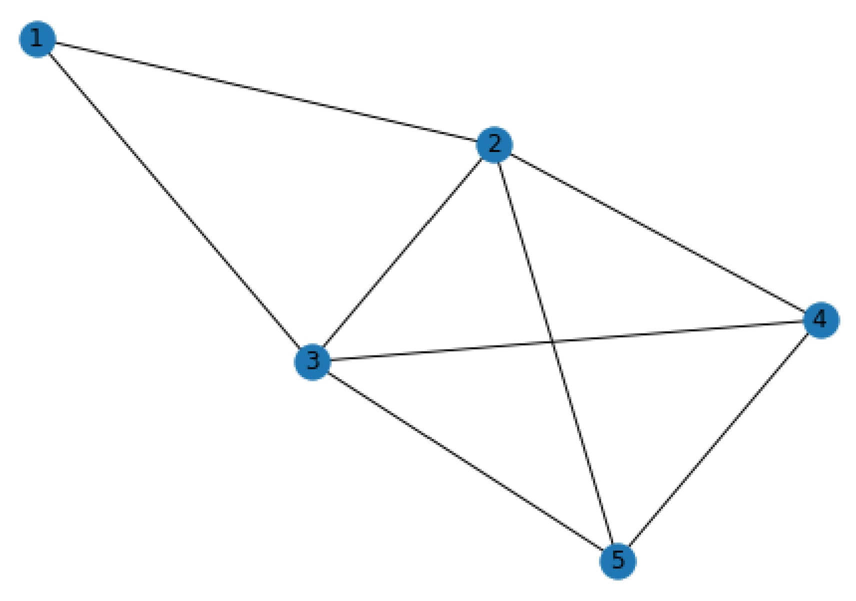
In [1]:
```
import networkx as nx
import numpy as np
G = nx.Graph()
G.add_nodes_from([1,2,3,4,5])
G.add_edges_from([(1,2),(1,3),(2,3),(2,4),(3,4),(2,5),(3,5),(4,5)])
A = nx.adjacency_matrix(G).todense()
A = np.array(A) # adjacency matrix
D = [G.degree[node] for node in G.nodes()]
D = np.diag(D) # degree matrix
```

In [2]:
```
nx.draw(G, with_labels = True)
```



In [3]:
```
# from scipy.linalg import fractional_matrix_power
# D_inv = fractional_matrix_power(D, -1)
D_inv = np.linalg.inv(D)
D_inv
```

Out[3]:
```
array([[ 0.5       ,  0.        ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.25      ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.25      ,  0.        ,  0.        ],
       [-0.        , -0.        , -0.        ,  0.33333333, -0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.        ,  0.33333333]])
```

In [4]:
```
L = D-A
L
```

Out[4]:
```
array([[ 2, -1, -1,  0,  0],
       [-1,  4, -1, -1, -1],
       [-1, -1,  4, -1, -1],
       [ 0, -1, -1,  3, -1],
       [ 0, -1, -1, -1,  3]])
```

In [5]:
```
L_rw = D_inv.dot(L)
L_rw
```

Out[5]:
```
array([[ 1.        , -0.5       , -0.5       ,  0.        ,  0.        ],
       [-0.25      ,  1.        , -0.25      , -0.25      , -0.25      ],
       [-0.25      , -0.25      ,  1.        , -0.25      , -0.25      ],
       [ 0.        , -0.33333333, -0.33333333,  1.        , -0.33333333],
       [ 0.        , -0.33333333, -0.33333333, -0.33333333,  1.        ]])
```

In [6]:
```
from numpy import linalg as LA
eigenvalues, eigenvectors = LA.eig(L_rw)
np.round(eigenvalues,3)
```

Out[6]:
```
array([ 1.564,  0.852, -0.   ,  1.25 ,  1.333])
```

In [7]:
```
eigenvectors
```

Out[7]:
```
array([[ 7.09192539e-01, -7.89120089e-01,  4.47213595e-01,
        -1.20816255e-15, -1.83183632e-30],
       [-4.00221100e-01, -1.16526856e-01,  4.47213595e-01,
         7.07106781e-01,  9.81307787e-16],
       [-4.00221100e-01, -1.16526856e-01,  4.47213595e-01,
        -7.07106781e-01, -9.81307787e-16],
       [ 2.97230621e-01,  4.18409171e-01,  4.47213595e-01,
        -2.63217411e-16, -7.07106781e-01],
       [ 2.97230621e-01,  4.18409171e-01,  4.47213595e-01,
        -1.52195109e-16,  7.07106781e-01]])
```

In [10]:
```
# from scipy.linalg import fractional_matrix_power
# D_ = fractional_matrix_power(D, -0.5)
# D_
D_ = D.tolist()
for i,item in enumerate(D_):
    for j,num in enumerate(item):
        if num!=0:
            D_[i][j] = 1/np.power(num,0.5)
D_ = np.array(D_)
D_  # D^(-1/2)
```

Out[10]:
```
array([[0.70710678, 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.5       , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.5       , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.57735027, 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.57735027]])
```

In [11]:
```
L_sym = D_.dot(L).dot(D_)
L_sym
```

Out[11]:
```
array([[ 1.        , -0.35355339, -0.35355339,  0.        ,  0.        ],
       [-0.35355339,  1.        , -0.25      , -0.28867513, -0.28867513],
       [-0.35355339, -0.25      ,  1.        , -0.28867513, -0.28867513],
       [ 0.        , -0.28867513, -0.28867513,  1.        , -0.33333333],
       [ 0.        , -0.28867513, -0.28867513, -0.33333333,  1.        ]])
```

In [12]:
```
from numpy import linalg as LA
eigenvalues2, eigenvectors2 = LA.eig(L_sym)
np.round(eigenvalues2,3)
```

Out[12]:
```
array([ 1.564,  0.852, -0.   ,  1.25 ,  1.333])
```

In [13]:
```
eigenvectors2
```

Out[13]:
```
array([[-5.97523398e-01,  7.19698401e-01,  3.53553391e-01,
         3.04453569e-16, -3.05941490e-16],
       [ 4.76876298e-01,  1.50296361e-01,  5.00000000e-01,
        -7.07106781e-01,  1.17646874e-16],
       [ 4.76876298e-01,  1.50296361e-01,  5.00000000e-01,
         7.07106781e-01,  1.19409513e-16],
       [-3.06711412e-01, -4.67362931e-01,  4.33012702e-01,
        -7.44551294e-16,  7.07106781e-01],
       [-3.06711412e-01, -4.67362931e-01,  4.33012702e-01,
         1.37281932e-15, -7.07106781e-01]])
```

In [15]:
```
w = eigenvectors2[:,2]
w  # eigenvector corresponding to 0 eigenvalue using L_sym
```

Out[15]:
```
array([0.35355339, 0.5       , 0.5       , 0.4330127 , 0.4330127 ])
```

In [16]:
```
u = eigenvectors[:,2]
u # eigenvector corresponding to 0 eigenvalue using L_rw
```

Out[16]:
```
array([0.4472136, 0.4472136, 0.4472136, 0.4472136, 0.4472136])
```

In [19]:
```
# from scipy.linalg import fractional_matrix_power
# Di = fractional_matrix_power(D, 0.5)
# Di
Di = D.tolist()
for i,item in enumerate(Di):
    for j,num in enumerate(item):
        if num!=0:
            Di[i][j] = np.power(num,0.5)
Di = np.array(Di)
Di # # D^(1/2)
```

Out[19]:
```
array([[1.41421356, 0.        , 0.        , 0.        , 0.        ],
       [0.        , 2.        , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 2.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 1.73205081, 0.        ],
       [0.        , 0.        , 0.        , 0.        , 1.73205081]])
```

In [22]:
```
w_ = Di.dot(u.T) # w = D^(1/2)*u
w_
```

Out[22]:
```
array([0.63245553, 0.89442719, 0.89442719, 0.77459667, 0.77459667])
```

In [23]:
```
w_/w
```

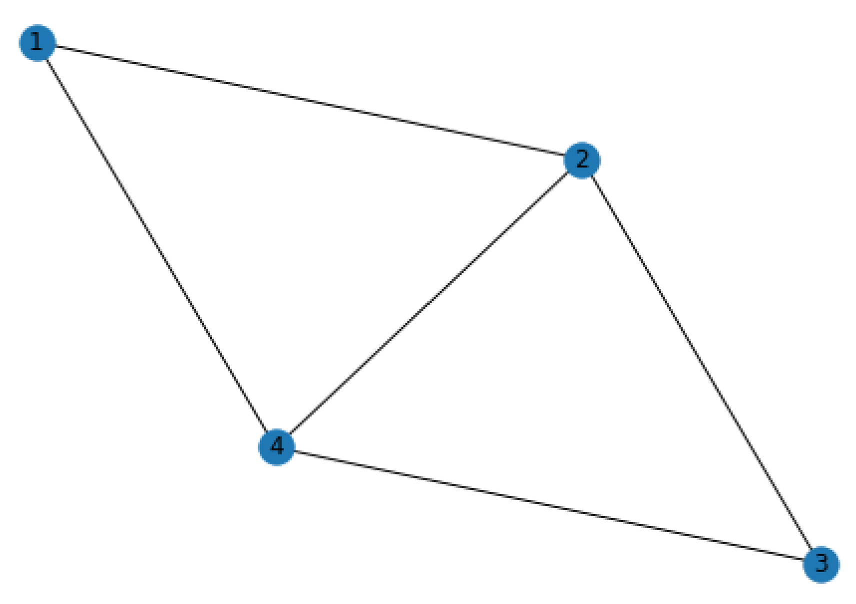
Out[23]:
```
array([1.78885438, 1.78885438, 1.78885438, 1.78885438, 1.78885438])
```

In [24]:
```
# constant values
```

Another Graph Solution

In [27]:
```
G1 = nx.Graph()
G1.add_nodes_from('1234')
G1.add_edges_from([('1','2'),('1','4'),('2','3'),('2','4'),('3','4')])
A1 = nx.adjacency_matrix(G1).todense() # adjacency matrix
A1 = np.array(A1)
D1 = [G1.degree[node] for node in G1.nodes()]
D1 = np.diag(D1)
```

In [28]:
```
nx.draw(G1, with_labels = True)
```



In [32]:
```
# from scipy.linalg import fractional_matrix_power
# D1_inv = fractional_matrix_power(D1, -1)
D1_inv = np.linalg.inv(D1)
D1_inv
```

Out[32]:
```
array([[0.5       , 0.        , 0.        , 0.        ],
       [0.        , 0.33333333, 0.        , 0.        ],
       [0.        , 0.        , 0.5       , 0.        ],
       [0.        , 0.        , 0.        , 0.33333333]])
```

In [34]:
```
L1 = D1-A1
L1
```

Out[34]:
```
array([[ 2, -1,  0, -1],
       [-1,  3, -1, -1],
       [ 0, -1,  2, -1],
       [-1, -1, -1,  3]])
```

In [35]:
```
L_rw = D1_inv.dot(L1)
L_rw
```

Out[35]:
```
array([[ 1.        , -0.5       ,  0.        , -0.5       ],
       [-0.33333333,  1.        , -0.33333333, -0.33333333],
       [ 0.        , -0.5       ,  1.        , -0.5       ],
       [-0.33333333, -0.33333333, -0.33333333,  1.        ]])
```

In [36]:
```
from numpy import linalg as LA
eigenvalues, eigenvectors = LA.eig(L_rw)
np.round(eigenvalues,3)
```

Out[36]:
```
array([0.   , 1.   , 1.667, 1.333])
```

In [37]:
```
eigenvectors
```

Out[37]:
```
array([[ 5.00000000e-01,  7.07106781e-01, -5.88348405e-01,
        -2.20294064e-16],
       [ 5.00000000e-01,  5.58454537e-17,  3.92232270e-01,
        -7.07106781e-01],
       [ 5.00000000e-01, -7.07106781e-01, -5.88348405e-01,
        -2.20294064e-16],
       [ 5.00000000e-01, -8.50996706e-17,  3.92232270e-01,
         7.07106781e-01]])
```

In [38]:
```
# from scipy.linalg import fractional_matrix_power
# D1_ = fractional_matrix_power(D1, -0.5)
# D1_
D1_ = D1.tolist()
for i,item in enumerate(D1_):
    for j,num in enumerate(item):
        if num!=0:
            D1_[i][j] = 1/np.power(num,0.5)
D1_ = np.array(D1_)
D1_  # D^(-1/2)
```

Out[38]:
```
array([[0.70710678, 0.        , 0.        , 0.        ],
       [0.        , 0.57735027, 0.        , 0.        ],
       [0.        , 0.        , 0.70710678, 0.        ],
       [0.        , 0.        , 0.        , 0.57735027]])
```

In [39]:
```
L_sym = D1_.dot(L1).dot(D1_)
L_sym
```

Out[39]:
```
array([[ 1.        , -0.40824829,  0.        , -0.40824829],
       [-0.40824829,  1.        , -0.40824829, -0.33333333],
       [ 0.        , -0.40824829,  1.        , -0.40824829],
       [-0.40824829, -0.33333333, -0.40824829,  1.        ]])
```

In [40]:
```
from numpy import linalg as LA
eigenvalues2, eigenvectors2 = LA.eig(L_sym)
np.round(eigenvalues2,3)
```

Out[40]:
```
array([0.   , 1.   , 1.667, 1.333])
```

In [41]:
```
eigenvectors2
```

Out[41]:
```
array([[ 4.47213595e-01,  7.07106781e-01, -5.47722558e-01,
         9.10626451e-17],
       [ 5.47722558e-01,  2.28871080e-16,  4.47213595e-01,
        -7.07106781e-01],
       [ 4.47213595e-01, -7.07106781e-01, -5.47722558e-01,
         9.10626451e-17],
       [ 5.47722558e-01,  2.28871080e-16,  4.47213595e-01,
         7.07106781e-01]])
```

In [52]:
```
w = eigenvectors2[:,0]
w  # eigenvector corresponding to 0 eigenvalue using L_sym
```

Out[52]:
```
array([0.4472136 , 0.54772256, 0.4472136 , 0.54772256])
```

In [53]:
```
u = eigenvectors[:,0]
u # eigenvector corresponding to 0 eigenvalue using L_rw
```

Out[53]:
```
array([0.5, 0.5, 0.5, 0.5])
```

In [54]:
```
# from scipy.linalg import fractional_matrix_power
# Di = fractional_matrix_power(D1, 0.5)
# Di
Di = D1.tolist()
for i,item in enumerate(Di):
    for j,num in enumerate(item):
        if num!=0:
            Di[i][j] = np.power(num,0.5)
Di = np.array(Di)
Di # # D^(1/2)
```

Out[54]:
```
array([[1.41421356, 0.        , 0.        , 0.        ],
       [0.        , 1.73205081, 0.        , 0.        ],
       [0.        , 0.        , 1.41421356, 0.        ],
       [0.        , 0.        , 0.        , 1.73205081]])
```

In [55]:
```
w_ = Di.dot(u.T) # w = D^(1/2)*u
w_
```

Out[55]:
```
array([0.70710678, 0.8660254 , 0.70710678, 0.8660254 ])
```

In [56]:
```
w_/w
```

Out[56]:
```
array([1.58113883, 1.58113883, 1.58113883, 1.58113883])
```

In [47]:
```
# constant values
```