lab-7-22mcb1002

April 23, 2023

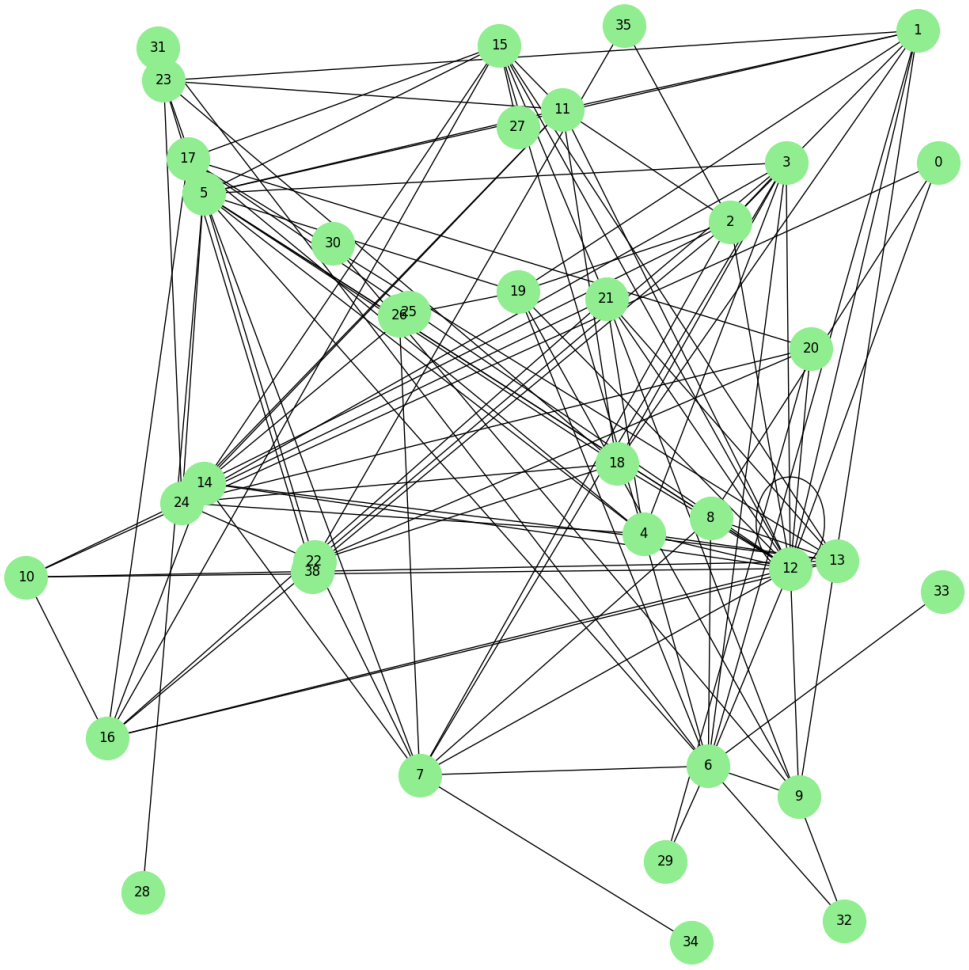
1 Visualization of friendship dataset

[1]:
```
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt
from networkx.algorithms.community import girvan_newman
```

[7]:
```
G = nx.Graph()
df = pd.read_csv("edgelist.txt", header = None).apply(lambda edgeData:␣
↪(edgeData[0], edgeData[1]), axis=1)
G.add_edges_from(df)
```

i. Network Visualization

[10]:
```
plt.figure(figsize = (14,14))
nx.draw(G,nx.random_layout(G), with_labels = True, node_size=1500, node_color =␣
↪'#90ee90')
plt.show()
```



ii. Centrality measures

[11]:
```
nx.closeness_centrality(G)
```

[11]:
```
{0: 0.4235294117647059,
 8: 0.5070422535211268,
 10: 0.43902439024390244,
 1: 0.5454545454545454,
 2: 0.5070422535211268,
 5: 0.6101694915254238,
 6: 0.5714285714285714,
 9: 0.48,
 11: 0.5142857142857142,
```

```
 18: 0.4931506849315068,
 19: 0.5217391304347826,
 23: 0.43902439024390244,
 7: 0.5538461538461539,
 3: 0.5373134328358209,
 4: 0.5217391304347826,
 12: 0.6923076923076923,
 14: 0.5070422535211268,
 16: 0.47368421052631576,
 22: 0.45569620253164556,
 24: 0.4675324675324675,
 13: 0.48,
 15: 0.5142857142857142,
 17: 0.46153846153846156,
 21: 0.46153846153846156,
 20: 0.4675324675324675,
 26: 0.37894736842105264,
 29: 0.391304347826087,
 32: 0.3956043956043956,
 35: 0.3673469387755102,
 38: 0.3076923076923077,
 25: 0.33962264150943394,
 27: 0.34285714285714286,
 28: 0.3829787234042553,
 30: 0.32727272727272727,
 31: 0.3673469387755102,
 33: 0.3673469387755102,
 34: 0.36}
```

[12]:
```
nx.betweenness_centrality(G)
```

[12]:
```
{0: 0.000529100529100529,
 8: 0.013569974998546426,
 10: 0.0025132275132275133,
 1: 0.03941317941317941,
 2: 0.048016674683341344,
 5: 0.14578312780693728,
 6: 0.19515529444100874,
 9: 0.05555555555555555,
 11: 0.03349773242630386,
 18: 0.01187188252664443,
 19: 0.02978784443070158,
 23: 0.05617913832199546,
 7: 0.10986720510530033,
 3: 0.020551926028116504,
 4: 0.02967248580343818,
 12: 0.2755905734477164,
```

```
 14: 0.066578128661462,
 16: 0.01241118669690098,
 22: 0.026445578231292514,
 24: 0.01675979223598271,
 13: 0.012667233560090701,
 15: 0.07248389837675552,
 17: 0.005967458467458468,
 21: 0.018154761904761906,
 20: 0.02175866461580747,
 26: 0.0006349206349206349,
 29: 0.00291005291005291,
 32: 0.001984126984126984,
 35: 0.0006734006734006733,
 38: 0.0,
 25: 0.0,
 27: 0.0,
 28: 0.0,
 30: 0.0,
 31: 0.0,
 33: 0.0,
 34: 0.0}
```

[13]:
```
nx.edge_betweenness_centrality(G)
```

[13]:
```
{(0, 8): 0.010378235378235378,
 (0, 10): 0.004379379379379379,
 (0, 12): 0.040297440297440285,
 (8, 7): 0.011841511841511844,
 (8, 5): 0.016585541585541583,
 (8, 6): 0.01569069069069069,
 (8, 13): 0.011592361592361593,
 (8, 12): 0.01363863863863864,
 (10, 12): 0.037412412412412416,
 (10, 13): 0.005442942942942942,
 (10, 16): 0.006006006006006005,
 (10, 21): 0.0055680680680680675,
 (1, 2): 0.01319831736498403,
 (1, 5): 0.00967168467168467,
 (1, 6): 0.02238845988845989,
 (1, 9): 0.01691951691951692,
 (1, 11): 0.008260045760045761,
 (1, 18): 0.007648232648232647,
 (1, 19): 0.006783569283569284,
 (1, 23): 0.024989274989274987,
 (1, 12): 0.01876042709376042,
 (2, 11): 0.01068777110443777,
 (2, 7): 0.013689338689338686,
```

```
(2, 3): 0.010467366717366717,
(2, 12): 0.02186293436293436,
(2, 14): 0.011861861861861861,
(2, 16): 0.015468011301344632,
(2, 19): 0.012927212927212923,
(2, 35): 0.03473359723359723,
(5, 3): 0.006856856856856856,
(5, 6): 0.03594974844974845,
(5, 4): 0.012656612656612658,
(5, 11): 0.010535535535535535,
(5, 15): 0.03353736895403561,
(5, 18): 0.008451308451308452,
(5, 19): 0.018446530946530947,
(5, 22): 0.022964210464210457,
(5, 23): 0.039330997664331,
(5, 24): 0.013924638924638922,
(5, 7): 0.024715787215787214,
(5, 12): 0.022179095095761766,
(5, 28): 0.05405405405405406,
(6, 4): 0.022654797654797654,
(6, 3): 0.01885944385944386,
(6, 9): 0.02548652548652548,
(6, 19): 0.01553160303160303,
(6, 7): 0.03516146016146016,
(6, 29): 0.04062276562276562,
(6, 31): 0.05405405405405406,
(6, 32): 0.03908908908908909,
(6, 33): 0.05405405405405406,
(6, 12): 0.043724081224081224,
(9, 19): 0.009997009997009997,
(9, 30): 0.05405405405405406,
(9, 12): 0.0527020527020527,
(11, 15): 0.014850266933600264,
(11, 18): 0.005180180180180179,
(11, 23): 0.01986986986986987,
(11, 24): 0.010315672815672815,
(11, 14): 0.01855814147480814,
(11, 12): 0.019170658753992084,
(18, 3): 0.00646003146003146,
(18, 22): 0.009316042649375982,
(18, 23): 0.009050717384050716,
(18, 24): 0.004142535392535392,
(18, 12): 0.02626532418199085,
(19, 26): 0.023709423709423708,
(19, 12): 0.02301408551408551,
(23, 24): 0.013043996377329713,
(23, 38): 0.05405405405405406,
```

```
(7, 3): 0.009187759187759187,
(7, 14): 0.027992240492240493,
(7, 22): 0.020693416526749855,
(7, 26): 0.03154583154583154,
(7, 34): 0.05405405405405406,
(7, 12): 0.033029528862862194,
(3, 4): 0.008229566562899896,
(3, 22): 0.008968038134704802,
(3, 24): 0.008576433576433578,
(3, 12): 0.015330579913913248,
(4, 13): 0.009712115962115962,
(4, 15): 0.010379129129129128,
(4, 17): 0.008547535630868965,
(4, 21): 0.010606141856141857,
(4, 24): 0.015279771529771526,
(4, 12): 0.012125518375518375,
(12, 13): 0.01889581889581889,
(12, 14): 0.033533533533533534,
(12, 15): 0.028528528528528524,
(12, 16): 0.022375026541693203,
(12, 21): 0.0294663711330378,
(12, 12): 0.0,
(12, 17): 0.025086991753658416,
(12, 20): 0.03804257762591096,
(14, 13): 0.011301686301686301,
(14, 16): 0.008246693663360331,
(14, 15): 0.014464464464464467,
(14, 25): 0.05405405405405406,
(16, 13): 0.0036473973973973972,
(16, 15): 0.008367647950981284,
(16, 21): 0.0070293507793507785,
(16, 17): 0.006394543894543894,
(22, 20): 0.01398544731878065,
(22, 24): 0.007564615897949231,
(22, 35): 0.020594458094458097,
(24, 20): 0.012914104580771248,
(13, 15): 0.006722106722106722,
(13, 17): 0.005184855184855185,
(13, 21): 0.00551980551980552,
(15, 17): 0.008789742123075456,
(15, 21): 0.011492444825778159,
(15, 27): 0.05405405405405406,
(17, 20): 0.011340171756838423,
(21, 32): 0.018718718718718715,
(20, 29): 0.01893679393679393}
```

iii. CLIQUES Analysis

[14]:
```
cliques = list(nx.find_cliques(G))

one_cliques = [c for c in cliques if len(c) == 1]
print("1-clique(s):", one_cliques)
```

```
1-clique(s): []
```

[15]:
```
two_cliques = [c for c in cliques if len(c) == 2]
print("2-clique(s):", two_cliques)
```

```
2-clique(s): [[32, 21], [32, 6], [33, 6], [34, 7], [35, 2], [35, 22], [38, 23],
[25, 14], [26, 19], [26, 7], [27, 15], [28, 5], [29, 20], [29, 6], [30, 9], [31,
6]]
```

[16]:
```
three_cliques = [c for c in cliques if len(c) == 3]
print("3-clique(s):", three_cliques)
```

```
3-clique(s): [[12, 0, 8], [12, 0, 10], [12, 13, 8], [12, 17, 20], [22, 20, 24]]
```

[18]:
```
# Maximal Cliques
maximal_cliques = [c for c in cliques if len(c) == len(max(cliques, key=len))]
print("Maximal clique(s):", maximal_cliques)
```

```
Maximal clique(s): [[12, 5, 18, 11, 1], [12, 5, 6, 8, 7], [12, 5, 6, 1, 19],
[12, 5, 6, 3, 4], [12, 5, 6, 3, 7], [12, 9, 1, 19, 6], [12, 10, 16, 13, 21],
[12, 13, 15, 16, 17], [12, 13, 15, 16, 21], [12, 13, 15, 16, 14], [12, 13, 15,
4, 17], [12, 13, 15, 4, 21], [22, 3, 5, 24, 18], [23, 5, 18, 11, 24], [23, 5,
18, 11, 1]]
```

[19]:
```
# Cohesive subgroups
cohesive_subgroups = []
for c in cliques:
    subgraph = G.subgraph(c)
    subgraph_density = nx.density(subgraph)
    if subgraph_density >= 0.8:
        cohesive_subgroups.append(subgraph)
print("Cohesive subgroups formed from cliques:")
for sg in cohesive_subgroups:
    print(sg.nodes())
```

```
Cohesive subgroups formed from cliques:
[32, 21]
[32, 6]
[33, 6]
[34, 7]
[2, 35]
[35, 22]
[38, 23]
```

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

[20]:
```
# overlapping cliques
communities = girvan_newman(G)

overlapping_subgroups = []
for c in communities:
```

```
    if len(c) > 1:
        overlapping_subgroups.append(c)

print(overlapping_subgroups)
```

```
[({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, 29, 30, 31, 32, 33, 34, 35, 38}, {28}), ({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, 29, 30, 32, 33, 34, 35, 38}, {28}, {31}), ({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, 29,
30, 32, 34, 35, 38}, {28}, {31}, {33}), ({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, 29, 32, 34, 35,
38}, {28}, {30}, {31}, {33}), ({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, 29, 32, 34, 35}, {38},
{28}, {30}, {31}, {33}), ({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, 29, 32, 35}, {38}, {28}, {30},
{31}, {33}, {34}), ({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, 26, 27, 29, 32, 35}, {38}, {25}, {28}, {30},
{31}, {33}, {34}), ({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, 26, 29, 32, 35}, {38}, {25}, {27}, {28}, {30},
{31}, {33}, {34}), ({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, 26, 29, 32, 35}, {38}, {25}, {27}, {28}, {30},
{31}, {33}, {34}), ({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, 26, 29, 35}, {32}, {38}, {25}, {27}, {28}, {30},
{31}, {33}, {34}), ({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, 26, 35}, {29}, {32}, {38}, {25}, {27}, {28},
{30}, {31}, {33}, {34}), ({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, 26}, {29}, {32}, {35}, {38}, {25}, {27},
{28}, {30}, {31}, {33}, {34}), ({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}, {26}, {29}, {32}, {35}, {38}, {25},
{27}, {28}, {30}, {31}, {33}, {34}), ({0}, {1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12,
14, 18, 19, 20, 22, 23, 24}, {10, 13, 15, 16, 17, 21}, {26}, {29}, {32}, {35},
{38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {1, 2, 3, 4, 5, 6, 7, 8,
9, 11, 12, 14, 18, 19, 22, 23, 24}, {10, 13, 15, 16, 17, 21}, {20}, {26}, {29},
{32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {1, 2, 3, 4,
5, 6, 7, 8, 11, 12, 14, 18, 19, 22, 23, 24}, {10, 13, 15, 16, 17, 21}, {9},
{20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}),
({0}, {8}, {10, 13, 15, 16, 17, 21}, {1, 2, 3, 4, 5, 6, 7, 11, 12, 14, 18, 19,
22, 23, 24}, {9}, {20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30},
{31}, {33}, {34}), ({0}, {8}, {10, 13, 15, 16, 17, 21}, {1, 2, 3, 4, 5, 6, 7,
11, 12, 14, 18, 22, 23, 24}, {9}, {19}, {20}, {26}, {29}, {32}, {35}, {38},
{25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10, 13, 15, 16, 17, 21},
{1, 2, 3, 4, 5, 6, 7, 11, 12, 18, 22, 23, 24}, {9}, {19}, {14}, {20}, {26},
{29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8},
{10, 13, 15, 16, 17, 21}, {1, 2, 3, 4, 5, 6, 7, 11, 12, 18, 22, 24}, {9}, {19},
{23}, {14}, {20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31},
{33}, {34}), ({0}, {8}, {10, 13, 15, 16, 17, 21}, {1, 3, 4, 5, 6, 7, 11, 12, 18,
```

```
22, 24}, {2}, {9}, {19}, {23}, {14}, {20}, {26}, {29}, {32}, {35}, {38}, {25},
{27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10, 13, 15, 16, 17, 21}, {1, 3,
4, 5, 6, 7, 11, 12, 18, 24}, {2}, {9}, {19}, {23}, {14}, {22}, {20}, {26}, {29},
{32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10, 13,
15, 16, 17, 21}, {1, 3, 4, 5, 6, 11, 12, 18, 24}, {2}, {9}, {19}, {23}, {7},
{14}, {22}, {20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31},
{33}, {34}), ({0}, {8}, {10, 13, 15, 16, 17, 21}, {1}, {2}, {3, 4, 5, 6, 11, 12,
18, 24}, {9}, {19}, {23}, {7}, {14}, {22}, {20}, {26}, {29}, {32}, {35}, {38},
{25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10, 13, 15, 16, 17, 21},
{1}, {2}, {3, 4, 5, 6, 12, 18, 24}, {9}, {11}, {19}, {23}, {7}, {14}, {22},
{20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}),
({0}, {8}, {10}, {1}, {2}, {3, 4, 5, 6, 12, 18, 24}, {9}, {11}, {19}, {23}, {7},
{14}, {13, 15, 16, 17, 21}, {22}, {20}, {26}, {29}, {32}, {35}, {38}, {25},
{27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10}, {1}, {2}, {3, 4, 5, 6, 12,
24}, {9}, {11}, {18}, {19}, {23}, {7}, {14}, {13, 15, 16, 17, 21}, {22}, {20},
{26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0},
{8}, {10}, {1}, {2}, {3, 4, 5, 6, 12}, {9}, {11}, {18}, {19}, {23}, {7}, {14},
{13, 15, 16, 17, 21}, {22}, {24}, {20}, {26}, {29}, {32}, {35}, {38}, {25},
{27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10}, {1}, {2}, {3, 4, 5, 6,
12}, {9}, {11}, {18}, {19}, {23}, {7}, {14}, {16, 17, 13, 15}, {22}, {24}, {21},
{20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}),
({0}, {8}, {10}, {1}, {2}, {5}, {3, 4, 12, 6}, {9}, {11}, {18}, {19}, {23}, {7},
{14}, {16, 17, 13, 15}, {22}, {24}, {21}, {20}, {26}, {29}, {32}, {35}, {38},
{25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10}, {1}, {2}, {5}, {6},
{9}, {11}, {18}, {19}, {23}, {7}, {3, 4, 12}, {14}, {16, 17, 13, 15}, {22},
{24}, {21}, {20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31},
{33}, {34}), ({0}, {8}, {10}, {1}, {2}, {5}, {6}, {9}, {11}, {18}, {19}, {23},
{7}, {3}, {4, 12}, {14}, {16, 17, 13, 15}, {22}, {24}, {21}, {20}, {26}, {29},
{32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10},
{1}, {2}, {5}, {6}, {9}, {11}, {18}, {19}, {23}, {7}, {3}, {4}, {12}, {14}, {16,
17, 13, 15}, {22}, {24}, {21}, {20}, {26}, {29}, {32}, {35}, {38}, {25}, {27},
{28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10}, {1}, {2}, {5}, {6}, {9}, {11},
{18}, {19}, {23}, {7}, {3}, {4}, {12}, {14}, {16}, {22}, {24}, {17, 13, 15},
{21}, {20}, {26}, {29}, {32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33},
{34}), ({0}, {8}, {10}, {1}, {2}, {5}, {6}, {9}, {11}, {18}, {19}, {23}, {7},
{3}, {4}, {12}, {14}, {16}, {22}, {24}, {13}, {17, 15}, {21}, {20}, {26}, {29},
{32}, {35}, {38}, {25}, {27}, {28}, {30}, {31}, {33}, {34}), ({0}, {8}, {10},
{1}, {2}, {5}, {6}, {9}, {11}, {18}, {19}, {23}, {7}, {3}, {4}, {12}, {14},
{16}, {22}, {24}, {13}, {15}, {17}, {21}, {20}, {26}, {29}, {32}, {35}, {38},
{25}, {27}, {28}, {30}, {31}, {33}, {34})]
```

[22]:
```python
# Density of maximal cliques
mcn = nx.Graph()
for clique in cliques:
    if len(clique) > 1:
        mcn.add_edges_from([(u, v) for u in clique for v in clique if u < v])
```

```
density = nx.density(mcn)

print("Density of maximal clique network:", density)
```

```
Density of maximal clique network: 0.17117117117117117
```

[25]:
```
max_clique = max(cliques, key=len)
print("Maximum clique:", max_clique)
print("Maximal clique(s):", maximal_cliques)
```

```
Maximum clique: [12, 5, 18, 11, 1]
Maximal clique(s): [[12, 5, 18, 11, 1], [12, 5, 6, 8, 7], [12, 5, 6, 1, 19],
[12, 5, 6, 3, 4], [12, 5, 6, 3, 7], [12, 9, 1, 19, 6], [12, 10, 16, 13, 21],
[12, 13, 15, 16, 17], [12, 13, 15, 16, 21], [12, 13, 15, 16, 14], [12, 13, 15,
4, 17], [12, 13, 15, 4, 21], [22, 3, 5, 24, 18], [23, 5, 18, 11, 24], [23, 5,
18, 11, 1]]
```

iv. Hubs and Authority using HITS algorithm

[27]:
```
hub, authority = nx.hits(G)
print("Hubs:", sorted(hub, key = hub.get, reverse = True))
print("Authorities:", sorted(authority, key = authority.get, reverse = True))
```

```
Hubs: [12, 5, 6, 3, 4, 15, 11, 1, 7, 2, 18, 13, 14, 16, 19, 24, 8, 21, 17, 22,
23, 9, 10, 20, 0, 32, 26, 29, 35, 28, 31, 33, 27, 34, 25, 38, 30]
Authorities: [12, 5, 6, 3, 4, 15, 11, 1, 7, 2, 18, 13, 14, 16, 19, 24, 8, 21,
17, 22, 23, 9, 10, 20, 0, 32, 26, 29, 35, 28, 31, 33, 27, 34, 25, 38, 30]
```

v. Cliques, Clans, Plexes, Cores

[28]:
```
n = 3
cliques = list(nx.find_cliques(G))
n_cliques = [c for c in cliques if len(c) == n]

print("n-cliques:", n_cliques)
```

```
n-cliques: [[12, 0, 8], [12, 0, 10], [12, 13, 8], [12, 17, 20], [22, 20, 24]]
```

[30]:
```
k = 3
clique_communities = list(nx.find_cliques(G))
k_plexes = [c for c in clique_communities if len(c) >= k and
            all(len(set(G.neighbors(n)).intersection(set(c))) >= k-1 for n in␣
 ↪c)]

print("k-plexes:", k_plexes)
```

```
k-plexes: [[12, 0, 8], [12, 0, 10], [12, 2, 19, 1], [12, 2, 1, 11], [12, 2, 3,
7], [12, 2, 14, 16], [12, 2, 14, 11], [12, 2, 14, 7], [12, 5, 18, 3], [12, 5,
18, 11, 1], [12, 5, 11, 15], [12, 5, 6, 8, 7], [12, 5, 6, 1, 19], [12, 5, 6, 3,
```

```
4], [12, 5, 6, 3, 7], [12, 5, 15, 4], [12, 9, 1, 19, 6], [12, 10, 16, 13, 21],
[12, 13, 8], [12, 13, 15, 16, 17], [12, 13, 15, 16, 21], [12, 13, 15, 16, 14],
[12, 13, 15, 4, 17], [12, 13, 15, 4, 21], [12, 14, 15, 11], [12, 17, 20], [22,
3, 5, 24, 18], [22, 3, 5, 7], [22, 20, 24], [23, 5, 18, 11, 24], [23, 5, 18, 11,
1], [24, 5, 3, 4]]
```

[31]:
```
G.remove_edges_from(nx.selfloop_edges(G))
```

[32]:
```
k = 3
k_core = nx.k_core(G, k)
print("Nodes in the 3-core:", k_core.nodes())
```

```
Nodes in the 3-core: [0, 8, 10, 1, 2, 5, 6, 9, 11, 18, 19, 23, 7, 3, 4, 12, 14,
16, 22, 24, 13, 15, 17, 21, 20]
```