# [CS6240] Large-Scale Parallel Data Processing Homework03

Hwijong Im

## ***Problem01***

***(1). RDD-G Pseudo Code***

```
val conf = new SparkConf().setAppName("RDD-G").setMaster("local")
val sc = new SparkContext(conf)

val lines = sc.textFile("edges.csv")
val pairs = lines.map(line => {
  val parts = line.split(",")
  (parts(1).toInt, 1)
}).filter(_._1 % 100 == 0)

val grouped = pairs.groupByKey()
val counts = grouped.mapValues(_.sum)

counts.saveAsTextFile("output/RDD-G")
sc.stop()
```

***(2). RDD-R Pseudo Code***

```
val conf = new SparkConf().setAppName("RDD-R").setMaster("local")
val sc = new SparkContext(conf)

val lines = sc.textFile("edges.csv")
val pairs = lines.map(line => {
  val parts = line.split(",")
  (parts(1).toInt, 1)
}).filter(_._1 % 100 == 0)

val counts = pairs.reduceByKey(_ + _)

counts.saveAsTextFile("output/RDD-R")
sc.stop()
```

***(3). RDD-F Pseudo Code***

```
val conf = new SparkConf().setAppName("RDD-F").setMaster("local")
val sc = new SparkContext(conf)

val lines = sc.textFile("edges.csv")
```

```
val pairs = lines.map(line => {
  val parts = line.split(",")
  (parts(1).toInt, 1)
}).filter(_._1 % 100 == 0)

val counts = pairs.foldByKey(0)(_ + _)

counts.saveAsTextFile("output/RDD-F")
sc.stop()
```

***(4). RDD-A Pseudo Code***

```
val conf = new SparkConf().setAppName("RDD-A").setMaster("local")
val sc = new SparkContext(conf)

val lines = sc.textFile("edges.csv")
val pairs = lines.map(line => {
  val parts = line.split(",")
  (parts(1).toInt, 1)
}).filter(_._1 % 100 == 0)

val zeroValue = 0
val counts = pairs.aggregateByKey(zeroValue)(
  (acc, value) => acc + value,
  (acc1, acc2) => acc1 + acc2
)

counts.saveAsTextFile("output/RDD-A")
sc.stop()
```

***(5). DSET Pseudo Code***

```
val spark = SparkSession.builder()
  .appName("DSET")
  .master("local")
  .getOrCreate()

import spark.implicits._

val df = spark.read.csv("edges.csv")
  .toDF("user", "follower")
  .select($"follower".cast("int").as("follower"))
  .filter($"follower" % 100 === 0)
  .groupBy($"follower")
  .agg(count($"follower").as("num_followers"))

df.write.format("csv").save("output/DSET")
spark.stop()
```

## Problem02

Code Link of Aggregations

***RDD-G:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/spark_g/RDD_G.java

***RDD-R:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/spark_r/RDD_R.java

***RDD-F:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/spark_f/RDD_F.java

***RDD-A:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/spark_a/RDD_A.java

***DSET:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/spark_dset/RDD_DSET.java

## Problem03

### *(1). RDD-G*

The shuffling has been proceeded before using GroupByKey.

```
24/02/16 21:16:29 INFO SparkContext: Created broadcast 0 from textFile at RDD_G.java:54
24/02/16 21:16:29 INFO FileInputFormat: Total input files to process : 1
(40) MapPartitionsRDD[6] at mapValues at RDD_G.java:68 []
 |   MapPartitionsRDD[5] at groupByKey at RDD_G.java:67 []
 |   ShuffledRDD[4] at groupByKey at RDD_G.java:67 []
 +-(40) MapPartitionsRDD[3] at filter at RDD_G.java:64 []
    |   MapPartitionsRDD[2] at mapToPair at RDD_G.java:58 []
    |   input/edges.csv MapPartitionsRDD[1] at textFile at RDD_G.java:54 []
    |   input/edges.csv HadoopRDD[0] at textFile at RDD_G.java:54 []
```

### *(2). RDD-R*

ReduceByKey operation includes a shuffling stage where local aggregation is performed before data is shuffled across the network. the “ShuffledRDD[4]” indicates that shuffling occurs as part of the reduceByKey operation.

```
Execution Plan for followerCounts:
(40) ShuffledRDD[4] at reduceByKey at RDD_R.java:56 []
 +-(40) MapPartitionsRDD[3] at filter at RDD_R.java:53 []
    |   MapPartitionsRDD[2] at mapToPair at RDD_R.java:47 []
    |   input/edges.csv MapPartitionsRDD[1] at textFile at RDD_R.java:43 []
    |   input/edges.csv HadoopRDD[0] at textFile at RDD_R.java:43 []
24/02/16 21:50:22 INFO SparkContext: Starting job: foreach at RDD_R.java:66
24/02/16 21:50:22 INFO DAGScheduler: Registering RDD 3 (filter at RDD_R.java:53) as input to shuffle 0
24/02/16 21:50:22 INFO DAGScheduler: Got job 0 (foreach at RDD_R.java:66) with 40 output partitions
24/02/16 21:50:22 INFO DAGScheduler: Final stage: ResultStage 1 (foreach at RDD_R.java:66)
24/02/16 21:50:22 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 0)
24/02/16 21:50:22 INFO DAGScheduler: Missing parents: List(ShuffleMapStage 0)
```

### *(3). RDD-F*

FoldbyKey operation also includes a shuffling stage to aggregate effectively.

```
Execution Plan for FoldByKey:
(40) ShuffledRDD[4] at foldByKey at RDD_F.java:55 []
 +-(40) MapPartitionsRDD[3] at filter at RDD_F.java:52 []
    |   MapPartitionsRDD[2] at mapToPair at RDD_F.java:46 []
    |   input/edges.csv MapPartitionsRDD[1] at textFile at RDD_F.java:42 []
    |   input/edges.csv HadoopRDD[0] at textFile at RDD_F.java:42 []
```

### *(4). RDD-A*

AggregatebyKey operation starts at the nearly same with shuffling stage.

```
Execution Plan for Aggregate:
(40) ShuffledRDD[4] at aggregateByKey at RDD_A.java:57 []
 +-(40) MapPartitionsRDD[3] at filter at RDD_A.java:53 []
    |   MapPartitionsRDD[2] at mapToPair at RDD_A.java:47 []
    |   input/edges.csv MapPartitionsRDD[1] at textFile at RDD_A.java:43 []
    |   input/edges.csv HadoopRDD[0] at textFile at RDD_A.java:43 []
```

***(5). DSET***

As you can see the image, the aggregation operation, groupby, has been performed after shuffling stage. This indicates that Spark collects and redistributes the data based on the grouping key("userid") through shuffling, and applies the aggregation function "count" to compute the final results.

```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[userId#21], functions=[count(followerId#22)])
   +- Exchange hashpartitioning(userId#21, 200), ENSURE_REQUIREMENTS, [plan_id=34]
      +- HashAggregate(keys=[userId#21], functions=[partial_count(followerId#22)])
         +- Project [_c0#17 AS userId#21, _c1#18 AS followerId#22]
            +- Filter (isnotnull(_c0#17) AND ((_c0#17 % 100) = 0))
               +- FileScan csv [_c0#17,_c1#18] Batched: false, DataFilters: [isnotnull(_c0#17), ((_c0#17 % 100) = 0)],
 Format: CSV, Location: InMemoryFileIndex(1 paths)[file:/Users/minsungim/Desktop/Desktop/hw-3-bellwood22-master/input/
edg..., PartitionFilters: [], PushedFilters: [IsNotNull(_c0)], ReadSchema: struct<_c0:int,_c1:int>


== Parsed Logical Plan ==
'Aggregate [userId#21], [userId#21, count('followerId) AS followercount#30]
+- Filter ((userId#21 % 100) = 0)
   +- Project [_c0#17 AS userId#21, _c1#18 AS followerId#22]
      +- Relation [_c0#17,_c1#18] csv

== Analyzed Logical Plan ==
userId: int, followercount: bigint
Aggregate [userId#21], [userId#21, count(followerId#22) AS followercount#30L]
+- Filter ((userId#21 % 100) = 0)
   +- Project [_c0#17 AS userId#21, _c1#18 AS followerId#22]
      +- Relation [_c0#17,_c1#18] csv

== Optimized Logical Plan ==
Aggregate [userId#21], [userId#21, count(followerId#22) AS followercount#30L]
+- Project [_c0#17 AS userId#21, _c1#18 AS followerId#22]
   +- Filter (isnotnull(_c0#17) AND ((_c0#17 % 100) = 0))
      +- Relation [_c0#17,_c1#18] csv

== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[userId#21], functions=[count(followerId#22)], output=[userId#21, followercount#30L])
   +- Exchange hashpartitioning(userId#21, 200), ENSURE_REQUIREMENTS, [plan_id=34]
      +- HashAggregate(keys=[userId#21], functions=[partial_count(followerId#22)], output=[userId#21, count#34L])
         +- Project [_c0#17 AS userId#21, _c1#18 AS followerId#22]
            +- Filter (isnotnull(_c0#17) AND ((_c0#17 % 100) = 0))
               +- FileScan csv [_c0#17,_c1#18] Batched: false, DataFilters: [isnotnull(_c0#17), ((_c0#17 % 100) = 0)],
 Format: CSV, Location: InMemoryFileIndex(1 paths)[file:/Users/minsungim/Desktop/Desktop/hw-3-bellwood22-master/input/
edg..., PartitionFilters: [], PushedFilters: [IsNotNull(_c0)], ReadSchema: struct<_c0:int,_c1:int>
```

## ***Problem04***

***(1). RS-R***

```
object RS_R_Triangles {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("RS_R_Triangles").setMaster("local")
    val sc = new SparkContext(conf)

    // Load data
    val rawData: RDD[String] = sc.textFile("input/edges.csv")

    // Create (a, b) and (b, a) pair RDDs
    val edges: RDD[(String, String)] = rawData.flatMap(line => {
```

```
    val parts = line.split(",")
    Seq((parts(0), parts(1)), (parts(1), parts(0)))
  })

  // Create pairs for potential triangles (a, b) => (b, a)
  val possibleTriangles: RDD[((String, String), String)] = edges.map {
    case (a, b) => ((a, b), b)
  }

  // Join on (a, b) to find all potential triangles
  val joined: RDD[((String, String), ((String, String), String))] = possibleTriangles.join(possibleTriangles)

  // Filter to find actual triangles
  // This step filters out false positives by ensuring that the third edge exists to close the triangle
  val triangles: RDD[(String, String, String)] = joined.flatMap {
    case ((a, b), ((_, _), c)) if a != c && b != c =>
      Seq((a, b, c))
    case _ => Seq.empty[(String, String, String)]
  }.distinct()

  // Count the triangles
  val triangleCount: Long = triangles.count()

  println(s"Total Triangles: $triangleCount")
  triangles.saveAsTextFile("output/RS_R_Triangles")

  sc.stop()
 }
}
```

***(2). RS-D***

```
object RS_D {
 def main(args: Array[String]): Unit = {
  // Spark setting
  val spark = SparkSession.builder.appName("RS_D").master("local").getOrCreate()

  // Dataset import
  val edges = spark.read.option("inferSchema", "true").option("header", "false")
      .csv("input/edges.csv").toDF("src", "dst")
  val reversedEdges = edges.select(col("dst").alias("src2"), col("src").alias("dst2"))

  // join_01: Find pairs of edges that could potentially form triangles
  val joined = edges.as("edges1").join(reversedEdges.as("edges2"),
      col("edges1.dst") === col("edges2.src2"))

  // join_02: Complete triangles by finding the third edge
```

```
    val triangles = joined.as("joined")
                .join(edges.as("edges3"), col("joined.dst2") === col("edges3.src"))
                .where(col("joined.src") === col("edges3.dst"))

    // Output
    triangles.show()
    val trianglesCount = triangles.count()
    println(s"Total count: $trianglesCount")

    // Save output
                                                triangles.write.format("csv").option("header",
"true").mode("overwrite").save("output/RS_D/triangles")

    spark.stop()
  }
```

***(3). REP-R***

```
object REP_R {
  def main(args: Array[String]): Unit = {
    // Spark setting
    val conf = new SparkConf().setAppName("REP_R").setMaster("local")
    val sc = new SparkContext(conf)

    // data load
    val rawData: RDD[String] = sc.textFile("input/edges.csv")

    // (a, b), (b, a) pair
    val df01: RDD[(String, String)] = rawData.map(line => {
      val parts = line.split(",")
      (parts(0), parts(1))
    })
    val df02: RDD[(String, String)] = rawData.map(line => {
      val parts = line.split(",")
      (parts(1), parts(0))
    })

    // df02 mapping and broadcasting
    val broad_df02: Broadcast[Map[String, String]] = sc.broadcast(df02.collectAsMap())

    // perform join
    val triangles: RDD[String] = df01.flatMap {
      case (a, b) =>
        val cOption = broad_df02.value.get(b)
                    if (cOption.isDefined && broad_df02.value.contains(cOption.get) &&
broad_df02.value(cOption.get) == a) {
          Some(s"$a,$b,${cOption.get}")
        } else {
          None
```

```
      }
    }

    // triangle counts
    val triangleCount: Long = triangles.count()
    println(s"Total Triangles: $triangleCount")

    // result
    triangles.saveAsTextFile("output/REP_R")

    sc.stop()
  }
}
```

***(4). REP-D***

```
object REP_D {
  def main(args: Array[String]): Unit = {
    // Spark setting
    val spark = SparkSession.builder()
                        .appName("REP_D")
                        .master("local")
                        .getOrCreate()

    import spark.implicits._

    // data load
    val edges: Dataset[Row] = spark.read
                              .option("inferSchema", "true")
                              .csv("input/edges.csv")
                              .toDF("src", "dst")

    // create opposite dataset
    val reversedEdges = edges.select($"dst".alias("b"), $"src".alias("a"))

    // broadcasting
    val broadcastedReversedEdges = broadcast(reversedEdges)

    // first join
    val join01 = edges.join(broadcastedReversedEdges, $"dst" === $"b")
                  .select($"src".alias("a"), $"dst".alias("b"), $"a".alias("c"))

    // second join
    val triangles = join01.join(edges, $"c" === edges("dst") && $"a" === edges("src"))
                        .select($"a", $"b", $"c")

    // output
    triangles.show(false)
    val triangleCount = triangles.count()
```

```
    println(s"Total Triangles: $triangleCount")

    // save
    triangles.write
            .format("csv")
            .option("header", "true")
            .mode("overwrite")
            .save("output/REP_D/triangles")

    spark.stop()
  }
}
```

## *Problem05*

Github Link of Join Code

***RS-R:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/join_rs_r/RS_R.java

***RS-D:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/join_rs_d/RS_D.java

***REP-R:***

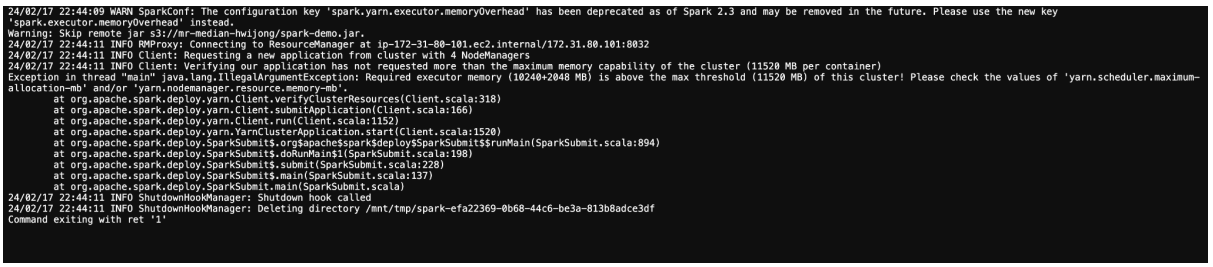
https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/join_rep_r/REP_R.java

***REP-D:***

https://github.com/CS6240/hw-3-bellwood22/blob/master/src/main/java/join_rep_d/REP_D.java

## *Problem06*

### *(1). RS-R*




Tried to solve the memory issue, however, even when I tried to extend memory, the same issue happened repeatedly. I lowered the max value of executor memory to 8GB with a memory overhead of 3GB.

**machine type**: 1 master node, 4 worker node
**max value:** 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time:** NA
**triangle count:** NA

### *(2). RS_D*

Attempted to process the tasks for over 3 hours, but the jobs continued without completing, leading me to conclude they wouldn't finish in a reasonable timeframe. Despite the code being correctly implemented and capable of calculating the number of triangles accurately, the main challenges encountered were the substantial size of the data set and the complexity of the task, compounded by the insufficient number of worker nodes available. This situation highlights the critical impact of resource limitations on processing large-scale data tasks and underscores the need for adequate computational resources to efficiently handle such demanding workloads.

**machine type:** 1 master node, 4 worker node
**max value:** 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time:** NA
**triangle count:** NA

***(3).REP_R***


Tried several ways to manage the memory issue, however, the memory of the master node has been exceeded.

**machine type**: 1 master node, 4 worker node
**max value**: 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time**: NA
**triangle count**: NA

***(4). REP_D***

**machine type**: 1 master node, 4 worker node
**max value**: 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time**: 143343 milliseconds
**triangle count**: 2336

## ***Problem07***

***(1). RS_R Link:***

https://mr-median-hwijong.s3.amazonaws.com/log/j-33DCSHZPY8INN/steps/s-0559237WGG845J0CYCR/stderr.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIAWVEGCM7SYYTVDMWD%2F20240218%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240218T204016Z&X-Amz-Expires=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEB0aCXVzLWVhc3QtMSJHMEUCIBix1QG%2F8PahhIEoz%2FUm0orwLxhy9Dj1L2V46zKOlssMAiEAmdcD5yn7z9yID%2F44rRLgIEjAH7zp0hQ44bWXzfbHdUEqigMI9v%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw0NTc2OTUxOTMwNjEiDIdhK57MRSCBfPC4JyreAvbI%2FezHN54Z39gHVt8A5RBZKI0pRWBm2fTsRlWbB0VM67VFuSxVcp%2BERGeaBtdOFvKYx9leGcsfPgwfrwStnI8XMp9spLcizJdK6zFcmf3uogT3fMfUCmG3gBWcMWxt%2F4K2n5Gwd5tu3R1PNOTUZEHFKnhkuQEmHczvJ65hRfd88wrtcrhS2VXjm1jl4Q%2BvjWrII4ZVBd1JM

X0BnPCXJVAXN%2FuUumtPMguKU2FyT8%2B7K0uI0knAWi%2BcAxGR6DoNXN25VNWR
sgjw%2FDq1MKcAbUZU6KI3iXW3e9sG39X9nWqWsiyhjrkIiJyq%2BSheMswnlZDLdM8LQ4
FHpK6xHZl9SxDcMf1nm8l96UtUPAXeA37XgmwWsrLxlUcZvPhRrJOpZ%2Bk0fCNlvp3GgL
7Ded%2FY4773%2FjJC1ni3jvWvdBoQEvN92uubwLpZMA8HLnWq3EwlSyqoctsMxNBD2Q
%2Bk%2BBX8MLm8ya4GOocCGfFAIkZJ33%2BbkvOhYkTkkgFhrZYl09PCtvxyLT1u%2Bae
XNVsoDuwNi0YLoCQ8e1aTMocUdKCLHkZ3pRzcqsBbKtfgIII%2BQBLT01H6m09cSmWtDF
UJZqYHzLggCZYRfpYw54ud9L4iogm4w%2FBB5x890ILhl8zIIM%2BxcLpanB5tFNQbMSOP
91J6juCl6kXlztdbkR27auD1ClqCRnOJtJlzH0540TAtx%2BziuLGdE2EYsBk%2Bd88QRZaRe
U7OPASHd8KQIPEYqlnVg7r3trxeNxJN%2F7Su%2F2nu1kNt26Yb2TdZC7lWV9LQzc7lzXUj
ZGq86d5X36fxWIVKp2E2bUADk6sJURHjS2qEacY%3D&X-Amz-Signature=ee9068443750
d79b6001aee2261ff3252b3fb67c0b8b42c55bfcb7fcf9b0b0b9&X-Amz-SignedHeaders=host&
response-content-disposition=inline

***(2). RS_D Link:***

https://mr-median-hwijong.s3.amazonaws.com/log/j-3MH5ZMSM4EAVF/steps/s-07210232E
5WS81GL3XBR/stderr.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASI
AWVEGCM7SWBYO2GX6%2F20240218%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-D
ate=20240218T211009Z&X-Amz-Expires=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjE
B0aCXVzLWVhc3QtMSJGMEQCID53DY%2BbiBOYUH8jBOVw2qSma8f3gdg4bywm9zc8M
K2TAiBdEvrsYHY6DsufHvOMOo1wkrnosij7lA%2FYNsDXcyhJvyqKAwj2%2F%2F%2F%2F
%2F%2F%2F%2F%2F%2F8BEAAaDDQ1NzY5NTE5MzA2MSIM9sDVnA2G6Y7ng0aOKt4
CXW0gSkq4Px59dJ%2BIaPJ4vL1USiIRptmXhj88JyP7KgmBQZfx%2BY8CZyzqZGsYNxoN
ujRGDOVthW3VWMi%2BlPm7MlImUCqp0nyG0sAo8LB7MLRwb8KSPzkxgyksI3kjYu79v5%
2FwoF0WT7IxnIryZT5IQF4VkG2O3snk7R75nybEB2TF5Z5w5xfWoQ4PLm0sullSxNW81pK
FJLj34rfU6%2FczYI%2F208Y9KENZJAPhUIpZVXzaM6ogCuREhrStKJxNUp9MQqEwgZ9h
eZWHZjfY4b01tX1pVAOEscqhivKyes%2Fn74qJ4Zt4FzlyTcRvZoqjT%2B2ny6QeuI5r0MDWx
92FXTiGp%2F7bjHlwF0woWlYApe%2Fw8v0eJ%2BosD%2FiiMm8AdKoKIT%2FsHAu%2Bg
37QUora8NecpLa2bSJ9Sa%2BMOVFyPfr1HZ9S1NFkGv9Mf544t5jaY5gFYGLAQXATemqI0
%2FmDBEiKEI0wubzJrgY6iAL8b5S3ZuEQInnMmjFEokRfRhRxKes2R%2BHmRML3shfjdh
RvhOPmvdPVNszwC8QAmdABxOizk3DqxQsX0Xb0QW5xQCmAlVnMt0WD3V%2FY2zRfP
owYRutGB%2B7PwaomHZ997OC8XW%2FN4LdD6Vr3Lg9gjvd52t5AdqC4BeEeQgGNkvVX
FDl51m2m8PxhGziSSPoREA4dM17vWcl2h0A7%2Bexh06VoBz%2BSqG4IiW4ogunuTzASh
e%2B4hwjzOpVu%2B395u%2F%2B66eISTE%2F9uSo3junyQEkTmlP9vwDwUaeMsl6oGsm
ZzneEzIGFRxaaY4k4njFsMZvJHh9nF3s1UGOF6BCEZRsx%2FQNSofcllrtBu1o%3D&X-Am
z-Signature=1ff102dd3cf2e43b53ca9f862672ef104c8a4eeb367014e72c4d1381990edf3e&X-
Amz-SignedHeaders=host&response-content-disposition=inline

***(3). REP_R Link:***

https://mr-median-hwijong.s3.amazonaws.com/log/j-3A1HL75C67P4R/steps/s-02438524O6
A90GGESBV/stderr.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIA
WVEGCM7SSOADV3HI%2F20240218%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Dat
e=20240218T225029Z&X-Amz-Expires=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEB
8aCXVzLWVhc3QtMSJGMEQCIBWiralhPBCGkuEottcagTrVjwZSkZICAeq%2BXxoqO2eXAi
BZqefAPtAiJDq%2Ft%2FFh%2BKGWhHruW94KN%2FgnTQhlDVM90yqKAwj4%2F%2F%2
F%2F%2F%2F%2F%2F%2F%2F8BEAAaDDQ1NzY5NTE5MzA2MSIMadb26wbSh3PIycsV
Kt4CPPfCWu6ujBOobrDvVk%2B5iy2EccdVbxJAQR49s286Dcd04Vs%2Fz0s6Dpl0gfUB9J3
be%2B1W5w%2Bkpdzj6AlQBuiS%2B6qVl3ZYUYkBj2rtFXU1SvxcXb%2BcJ268Fg1tUUzTD
phzHUuVa1z3OA5Inum35MsQx8yNaq2hzLQgWc3FgDYb0cYuOjAqO5cbJhL4vTU2d7jU9Ve

%2BmnDsgr9WCmlee1ZjrRTvGzn93K5S7nT%2BJaQ1w6On5PVHQKsGJ5ONLJ3Qd3sj9C
7yZGizVSY8BJ1pqEMhxQKtMcBX3YIJEp8DSGsID4AhgNs1liS0CT3j5yaZCZi4frlnIPuzkbEE
BvgxgwUHWSxZ10EoiWn7vvJprz01WwpHDs%2Btm16GfO%2Bmc5DPIYdrmmi%2FO8JIn1
POEu7O054gkh4SC0nyDWHUoZcc6zDt4Hi8t%2BeIUaZwp7oLSW14%2BIUNto1d7tZVeW
NqJ3QKEh4wubzJrgY6iAJ1s7XOQM8Krw77f%2BwsIx3%2FZCbi%2BJ5rh0B8yR4w7NFDUf
jw9OMzu3rOzwl8qRcSjQnRk8QaBBPPKfsZXxeFSN1ljzqxg0evdovzT%2FiHiCMhZ4d75Adr
kNGFW8r8G9yGtfKTDmW7aePpMUTa1YKByljndKtk5jdNmhqYgOHUCG2nYqtG%2B4XN%
2Fvj196%2BGBZtpw5Vu%2BAjCP7ph2Jf82VgiqxJYkM5fqC2RPXJQ%2FjZNfQ3DhvWH2Xa
hEl3gjvMglAHkwZe2OLgpf6nJqo4i%2FvGqzG0n9ghA9yFrN1lsxLCrvoZCMI4t3tC3%2FhrY3
5jbvcfkdRfueFgcgWi%2BdXjoBPis6DDlo3WQwo6LaJQ%3D&X-Amz-Signature=015e33297
089cc7fd8ea3c7dceb89186cfb2a43eb8a22f0a5e39c9a1a3be9d1d&X-Amz-SignedHeaders=
host&response-content-disposition=inline

***(4). REP_D Link:***
https://mr-median-hwijong.s3.amazonaws.com/log/j-3562A00IZRIC1/containers/application_
1708297026350_0001/container_1708297026350_0001_01_000001/stderr.gz?X-Amz-Algor
ithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIAWVEGCM7SSFJEFZ5K%2F2024021
8%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240218T230926Z&X-Amz-Expire
s=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEB8aCXVzLWVhc3QtMSJGMEQCIE9%2
BTLcTTZBOjyPW2j393mBxElrp3CgJfDphYDQwmdUgAiA0IWvQDwNlKaGhP0JKUhwuX2B
4bhNg5wvsEKTsKZC1uSqKAwj4%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAAaDD
Q1NzY5NTE5MzA2MSIMg1HX9vbyUwl1gslyKt4CyUnyv9zpOIHRqB6T9BuUiL84Qaqp%2B
8hPwLME%2FkuPlzTrjgoleNhNVIR0wfbReolQC9IyLZ3Kx5NhdfcQFEvNKUbr%2B4agzPnx5
uhYWZ5cuehLU2uQ4jw%2BMIQWnZl%2BpaUi0z%2BtrerozEo%2FL4R3RTVSTE9fXo8Pk9
Wt%2FxTkG8bIu2oI%2BdrMSYuch1pxvwW4AzAd0V5y1YxukE6jbjKYkCtdSZHZKhzr%2F1
UsYfH0BU%2BmT0PTlqIefKeAkDsxGEhGv9FYzrb5bzNJiFyMtOr6oFGJmEHRZBFTU0uVnr
i28GqXc8DO%2FvW%2B%2BcBaEFiuDVjWcCoun14wVX4k6gN8mvOxFJNP%2Fc59kZXS
CIERJrHN6xYuS%2FtG9sJcWrzDMC8dqJR4gzh2oFzWAnpvzTHKO4KNLrInlkTCZplK1RB6
ExNPjlsDCEfIVnt9Vir4pojbJCtLlEYgvQAcVaSrCnXjTmrMC%2FkwubzJrgY6iAKi83pyefpB3L
10euwQw0x9fWUGLDJpl9HBXOX7pVo8XKRW70WuQoycBcgyY0UF%2Bm%2BdYfJfEmAy
FuR6FEVznAvFFYGjJFWi%2F0q7jE3nrUMwi2R%2BospBcJiP9TosUsWj5uiCGVVJKwYJhG
rhSsblbKIdWBGMC1BFTfy51vf5JjC25ZyDxtmnkFZUbbry%2F9M5m4P2TyJD4owOiT9Cozt8
FaM9C%2FwMyQRiXwEJ6y8KAEqSYWvVSI%2BVpGJBvADYU10EEFvhYRPk9yRFVjkHf8
%2Fo7tngmNT%2FVV8nvGecEmTz4YVB8t55GA9%2B36wdMl%2BHiMwjezvHi6BqRMpgW
yKboGHACbcudMseK%2B%2FAOdU%3D&X-Amz-Signature=6bdb03f490605ca5c9c8a875
3756f52b9412788418e0fca13b08d24c14906cf8&X-Amz-SignedHeaders=host&response-co
ntent-disposition=inline

## ***Problem08***

***(1). RS-R***
Nevertheless ran 2 hours to perform the program, it did not completed but only keep running.

**machine type**: 1 master node, 8 worker node
**max value**: 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time**: na
**triangle count**: na

***(2). RS-D***

Nevertheless ran 2 hours to perform the program, it did not completed but only keep running. Because of the join inefficiency, the program code is hard to get the result of the number of triangles.

**machine type**: 1 master node, 8 worker node
**max value**: 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time**: na
**triangle count**: na

***(3). REP-R***

Tried several ways to manage the memory issue, however, the memory of the master node has been exceeded.

**machine type**: 1 master node, 8 worker node
**max value**: 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time**: na
**triangle count**: na

***(4). REP-D***

**machine type**: 1 master node, 8 worker node
**max value**: 8 driver memory, 8 executor memory, 3 of each memory overhead
**running time**: 2336
**triangle count**: 135588 milliseconds

## *Problem09*

***RS-R Link:***

https://mr-median-hwijong.s3.amazonaws.com/log/j-2CHI9HQXOYUM2/containers/application_1708297787674_0001/container_1708297787674_0001_01_000001/stderr.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIAWVEGCM7STPLDEEOE%2F20240218%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240218T233318Z&X-Amz-Expires=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjECAaCXVzLWVhc3QtMSJHMEUCIGqb8d2sZ9m%2BNoGXdQ7CETykE7GOkI%2BS5aUKKRXgYmtFAiEAlXuOHWXkF4%2BwrNvXIaAaqbs7A8S2LWhzgs6VLvhqqyEqigMI%2Bf%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw0NTc2OTUxOTMwNjEiDDLvlkJH6G%2B%2BorPqlireAprmsHNjntYW4R5zU9%2BLml%2Fd4YXO68yUYw3mynoQp9%2B2AoHDRT1w558bH7Ww8B12aZfGljgKAx0bEjLjuJiL00I5cBtf5OzgTiIziImEkax6tqcl657Cn9JlZRXUFXwXIsWs3y7%2B2bQ92%2B3l95WsNQQnZ%2FlahAaVh9W2%2FQxhFwfYBQ23hEmrsmzy7Wi1UyrRqexmUZ3gFgddIedQWce99ajIvSHqXLC%2BoQiI8K9c8iJ%2By9Q4FVGskqQRu0WiKbQb1%2FugncEBaUhS5FyqHfBXu4%2FUeuU9rrt3wspk%2Femjn7UocR9nYdBX43T9k9PSz7PvYm1%2BvjLDgV3rbBlO9BVhE3y2PbqRiTzMEC9zk5NZz4u8LWbWal9UypgD9TAPdBe18jtdESBk%2FsPlc1jdHH5gcCScO0walb1GwbvtwE6RTUHLEkvaA4p0%2BLbhE%2BDPK1yf642lWfdX%2FNVnCS%2FNbKQTMLm8ya4GOocC8xpTu4kx6S6LKDDoan%2FvyUH%2BNQ%2FsfZh8w8DibsFDPRnV%2BdsKCxsf%2Fs6FvBKanzCuB1hzJYoyip7ucvRppJUS0i7M1neWLS%2FFF8m7lRZFCNP5o4hNP5LF2mF83k%2Fb0pRubGMxYuB6Fa997kuJimUhEclPerDn3oOw3lx3IrIoz7HHjM5fcJsru

HAd1sC%2F6W3dqKgu5f5gYYvuybQm290JDk9BIHwxxIrRtrHsu8rbS4fw7SVCgHQXbzPxJ8swmc1nfGUEVhkDzgGI5D0jMXKMrFFh0O0Dq%2Fe7GHQhKCJ1zLK7qNaF9qFg2W%2FJ5v1M%2Fm2P1fTOtI40wKuEKXLksO%2F6QH6%2BBHPT1no%3D&X-Amz-Signature=ca2241750011a1ee60dd7aecd3d93e8f71a49c493be8c554cb82b18f4cf56b36&X-Amz-SignedHeaders=host&response-content-disposition=inline

***RS-D Link:***

https://mr-median-hwijong.s3.amazonaws.com/log/j-31Y7CKCZ5FZO9/steps/s-09554103682H2QSEBE4C/stderr.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIAWVEGCM7SXIQMQIOQ%2F20240219%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240219T002553Z&X-Amz-Expires=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjECEaCXVzLWVhc3QtMSJHMEUCIBZkaD1EzjsD0AbDQCbNt8cDUel2JoF1OqRRJD8%2FkK%2B1AiEApuAYx1nCcHyP9DkVpzPySUB%2Ft8LtSRgbbVP%2B9X0h%2B%2BMqigMI%2Bf%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw0NTc2OTUxOTMwNjEiDNwglJgowuDAn9cgdCreArdGopIo21TAk8WtH0PcWVMj%2BXYS88NFVUgz8xJCg%2FZafceyS0M4l0LOM4JLD4eHa3nCCHsONekTo4mIoPwGqqFyY5qEzGsj1YTkWvv%2FoFx4DX2cRf3RDD9Ez3VZ77qjb2L85xSEUiRP6sr09Bn8mFmBkXyELCYvk1docBWlDieyaO0eon89TR2YtAytL9ggcMnOjjU3Nwk9ImSE2MVQxk5sv9QRGK7RHjlViZwg1unt5FNR1iXtknM4ngfJUFakZRf5AIHmLwRDFJqvs5KKUXepAi3C93SFx6G03m5nnLu7auAyipPYlaltlXZ1J7hbKKkfv7SNOj5E5%2F8Nkt%2BrBEWGHOks4zSSqoyS1REn7s2AgoygrS7P4lpzoiI1KxO9cP3EhkPwH1h0JL2Vn75iRX%2BZ2j6iSEEin4ovPc%2BZG5JJlhIgimRMKf4LC5XJJWsHpfPWk9H2nJ9qzQ3DDe0LMI6yyq4GOocC7e%2FdQKk5pKJW%2Bbo6TrWjwtsfzruF3ubE5RknDCu6QuNC2oYs1Nt0ToIMmYplKGnhbpRC3fVbunL%2FHjSUbDLILh095XjYHVV0jyp3MZftuK4KQfvO8ff79YRMIoQCp52kmD9x%2BBOtnWvfz4fWUR4FzKCSu7zqQLLsPdnEnd8NP7LrAEstS%2BFSgzjWiei6UvOAOq7YDII84uetdZBODSgl6CK%2Fyyd73HSG5K84F15koLAV7mcLzYomxtdrtnly%2FweppHHqFK0dlH02ty7gO3TLhXuyPNM6lKHSz%2FvangPIGVA2IILgSkPetiUHS6lO4rPBC7FlZzQ8KRVKq8A7RCiT6nknMhEKWA8%3D&X-Amz-Signature=7249880221694fe20e8290713a05668d68770a4b3736e8094c6625277a70714b&X-Amz-SignedHeaders=host&response-content-disposition=inline

***REP-R:***

https://mr-median-hwijong.s3.amazonaws.com/log/j-D9UWLDORVFAY/steps/s-00285333916G9OH5CJZZ/stderr.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIAWVEGCM7SWNRAVQXJ%2F20240219%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240219T005318Z&X-Amz-Expires=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjECEaCXVzLWVhc3QtMSJIMEYCIQCcTm8fE1nvCb%2BpPhUd5%2Frba8ymS1bnUUXJwZA9eKgIfQIhAPURTMxfyFYsVEZ7YNQrXoH%2BCCH6p2%2BJV8h4uoVdeFimKooDCPr%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEQABoMNDU3Njk1MTkzMDYxIgx2vpo7bHBbjf%2BJ0GAq3gJMw5YsMUgqx3gXCIsiOIqTjX10Uo2kn%2Fgf52HtFtTJ8t9sWRaA2GxRBZlX4gLC7EZ30RdxD6CSWvffoX%2B2DK9%2Bz62eztOJVn3yeQkpC4SmW11wINVYOnrIWJnVgJ%2FymRBKjVn9X2euW1uAZcmGHPtvou0FiwkwxYnYNFdOnq%2Fg8OEKxJN0LSEfampzmJE2D6kX9Q7nwtj5c5KVxKUk3oGC7iulB6KaLOVb%2FCWK%2B0QWaz2tjcKYAZV69pb1RrCD6LDm6%2F4%2FIeN%2B53UFHI5qTjPY5htfkeUzKVIZW26CrFCPgxDxFcX8OYPlv896rEjYgH%2F%2BhbRud3RGYWj7HVyax%2FJkK3M2syNEKCJIYbS7uapPNfLwT%2B%2B6EsW6TH%2FBExLIQxS2XEWOnrKKfDxEOJ9IbzTDNiXywfIdNGFKvQO7RwGxT3sMIIx7xIbpUC1CGdbIzSAGlLmclvoCLBVtmQDxCjCOssquBjqGAr8pG8hfcuuTpQsqa06Pvt1JrUSQvmwkBO8%2FQAFrUeOOSgDiygeXZAFstNPEG0iL4szAwk6afoXm3TdoqLlHTBTHW51yu93QBB8i3kKm3BijMZE8BTpqwcnFLrhrQQumhKrLSg%2B%2BrfmXQFZs3s5oiANLCttbrBa7rmtliVC

PRml8HxO%2Bo6RWc13jRPOv1fwjcv8zEt%2FiSRp32AoYv94ZHenHlcUJhqZxHYvyl98bU%2BZekAyTn76y%2FDwjJSdHGl0CRWJPN3mk31wPHKizTC%2Bk9OHdtJYPm1j7HX7dHtr5cw2SU0IzZEMFrD8ytxKCrtnK7B%2F7AkkbTs9IIbqOMHZgg4RftMKI4Zo%3D&X-Amz-Signature=9e44baa835f14e72f931e4c3f532ebaa8843e12beaec241f5c81d324855ab762&X-Amz-SignedHeaders=host&response-content-disposition=inline

***REP-D:***

https://mr-median-hwijong.s3.amazonaws.com/log/j-F1JXU1D1JSKI/containers/application_1708304365290_0001/container_1708304365290_0001_01_000001/stderr.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIAWVEGCM7S66VSZVPG%2F20240219%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240219T010814Z&X-Amz-Expires=900&X-Amz-Security-Token=IQoJb3JpZ2luX2VjECIaCXVzLWVhc3QtMSJGMEQCIAa8lwLrK09tCgBK61bo4DkIXymCSsH3jcKH0PH4JwfWAiA33uZszse374URDdDwLI7x0us8RDvn4QJ0x3VS8GF61yqKAwj6%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAAaDDQ1NzY5NTE5MzA2MSIMlnZY8Xgvm4zBqp5gKt4ChpxF76DsAUsVIFRTjwItikeaUvwm7Rh9ML36QvPrSI%2BLTqotYgXEo%2BqQA8vFgOV1gh4CeYWCoQMHeSbi8ACytgX%2BYC7mBopuygbFJpY7kHKtfOJhCokxMDL1ejoe2BiGBXA%2BFngy3Lx%2FMmebMMkgFrY%2F4UH8NPrm7EkIRM9ek%2FxDyzlkKj%2FeE8xTEQHAOwVipXQ6BrsVa0R51K9dMpRd8k4v%2BerIsNekSxhagVqnYaZrmy059uRdtjSvbIur%2B4njbuxLzbEjqJMbSpdn2Q5dmUu8D00XyKKOaEd5GwGeQgkl5eFeedsZhJD%2FRM6j3EXuQiURakp%2FD0ZHTFGePCJ6Y7IQtA7oz4VFf%2F2xyQyXWyRi4uD0nTmd25DnYICUclsUevS%2BdIugS0rkTCLkbN%2BnURKgcGA7CqqFW97lLBPilrQCpDE%2FZpe5Vy1VtAO8j7r%2FKlMndr3Ge%2ByaTIuYYLswjrLKrgY6iALoOzCEZU2b1PeO5rSGEUIvpXLldDIKyUa447oweozdFUcvl6qsnOTMeIYma4lWo8mNXGje1bhTJBsYEmDMUizicDN%2BuutHXZBVLnsKYFnnoBwO3k7wmieKIAtW5HkN672Hg%2Fz06eYL9Y%2FvVQ5%2FZPpdJ1%2B67VQBZ4Po7KpHDIi8PzkfMy%2BkZV1M0Oak7jblMutKwSpoYPb9qYbXW1q4BchclrBenZcv%2B0Yc8yfJI38IA5ykRwmUSIPuanvooc14rAXITgKqld10jtkekPTJ%2BmBJKs5qpfnPPVpH%2BNqnvgoNrQ%2BznnmkDHkMIDKE4MBQgX9FNEXhVj3xBReYRYtS7eE0ddBbPqlgUzM%3D&X-Amz-Signature=6d588f3a6910ae38d4b4511d1143d54a077f868ef40c3a91f4fa450483152bea&X-Amz-SignedHeaders=host&response-content-disposition=inline