NAME:- DUBEY KARAN SANJEEV
CLASS:- B.E - 4
ROLL NO:- 04
BATCH:- A

Experiment No-5

AIM: To implement Matrix multiplication using Map-Reduce.

THEORY:

What is Matrix Multiplication?

Matrix multiplication is a binary operation that takes a pair of matrices, and produces another matrix. Numbers such as the real or complex numbers can be multiplied according to elementary arithmetic.

Let A be an m x n matrix and B an n x p matrix.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$$

We want to compute the product AB, an m x p matrix.

$$\mathbf{AB} = \begin{bmatrix} \sum_{j=1}^{n} a_{1j}b_{j1} & \sum_{j=1}^{n} a_{1j}b_{j2} & \cdots & \sum_{j=1}^{n} a_{1j}b_{jp} \\ \sum_{j=1}^{n} a_{2j}b_{j1} & \sum_{j=1}^{n} a_{2j}b_{j2} & \cdots & \sum_{j=1}^{n} a_{2j}b_{jp} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^{n} a_{mj}b_{j1} & \sum_{j=1}^{n} a_{mj}b_{j2} & \cdots & \sum_{j=1}^{n} a_{mj}b_{jp} \end{bmatrix}$$

INPUT

The input file has one line of the following format for each non-zero element m_{ij} of a matrix M:
<M><i><j><m_ij>

Suppose

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \end{bmatrix}$$

The input file that represents A and B has the following lines:

A,0,1,1.0
A,0,2,2.0
A,0,3,3.0
A,0,4,4.0
A,1,0,5.0
A,1,1,6.0
A,1,2,7.0
A,1,3,8.0
A,1,4,9.0
B,0,1,1.0
B,0,2,2.0
B,1,0,3.0
B,1,1,4.0
B,1,2,5.0
B,2,0,6.0
B,2,1,7.0
B,2,2,8.0
B,3,0,9.0
B,3,1,10.0
B,3,2,11.0
B,4,0,12.0
B,4,1,13.0
B,4,2,14.0

The output file has one line of the following format for each non-zero element m_{ij} of a matrix M:
<i><j><m_ij>

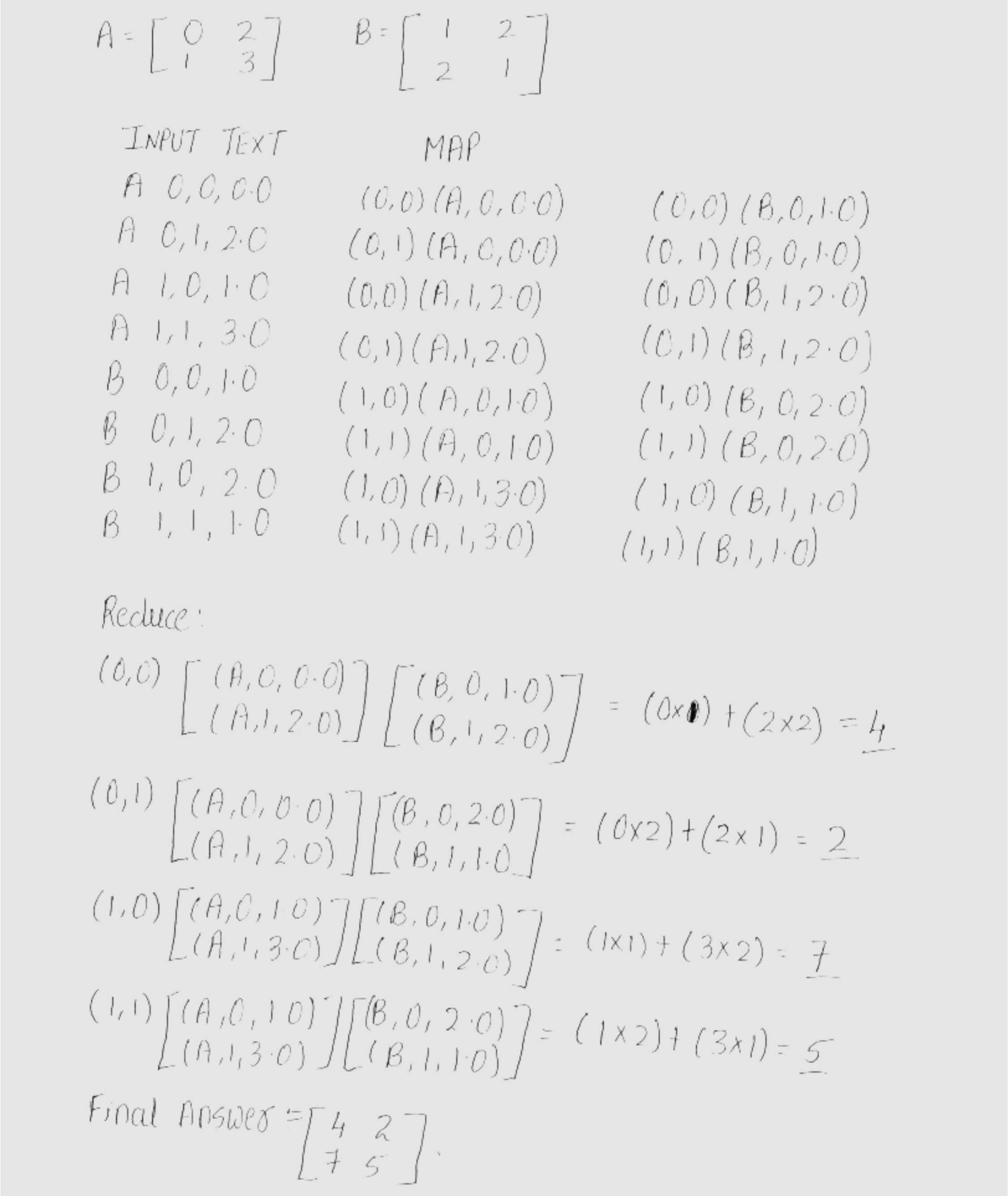
In our example, the output file that represents AB should have the following lines:

0,0,90.0
0,1,100.0
0,2,110.0
1,0,240.0
1,1,275.0
1,2,310.0

ALGORITHM:

```
map(key, value):
    // value is ("A", i, j, a_ij) or ("B", j, k, b_jk)
    if value[0] == "A":
        i = value[1]
        j = value[2]
        a_ij = value[3]
        for k = 1 to p:
            emit((i, k), (A, j, a_ij))
    else:
        j = value[1]
        k = value[2]
        b_jk = value[3]
        for i = 1 to m:
            emit((i, k), (B, j, b_jk))

reduce(key, values):
    // key is (i, k)
    // values is a list of ("A", j, a_ij) and ("B", j, b_jk)
    hash_A = {j: a_ij for (x, j, a_ij) in values if x == A}
    hash_B = {j: b_jk for (x, j, b_jk) in values if x == B}
    result = 0
    for j = 1 to n:
        result += hash_A[j] * hash_B[j]
    emit(key, result)
```



$A = \begin{bmatrix} 0 & 2 \\ 1 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$

INPUT TEXT	MAP	
A 0,0,0·0	(0,0) (A,0,0·0)	(0,0) (B,0,1·0)
A 0,1,2·0	(0,1) (A,0,0·0)	(0,1) (B,0,1·0)
A 1,0,1·0	(0,0) (A,1,2·0)	(0,0) (B,1,2·0)
A 1,1,3·0	(0,1) (A,1,2·0)	(0,1) (B,1,2·0)
B 0,0,1·0	(1,0) (A,0,1·0)	(1,0) (B,0,2·0)
B 0,1,2·0	(1,1) (A,0,1·0)	(1,1) (B,0,2·0)
B 1,0,2·0	(1,0) (A,1,3·0)	(1,0) (B,1,1·0)
B 1,1,1·0	(1,1) (A,1,3·0)	(1,1) (B,1,1·0)

Reduce:

$(0,0) \begin{bmatrix} (A,0,0\cdot0) \\ (A,1,2\cdot0) \end{bmatrix} \begin{bmatrix} (B,0,1\cdot0) \\ (B,1,2\cdot0) \end{bmatrix} = (0 \times [illegible]) + (2 \times 2) = \underline{4}$

$(0,1) \begin{bmatrix} (A,0,0\cdot0) \\ (A,1,2\cdot0) \end{bmatrix} \begin{bmatrix} (B,0,2\cdot0) \\ (B,1,1\cdot0 \end{bmatrix} = (0 \times 2) + (2 \times 1) = \underline{2}$

$(1,0) \begin{bmatrix} (A,0,1\cdot0) \\ (A,1,3\cdot0) \end{bmatrix} \begin{bmatrix} (B,0,1\cdot0) \\ (B,1,2\cdot0) \end{bmatrix} = (1 \times 1) + (3 \times 2) = \underline{7}$

$(1,1) \begin{bmatrix} (A,0,1\cdot0) \\ (A,1,3\cdot0) \end{bmatrix} \begin{bmatrix} (B,0,2\cdot0) \\ (B,1,1\cdot0) \end{bmatrix} = (1 \times 2) + (3 \times 1) = \underline{5}$

Final Answer $= \begin{bmatrix} 4 & 2 \\ 7 & 5 \end{bmatrix}$.

CONCLUSION:

Thus, Map-Reduce technique is used to perform Matrix Multiplication. Map function reads the input file and generates the intermediate key-value pairs of matrix. Reduce function multiplies and generates final result in key-value pairs which can be viewed in HDFS.

Program formation/ Execution/ ethical practices (06)	**Timely Submission and Documentation (02)**	**Viva Answer (02)**	**Experiment Marks (10)**	**Teacher Signature with date**

MATRIX MULTIPLICATION:-

```
package package1;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Matrixmulti{

    public static class Map extends Mapper<LongWritable, Text, Text, Text> {
        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            Configuration conf = context.getConfiguration();
            int m = Integer.parseInt(conf.get("m"));
            int p = Integer.parseInt(conf.get("p"));
            String line = value.toString();
            String[] indicesAndValue = line.split(",");
            Text outputKey = new Text();
            Text outputValue = new Text();
            if (indicesAndValue[0].equals("A")) {
                for (int k = 0; k < p; k++) {
                    outputKey.set(indicesAndValue[1] + "," + k);
                    outputValue.set("A," + indicesAndValue[2] + "," +
indicesAndValue[3]);
                    context.write(outputKey, outputValue);
                }
            } else {
                for (int i = 0; i < m; i++) {
                    outputKey.set(i + "," + indicesAndValue[2]);
                    outputValue.set("B," + indicesAndValue[1] + "," +
indicesAndValue[3]);
                    context.write(outputKey, outputValue);
                }
            }
        }
    }

   public static class Reduce extends Reducer<Text, Text, Text, Text> {
        public void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException {
            String[] value;
            HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();
            HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
            for (Text val : values) {
                value = val.toString().split(",");
                if (value[0].equals("A")) {
                    hashA.put(Integer.parseInt(value[1]),
Float.parseFloat(value[2]));
                } else {
                    hashB.put(Integer.parseInt(value[1]),
Float.parseFloat(value[2]));
                }
            }
            int n = Integer.parseInt(context.getConfiguration().get("n"));
            float result = 0.0f;
            float a_ij;
            float b_jk;
            for (int j = 0; j < n; j++) {
                a_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
                b_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
                result += a_ij * b_jk;
            }
            if (result != 0.0f) {
                context.write(null, new Text(key.toString() + "," +
Float.toString(result)));
            }
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        // A is an m-by-n matrix; B is an n-by-p matrix.
        conf.set("m", "2");
        conf.set("n", "3");
        conf.set("p", "3");

        Job job = new Job(conf, "MatrixMatrixMultiplicationOneStep");
        job.setJarByClass(Matrixmulti.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

input file matrix.txt in HDFS:-

```
A,0,0,0.0
A,0,1,2.0
A,1,0,1.0
A,1,1,3.0
B,0,0,1.0
B,0,1,2.0
B,1,0,2.0
B,1,1,1.0
```

```
[training@localhost ~]$ hadoop jar mul.jar package1.Matrixmulti matrix_in
matrix_out

[training@localhost ~]$ hadoop fs -ls matrix_out
Found 3 items
-rw-r--r--   1 training supergroup          0 2016-02-25 00:19
/user/training/matrix_out/_SUCCESS
drwxr-xr-x   - training supergroup          0 2016-02-25 00:18
/user/training/matrix_out/_logs
-rw-r--r--   1 training supergroup         32 2016-02-25 00:19
/user/training/matrix_out/part-r-00000


[training@localhost ~]$ hadoop fs -cat matrix_out/part-r-00000
```

output file matrix_out in HDFS:-

```
0,0,4.0
0,1,2.0
1,0,7.0
1,1,5.0
```