HW2_308315092_206120545

May 9, 2020

1 Algorithms and Applications in Social Networks

1.1 Homework 2

1.1.1 Ido Lerer 206120545, Offek Gil 308315092

1.2 Question 1

1.2.1 Question 1a

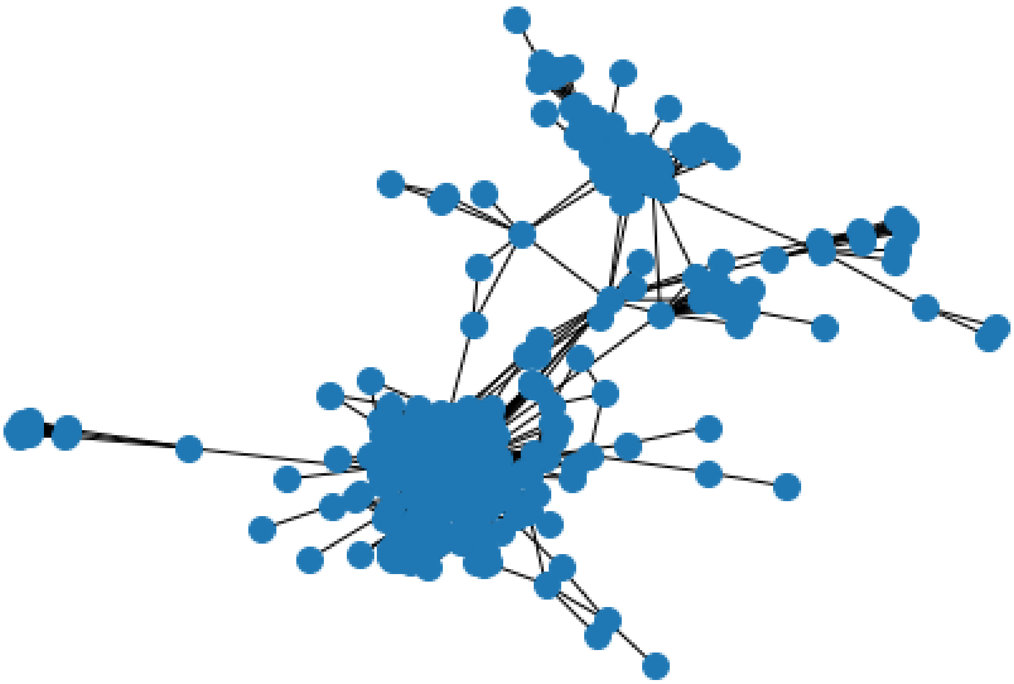
[1]:
```
import networkx as nx
import matplotlib.pyplot as plt
from operator import itemgetter
```

[2]:
```
def newman_girvan(G, k):
    Gc = G.copy()
    while (nx.number_connected_components(Gc) < k):
        edge_betweeness = nx.edge_betweenness_centrality(Gc)
        e = max(edge_betweeness.items(), key=itemgetter(1))[0]
        Gc.remove_edge(e[0], e[1])
    return list(nx.connected_components(Gc))
```

1.2.2 Question 1b

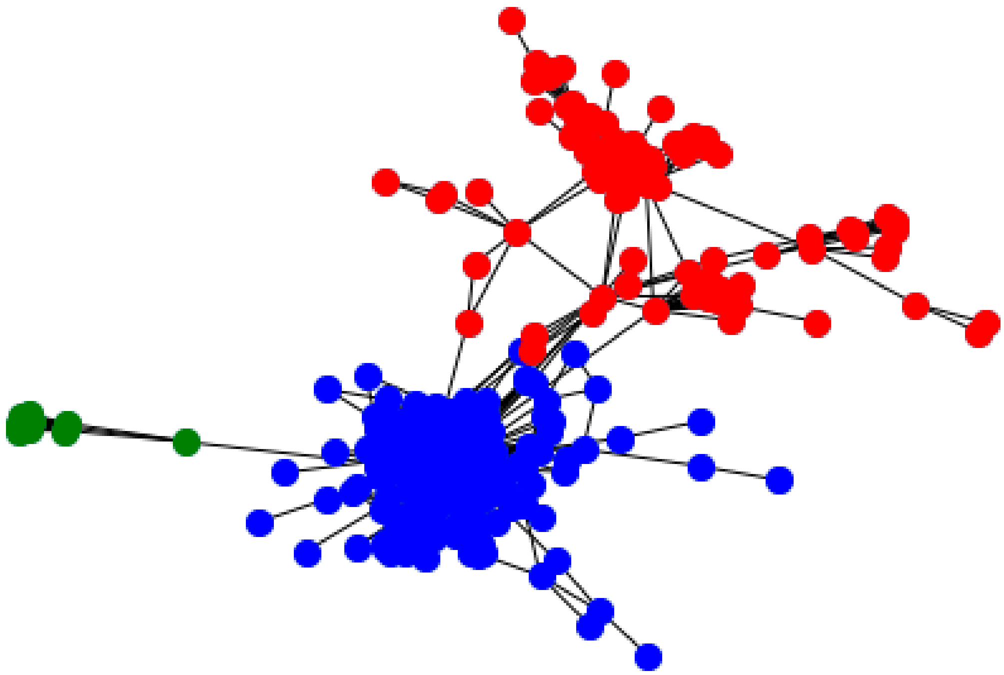
[3]:
```
G = nx.read_edgelist("communities.txt")
```

[4]:
```
largest_connected_component = G.subgraph(max(nx.connected_components(G),␣
 ↪key=len))
pos = nx.spring_layout(largest_connected_component)
nx.draw(largest_connected_component, pos, node_size=100)
plt.show()
```



[5]:
```
Gc = newman_girvan(largest_connected_component, 3)
```

[6]:
```
nx.draw(largest_connected_component, pos, node_size=100)
nx.draw_networkx_nodes(largest_connected_component, pos, nodelist=Gc[0],␣
 ↪node_color='b', node_size=100)
nx.draw_networkx_nodes(largest_connected_component, pos, nodelist=Gc[1],␣
 ↪node_color='r', node_size=100)
nx.draw_networkx_nodes(largest_connected_component, pos, nodelist=Gc[2],␣
 ↪node_color='g', node_size=100)
plt.show()
```



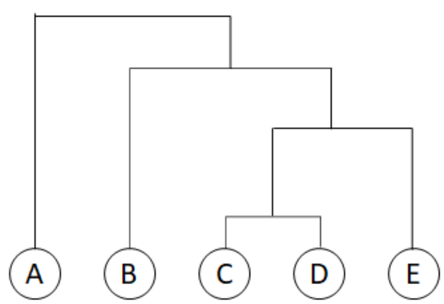
[7]:
```
print(Gc)
```

```
[{'117', '34', '301', '212', '228', '164', '200', '53', '82', '199', '129',
'299', '344', '261', '266', '36', '252', '272', '81', '183', '10', '202', '96',
'281', '339', '136', '274', '69', '79', '198', '64', '55', '332', '133', '325',
'187', '236', '165', '237', '54', '204', '67', '290', '249', '21', '342', '248',
'130', '338', '260', '180', '257', '80', '280', '270', '314', '126', '229',
'206', '336', '288', '197', '158', '76', '123', '73', '128', '120', '59', '239',
'105', '191', '51', '100', '297', '247', '24', '1', '65', '173', '294', '251',
'235', '31', '156', '160', '7', '171', '72', '159', '104', '346', '203', '224',
'127', '284', '57', '347', '322', '75', '83', '211', '26', '118', '330', '47',
'40', '208', '63', '50', '169', '135', '94', '139', '161', '250', '276', '217',
'25', '106', '153', '186', '223', '300', '190', '170', '142', '29', '189',
'176', '62', '30', '291', '22', '318', '185', '309', '146', '213', '150', '221',
'311', '320', '119', '234', '283', '9', '323', '324', '295', '277', '172',
'141', '265', '308', '254', '331', '48', '113', '238', '341', '60', '101',
'246', '194', '316', '92', '121', '242', '5', '345', '87', '269', '271', '16',
'66', '45', '27', '222', '334', '317', '178', '286', '109', '268', '340', '122',
'329', '303', '85', '315', '107', '207', '166', '108', '184', '38', '232',
'103', '56', '188', '148', '285', '313', '231', '77', '298', '302', '125',
'132', '168', '304', '3', '134', '163', '58', '196', '88', '84', '39', '13',
'258', '98'}, {'278', '144', '95', '97', '262', '227', '46', '111', '175',
'240', '20', '337', '326', '333', '310', '23', '167', '68', '182', '216', '321',
```

```
'91', '147', '279', '177', '61', '41', '264', '19', '89', '86', '6', '289', '8',
'219', '201', '140', '110', '71', '99', '49', '293', '28', '267', '205', '32',
'115', '193', '143', '343', '192', '162', '263', '14', '253', '241', '296',
'70', '259', '17', '226', '307', '230', '312', '154', '245', '305', '102',
'112', '243', '319', '174', '255', '137', '138', '151', '44', '124', '35',
'220', '155', '225', '149', '2', '131', '116', '52', '93', '157', '214', '327'},
{'4', '273', '78', '152', '306', '181', '218', '275', '328', '195'}]
```

1.2.3 Question 1c

1. Calculating betweens: BFS Weights: A -> {(A,B):4, (B,D):2, (B,C):1, (D,E):1} B -> {(A,B):1, (B,D):2, (B,C):1, (D,E):1} C -> {(C,B):2, (C,D):1, (C,E):1, (B,A):1} D -> {(D,B):2, (D,C):1, (D,E):1, (B,A):1} E -> {(E,C):3, (E,D):1, (C,B):2, (B,A):1} Edge betweeness: (sum/2) {(A,B):4, (B,C):3, (B,D):3, (C,D):1, (C,E):2, (D,E):2}
2. Removing edge with largest EB: (A,B) We are left with the two communities {A}, {B, C, D, E} and we are done.



Dendogram:

2 Question 2

Proof: Let G be the complete graph with 20 vertices which 18 of its edges were removed, we'll show it is still connected. Assume by contradiction that the graph isn't connected. Given one of the graph's connected components, marked C and having k verticies, there are 20-k>0 verticies not in C (k=20 contradicts the assumption). Since in the original graph all verticies in C had edges to all verticies outside of it, at least k*(20-k) edges had to be removed. We get:

$$k * (20 - k) \leq 180 \leq k^2 - 20k + 18k \geq 19.05 \lor k \leq 0.944$$

Which are both not possible. Contradiction.

3 Question 3

Proof: Let G be the complete graph with n vertices, some of its edges marked as 'phone' and some as 'mail'. We'll show either the graph is connected using only the 'phone' edges or connected using only the 'mail' edges. If the graph is connected using the 'phone' we are done. Assuming it isn't, we'll show it is connected using the 'mail' edges:

Let u,v be verticies in the graph. If (u,v) is a ‘mail’ edge we are done. Else, they are connected in the ‘phone’ graph, and there exists a vertex w that isn’t a neighbour of either of them in the ‘phone’ graph (otherwise the entire graph is in the same connected component, contradicting the assumption that the ‘phone’ graph is not connected). Therefore (u,w),(v,w) are edges in the ‘mail’ graph and u,v are connected though w. We’ve shown the ‘mail’ graph is connected and the proof is complete.

4 Question 4

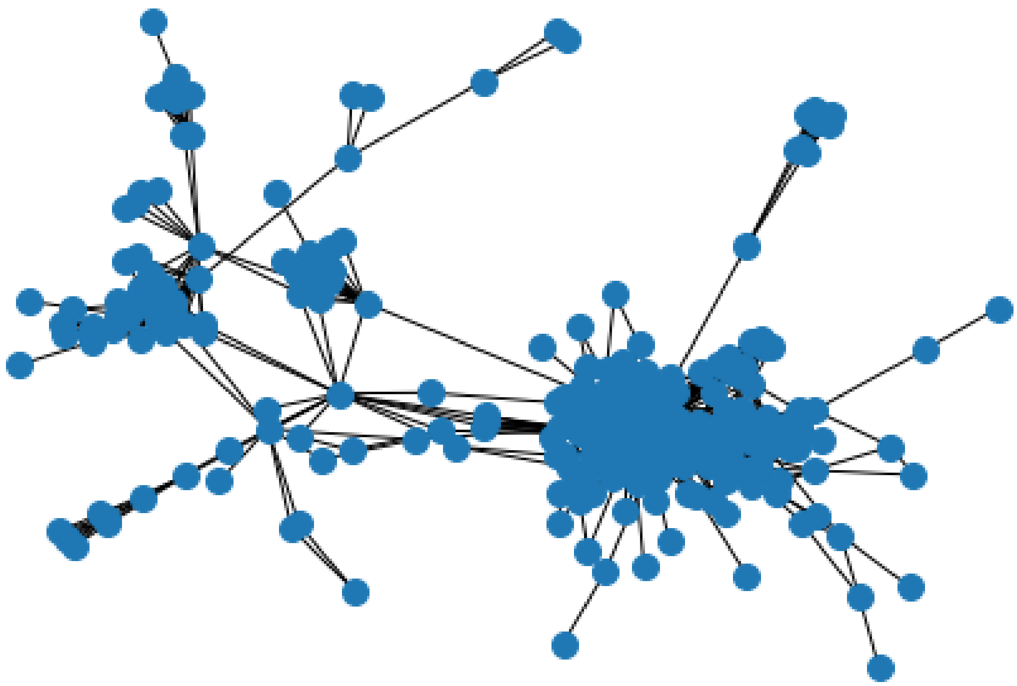
4.0.1 Question 4a

[8]:
```
def find_communities(threshold_matrix, clique_list):
    #gets threshold_matrix and clique_list, goes over the matrix to find all␣
 ↪connected communities
    used_cliques = set()
    length = len(threshold_matrix)
    communities = []

    for i in range(length):
        if i in used_cliques:
            continue
        clique_queue = {i}
        community = set()
        community = community.union(clique_list[i])

        while len(clique_queue) > 0:
            curr_clique = clique_queue.pop()
            if curr_clique not in used_cliques:
                for j in range(length):
                    if threshold_matrix[curr_clique][j] == 1:
                        clique_queue.add(j)
                        community = community.union(clique_list[j])
                used_cliques.add(curr_clique)

        communities.append(community)
    return communities


def k_clique_communities(G, k):
    max_cliques = nx.find_cliques(G)
    threshold_matrix = []
    clique_list = []

    #find al cliques that are at least size k
    for clique in max_cliques:
        if len(clique) >= k:
            clique_list.append(set(clique))
```

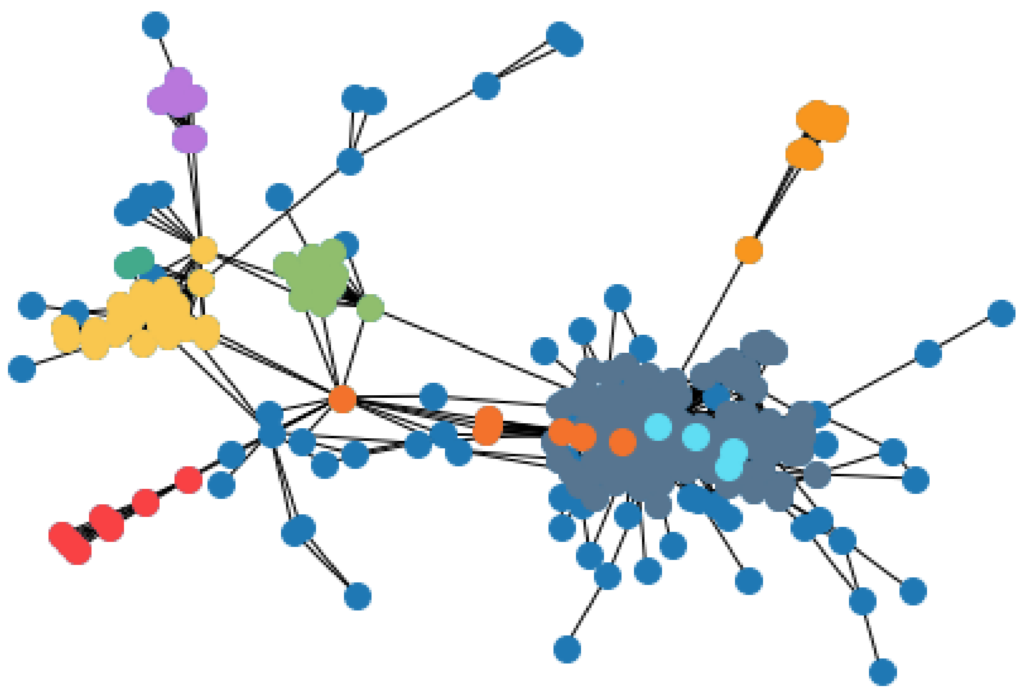
```
    #preset the thershold matrix with all zeros
    for i in range(len(clique_list)):
         threshold_matrix.append([0 for i in range(len(clique_list))])

    for i in range(len(clique_list)):
        for j in range(i,len(clique_list)):
            overlap =  len(clique_list[i] & clique_list[j])
            if overlap >= (k-1):
                #if two cliques have at least k-1 overlap, mark them as␣
↪conncetng
                threshold_matrix[i][j] = 1
                if(i != j):
                    threshold_matrix[j][i] = 1

    communities = find_communities(threshold_matrix, clique_list)
    return(communities)
```

4.0.2 Question 4b

[9]:
```
G = nx.read_edgelist("communities.txt")
largest_connected_component = G.subgraph(max(nx.connected_components(G),␣
↪key=len))
pos = nx.spring_layout(largest_connected_component)
nx.draw(largest_connected_component, pos, node_size=100)
plt.show()
```

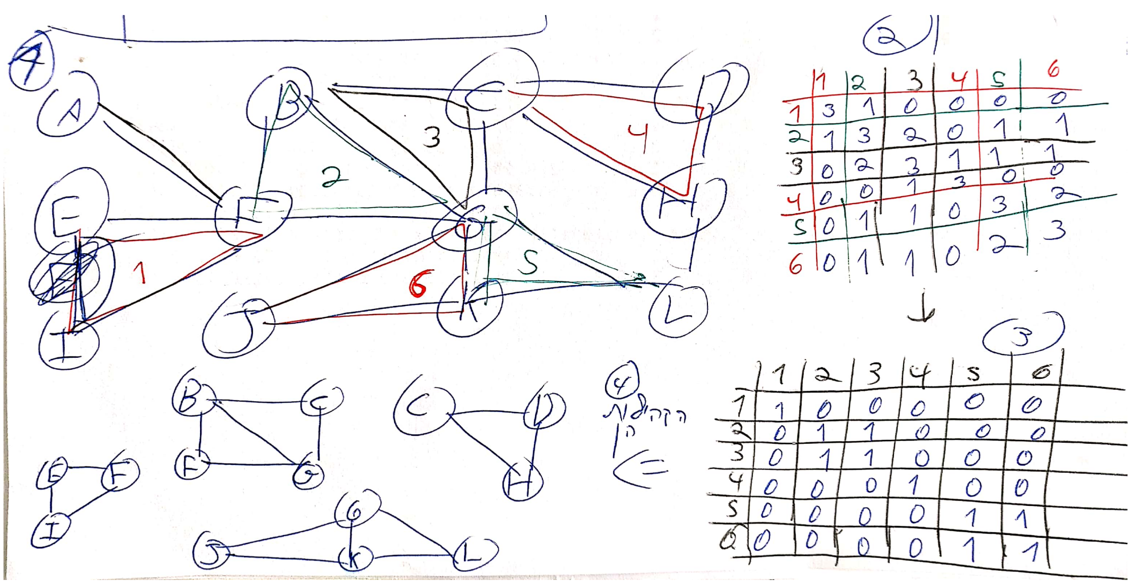


[11]:
```
communs = k_clique_communities(largest_connected_component, 4)
colors = ['#577590', '#43AA8B', '#90BE6D', '#F9C74F', '#F8961E', '#F3722C',␣
↪'#F94144', '#B977DC', '#5FDDF3']
nx.draw(largest_connected_component, pos, node_size=100)
for i in range(len(communs)):
    nx.draw_networkx_nodes(largest_connected_component, pos,␣
↪nodelist=communs[i], node_color=colors[i], node_size=100)
```



[ ]:
```
print("There are", len(communs),"4-clique comunities in the largest connected␣
↪component:\n")
print(communs)
```

4.0.3 Question 4c



Using the algorithm manualy, we get that there are 4 3-clique communities in the given graph:
1 - {E,F,I}
2 - {B,C,F,G}
3 - {C,D,H}
4 - {G,J,K,L}

4.0.4 Question 4d

The graphs central node is G.
This is because the maximum distance between it and any other node in the graph, is 2, and this is less then all other nodes.

```
[ ]:
```