Matthew Virgin
Dr. Chaufen Chen
COS 480
24 April 2023

WRITEUP:

Task 1:

a) I decided to use MongoDB. I read the data in from the text files in much the same way I did for homework 2. This time, however, every row has a unique ID generated by ObjectID(). I also re-named the id attributes for person, movie, and director to make them easily distinguishable from one another, and to prevent querying for an ID of 54 giving you person 54, director 54, and movie 54. For the shard key, I would use the name of the movie. This would ensure the movie names are evenly distributed with the rest of the data, allowing for fast access to everything else.

b) See code comments. I defined a function that takes a movies name and does a sequence of queries to return the movies information, its cast information, and its directors information.

c) I would use the year as the shard key. Since this value is numerical, it can be distributed by range instead of with HASH, grouping movies of similar years, which would be good for quickly accessing similarly dated movies.

Task 2:

0. To add an edge from the node to every other node in the input graph when a node has no outgoing edge, I first got all the edges by using the .textFile method and .map to split each line of the text file into an rdd that is essentially a list of tuples, where each line is a tuple. I did the same thing for only the left values in the text file and the right values. Using .subtract, all nodes without any outgoing edges can be found because a node has no outgoing edge if a node on the right never appears on the left in the input file. Then, if the rdd containing these nodes without outgoing edges is not empty, .cartesian and .filter are used to put each outgoing edgeless node into a tuple with another, different node.
1. I used a simple .map on the nodes to initialize all ranks to 1, stored on a rdd called pRanks. Nodes was created with .flatmap and .distinct on the edges rdd
2. Nodes and their neighbors are first found and stored in an rdd. Then, pRanks is joined onto that rdd. Then, a contributions rdd is created that uses the contribs(someTuple) function that I made to calculate the contribution to each neighbor node.
3. pRanks is then updated by unioning the contributions and multiplying by them

Task 3:

a) The values I got from task 2 are put into a rdd using .parallelize. A spark dataframe is then created using the schema [‘id’, ‘val’].

```
b) query = spark.sql("SELECT val FROM data WHERE data.id=2")
   print(query.collect())
c) query2 = spark.sql("SELECT MAX(val) FROM data")
```

d) From here on, I ran out of time and couldn't get spark to cooperate with me. I kept getting errors with calling "o25.sql"