Project Report

Question 1:

1. Chosen Function - get_out_degrees()

```
def get_out_degrees(rdd):
        rdd5 = rdd.map(lambda x: (x[0],x[2]))
        rdd61 = rdd.map(lambda x: (x[1],0))
        rdd71 = rdd61.union(rdd5).reduceByKey(lambda x,y: x+y).map(lambda x: (x[1],x[0])).sortBy(lambda x: (x[0],x[1]),False)
        return rdd71
```

Brief Explanation:

get_out_degrees is function in Question 3.2 which is used to get the out_degree of each node (i.e. one email id) from convert_to_weighted_network() created in Question 2. So basically, we are extracting total number of emails sent by each email id and sorting it in a decreasing lexicographical order of integers. We need to consider Email id's from Sender as well as in Receipent and get their out_degree (total no. of email's sent).

Function get_out_degrees takes a parameter rdd and rdd is passed in the function and a rdd is returned.

rdd5 is mapped with rdd which contains sender email id & no. of emails exchanged i.e. x[0] and x[2]. rdd61 is mapped with x[1] which is receiver email id & with 0. Further, rdd71 undergoes the union of rdd61 and rdd5 which is then reduceByKey by adding the numbers which gives us the out_degree of each email id which is then mapped as the format of out_degree, email_id and lastly sorted in descending lexicographical order of integers/string pairs.

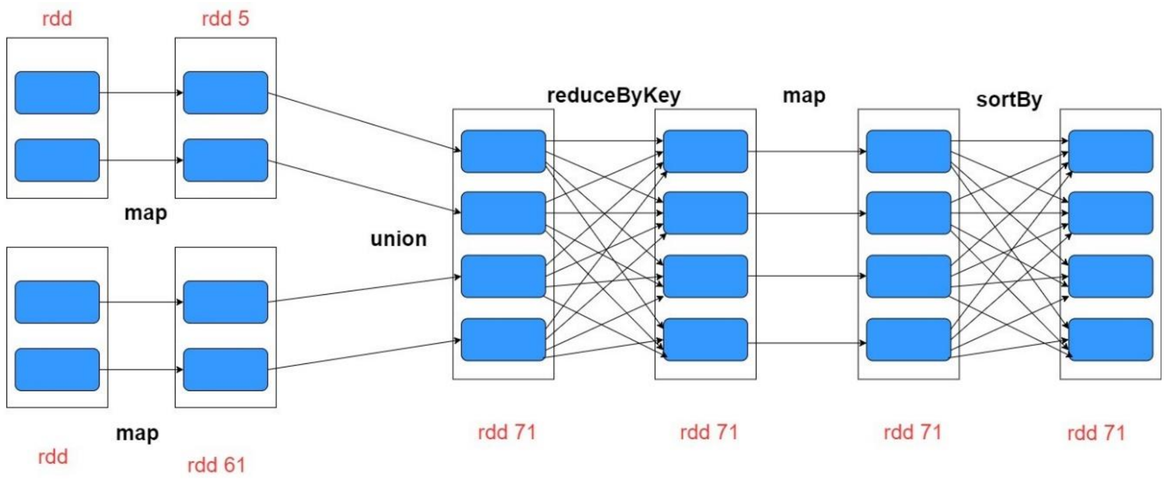
Finally, rdd71 is returned by the function get_out_degrees that gives total no. of emails sent to the corresponding email id.

Challenges Faced:

I first wrote the function that gave the final out_degree of just x[0] i.e. I only mapped the sender email id's and I forgot to map x[1] i.e. receiver email id's that gave me output something which is a lot like the expected output in test file provided but the integer values were incorrect and few email id's were missing. After finetuning and doing the required changes I got the expected result.

2.

Lineage Graph (DAG)



3. Yes, there are narrow dependencies. As we can see from above lineage graph there are 4 narrow dependencies.

a. rdd ----> rdd5 (map)

b. rdd ----> rdd61 (map)

c. rdd5 & rdd61 ----> rdd71 (union)

d. rdd71 ----> rdd71 (map)

4. Yes, there are wide dependencies. As we can see from above lineage graph there are 6 wide dependencies.

a. rdd71 ----> rdd71 (reduceByKey)

b. rdd71 ----> rdd71 (sortBy)

Question 2.

Chosen Slice: 01 Jan 2000 – 31 Dec 2000

1.

Total_weigted_degrees = 264,408

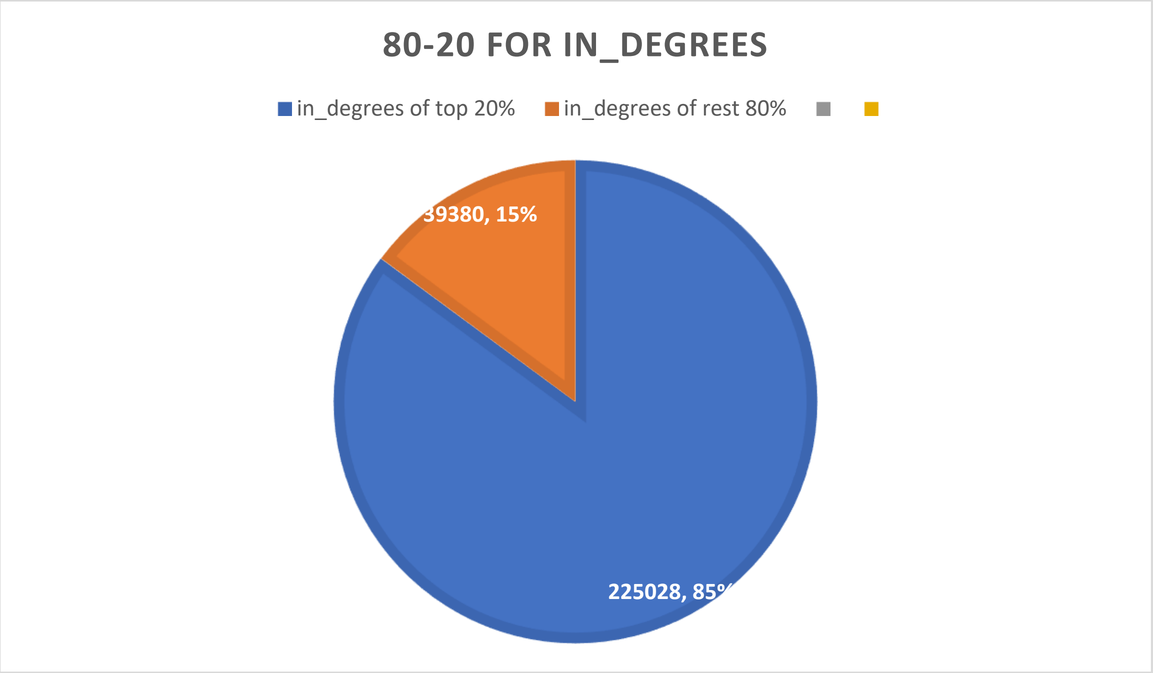
weighted_out_degrees of top 20% = 263,372

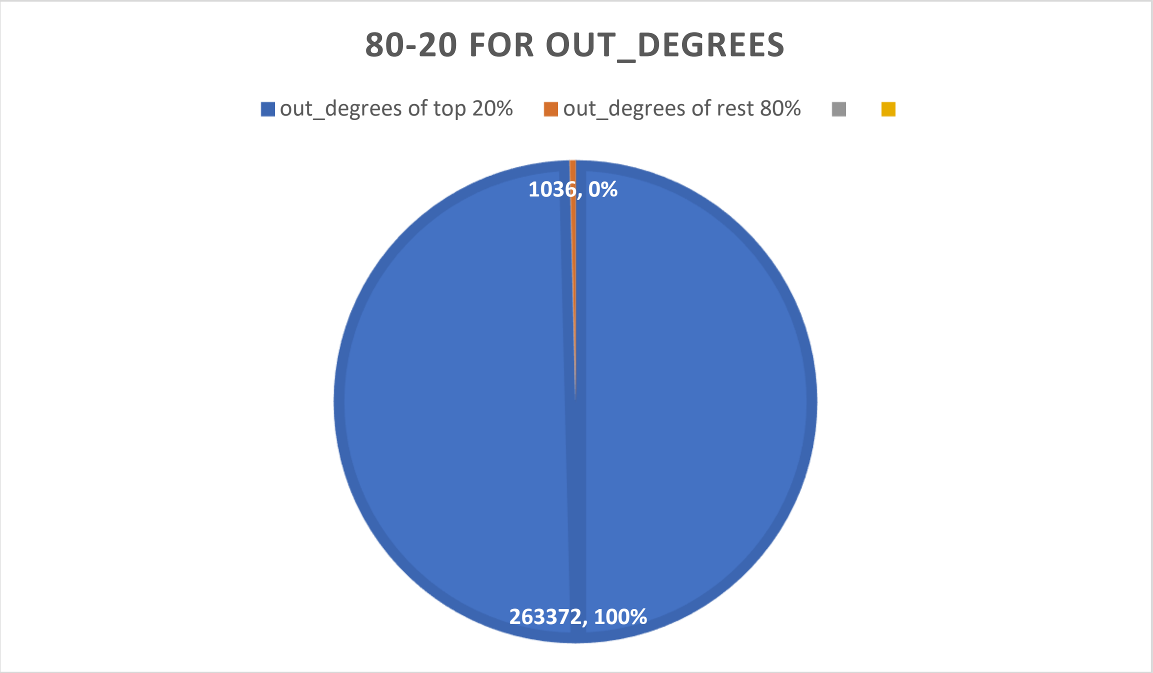
weighted_in_degrees of top 20% = 225,028

Hence,

Ratio of weighted_out_degrees of top 20% to total_weighted_degrees = 99.60%

Ratio of weighted_in_degrees of top 20% to total_weighted_degrees = 85.10%





P.S – Graph depicts rounded off percentage.

Hence from the above details we can say that it does not follow 80/20 rule as for in_degree it is 85.10% and out_degree it is 99.60% which is higher than the 80%

Explanation – Used the above mentioned slice to get the in_degrees and out_degrees of all the email ids by using the command ‘coalesce(1).saveAsTextFile("file_name")’ and then wrote a small snippet of code to extract the in_degrees and out_degrees and sum it & later on only extracting the top 20% and sum it and divide with total weighted degrees.

2.

From 1 Jan 2000 – 30 Apr 2000 (consecutive 4 months)

Kmax_out_degree: 3876

Kmax_in_degree: 1627

From 1 Jan 2000 – 31 Aug 2000 (consecutive 8 months)

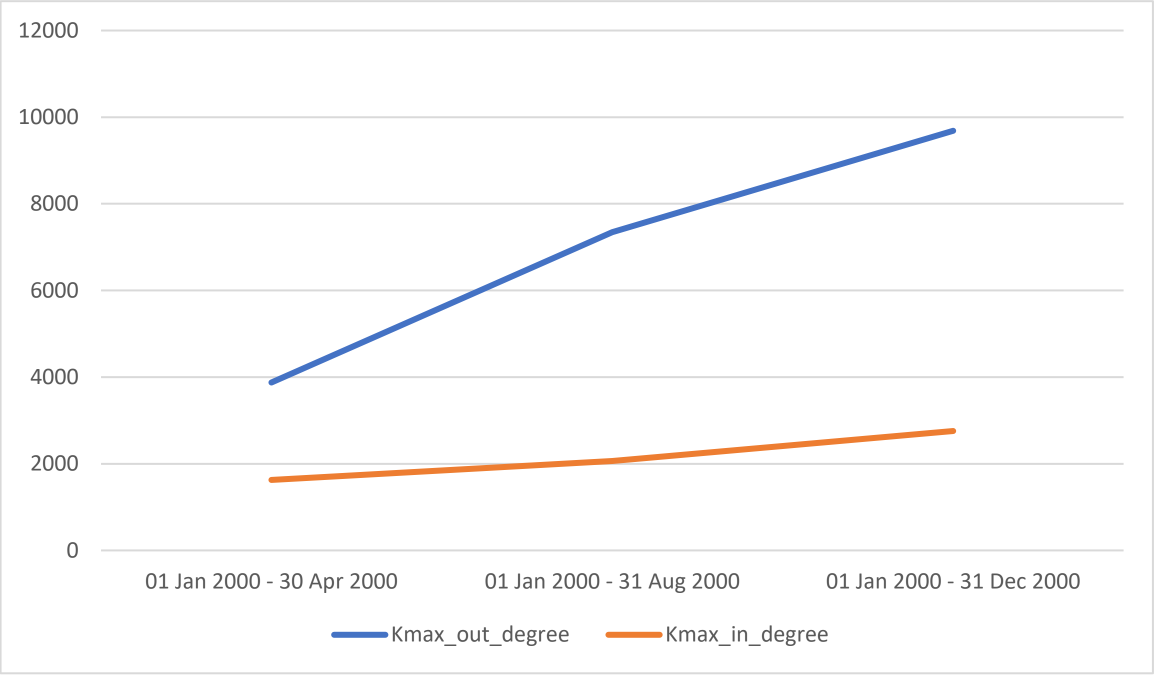
Kmax_out_degree: 7345

Kmax_in_degree: 2065

From 1 Jan 2000 – 31 Dec 2000 (consecutive 12 months)

Kmax_out_degree: 9686

Kmax_in_degree: 2756



Kmax for out_degree linearly increases with high growth rate whereas Kmax for in_degree does increase linearly but with very less growth rate after every consecutive 4 months.

Explanation: Used three different slices for both in_degrees and out_degrees which were 1 Jan 2000 – 30 Apr 2000, 1 Jan 2000 – 31 Aug 2000 & 1 Jan 2000 – 31 Dec 2000 which is 4 months, 8 months and 12 consecutive months respectively. Just had to take the kmax for both in_degrees and out_degrees in all three slices and plotted it which shows us a linear increase. Hence the rich getting richer analogy holds true here.