Master Informatique, parcours MALIA & MIASHS

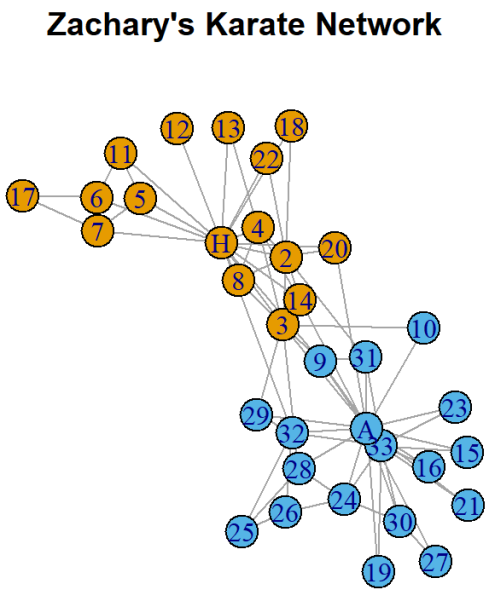
Carnets de note Python pour le cours de Network Analysis for Information Retrieval

Julien Velcin, laboratoire ERIC, Université Lyon 2

Prérequis

In [5]:
```
from torch_geometric.datasets import karate
import networkx as nx
from torch_geometric.utils import to_networkx
```

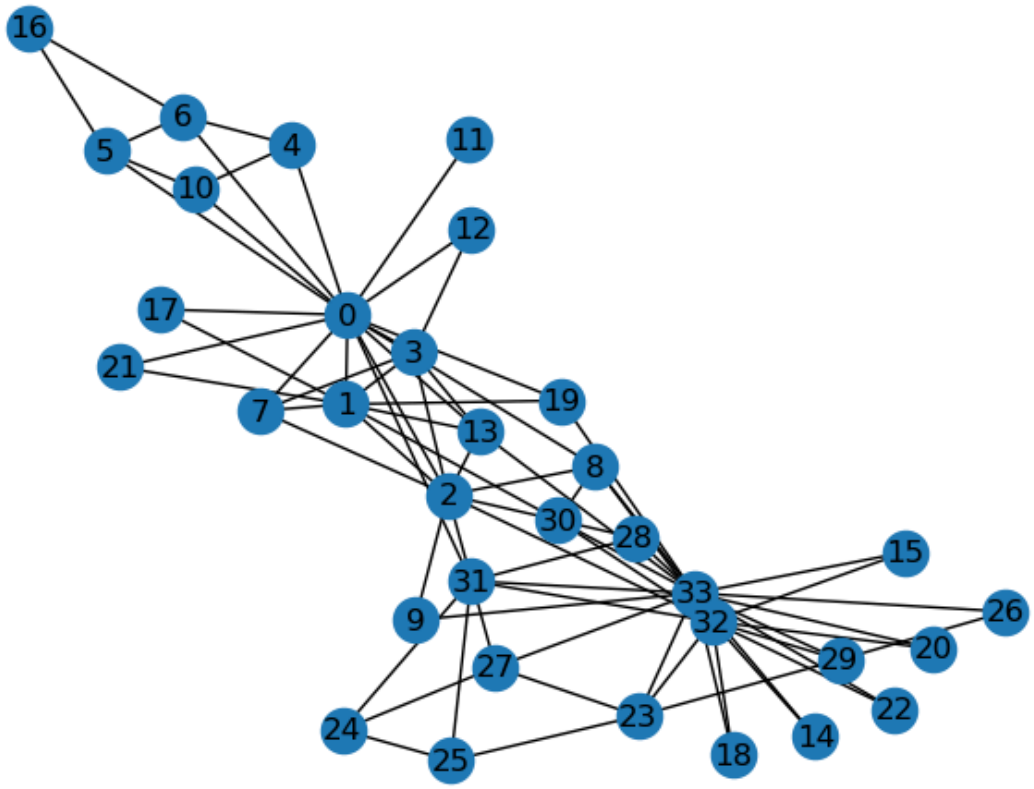
Visualisation d'un graphe "jouet"



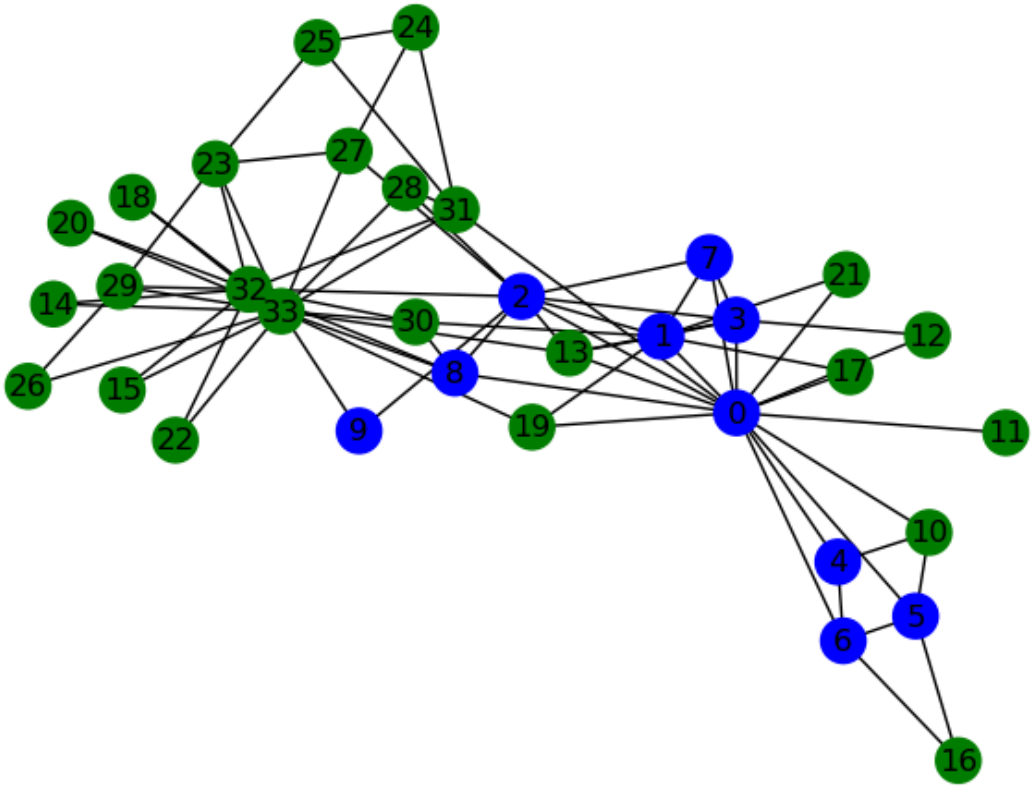
In [1]:
```
dataset = karate.KarateClub()
dataset._data
```

Out[1]:
```
Data(x=[34, 34], edge_index=[2, 156], y=[34], train_mask=[34])
```

In [2]:
```
g = to_networkx(dataset._data, to_undirected=True)
nx.draw(g, with_labels=True)
```



In [3]:
```
color_map = []
for node in g:
    if node < 10:
        color_map.append('blue')
    else:
        color_map.append('green')
nx.draw(g, node_color=color_map, with_labels=True)
```



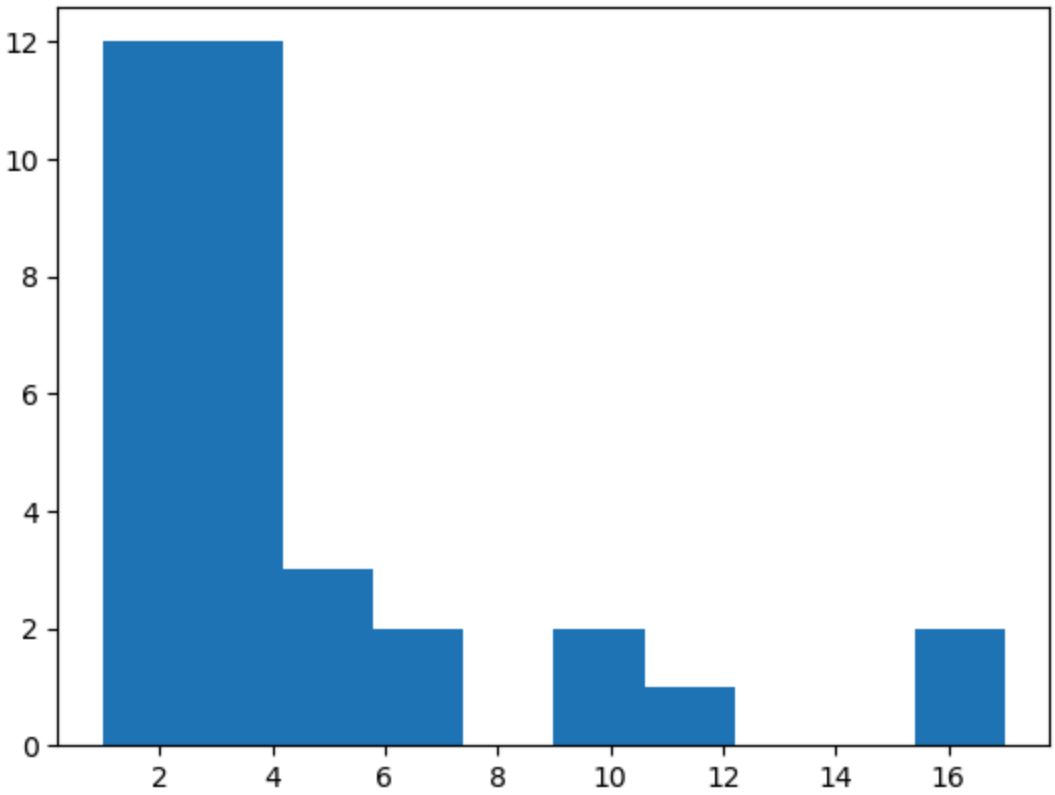
In [20]:
```
list(g.adj[1])
```

Out[20]:
```
[0, 2, 3, 7, 13, 17, 19, 21, 30]
```

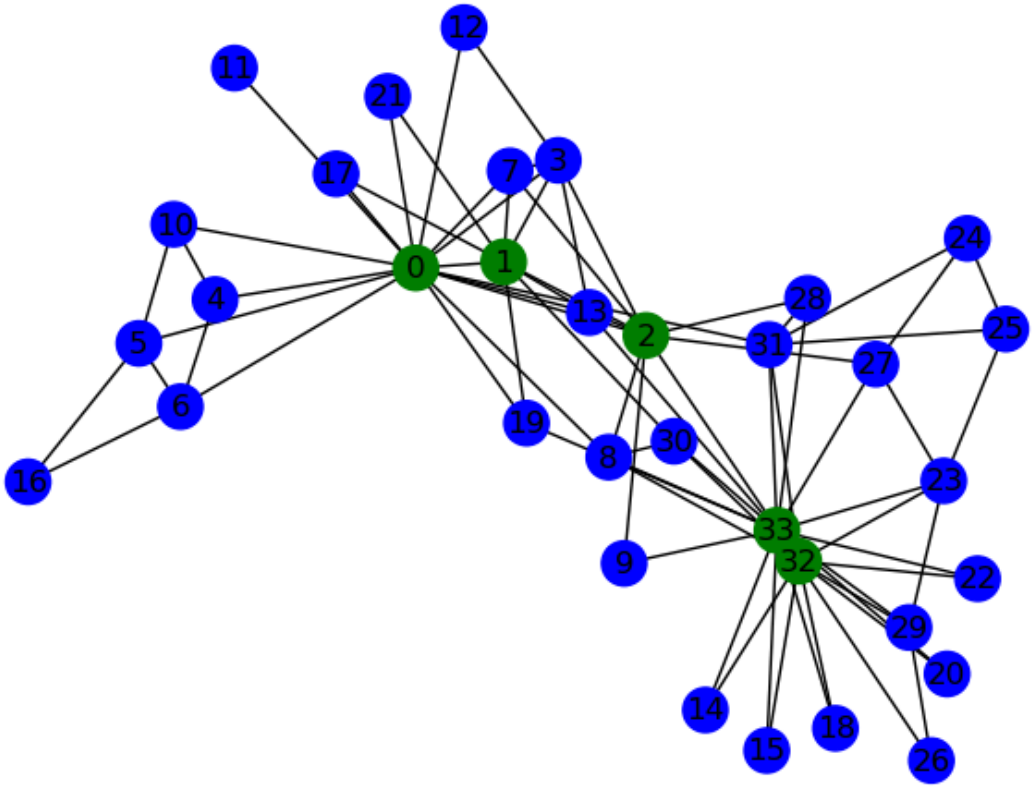
In [21]:
```
g.degree()
```

Out[21]:
```
DegreeView({0: 16, 1: 9, 2: 10, 3: 6, 4: 3, 5: 4, 6: 4, 7: 4, 8: 5, 9: 2, 10: 3,
1: 1, 12: 2, 13: 5, 14: 2, 15: 2, 16: 2, 17: 2, 18: 2, 19: 3, 20: 2, 21: 2, 22: 2,
23: 5, 24: 3, 25: 3, 26: 2, 27: 4, 28: 3, 29: 4, 30: 4, 31: 6, 32: 12, 33: 17})
```

In [22]:
```
import numpy as np
import matplotlib.pyplot as plt
d_degree = dict(g.degree())
n, bins, patches = plt.hist(d_degree.values())
plt.show()
```



In [23]:
```
color_map = []
for node in g:
    if d_degree[node]<8:
        color_map.append('blue')
    else:
        color_map.append('green')
nx.draw(g, node_color=color_map, with_labels=True)
```



In [24]:
```
g.adj
```

Out[24]:
```
AdjacencyView({0: {1: {}, 2: {}, 3: {}, 4: {}, 5: {}, 6: {}, 7: {}, 8: {}, 10: {},
11: {}, 12: {}, 13: {}, 17: {}, 19: {}, 21: {}, 31: {}}, 1: {0: {}, 2: {}, 3: {},
7: {}, 13: {}, 17: {}, 19: {}, 21: {}, 30: {}}, 2: {0: {}, 1: {}, 3: {}, 7: {}, 8:
{}, 9: {}, 13: {}, 27: {}, 28: {}, 32: {}}, 3: {0: {}, 1: {}, 2: {}, 7: {}, 12:
{}, 13: {}}, 4: {0: {}, 6: {}, 10: {}}, 5: {0: {}, 6: {}, 10: {}, 16: {}}, 6: {0:
{}, 4: {}, 5: {}, 16: {}}, 7: {0: {}, 1: {}, 2: {}, 3: {}}, 8: {0: {}, 2: {}, 30:
{}, 32: {}, 33: {}}, 9: {2: {}, 33: {}}, 10: {0: {}, 4: {}, 5: {}}, 11: {0: {}}, 1
2: {0: {}, 3: {}}, 13: {0: {}, 1: {}, 2: {}, 3: {}, 33: {}}, 14: {32: {}, 33: {}},
15: {32: {}, 33: {}}, 16: {5: {}, 6: {}}, 17: {0: {}, 1: {}}, 18: {32: {}, 33:
{}}, 19: {0: {}, 1: {}, 33: {}}, 20: {32: {}, 33: {}}, 21: {0: {}, 1: {}}, 22: {3
2: {}, 33: {}}, 23: {25: {}, 27: {}, 29: {}, 32: {}, 33: {}}, 24: {25: {}, 27: {},
31: {}}, 25: {23: {}, 24: {}, 31: {}}, 26: {29: {}, 33: {}}, 27: {2: {}, 23: {}, 2
4: {}, 33: {}}, 28: {2: {}, 31: {}, 33: {}}, 29: {23: {}, 26: {}, 32: {}, 33: {}},
30: {1: {}, 8: {}, 32: {}, 33: {}}, 31: {0: {}, 24: {}, 25: {}, 28: {}, 32: {}, 3
3: {}}, 32: {2: {}, 8: {}, 14: {}, 15: {}, 18: {}, 20: {}, 22: {}, 23: {}, 29: {},
30: {}, 31: {}, 33: {}}, 33: {8: {}, 9: {}, 13: {}, 14: {}, 15: {}, 18: {}, 19:
{}, 20: {}, 22: {}, 23: {}, 26: {}, 27: {}, 28: {}, 29: {}, 30: {}, 31: {}, 32:
{}}})
```

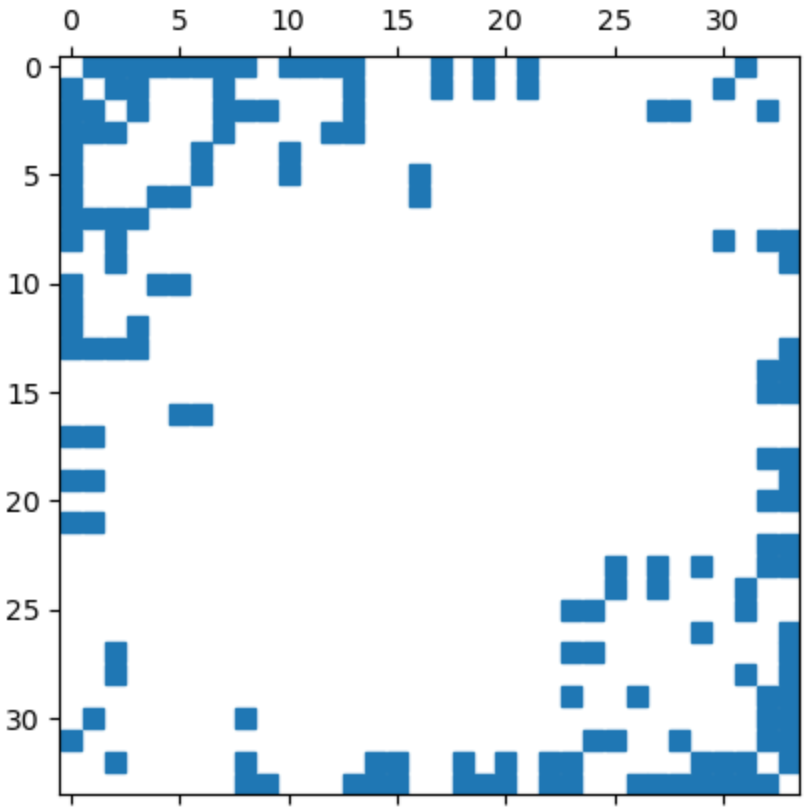
In [25]:
```
A = nx.to_scipy_sparse_array(g)
```

In [26]:
```
A.todense()
```

Out[26]:
```
array([[0, 1, 1, ..., 1, 0, 0],
       [1, 0, 1, ..., 0, 0, 0],
       [1, 1, 0, ..., 0, 1, 0],
       ...,
       [1, 0, 0, ..., 0, 1, 1],
       [0, 0, 1, ..., 1, 0, 1],
       [0, 0, 0, ..., 1, 1, 0]])
```

In [27]:
```
import matplotlib.pylab as plt
import scipy.sparse as sparse
plt.spy(A, markersize=7)
```

Out[27]:
```
<matplotlib.lines.Line2D at 0x17fb7e350>
```

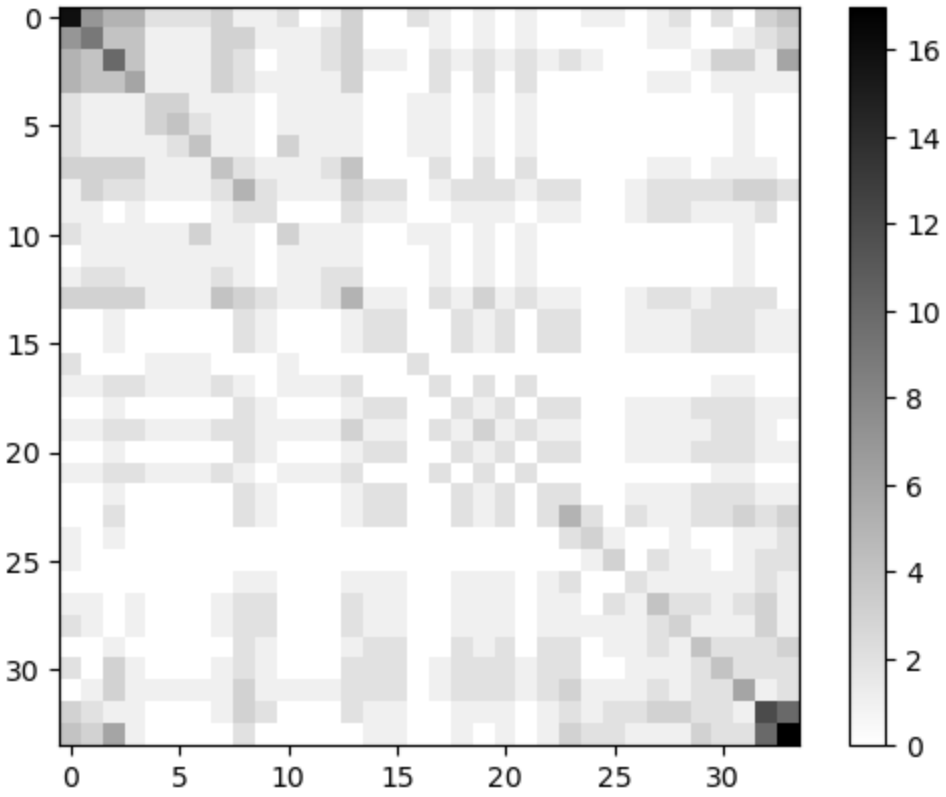


In [28]:
```
A2 = A@A
A2.todense()
```

Out[28]:
```
array([[16,  7,  5, ...,  0,  3,  4],
       [ 7,  9,  4, ...,  1,  2,  3],
       [ 5,  4, 10, ...,  3,  1,  6],
       ...,
       [ 0,  1,  3, ...,  6,  1,  2],
       [ 3,  2,  1, ...,  1, 12, 10],
       [ 4,  3,  6, ...,  2, 10, 17]])
```

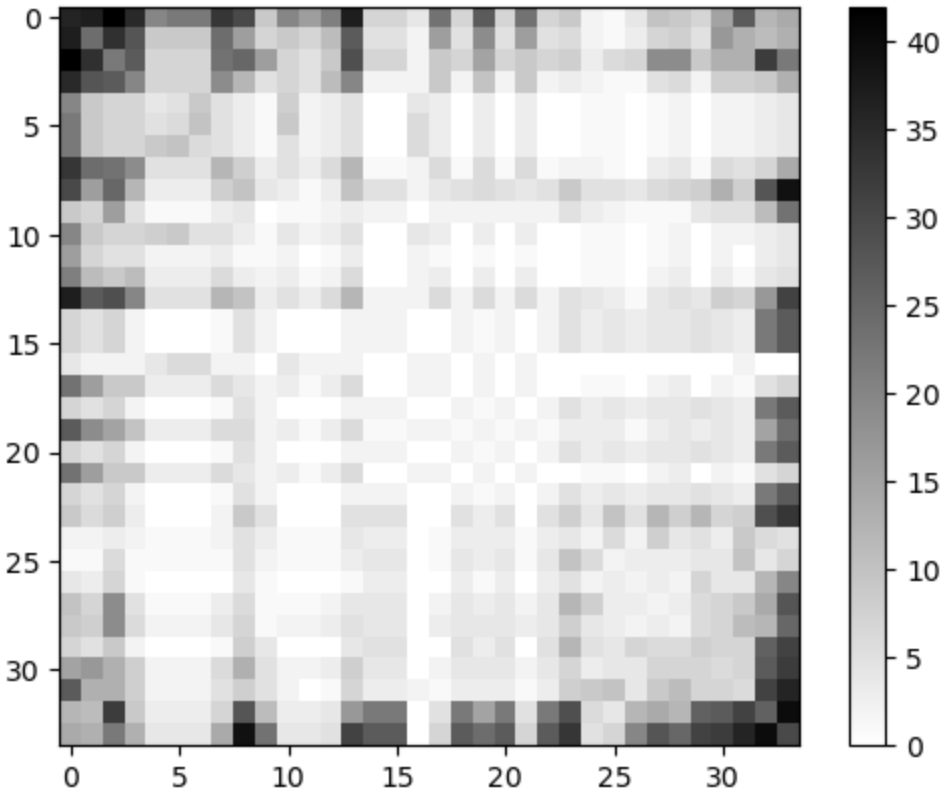
In [29]:
```
d = A2.todense()
plt.imshow(d,interpolation='none',cmap='binary')
plt.colorbar()
```

Out[29]:
```
<matplotlib.colorbar.Colorbar at 0x17fbbf090>
```



In [30]:
```
A3 = A@A2
d3 = A3.todense()
plt.imshow(d3,interpolation='none',cmap='binary')
plt.colorbar()
```

Out[30]: <matplotlib.colorbar.Colorbar at 0x17fbea150>



Mesures de centralité

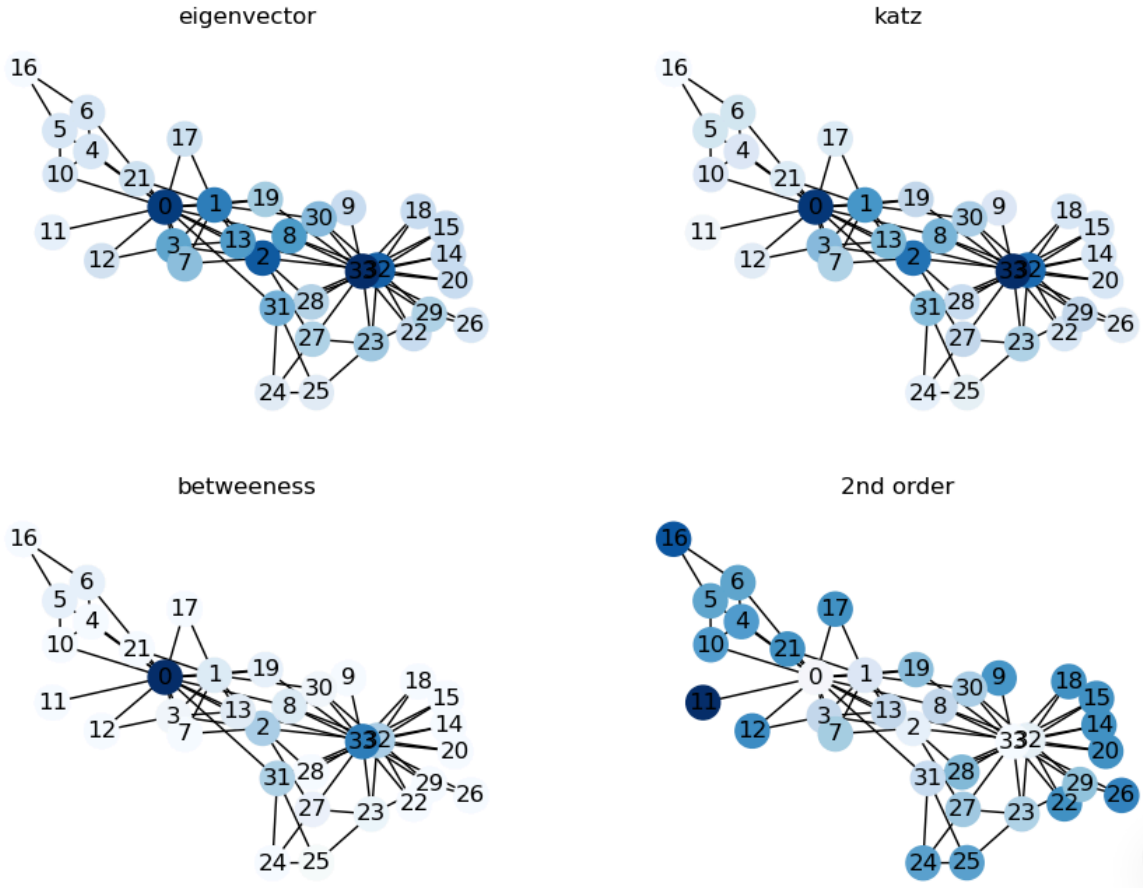
On va à présent calculer et afficher un certain nombre de mesures de centralité déjà implémentées dans la librairie.

In [65]:
```
centralities = [nx.eigenvector_centrality, nx.katz_centrality, nx.betweenness_ce
centralities_names = ["eigenvector", "katz", "betweeness", "2nd order"]
val_cent = []

for c in centralities:
    val_cent.append(c(g))
```

In [69]:
```
# on enregistre le layout du graphe pour la visualisation
my_pos = nx.spring_layout(g)
```

In [68]:
```
figure = plt.figure(figsize=(11, 8))
for i in range(len(centralities)):
    figure.add_subplot(2, 2, i+1)
    plt.title(centralities_names[i])
    nx.draw(g, pos = my_pos, with_labels=True, node_color=[*val_cent[i].values()],
plt.show()
```



In [ ]: