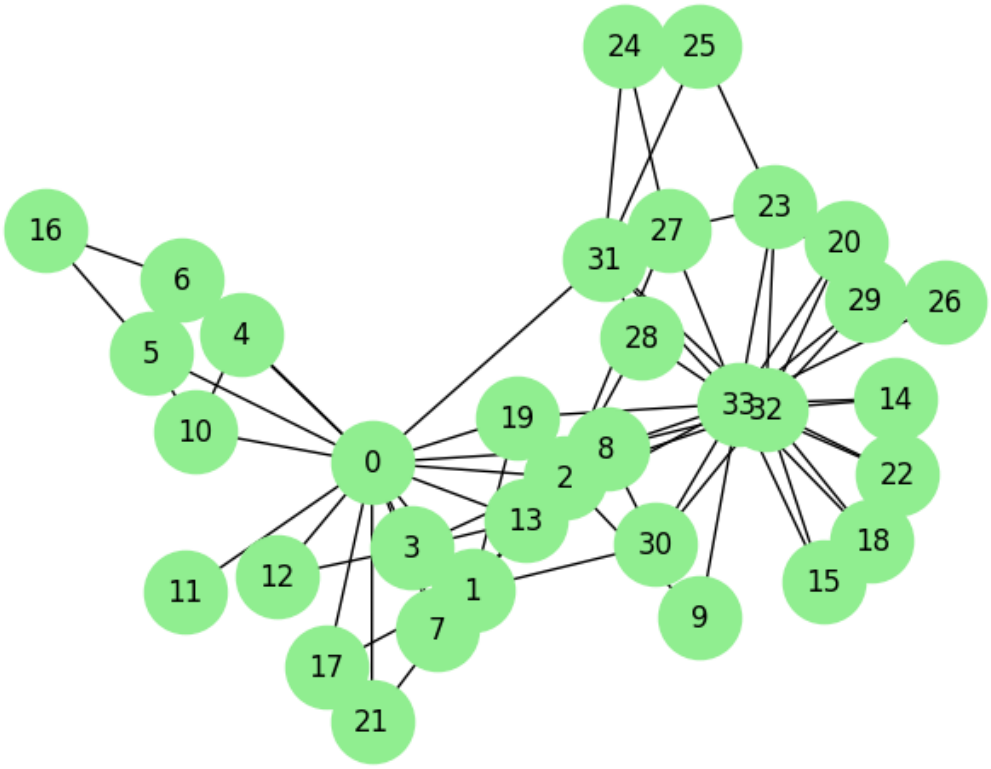
lab-8-22mcb1002

April 30, 2023

[13]:
```
import pandas as pd
import numpy as np
import networkx as nx
from math import log
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import plotly.express as px
```

[9]:
```
nodes = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
         20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]
edges = [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8),
         (0, 10), (0, 11), (0, 12), (0, 13), (0, 17), (0, 19), (0, 21), (0, 31),
         (1, 2), (1, 3), (1, 7), (1, 13), (1, 17), (1, 19), (1, 21), (1, 30),
         (2, 3), (2, 7), (2, 8), (2, 9), (2, 13), (2, 27), (2, 28), (2, 32),
         (3, 7), (3, 12), (3, 13), (4, 6), (4, 10), (5, 6), (5, 10), (5, 16),
         (6, 16), (8, 30), (8, 32), (8, 33), (9, 33), (13, 33), (14, 32), (14,␣
 ↪33),
         (15, 32), (15, 33), (18, 32), (18, 33), (19, 33), (20, 32), (20, 33),
         (22, 32), (22, 33), (23, 25), (23, 27), (23, 29), (23, 32), (23, 33),
         (24, 25), (24, 27), (24, 31), (25, 31), (26, 29), (26, 33), (27, 33),
         (28, 31), (28, 33), (29, 32), (29, 33), (30, 32), (30, 33), (31, 32),
         (31, 33), (32, 33)]
G = nx.Graph()
G.add_nodes_from(nodes)
G.add_edges_from(edges)

nx.draw(G, with_labels = True, node_size = 1200, node_color = '#90ee90')
plt.show()
```



i) Node degree of network

[5]:
```
degrees = dict(G.degree())
for node, degree in degrees.items():
    print("Node", node, "has degree", degree)
```

```
Node 0 has degree 16
Node 1 has degree 9
Node 2 has degree 10
Node 3 has degree 6
Node 4 has degree 3
Node 5 has degree 4
Node 6 has degree 4
Node 7 has degree 4
Node 8 has degree 5
Node 9 has degree 2
Node 10 has degree 3
Node 11 has degree 1
Node 12 has degree 2
Node 13 has degree 5
Node 14 has degree 2
```

```
Node 15 has degree 2
Node 16 has degree 2
Node 17 has degree 2
Node 18 has degree 2
Node 19 has degree 3
Node 20 has degree 2
Node 21 has degree 2
Node 22 has degree 2
Node 23 has degree 5
Node 24 has degree 3
Node 25 has degree 3
Node 26 has degree 2
Node 27 has degree 4
Node 28 has degree 3
Node 29 has degree 4
Node 30 has degree 4
Node 31 has degree 6
Node 32 has degree 12
Node 33 has degree 17
```

ii) Edges which will be formed in future

[15]:
```
# Using Adamic-Adar algorithm
future_edges = []
for u, v in nx.non_edges(G):
    score = sum(1 / log(G.degree(w)) for w in nx.common_neighbors(G, u, v))
    future_edges.append((u, v, {'score': score}))
sorted_edges = sorted(future_edges, key=lambda x: x[2]['score'], reverse = True)
future_edges
```

[15]:
```
[(0, 32, {'score': 1.613740043014111}),
 (0, 33, {'score': 2.7110197222973085}),
 (0, 9, {'score': 0.43429448190325176}),
 (0, 14, {'score': 0}),
 (0, 15, {'score': 0}),
 (0, 16, {'score': 1.4426950408889634}),
 (0, 18, {'score': 0}),
 (0, 20, {'score': 0}),
 (0, 22, {'score': 0}),
 (0, 23, {'score': 0}),
 (0, 24, {'score': 0.5581106265512472}),
 (0, 25, {'score': 0.5581106265512472}),
 (0, 26, {'score': 0}),
 (0, 27, {'score': 0.43429448190325176}),
 (0, 28, {'score': 0.9924051084544989}),
 (0, 29, {'score': 0}),
 (0, 30, {'score': 1.0764545478730305}),
 (1, 4, {'score': 0.36067376022224085}),
```

```
(1, 5, {'score': 0.36067376022224085}),
(1, 6, {'score': 0.36067376022224085}),
(1, 8, {'score': 1.5163157625699744}),
(1, 9, {'score': 0.43429448190325176}),
(1, 10, {'score': 0.36067376022224085}),
(1, 11, {'score': 0.36067376022224085}),
(1, 12, {'score': 0.9187843867734881}),
(1, 14, {'score': 0}),
(1, 15, {'score': 0}),
(1, 16, {'score': 0}),
(1, 18, {'score': 0}),
(1, 20, {'score': 0}),
(1, 22, {'score': 0}),
(1, 23, {'score': 0}),
(1, 24, {'score': 0}),
(1, 25, {'score': 0}),
(1, 26, {'score': 0}),
(1, 27, {'score': 0.43429448190325176}),
(1, 28, {'score': 0.43429448190325176}),
(1, 29, {'score': 0}),
(1, 31, {'score': 0.36067376022224085}),
(1, 32, {'score': 1.1556420023477334}),
(1, 33, {'score': 2.252921681630931}),
(2, 4, {'score': 0.36067376022224085}),
(2, 5, {'score': 0.36067376022224085}),
(2, 6, {'score': 0.36067376022224085}),
(2, 10, {'score': 0.36067376022224085}),
(2, 11, {'score': 0.36067376022224085}),
(2, 12, {'score': 0.9187843867734881}),
(2, 14, {'score': 0.40242960438184466}),
(2, 15, {'score': 0.40242960438184466}),
(2, 16, {'score': 0}),
(2, 17, {'score': 0.8157933735356595}),
(2, 18, {'score': 0.40242960438184466}),
(2, 19, {'score': 0.8157933735356595}),
(2, 20, {'score': 0.40242960438184466}),
(2, 21, {'score': 0.8157933735356595}),
(2, 22, {'score': 0.40242960438184466}),
(2, 23, {'score': 1.1237771248263264}),
(2, 24, {'score': 0.7213475204444817}),
(2, 25, {'score': 0}),
(2, 26, {'score': 0}),
(2, 29, {'score': 0.40242960438184466}),
(2, 30, {'score': 1.4788841522548752}),
(2, 31, {'score': 1.6733425912309228}),
(2, 33, {'score': 4.719381261461351}),
(3, 4, {'score': 0.36067376022224085}),
```

```
(3, 5, {'score': 0.36067376022224085}),
(3, 6, {'score': 0.36067376022224085}),
(3, 8, {'score': 0.7949682421254927}),
(3, 9, {'score': 0.43429448190325176}),
(3, 10, {'score': 0.36067376022224085}),
(3, 11, {'score': 0.36067376022224085}),
(3, 14, {'score': 0}),
(3, 15, {'score': 0}),
(3, 16, {'score': 0}),
(3, 17, {'score': 0.8157933735356595}),
(3, 18, {'score': 0}),
(3, 19, {'score': 0.8157933735356595}),
(3, 20, {'score': 0}),
(3, 21, {'score': 0.8157933735356595}),
(3, 22, {'score': 0}),
(3, 23, {'score': 0}),
(3, 24, {'score': 0}),
(3, 25, {'score': 0}),
(3, 26, {'score': 0}),
(3, 27, {'score': 0.43429448190325176}),
(3, 28, {'score': 0.43429448190325176}),
(3, 29, {'score': 0}),
(3, 30, {'score': 0.45511961331341866}),
(3, 31, {'score': 0.36067376022224085}),
(3, 32, {'score': 0.43429448190325176}),
(3, 33, {'score': 0.6213349345596119}),
(4, 5, {'score': 1.9922605072935597}),
(4, 7, {'score': 0.36067376022224085}),
(4, 8, {'score': 0.36067376022224085}),
(4, 9, {'score': 0}),
(4, 11, {'score': 0.36067376022224085}),
(4, 12, {'score': 0.36067376022224085}),
(4, 13, {'score': 0.36067376022224085}),
(4, 14, {'score': 0}),
(4, 15, {'score': 0}),
(4, 16, {'score': 0.7213475204444817}),
(4, 17, {'score': 0.36067376022224085}),
(4, 18, {'score': 0}),
(4, 19, {'score': 0.36067376022224085}),
(4, 20, {'score': 0}),
(4, 21, {'score': 0.36067376022224085}),
(4, 22, {'score': 0}),
(4, 23, {'score': 0}),
(4, 24, {'score': 0}),
(4, 25, {'score': 0}),
(4, 26, {'score': 0}),
(4, 27, {'score': 0}),
```

```
(4, 28, {'score': 0}),
(4, 29, {'score': 0}),
(4, 30, {'score': 0}),
(4, 31, {'score': 0.36067376022224085}),
(4, 32, {'score': 0}),
(4, 33, {'score': 0}),
(5, 7, {'score': 0.36067376022224085}),
(5, 8, {'score': 0.36067376022224085}),
(5, 9, {'score': 0}),
(5, 11, {'score': 0.36067376022224085}),
(5, 12, {'score': 0.36067376022224085}),
(5, 13, {'score': 0.36067376022224085}),
(5, 14, {'score': 0}),
(5, 15, {'score': 0}),
(5, 17, {'score': 0.36067376022224085}),
(5, 18, {'score': 0}),
(5, 19, {'score': 0.36067376022224085}),
(5, 20, {'score': 0}),
(5, 21, {'score': 0.36067376022224085}),
(5, 22, {'score': 0}),
(5, 23, {'score': 0}),
(5, 24, {'score': 0}),
(5, 25, {'score': 0}),
(5, 26, {'score': 0}),
(5, 27, {'score': 0}),
(5, 28, {'score': 0}),
(5, 29, {'score': 0}),
(5, 30, {'score': 0}),
(5, 31, {'score': 0.36067376022224085}),
(5, 32, {'score': 0}),
(5, 33, {'score': 0}),
(6, 7, {'score': 0.36067376022224085}),
(6, 8, {'score': 0.36067376022224085}),
(6, 9, {'score': 0}),
(6, 10, {'score': 1.9922605072935597}),
(6, 11, {'score': 0.36067376022224085}),
(6, 12, {'score': 0.36067376022224085}),
(6, 13, {'score': 0.36067376022224085}),
(6, 14, {'score': 0}),
(6, 15, {'score': 0}),
(6, 17, {'score': 0.36067376022224085}),
(6, 18, {'score': 0}),
(6, 19, {'score': 0.36067376022224085}),
(6, 20, {'score': 0}),
(6, 21, {'score': 0.36067376022224085}),
(6, 22, {'score': 0}),
(6, 23, {'score': 0}),
```

```
(6, 24, {'score': 0}),
(6, 25, {'score': 0}),
(6, 26, {'score': 0}),
(6, 27, {'score': 0}),
(6, 28, {'score': 0}),
(6, 29, {'score': 0}),
(6, 30, {'score': 0}),
(6, 31, {'score': 0.36067376022224085}),
(6, 32, {'score': 0}),
(6, 33, {'score': 0}),
(7, 8, {'score': 0.7949682421254927}),
(7, 9, {'score': 0.43429448190325176}),
(7, 10, {'score': 0.36067376022224085}),
(7, 11, {'score': 0.36067376022224085}),
(7, 12, {'score': 0.9187843867734881}),
(7, 13, {'score': 1.8081984819901584}),
(7, 14, {'score': 0}),
(7, 15, {'score': 0}),
(7, 16, {'score': 0}),
(7, 17, {'score': 0.8157933735356595}),
(7, 18, {'score': 0}),
(7, 19, {'score': 0.8157933735356595}),
(7, 20, {'score': 0}),
(7, 21, {'score': 0.8157933735356595}),
(7, 22, {'score': 0}),
(7, 23, {'score': 0}),
(7, 24, {'score': 0}),
(7, 25, {'score': 0}),
(7, 26, {'score': 0}),
(7, 27, {'score': 0.43429448190325176}),
(7, 28, {'score': 0.43429448190325176}),
(7, 29, {'score': 0}),
(7, 30, {'score': 0.45511961331341866}),
(7, 31, {'score': 0.36067376022224085}),
(7, 32, {'score': 0.43429448190325176}),
(7, 33, {'score': 0}),
(8, 9, {'score': 0.7872506057680129}),
(8, 10, {'score': 0.36067376022224085}),
(8, 11, {'score': 0.36067376022224085}),
(8, 12, {'score': 0.36067376022224085}),
(8, 13, {'score': 1.1479243659902538}),
(8, 14, {'score': 0.7553857282466059}),
(8, 15, {'score': 0.7553857282466059}),
(8, 16, {'score': 0}),
(8, 17, {'score': 0.36067376022224085}),
(8, 18, {'score': 0.7553857282466059}),
(8, 19, {'score': 0.713629884087002}),
```

```
(8, 20, {'score': 0.7553857282466059}),
(8, 21, {'score': 0.36067376022224085}),
(8, 22, {'score': 0.7553857282466059}),
(8, 23, {'score': 0.7553857282466059}),
(8, 24, {'score': 0}),
(8, 25, {'score': 0}),
(8, 26, {'score': 0.35295612386476116}),
(8, 27, {'score': 0.7872506057680129}),
(8, 28, {'score': 0.7872506057680129}),
(8, 29, {'score': 0.7553857282466059}),
(8, 31, {'score': 1.1160594884688466}),
(9, 10, {'score': 0}),
(9, 11, {'score': 0}),
(9, 12, {'score': 0}),
(9, 13, {'score': 0.7872506057680129}),
(9, 14, {'score': 0.35295612386476116}),
(9, 15, {'score': 0.35295612386476116}),
(9, 16, {'score': 0}),
(9, 17, {'score': 0}),
(9, 18, {'score': 0.35295612386476116}),
(9, 19, {'score': 0.35295612386476116}),
(9, 20, {'score': 0.35295612386476116}),
(9, 21, {'score': 0}),
(9, 22, {'score': 0.35295612386476116}),
(9, 23, {'score': 0.35295612386476116}),
(9, 24, {'score': 0}),
(9, 25, {'score': 0}),
(9, 26, {'score': 0.35295612386476116}),
(9, 27, {'score': 0.7872506057680129}),
(9, 28, {'score': 0.7872506057680129}),
(9, 29, {'score': 0.35295612386476116}),
(9, 30, {'score': 0.35295612386476116}),
(9, 31, {'score': 0.35295612386476116}),
(9, 32, {'score': 0.7872506057680129}),
(10, 11, {'score': 0.36067376022224085}),
(10, 12, {'score': 0.36067376022224085}),
(10, 13, {'score': 0.36067376022224085}),
(10, 14, {'score': 0}),
(10, 15, {'score': 0}),
(10, 16, {'score': 0.7213475204444817}),
(10, 17, {'score': 0.36067376022224085}),
(10, 18, {'score': 0}),
(10, 19, {'score': 0.36067376022224085}),
(10, 20, {'score': 0}),
(10, 21, {'score': 0.36067376022224085}),
(10, 22, {'score': 0}),
(10, 23, {'score': 0}),
```

```
(10, 24, {'score': 0}),
(10, 25, {'score': 0}),
(10, 26, {'score': 0}),
(10, 27, {'score': 0}),
(10, 28, {'score': 0}),
(10, 29, {'score': 0}),
(10, 30, {'score': 0}),
(10, 31, {'score': 0.36067376022224085}),
(10, 32, {'score': 0}),
(10, 33, {'score': 0}),
(11, 12, {'score': 0.36067376022224085}),
(11, 13, {'score': 0.36067376022224085}),
(11, 14, {'score': 0}),
(11, 15, {'score': 0}),
(11, 16, {'score': 0}),
(11, 17, {'score': 0.36067376022224085}),
(11, 18, {'score': 0}),
(11, 19, {'score': 0.36067376022224085}),
(11, 20, {'score': 0}),
(11, 21, {'score': 0.36067376022224085}),
(11, 22, {'score': 0}),
(11, 23, {'score': 0}),
(11, 24, {'score': 0}),
(11, 25, {'score': 0}),
(11, 26, {'score': 0}),
(11, 27, {'score': 0}),
(11, 28, {'score': 0}),
(11, 29, {'score': 0}),
(11, 30, {'score': 0}),
(11, 31, {'score': 0.36067376022224085}),
(11, 32, {'score': 0}),
(11, 33, {'score': 0}),
(12, 13, {'score': 0.9187843867734881}),
(12, 14, {'score': 0}),
(12, 15, {'score': 0}),
(12, 16, {'score': 0}),
(12, 17, {'score': 0.36067376022224085}),
(12, 18, {'score': 0}),
(12, 19, {'score': 0.36067376022224085}),
(12, 20, {'score': 0}),
(12, 21, {'score': 0.36067376022224085}),
(12, 22, {'score': 0}),
(12, 23, {'score': 0}),
(12, 24, {'score': 0}),
(12, 25, {'score': 0}),
(12, 26, {'score': 0}),
(12, 27, {'score': 0}),
```

```
(12, 28, {'score': 0}),
(12, 29, {'score': 0}),
(12, 30, {'score': 0}),
(12, 31, {'score': 0.36067376022224085}),
(12, 32, {'score': 0}),
(12, 33, {'score': 0}),
(13, 14, {'score': 0.35295612386476116}),
(13, 15, {'score': 0.35295612386476116}),
(13, 16, {'score': 0}),
(13, 17, {'score': 0.8157933735356595}),
(13, 18, {'score': 0.35295612386476116}),
(13, 19, {'score': 1.1687494974004207}),
(13, 20, {'score': 0.35295612386476116}),
(13, 21, {'score': 0.8157933735356595}),
(13, 22, {'score': 0.35295612386476116}),
(13, 23, {'score': 0.35295612386476116}),
(13, 24, {'score': 0}),
(13, 25, {'score': 0}),
(13, 26, {'score': 0.35295612386476116}),
(13, 27, {'score': 0.7872506057680129}),
(13, 28, {'score': 0.7872506057680129}),
(13, 29, {'score': 0.35295612386476116}),
(13, 30, {'score': 0.8080757371781798}),
(13, 31, {'score': 0.713629884087002}),
(13, 32, {'score': 0.7872506057680129}),
(14, 15, {'score': 0.7553857282466059}),
(14, 16, {'score': 0}),
(14, 17, {'score': 0}),
(14, 18, {'score': 0.7553857282466059}),
(14, 19, {'score': 0.35295612386476116}),
(14, 20, {'score': 0.7553857282466059}),
(14, 21, {'score': 0}),
(14, 22, {'score': 0.7553857282466059}),
(14, 23, {'score': 0.7553857282466059}),
(14, 24, {'score': 0}),
(14, 25, {'score': 0}),
(14, 26, {'score': 0.35295612386476116}),
(14, 27, {'score': 0.35295612386476116}),
(14, 28, {'score': 0.35295612386476116}),
(14, 29, {'score': 0.7553857282466059}),
(14, 30, {'score': 0.7553857282466059}),
(14, 31, {'score': 0.7553857282466059}),
(15, 16, {'score': 0}),
(15, 17, {'score': 0}),
(15, 18, {'score': 0.7553857282466059}),
(15, 19, {'score': 0.35295612386476116}),
(15, 20, {'score': 0.7553857282466059}),
```

```
(15, 21, {'score': 0}),
(15, 22, {'score': 0.7553857282466059}),
(15, 23, {'score': 0.7553857282466059}),
(15, 24, {'score': 0}),
(15, 25, {'score': 0}),
(15, 26, {'score': 0.35295612386476116}),
(15, 27, {'score': 0.35295612386476116}),
(15, 28, {'score': 0.35295612386476116}),
(15, 29, {'score': 0.7553857282466059}),
(15, 30, {'score': 0.7553857282466059}),
(15, 31, {'score': 0.7553857282466059}),
(16, 17, {'score': 0}),
(16, 18, {'score': 0}),
(16, 19, {'score': 0}),
(16, 20, {'score': 0}),
(16, 21, {'score': 0}),
(16, 22, {'score': 0}),
(16, 23, {'score': 0}),
(16, 24, {'score': 0}),
(16, 25, {'score': 0}),
(16, 26, {'score': 0}),
(16, 27, {'score': 0}),
(16, 28, {'score': 0}),
(16, 29, {'score': 0}),
(16, 30, {'score': 0}),
(16, 31, {'score': 0}),
(16, 32, {'score': 0}),
(16, 33, {'score': 0}),
(17, 18, {'score': 0}),
(17, 19, {'score': 0.8157933735356595}),
(17, 20, {'score': 0}),
(17, 21, {'score': 0.8157933735356595}),
(17, 22, {'score': 0}),
(17, 23, {'score': 0}),
(17, 24, {'score': 0}),
(17, 25, {'score': 0}),
(17, 26, {'score': 0}),
(17, 27, {'score': 0}),
(17, 28, {'score': 0}),
(17, 29, {'score': 0}),
(17, 30, {'score': 0.45511961331341866}),
(17, 31, {'score': 0.36067376022224085}),
(17, 32, {'score': 0}),
(17, 33, {'score': 0}),
(18, 19, {'score': 0.35295612386476116}),
(18, 20, {'score': 0.7553857282466059}),
(18, 21, {'score': 0}),
```

```
(18, 22, {'score': 0.7553857282466059}),
(18, 23, {'score': 0.7553857282466059}),
(18, 24, {'score': 0}),
(18, 25, {'score': 0}),
(18, 26, {'score': 0.35295612386476116}),
(18, 27, {'score': 0.35295612386476116}),
(18, 28, {'score': 0.35295612386476116}),
(18, 29, {'score': 0.7553857282466059}),
(18, 30, {'score': 0.7553857282466059}),
(18, 31, {'score': 0.7553857282466059}),
(19, 32, {'score': 0.35295612386476116}),
(19, 20, {'score': 0.35295612386476116}),
(19, 21, {'score': 0.8157933735356595}),
(19, 22, {'score': 0.35295612386476116}),
(19, 23, {'score': 0.35295612386476116}),
(19, 24, {'score': 0}),
(19, 25, {'score': 0}),
(19, 26, {'score': 0.35295612386476116}),
(19, 27, {'score': 0.35295612386476116}),
(19, 28, {'score': 0.35295612386476116}),
(19, 29, {'score': 0.35295612386476116}),
(19, 30, {'score': 0.8080757371781798}),
(19, 31, {'score': 0.713629884087002}),
(20, 21, {'score': 0}),
(20, 22, {'score': 0.7553857282466059}),
(20, 23, {'score': 0.7553857282466059}),
(20, 24, {'score': 0}),
(20, 25, {'score': 0}),
(20, 26, {'score': 0.35295612386476116}),
(20, 27, {'score': 0.35295612386476116}),
(20, 28, {'score': 0.35295612386476116}),
(20, 29, {'score': 0.7553857282466059}),
(20, 30, {'score': 0.7553857282466059}),
(20, 31, {'score': 0.7553857282466059}),
(21, 32, {'score': 0}),
(21, 33, {'score': 0}),
(21, 22, {'score': 0}),
(21, 23, {'score': 0}),
(21, 24, {'score': 0}),
(21, 25, {'score': 0}),
(21, 26, {'score': 0}),
(21, 27, {'score': 0}),
(21, 28, {'score': 0}),
(21, 29, {'score': 0}),
(21, 30, {'score': 0.45511961331341866}),
(21, 31, {'score': 0.36067376022224085}),
(22, 23, {'score': 0.7553857282466059}),
```

```
(22, 24, {'score': 0}),
(22, 25, {'score': 0}),
(22, 26, {'score': 0.35295612386476116}),
(22, 27, {'score': 0.35295612386476116}),
(22, 28, {'score': 0.35295612386476116}),
(22, 29, {'score': 0.7553857282466059}),
(22, 30, {'score': 0.7553857282466059}),
(22, 31, {'score': 0.7553857282466059}),
(23, 24, {'score': 1.631586747071319}),
(23, 26, {'score': 1.0743036443092429}),
(23, 28, {'score': 0.35295612386476116}),
(23, 30, {'score': 0.7553857282466059}),
(23, 31, {'score': 1.6656249548734432}),
(24, 32, {'score': 0.5581106265512472}),
(24, 33, {'score': 1.279458146995729}),
(24, 26, {'score': 0}),
(24, 28, {'score': 0.5581106265512472}),
(24, 29, {'score': 0}),
(24, 30, {'score': 0}),
(25, 32, {'score': 1.179445561110859}),
(25, 33, {'score': 1.179445561110859}),
(25, 26, {'score': 0}),
(25, 27, {'score': 1.531574161186449}),
(25, 28, {'score': 0.5581106265512472}),
(25, 29, {'score': 0.6213349345596119}),
(25, 30, {'score': 0}),
(26, 32, {'score': 1.0743036443092429}),
(26, 27, {'score': 0.35295612386476116}),
(26, 28, {'score': 0.35295612386476116}),
(26, 30, {'score': 0.35295612386476116}),
(26, 31, {'score': 0.35295612386476116}),
(27, 32, {'score': 1.4085855403276248}),
(27, 28, {'score': 0.7872506057680129}),
(27, 29, {'score': 0.974291058424373}),
(27, 30, {'score': 0.35295612386476116}),
(27, 31, {'score': 1.2631953504915985}),
(28, 32, {'score': 1.34536123231926}),
(28, 29, {'score': 0.35295612386476116}),
(28, 30, {'score': 0.35295612386476116}),
(29, 30, {'score': 0.7553857282466059}),
(29, 31, {'score': 0.7553857282466059}),
(30, 31, {'score': 0.7553857282466059})]
```

iii) Common neighbour measure

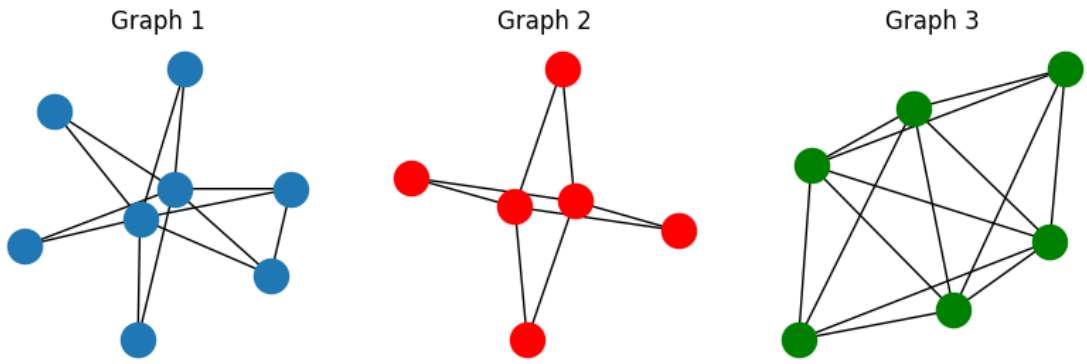
[18]:
```
def common_neighbor_measure(G, u, v):
    neighbors_u = set(G.neighbors(u))
```

```
    neighbors_v = set(G.neighbors(v))
    common_neighbors = neighbors_u.intersection(neighbors_v)
    return len(common_neighbors)
common_neighbor_measure(G, 6, 10)
```

[18]: 3

iv) Subgraph containing '2 nodes' common neighbors

[27]:
```
common_neighbors1 = list(nx.common_neighbors(G, 2, 33))
subgraph1 = G.subgraph(common_neighbors1 + [2, 33])
common_neighbors2 = list(nx.common_neighbors(G, 0, 33))
subgraph2 = G.subgraph(common_neighbors2 + [0, 33])
common_neighbors3 = list(nx.common_neighbors(G, 7, 13))
subgraph3 = G.subgraph(common_neighbors3 + [7, 13])

fig, axs = plt.subplots(1, 3, figsize = (10, 3))
nx.draw(subgraph1, ax=axs[0])
nx.draw(subgraph2, ax=axs[1], node_color = 'red')
nx.draw(subgraph3, ax=axs[2], node_color = 'green')

axs[0].set_title('Graph 1')
axs[1].set_title('Graph 2')
axs[2].set_title('Graph 3')

plt.show()
```



v) Jaccard Coefficient

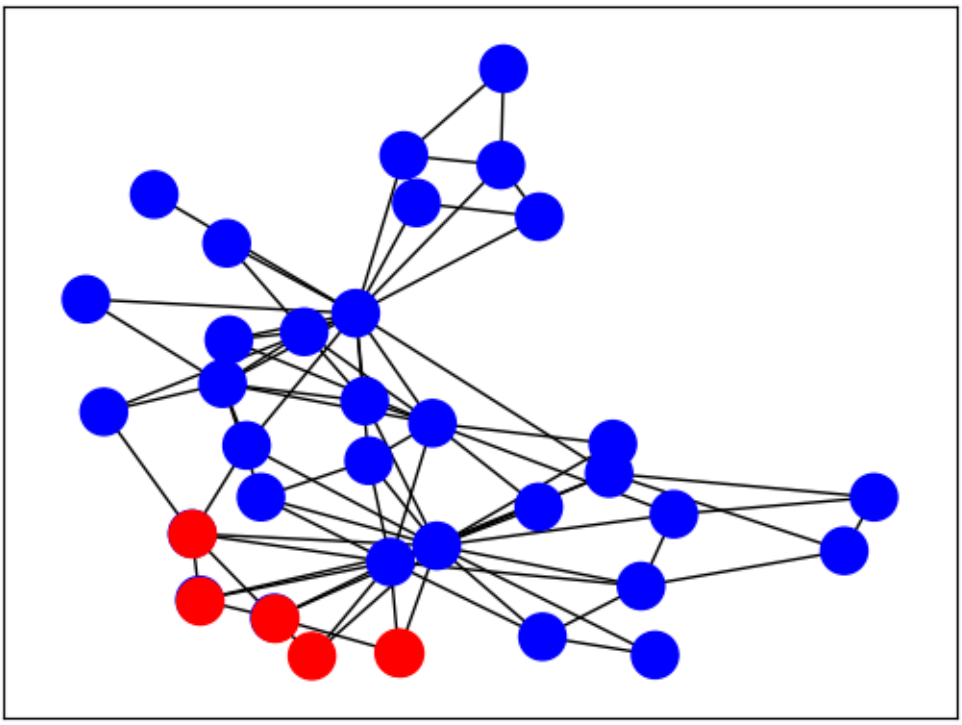
[43]:
```
G.add_edges_from([(14, 15), (14, 18), (14, 20), (14, 22), (15, 18)])
jaccard = list(nx.jaccard_coefficient(G))
df_jaccard = pd.DataFrame(jaccard, columns=['source', 'target',␣
 ↪'jaccard_coeff'])
df_jaccard = df_jaccard.sort_values('jaccard_coeff', ascending = False)
```

```
print(df_jaccard)
```

```
     source  target  jaccard_coeff
411      20      22           1.00
174       7      13           0.80
146       6      10           0.75
91        4       5           0.75
399      19      21           0.75
..      ...     ...            ...
292      12      20           0.00
116       4      32           0.00
294      12      22           0.00
295      12      23           0.00
331      14      25           0.00

[476 rows x 3 columns]
```

[46]:
```
def plot_jaccard(G, df_jaccard):
    df_jaccard = df_jaccard[df_jaccard['source'].isin(G.nodes()) &␣
 ↪df_jaccard['target'].isin(G.nodes())]

    node_color = ['b' for i in range(len(G))]

    max_coeff = df_jaccard['jaccard_coeff'].max()
    for i in range(len(df_jaccard)):
        source = df_jaccard['source'][i]
        target = df_jaccard['target'][i]
        coeff = df_jaccard['jaccard_coeff'][i]
        if coeff == max_coeff:
            node_color[source] = 'r'
            node_color[target] = 'r'

    pos = nx.spring_layout(G, seed = 42)
    nx.draw_networkx_nodes(G, pos, node_color=node_color, cmap = plt.cm.Blues)
    nx.draw_networkx_edges(G, pos)
    nx.draw_networkx_nodes(G, pos, nodelist = [14, 15, 18, 22], node_color =␣
 ↪'r')
    plt.show()

plot_jaccard(G, df_jaccard)
```

```
/usr/local/lib/python3.10/dist-packages/networkx/drawing/nx_pylab.py:433:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  node_collection = ax.scatter(
```



vi) Resource allocation index

[58]:
```
def resource_allocation_index(G, source, target):
    common_neighbors = set(G.neighbors(source)) & set(G.neighbors(target))

    ra_index = []
    for neighbor in common_neighbors:
        denominator = len(list(G.neighbors(neighbor)))
        if denominator == 0:
            ra_index.append(0)
        else:
            ra_index.append(sum(1 / len(list(G.neighbors(u))) for u in G.
 ↪neighbors(neighbor)))

    df_ra_index = pd.DataFrame({'source': [source]*len(common_neighbors),
                                'target': [target]*len(common_neighbors),
                                'resource_allocation_index': ra_index})

    df_ra_index.sort_values(by='resource_allocation_index', ascending=False,␣
 ↪inplace=True)
    df_ra_index.reset_index(drop=True, inplace=True)

    return df_ra_index
```

```
print(resource_allocation_index(G, 2, 33))
print(resource_allocation_index(G, 0, 33))
print(resource_allocation_index(G, 1, 33))
print(resource_allocation_index(G, 4, 5))
print(resource_allocation_index(G, 6, 10))
```

```
   source  target  resource_allocation_index
0       2      33                   2.475490
1       2      33                   0.692157
2       2      33                   0.554657
3       2      33                   0.499101
4       2      33                   0.325490
5       2      33                   0.158824
   source  target  resource_allocation_index
0       0      33                   1.204657
1       0      33                   0.554657
2       0      33                   0.499101
3       0      33                   0.399101
   source  target  resource_allocation_index
0       1      33                   0.499101
1       1      33                   0.453268
2       1      33                   0.399101
   source  target  resource_allocation_index
0       4       5                   4.944444
1       4       5                   1.145833
2       4       5                   0.645833
   source  target  resource_allocation_index
0       6      10                   4.944444
1       6      10                   1.145833
2       6      10                   0.645833
```

vii) Adamic-Adar index

[48]:
```
def adamic_adar_index(G, pairs):
    aa_dict = nx.adamic_adar_index(G, pairs)
    df_aa = pd.DataFrame(aa_dict, columns=['source', 'target',␣
 ↪'adamic_adar_index'])
    df_aa = df_aa.sort_values(by='adamic_adar_index', ascending=False)
    return df_aa

pairs = [(2, 33), (0, 33), (1, 33), (4, 5), (6, 10)]
df_aa = adamic_adar_index(G, pairs)
print(df_aa)
```

```
   source  target  adamic_adar_index
0       2      33           4.719381
1       0      33           2.522128
```

```
2       1      33           2.064030
3       4       5           1.992261
4       6      10           1.992261
```

viii) Preferential attachment values

[53]:
```
def preferential_attachment(G, source, target):
    preds = nx.preferential_attachment(G, [(source, target)])
    pa_score = list(preds)[0][2]
    df_pa = pd.DataFrame({'source': [source], 'target': [target],␣
 ↪'preferential_attachment': [pa_score]})
    df_pa = df_pa.sort_values('preferential_attachment', ascending=False)
    return df_pa

df_pa_1 = preferential_attachment(G, 0, 33)
df_pa_2 = preferential_attachment(G, 0, 32)
df_pa_3 = preferential_attachment(G, 2, 33)
df_pa_4 = preferential_attachment(G, 1, 33)
df_pa_5 = preferential_attachment(G, 1, 32)

print(df_pa_1)
print(df_pa_2)
print(df_pa_3)
print(df_pa_4)
print(df_pa_5)
```

```
   source  target  preferential_attachment
0       0      33                      272
   source  target  preferential_attachment
0       0      32                      192
   source  target  preferential_attachment
0       2      33                      170
   source  target  preferential_attachment
0       1      33                      153
   source  target  preferential_attachment
0       1      32                      108
```