```python
import networkx as nx
import matplotlib.pyplot as plt
import datetime
import numpy as np

import twitter_getting_data as tgd

def parse_net_report(path):
    file = open(path)
    relations = eval(file.readline())
    file.close()
    ids = []  # list of ids involved
    for relationship in relations:
        if not (relationship['source_user_id'] in ids):
            ids.append(relationship['source_user_id'])

    edges = {'followship': [],
             'likes': [],
             'rts': [],
             'replies': [],
             'matched_likes': [],
             'matched_rts': []}

    for relation in relations:
        if relation['source_follows_target']:
            edges['followship'].append((relation['source_user_id'],
                                        relation['target_user_id'],
                                        1))

        if relation['favorites']:
            edges['likes'].append((relation['source_user_id'],
                                   relation['target_user_id'],
                                   len(relation['favorites'])))

        if relation['retweeted']:
            edges['rts'].append((relation['source_user_id'],
                                 relation['target_user_id'],
                                 len(relation['retweeted'])))

        if relation['replies']:
            edges['replies'].append((relation['source_user_id'],
                                     relation['target_user_id'],
                                     len(relation['replies'])))

        if (relation['matched_likes'] and \
                relation['source_user_id'] < relation['target_user_id']):
            # we only add matched_likes once
            edges['matched_likes'].append((relation['source_user_id'],
                                           relations['target_user_id'],
                                           len(relation['matched_likes'])))
        if (relation['matched_rts'] and \
                relation['source_user_id'] < relation['target_user_id']):
            # we only add matched_likes once
            edges['matched_rts'].append((relation['source_user_id'],
                                         relation['target_user_id'],
                                         len(relation['matched_rts'])))
    return ids, edges


def get_directed_overall_edges(ids, edges_dict):
    dedges = []
    for source_id in ids:
        for target_id in ids:
            if source_id != target_id:
                weight = 0
                for pairs in edges_dict['followship']:
                    if source_id == pairs[0] and \
                            target_id == pairs[1]:
                        weight += pairs[2]
                for pairs in edges_dict['likes']:
                    if source_id == pairs[0] and \
                            target_id == pairs[1]:
                        weight += pairs[2]
                for pairs in edges_dict['rts']:
                    if source_id == pairs[0] and \
                            target_id == pairs[1]:
```

```python
                        weight += pairs[2]
                for pairs in edges_dict['replies']:
                    if source_id == pairs[0] and \
                            target_id == pairs[1]:
                        weight += pairs[2]
                if weight != 0:
                    dedges.append((source_id, target_id, weight))
    return dedges


def get_user_stats(likes, timeline):
    """
    time in days.
    """

    stats = {'like_time_period': datetime.datetime.strptime(likes[0]['created_at'], "%a %b %d %H:%M:%S %z %Y") \
                                 - datetime.datetime.strptime(likes[-1]['created_at'], "%a %b %d %H:%M:%S %z %Y"),
             'timeline_time_period': datetime.datetime.strptime(timeline[0]['created_at'], "%a %b %d %H:%M:%S %z %Y") \
                                 - datetime.datetime.strptime(timeline[-1]['created_at'], "%a %b %d %H:%M:%S %z %Y"),
             'total_likes': len(likes),
             'total_timeline': len(timeline),
             'likes_ranking': {},
             'rts_ranking': {},
             'replies_ranking': {}
             }

    for like in likes:
        stats['likes_ranking'][like['user']['id']] = 1 + stats['likes_ranking'].get(like['user']['id'], 0)

    for tweet in timeline:
        if tweet['in_reply_to_user_id']:
            stats['replies_ranking'][tweet['in_reply_to_user_id']] = 1 + stats['replies_ranking'].get(tweet['in_reply_to_user_id'], 0)

        if tweet.get('retweeted_status', False):
            stats['rts_ranking'][tweet['retweeted_status']['user']['id']] = 1 + stats['rts_ranking'].get(tweet['retweeted_status']['user']['id'], 0)

    return stats


def get_eigcentrality(potential_groupmates, net):
    id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}
    N = len(potential_groupmates)
    adjacency = np.zeros((N, N))
    for i, j in net:
        if i != j:
            adjacency[id_to_index[i], id_to_index[j]] = 1
            adjacency[id_to_index[j], id_to_index[i]] = 1
    vals, vecs = np.linalg.eigh(adjacency)
    index = np.where(vals == max(vals))[0]
    centrality = abs(np.transpose(vecs[:, index])[0])
    return centrality


def get_group_involvement(initial_friends, potential_groupmates, net,
                          max_iteration=100, partial_increment=0.1, decrease_factor=1.05):
    id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}
    N = len(potential_groupmates)
    adjacency = np.zeros((N, N))
    for i, j in net:
        if i != j:
            adjacency[id_to_index[i], id_to_index[j]] = 1
            adjacency[id_to_index[j], id_to_index[i]] = 1

    node_rank = np.zeros(len(potential_groupmates))
    for uid in initial_friends:
        if uid in potential_groupmates:
            node_rank[id_to_index[uid]] = 1
    iter = 0
    while iter < max_iteration:
        new_rank = np.zeros(len(potential_groupmates))
        for uid in potential_groupmates:
            score = node_rank[id_to_index[uid]]
            if sum(adjacency[id_to_index[uid], :]) > 0:# this should always happen
                new_rank += score /decrease_factor\
                            * adjacency[id_to_index[uid], :] / sum(adjacency[id_to_index[uid], :])
        node_rank = new_rank
        for uid in initial_friends:
            if uid in potential_groupmates:
                node_rank[id_to_index[uid]] += partial_increment
        iter += 1
    return node_rank


def get_like_involvement(initial_friends, potential_groupmates, net, mlikes=False,
                          max_iteration=100, partial_increment=0.1, decrease_factor=1.05):
    id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}
    N = len(potential_groupmates)
    adjacency = np.zeros((N, N))
    for i, j, likeij, likeji, mlikes in net:
        if i != j:
            if not mlikes:
                adjacency[id_to_index[i], id_to_index[j]] = likeij
                adjacency[id_to_index[j], id_to_index[i]] = likeji
            else:
                adjacency[id_to_index[i], id_to_index[j]] = mlikes
                adjacency[id_to_index[j], id_to_index[i]] = mlikes

    node_rank = np.zeros(len(potential_groupmates))
    for uid in initial_friends:
        if uid in potential_groupmates:
            node_rank[id_to_index[uid]] = 1
    iter = 0
    while iter < max_iteration:
        new_rank = np.zeros(len(potential_groupmates))
        for uid in potential_groupmates:
            score = node_rank[id_to_index[uid]]
            if sum(adjacency[id_to_index[uid], :]) > 0:  # this should always happen
                new_rank += score / decrease_factor \
                            * adjacency[id_to_index[uid], :] / sum(adjacency[id_to_index[uid], :])
        node_rank = new_rank
        for uid in initial_friends:
            if uid in potential_groupmates:
                node_rank[id_to_index[uid]] += partial_increment
        iter += 1
    return node_rank


def get_like_edges(ids, path):
    net = [] # (usr1, usr2, 1likes2, 2likes1, matched_likes)
    with open(path) as liked_file:
        liked = {}
        for line in liked_file:
            key, value = line.split(':', maxsplit=1)
            liked[int(key)] = eval(value)

    index = 0
    for source_id in ids:
        index += 1
        for target_id in ids[index:]:
            likes12 = 0
            likes21 = 0
            mlikes = 0

            for tweet in liked[source_id]:
                if target_id == tweet['user']['id']:
                    likes12 += 1
                #if tweet['id'] in liked[target_id]: # Don't think this is working
                #    mlikes += 1
            for tweet in liked[target_id]:
                if source_id == tweet['user']['id']:
                    likes21 += 1

            if likes12+likes21+mlikes > 0:
                net.append((source_id, target_id, likes12, likes21, mlikes))
    return net


def expanding_the_net():
    test_names = *** # Python list of length 5 which contains the Twitter user names, as strings, of each member in the seed-group
    test_ids = *** # Python list of length 5 which contains the Twitter user IDs, as integers, of each member in the seed-group

    file = open('large_net/qmisanz_friends_ids')
    potential_groupmates = eval(file.readline())
    file.close()
    file = open('large_net/qmisanz_friends_ids_netdata')
    net = []
    for pair in file:
        net.append(eval(pair))
    file.close()

    #centrality = get_eigcentrality(potential_groupmates, net)
    #node_rank = get_group_involvement(test_ids, potential_groupmates, net)
    like_net = get_like_edges(potential_groupmates, 'large_net/qmisanz_freinds_liked_2020-04-19')
    node_rank = get_like_involvement(test_ids, potential_groupmates, like_net)

    pairs = [(nr, uid) for nr, uid in zip(node_rank, potential_groupmates)]
    pairs.sort(reverse=True)
    node_rank = np.array([nr for nr, uid in pairs])
    potential_groupmates = np.array([uid for nr, uid in pairs])
    id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}

    show_plot = False
    if show_plot:
        for nr, uid in pairs:
            if uid in test_ids:
                plt.plot(id_to_index[uid], nr, '.C1')
            else:
                plt.plot(id_to_index[uid], nr, '.C0')
        plt.show()

    candidate_ranking = open('large_net/candidate_ranking_directed_likes', 'w')

    idtoname_file = open('large_net/qmisanz_friends_idtouser','r')
    id_to_name = eval(idtoname_file.readline())
    idtoname_file.close()

    for nr, uid in pairs:
        if uid in test_ids:
            print('!! {:17}, {:20}, {:8.6f}'.format(id_to_name[uid], uid, nr), file=candidate_ranking)
        else:
            print('{:20}, {:20}, {:8.6f}'.format(id_to_name[uid], uid, nr), file=candidate_ranking)
    candidate_ranking.close()


if __name__ == '__main__':
    test_names = *** # Python list of length 10 which contains the Twitter user names, as strings, of each member in the test group
    test_ids = *** # Python list of length 10 which contains the Twitter user IDs, as integers, of each member in the test group

    idtoname = {uid:name for uid, name in zip(test_ids, test_names)}
    idtonumber = {uid:number for number, uid in enumerate(test_ids)}

    # The time stamp in the name of the files should be adjusted, this is just an example
    ids, edges = parse_net_report('twitter_out/net_interactions_2020-06-08 16:29:58.015124.data')

    # printo name to number
    file_nodenames = open('twitter_out/edges_report_2020-06-08/node_names_2020-06-08.txt', 'w')
    print('#COMMENT: Each line contains the twitter screen name and the' +
          ' node number asigned to each user.', file=file_nodenames)
    for number, name in enumerate(test_names):
        print('{} {}'.format(name, number), file=file_nodenames)
    file_nodenames.close()

    # print followship relations
    file_followship = open('twitter_out/edges_report_2020-06-08/edges_followship_2020-06-08.txt', 'w')
    print('#COMMENT: Each line contains a follow relation from'+
          ' the first user to the second, each identified by their node number'+
          ' If the third number is 0 there is no relation.', file=file_followship)
    for src in ids:
        for trgt in ids:
            if src == trgt:
                continue
            if (src, trgt, 1) in edges['followship']:
                print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 1), file=file_followship)
            else:
                print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=file_followship)
    file_followship.close()

    # print like relations
    file_like = open('twitter_out/edges_report_2020-06-08/edges_like_2020-06-08.txt', 'w')
    print('#COMMENT: Each line contains the information about how many likes the first user' +
          ' gives to the second, each identified by their node number.', file=file_like)
    for src in ids:
        for trgt in ids:
            if src == trgt:
                continue
            for contact in edges['likes']:
                if (src, trgt) == contact[:2]:
                    print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], contact[2]), file=file_like)
                    break
            else:
                print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=file_like)
    file_like.close()

    # print matched like relations
    file_mlike = open('twitter_out/edges_report_2020-06-08/edges_matched_like_2020-06-08.txt', 'w')
    print('#COMMENT: Each line contains the information about how many times both users'+
          ' liked the same tweet, each user identified by their node number.', file=file_mlike)
    for src in ids:
        for trgt in ids:
            if src == trgt:
                continue
            for contact in edges['matched_likes']:
                if (src, trgt) == contact[:2]:
                    print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], contact[2]), file=file_mlike)
                    break
            else:
                print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=file_mlike)
    file_mlike.close()

    # print rt relations
    file_rt = open('twitter_out/edges_report_2020-06-08/edges_rt_2020-06-08.txt', 'w')
    print('#COMMENT: Each line contains the information about how many times the first user' +
          ' retweeted the second one, each identified by their node number.', file=file_rt)
    for src in ids:
        for trgt in ids:
            if src == trgt:
                continue
            for contact in edges['rts']:
                if (src, trgt) == contact[:2]:
                    print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], contact[2]), file=file_rt)
                    break
            else:
                print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=file_rt)
    file_rt.close()

    # print reply relations
    file_reply = open('twitter_out/edges_report_2020-06-08/edges_reply_2020-06-08.txt', 'w')
    print('#COMMENT: Each line contains the information about how many times the first user' +
          ' replied the second, each identified by their node number.', file=file_reply)
    for src in ids:
        for trgt in ids:
            if src == trgt:
                continue
            for contact in edges['replies']:
                if (src, trgt) == contact[:2]:
                    print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], contact[2]), file=file_reply)
                    break
            else:
                print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=file_reply)
    file_reply.close()
```