Problem Analysis

If you did not use a program, show the steps of the analysis you performed to estimate the number

We shall Analyze the values for below table using the approach explained below

Assume the Dataset to have a 2,000 values spread across 1000x2 (rows x columns)

Step 1

RS Join Input = equals the input size of the file = 2,000 = Z

RS Join Shuffled = equals the RS Join Input * RS Join Input = 2,000*2,000 = Z*Z

RS Join Output = equals A*B, where A and B are the sizes of 2 tables ~ 2,000*2,000 = Z*Z

Rep Join Input = equals the size of input

Rep Join File Cache = equals the size of the input = Y

Rep Join Output < Y*Y

Step 2

RS Join Input = equals the output from reducer 1 * input = Z

RS Join Shuffled = Z*Z

RS Join Output < Z*Z

Rep Join Input = size of the hashmap in Rep Join Output

Rep Join File Cache = size of the hashmap in Rep Join Output

Rep Join Output < size of the hashmap in Rep Join Output

Volume = Cardinality * 4

because each row is 4 bytes

	RS Join Input	**RS Join Shuffled**	**RS Join Output**	**Rep Join Input**	**Rep Join File Cache**	**Rep Join Output**
Step 1 (Join of edges with itself)	Z volume = 4Z	Z*Z volume =	Z*Z volume =	Y volume =	Y volume =	Y*Y volume =

		4*Z*Z	4*Z*Z	4*Y	4*Y	Y*Y
Step						
Step 2 (Join of path 2 with edges)	Z*Z volume = 4*Z*Z	Z*Z volume = 4*Z*Z	< Z*Z volume = 4*Z*Z	Y*Y volume = 4*Y*Y	Y volume = 4*Y	<Y volume = 4*Y

Join Implementation

Reduce Side Join algorithm

In the twitter data set, we have userid, followerid, where the userid follows the followerid with forward direction. Triangle count in the dataset is when we see $x \rightarrow y$, $y \rightarrow z$, $z \rightarrow x$ relationship.

Logic:

we will use two mapper and two reducers

step 1: First cycle of MapReduce produces first join, which is $x \rightarrow y, y \rightarrow z$

step 2: Pass the output of MapReduce from step 1 as input to mapper of second input
Second cycle of MapReduce, produces second join, which is $z \rightarrow x, x \rightarrow z$
count the matching entries there by finding the global count of triangles

The MapReduce framework guarantees, matching between like keys-values pairs. By maintaining the references as “table 1” and “table 2” while performing the self-join, we make sure we are not incorrectly matching same entries.

MAX-filter

The MapReduce program uses the above filter to filter ids which are greater than the mentioned threshold

First cycle of MapReduce

Enum
Used for a global triangle count

enum triangle {count}

```
map(String key, String value)
```

```
 {
      // key: user id (x)
      // value: follower id (y)
      // follower id is the second column

      for ( each row in the document )
      {
            emit(userid , followerid) // pass indicator as table 1 along with value
            emit(followerid , userid) // pass indicator as table 2 along with value

      }
}

reduce( String key, Iterator values )
{

      // key: user id (y)
      // values: ids ( different (x) and (z) )
      // create table 1
      // create table 2

      for (each val in values) {
            add the entry to table 1 if indicator is table 1
            add the entry to table 2 if indicator is table 2
      }
      for ( each value in table 1){
            for ( each value in table 2){
                  // get all the combinations for y and z
                  emit( y, z)
                  }
            }

}
```

Second cycle of MapReduce

```
map(String key, String value)
 {
      // key: user id (z)
      // value: follower id (x)
      // follower id is the second column

      for ( each row in the document )
      {

            if( the row is from join 1){
                  emit(z, x)
            }
            else{
                  emit(x, y)
            }

      }
}
```

```
reduce( String key, Iterator values )
{

      // key: user id (x)
      // values: ids ( (z) )
      // create table 1
      // create table 2

      for (each val in values) {
            add the entry to table 1 if indicator is join 1
            add the entry to table 2 if indicator is table 1
      }
      for ( each value in table 1){
            for ( each value in table 2){

                        if( the values are equal){
                        increment the enum count by 1


                  }
            }

}
```

Replicated Join Algorithm

Replicated join, works by replicating the same dataset across all the mappers. It is map only MapReduce. The join happens only in the mapper.

Logic

Create a HashMap by reading the input file
This HashMap will be available for all the mappers via distributed cache
The First join $x \rightarrow y, y \rightarrow z$ happens while we are creating the HashMap
Use the created HashMap and compare with the original input to get the final triangle count

MAX-filter

The MapReduce program uses the above filter to filter ids which are greater than the mentioned threshold

First cycle of MapReduce

Enum
Used for a global triangle count

```
mapsetup(String key, String value)
 {
      //key: user id (x)
```

```
      //value: follower id (y)

      create a HashMap

      for (each row in the input)
      {
            add the y,x to the HashMap
      }
}



map( String key, Iterator values )
{

      // key: user id (y)
      // values: ids ( different (x) and (z) )

      for ( each key in HashMap)
      {
            get the list of y
            if ( z→x, matches with x→z)
            {
                  count the triangles
            }
      }


}
```

Configuration	**Small Cluster Result** **5 data nodes and 1 master**	**Large Cluster Result** **8 data nodes and 1 master**
RS-join, MAX = 10000	Running time: 21 minutes, Triangle count: 520296	Running time: 22 minutes, Triangle count: 520296
Rep-join, MAX = 20000	Running time: 24 minutes, Triangle count: 9564591	Running time: 26 minutes, Triangle count: 9564591