Discussion:

As is shown above, it takes about **1.22 sec** to find the top 10 hashtags using map reduce approach in Python, while it only takes **0.71 sec** using Unix command. In this question, I solved it in different ways when using map reduce approach and Unix command. As for map reduce, I used the same algorithm as question 1. However, for Unix command, I used "awk" command to sort each line into ascending order at the begining. By doing so, if two users is pair of reciprocal follower, their ids will appear twice in the output file. Finally, by using "sort" command, we can easily find out pairs that are reciprocal followers. I think this method is faster than map reduce approach.

3 Finding Friends of Friends

In [111]:

```
# Read csv file
import pandas as pd

edges_orig = pd.read_csv("./Twitter-dataset/data/edges.csv")
follower_graph = pd.read_csv('follower_graph.csv')
edges = edges_orig.head(500000)
edges_test = edges_orig.head(5000)
```

In [139]:

```
pairs = list(map(list, follower_graph.values))
```

In [112]:

```
groups_edges = combiner_reciprocal(mapper_reciprocal(edges))
```

In [126]:

```
def mapper_findFriends(userID, group):
    if userID in group:
        return group[userID]

# Example;
mapper_findFriends(1, groups_edges)
```

Out[126]:

```
[8762940, 8762941, 688136, 8762942]
```

In [117]:

```
def mapper_commonFriends(list1, list2):
    return list(set(list1).intersection(list2))

# Example;
mapper_commonFriends([1,2,3,4,5],[2,4])
```

Out[117]:

```
[2, 4]
```

In [140]:

```
def reducer_numOfFriends(list1, list2):
    common = list(set(list1).intersection(list2))
    return len(common)

# Example;
reducer_numOfFriends([1,2,3,4,5],[2,4])
```

Out[140]:

2

In [147]:

```
def execute_commonFriends(edgesGraph, followerGraph, groups):
    output = []
    for pair in followerGraph:
#         print(pair)
        userID = pair[0]
        follerID = pair[1]
        friendOfUser = mapper_findFriends(userID, groups)
        friendOfFoller = mapper_findFriends(follerID, groups)
        output.append((pair, reducer_numOfFriends(friendOfUser,friendOfFoller)))
    return output

output = execute_commonFriends(edges, pairs, groups_edges)
output = sorted(output, key = lambda x: x[1], reverse = True)
output[:20]
```

Out[147]:

```
[([3682, 5276], 714),
 ([5276, 3682], 714),
 ([40704, 40997], 402),
 ([40997, 40704], 402),
 ([40997, 41039], 360),
 ([41039, 40997], 360),
 ([23503, 41422], 352),
 ([41422, 23503], 352),
 ([60887, 70696], 332),
 ([70696, 60887], 332),
 ([135546, 135684], 282),
 ([135684, 135546], 282),
 ([70696, 70772], 259),
 ([70772, 70696], 259),
 ([40704, 41039], 252),
 ([41039, 40704], 252),
 ([13232, 63255], 236),
 ([63255, 13232], 236),
 ([32173, 32452], 194),
 ([32452, 32173], 194)]
```