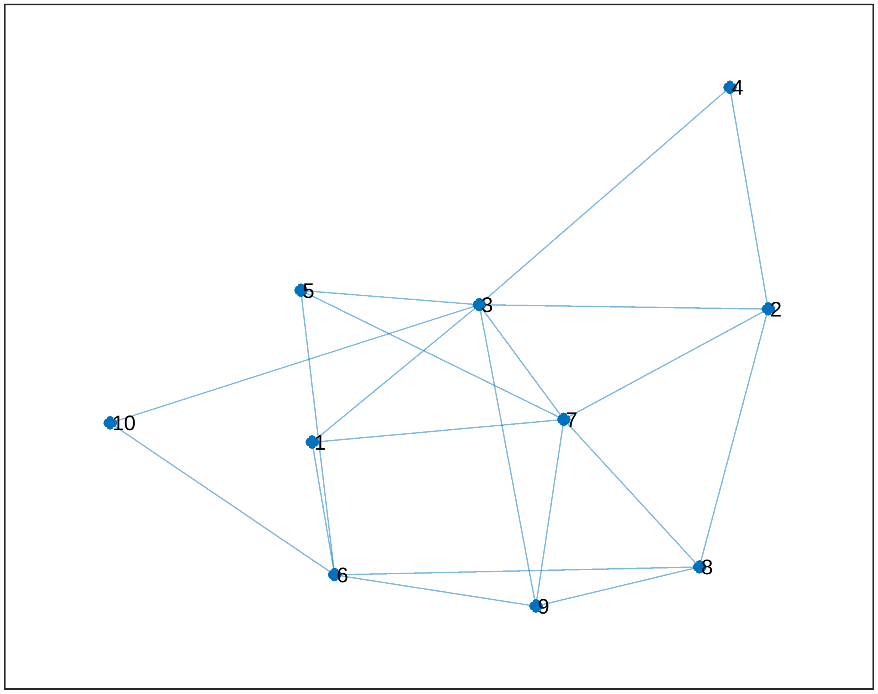
But we can also change the layout. Options include 'circle', 'force', 'subspace', ...

```
plot(G1, 'Layout','circle'); title("G1")
```



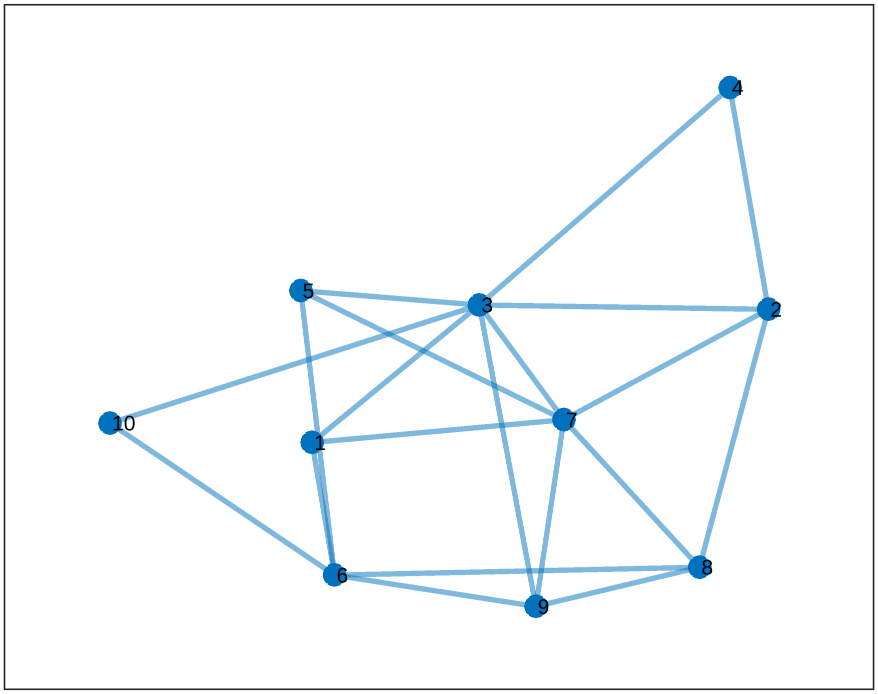
One can see the difference using a large graph. Let's make a random one with 20 vertices, and 40 edges:

```
v = 10; % number of vertices (try changing this)
e = 20; % number of edges (try changing this)
G2 = graph();
G2 = G2.addnode(v);
while (G2.numedges < e)
   s = randi(v);
   t = randi(v);
   if ((s ~= t) && (G2.findedge(s,t)==0)) % No loops
      G2=G2.addedge(s,t);
   end
end
h=G2.plot(); % h is a plot object!!!
layout(h, 'force') % try 'force', 'layered', 'circle','subspace','force3'
```



Since this is just a standard `plot` object, we can apply modifers has we've done before, such as 'LineWidth', 'MarkerSize', from Week 5. Example:

```
G2.plot('LineWidth',2, 'MarkerSize', 8)
```



There are some specifications that are particular to graphs, such as 'NodeFontSize', 'NodeColor', 'EdgeColor', etc. We'll see some of these below.

3. Adjacency Matrix

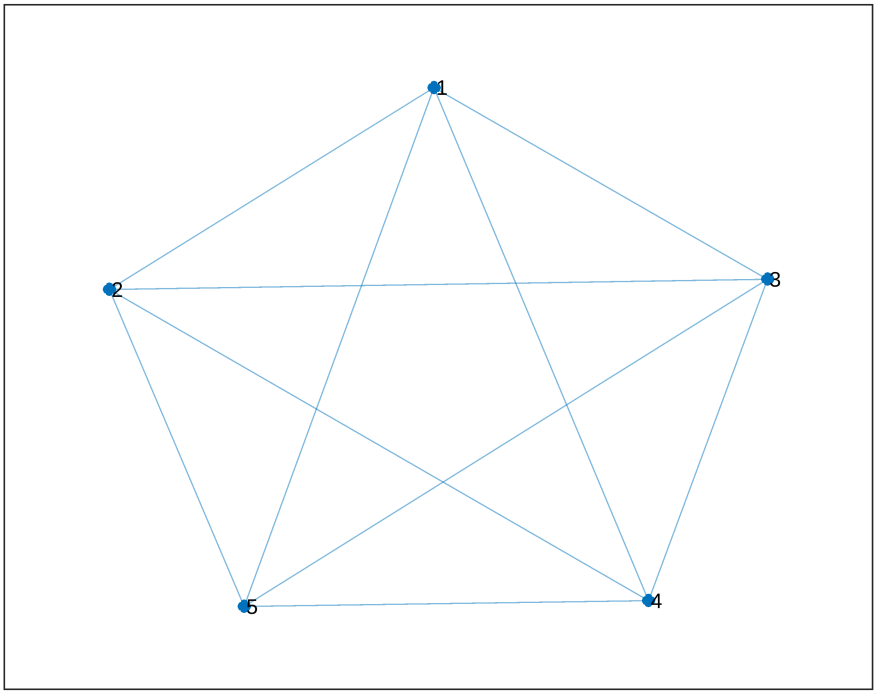
Since MATLAB is all about matrices, it is not surprising that it uses them to represent graphs. This can be done using the graph's adjacency matrix. This is a matrix, $A = (a_{ij})$ where $a_{ij} = \begin{cases} 1 & \{i, j\} \text{ is an edge} \\ 0 & \text{otherwise} \end{cases}$

Here is K_5, the complete graph on 5 vertices:

```
A5 = ones(5) - diag(ones(1,5))
```

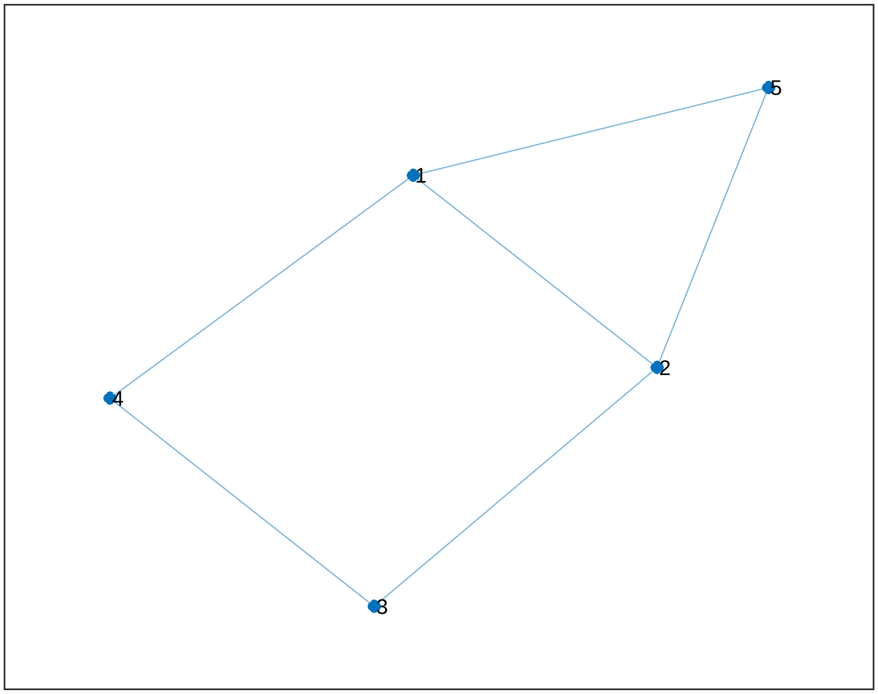
```
A5 = 5×5
     0     1     1     1     1
     1     0     1     1     1
     1     1     0     1     1
     1     1     1     0     1
     1     1     1     1     0
```

```
K5 = graph(A5);
plot(K5)
```



If we have a graph, we can extract the adjacency matrix. By default, this is a sparse matrx, so we will convert to full.

```
plot(G1)
```




```
A2 = full(G1.adjacency)
```

```
A2 = 5×5
     0     1     0     1     1
     1     0     1     0     1
     0     1     0     1     0
     1     0     1     0     0
     1     1     0     0     0
```

Notice that $A2$ is symmetric.

4. Directed Graphs

Use the digraph() function. Example: let's make the example shown on the cover page of today's slides.

```
s = [1 1 2 2 3 3 4 5 6]
```

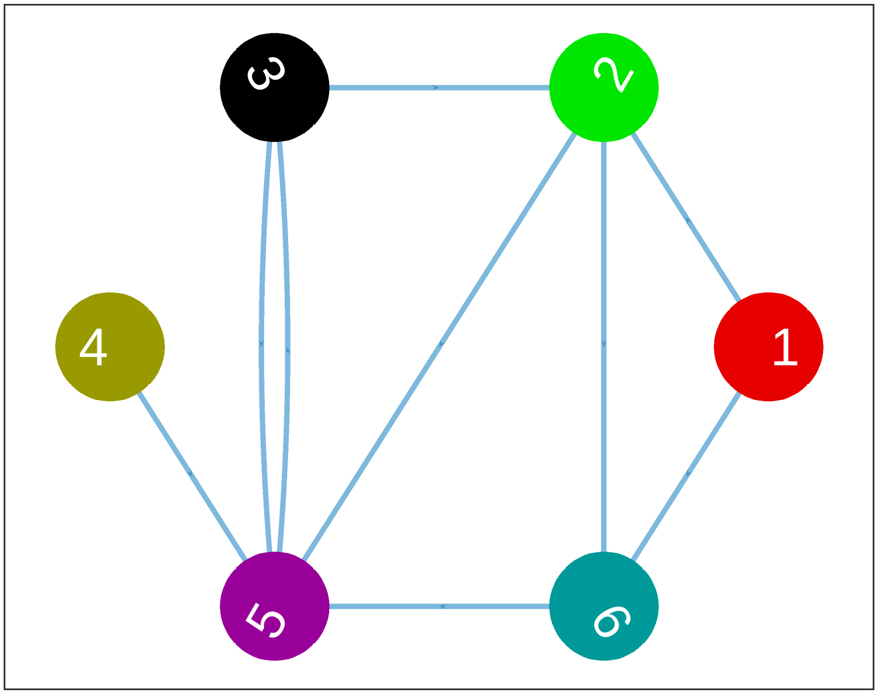
```
s = 1×9
     1     1     2     2     3     3     4     5     6
```

```
t = [2 6 5 6 2 5 5 3 5]
```

```
t = 1×9
     2     6     5     6     2     5     5     3     5
```

```
D = digraph(s,t);
plot(D, 'Layout','circle', 'MarkerSize', 40, ...
    'NodeColor', [.9 0 0; 0 .9 0; 0 0 0.0; .6 .6 0; .6 0 .6; 0 .6 .6], ...
    'LineWidth', 2, 'ArrowSize',16, 'NodeFontSize',20,...
```

```
    'NodeLabelColor', 'white');
```



5. PageRank

See notes from class.

```
A = full(D.adjacency)
```

```
A = 6×6
     0     1     0     0     0     1
     0     0     0     0     1     1
     0     1     0     0     1     0
     0     0     0     0     1     0
     0     0     1     0     0     0
     0     0     0     0     1     0
```

```
N = length(A);
S = A;
for i=1:N
   S(i,:)  = S(i,:)/sum(S(i,:));
end
disp(S)
```

```
         0    0.5000         0         0         0    0.5000
         0         0         0         0    0.5000    0.5000
         0    0.5000         0         0    0.5000         0
         0         0         0         0    1.0000         0
         0         0    1.0000         0         0         0
         0         0         0         0    1.0000         0
```

```
sigma = 0.85;
G = (sigma*S + (1-sigma)/N)';
u = ones(N,1)/N;
v = zeros(N,1);
d = norm(u-v);
TOL = 1.0e-3;
k=0; % iteration count
while(d > TOL)
   k=k+1;
   v = u;
   u = G*u;
   d = norm(u-v);
end
fprintf("Power method tool %d iterations\n", k);
```

```
Power method tool 15 iterations
```

```
[ranked_score,ranking]=sort(u, 'descend');
fprintf('The ranking is: ');
```

```
The ranking is:
```

```
for i=1:D.numnodes
   fprintf('Rank %d : Vertex %d (Value %5.3f)\n', ...
      i, ranking(i), ranked_score(i));
end
```

```
Rank 1 : Vertex 5 (Value 0.348)
Rank 2 : Vertex 3 (Value 0.321)
Rank 3 : Vertex 2 (Value 0.172)
Rank 4 : Vertex 6 (Value 0.109)
Rank 5 : Vertex 1 (Value 0.025)
Rank 6 : Vertex 4 (Value 0.025)
```