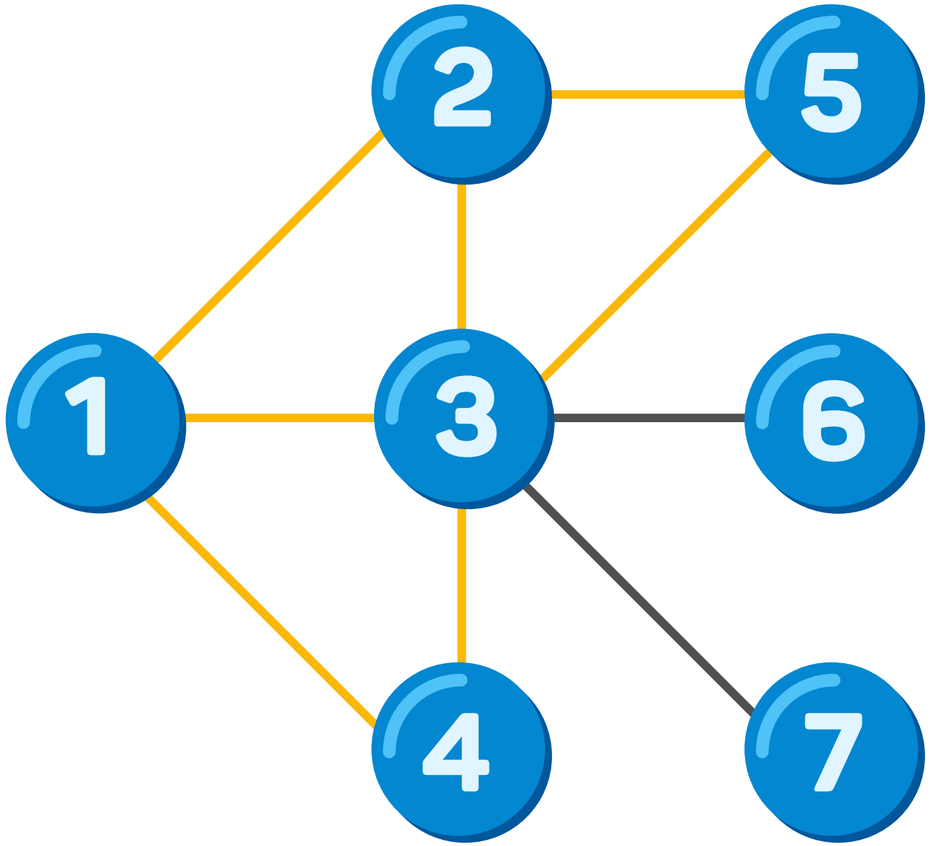
Count Triangles:edge list

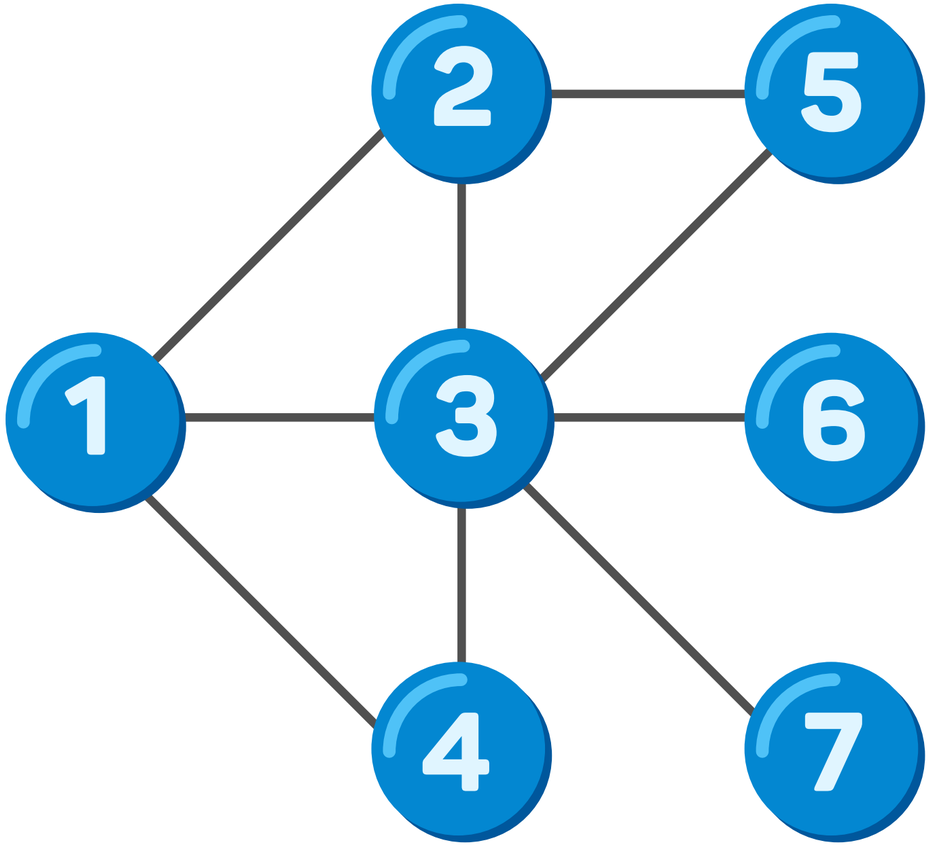
Mini social graph



1 - 2 triangles
2 - 2 triangles
3 - 3 triangles
4 - 1 triangle
5 - 1 triangle
6 - 0 triangles
7 - 0 triangles

Mini social graph

edgeList = [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 4), (3, 5), (3, 6), (3, 7)]



edgeList = [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 4), (3, 5), (3, 6), (3, 7)]

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
from pyspark.sql import Row

edgeList = [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 4), (3, 5), (3, 6), (3, 7)]

graphData = sparkSession.sparkContext.parallelize(edgeList).map(lambda (src, dst): Row(src, dst)

graphSchemaAB = StructType([StructField('A', IntegerType(), nullable=False),
                            StructField('B', StringType(), nullable=False)])

ab = sparkSession.createDataFrame(graphData, graphSchemaAB)

ab.show()
```

```
+---+---+
|  A|  B|
+---+---+
|  1|  2|
|  1|  3|
|  1|  4|
|  2|  3|
|  2|  5|
|  3|  4|
|  3|  5|
|  3|  6|
|  3|  7|
+---+---+
```

```
graphSchemaBC1 = StructType([StructField('B', IntegerType(), nullable=False),
                             StructField('C1', StringType(), nullable=False)])
bc1 = sparkSession.createDataFrame(graphData, graphSchemaBC1)

graphSchemaAC2 = StructType([StructField('A', IntegerType(), nullable=False),
                             StructField('C2', StringType(), nullable=False)])
ac2 = sparkSession.createDataFrame(graphData, graphSchemaAC2)
```

```
bc1.show()
```

```
+---+---+
|  B| C1|
+---+---+
|  1|  2|
|  1|  3|
|  1|  4|
|  2|  3|
|  2|  5|
|  3|  4|
|  3|  5|
|  3|  6|
|  3|  7|
+---+---+
```

```
ac2.show()
```

```
+---+---+
|  A| C2|
+---+---+
|  1|  2|
|  1|  3|
|  1|  4|
|  2|  3|
|  2|  5|
|  3|  4|
|  3|  5|
|  3|  6|
|  3|  7|
+---+---+
```

```
abc1 = ab.join(bc1, "B")
abc1.show()
```

```
+---+---+---+
|  B|  A| C1|
+---+---+---+
|  3|  1|  4|
|  3|  1|  5|
|  3|  1|  6|
|  3|  1|  7|
|  3|  2|  4|
|  3|  2|  5|
|  3|  2|  6|
|  3|  2|  7|
|  2|  1|  3|
|  2|  1|  5|
+---+---+---+
```

```
abc1c2 = abc1.join(ac2, "A")
abc1c2.show(15)
```

```
+---+---+---+---+
|  A|  B| C1| C2|
+---+---+---+---+
|  1|  3|  4|  2|
|  1|  3|  4|  3|
|  1|  3|  4|  4|
|  1|  3|  5|  2|
|  1|  3|  5|  3|
|  1|  3|  5|  4|
|  1|  3|  6|  2|
|  1|  3|  6|  3|
|  1|  3|  6|  4|
|  1|  3|  7|  2|
|  1|  3|  7|  3|
|  1|  3|  7|  4|
|  1|  2|  3|  2|
|  1|  2|  3|  3|
|  1|  2|  3|  4|
+---+---+---+---+
only showing top 15 rows
```

```
abc1c2.filter("C1 = C2").show()
```

```
+---+---+---+---+
|  A|  B| C1| C2|
+---+---+---+---+
|  1|  3|  4|  4|
|  1|  2|  3|  3|
|  2|  3|  5|  5|
+---+---+---+---+
```

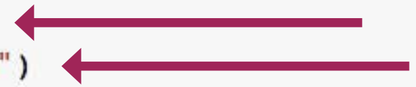
```
{
    abc1c2.filter("C1 = C2")
    .select(array(col("A"), col("B"), col("C1")).alias("triangleVertices"))
    .select(explode("triangleVertices").alias("triangleVertex"))
    .groupBy("triangleVertex")
    .count()
    .show()
}
```

```
+--------------+-----+
|triangleVertex|count|
+--------------+-----+
|             3|    3|
|             5|    1|
|             1|    2|
|             4|    1|
|             2|    2|
+--------------+-----+
```

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
from pyspark.sql import Row

edgeList = [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 4), (3, 5), (3, 6), (3, 7)]


graphData = sparkSession.sparkContext.parallelize(edgeList).map(lambda (src, dst): Row(src, dst)

graphSchemaAB = StructType([StructField('A', IntegerType(), nullable=False),
                        StructField('B', StringType(), nullable=False)])

ab = sparkSession.createDataFrame(graphData, graphSchemaAB)

graphSchemaBC1 = StructType([StructField('B', IntegerType(), nullable=False),
                        StructField('C1', StringType(), nullable=False)])
bc1 = sparkSession.createDataFrame(graphData, graphSchemaBC1)

graphSchemaAC2 = StructType([StructField('A', IntegerType(), nullable=False),
                        StructField('C2', StringType(), nullable=False)])
ac2 = sparkSession.createDataFrame(graphData, graphSchemaAC2)

abc1 = ab.join(bc1, "B")
abc1c2 = abc1.join(ac2, "A")


vertexTriangle = abc1c2.filter("C1 = C2") \
    .select(array(col("A"), col("B"), col("C1")).alias("triangleVertices")) \
    .select(explode("triangleVertices").alias("triangleVertex")) \
    .groupBy("triangleVertex") \
    .count() \
    .show()
```

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
from pyspark.sql import Row

edgeList = [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 4), (3, 5), (3, 6), (3, 7)]


graphData = sparkSession.sparkContext.parallelize(edgeList).map(lambda (src, dst): Row(src, dst)

graphSchemaAB = StructType([StructField('A', IntegerType(), nullable=False),
                        StructField('B', StringType(), nullable=False)])

ab = sparkSession.createDataFrame(graphData, graphSchemaAB)

graphSchemaBC1 = StructType([StructField('B', IntegerType(), nullable=False),
                        StructField('C1', StringType(), nullable=False)])
bc1 = sparkSession.createDataFrame(graphData, graphSchemaBC1)

graphSchemaAC2 = StructType([StructField('A', IntegerType(), nullable=False),
                        StructField('C2', StringType(), nullable=False)])
ac2 = sparkSession.createDataFrame(graphData, graphSchemaAC2)

abc1 = ab.join(bc1, "B")
abc1c2 = abc1.join(ac2, "A")


vertexTriangle = abc1c2.filter("C1 = C2") \
    .select(array(col("A"), col("B"), col("C1")).alias("triangleVertices")) \
    .select(explode("triangleVertices").alias("triangleVertex")) \
    .groupBy("triangleVertex") \
    .count() \
    .show()
```

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
from pyspark.sql import Row

edgeList = [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 4), (3, 5), (3, 6), (3, 7)]


graphData = sparkSession.sparkContext.parallelize(edgeList).map(lambda (src, dst): Row(src, dst)

graphSchemaAB = StructType([StructField('A', IntegerType(), nullable=False),
                            StructField('B', StringType(), nullable=False)])

ab = sparkSession.createDataFrame(graphData, graphSchemaAB)

graphSchemaBC1 = StructType([StructField('B', IntegerType(), nullable=False),
                            StructField('C1', StringType(), nullable=False)])
bc1 = sparkSession.createDataFrame(graphData, graphSchemaBC1)

graphSchemaAC2 = StructType([StructField('A', IntegerType(), nullable=False),
                            StructField('C2', StringType(), nullable=False)])
ac2 = sparkSession.createDataFrame(graphData, graphSchemaAC2)

abc1 = ab.join(bc1, "B")
abc1c2 = abc1.join(ac2, "A")


vertexTriangle = abc1c2.filter("C1 = C2") \
    .select(array(col("A"), col("B"), col("C1")).alias("triangleVertices")) \
    .select(explode("triangleVertices").alias("triangleVertex")) \
    .groupBy("triangleVertex") \
    .count() \
    .show()
```



```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
from pyspark.sql import Row

edgeList = [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 4), (3, 5), (3, 6), (3, 7)]


graphData = sparkSession.sparkContext.parallelize(edgeList).map(lambda (src, dst): Row(src, dst)

graphSchemaAB = StructType([StructField('A', IntegerType(), nullable=False),
                            StructField('B', StringType(), nullable=False)])

ab = sparkSession.createDataFrame(graphData, graphSchemaAB)

graphSchemaBC1 = StructType([StructField('B', IntegerType(), nullable=False),
                            StructField('C1', StringType(), nullable=False)])
bc1 = sparkSession.createDataFrame(graphData, graphSchemaBC1)

graphSchemaAC2 = StructType([StructField('A', IntegerType(), nullable=False),
                            StructField('C2', StringType(), nullable=False)])
ac2 = sparkSession.createDataFrame(graphData, graphSchemaAC2)

abc1 = ab.join(bc1, "B")
abc1c2 = abc1.join(ac2, "A")


vertexTriangle = abc1c2.filter("C1 = C2") \
    .select(array(col("A"), col("B"), col("C1")).alias("triangleVertices")) \
    .select(explode("triangleVertices").alias("triangleVertex")) \
    .groupBy("triangleVertex") \
    .count() \
    .show()
```

3 shuffles

```
edgeListBC = sparkSession.sparkContext.broadcast(set(edgeList))

# Define udf
from pyspark.sql.functions import udf

def isInEdgeList(src, dst):
    return (int(src), int(dst)) in edgeListBC.value

udfIsInEdgeList = udf(isInEdgeList, StringType())

abc1.withColumn("isTriangle", udfIsInEdgeList("A", "C1")).show()
```

```
+---+---+---+----------+
|  B|  A| C1|isTriangle|
+---+---+---+----------+
|  3|  1|  4|      true|
|  3|  1|  5|     false|
|  3|  1|  6|     false|
|  3|  1|  7|     false|
|  3|  2|  4|     false|
|  3|  2|  5|      true|
|  3|  2|  6|     false|
|  3|  2|  7|     false|
|  2|  1|  3|      true|
|  2|  1|  5|     false|
+---+---+---+----------+
```

Summary

- You have known how count number of triangles using edge lists

Summary

- You have known how count number of triangles using edge lists
- You have know how to reduce amount of shuffles of data in counting triangles algorithm using broadcast