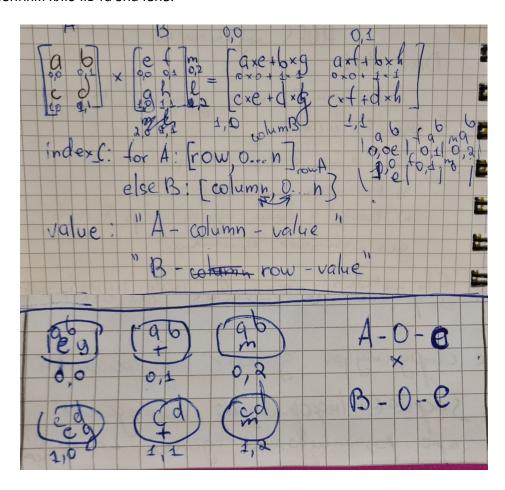
## Звіт до лабораторної роботи №3

Розмірковуючи над алгоритмом для реалізації множення матриць з використанням Hadoop MapReduce найважче було зрозуміти як правильно сформувати ключ-значення для reducer-a, щоб потім мати усі потрібні дані для виконання множення та сумування.

Але після того як намалював собі для візульного прикладу множення матриць, із зазначенням індексів (рядів-колонок). Знайшов доволі просту й помітну закономірність між значенням ключів та значень.



Далі потрібно було тільки імплементувати алгоритм, основні кроки:

1. Створити конфігурацію з вхідних аргументів:

```
// Set dimensions from args
String[] infoTupleA = args[1].split(argsDimensionSeparator);
rowA = Integer.parseInt(infoTupleA[1]);
columnA = Integer.parseInt(infoTupleA[2]);
String[] infoTupleB = args[3].split(argsDimensionSeparator);
columnB = Integer.parseInt(infoTupleB[2]);

conf.setInt(rowAConf, rowA);
conf.setInt(columnAConf, columnA);
conf.setInt(columnBConf, columnB);
```

2. Створення ключа та значення в залежності яка матриця прийшла на вхід:

```
String[] positionAndValue = value.toString().split(positionValueSeparator);
int val = Integer.parseInt(positionAndValue[1]);
String[] rowAndColumn = positionAndValue[0].split(rowColumnSeparator);
int row = Integer.parseInt(rowAndColumn[0]);
int column = Integer.parseInt(rowAndColumn[1]);

if (matrix.equals(firstMatrix)) {
    for (int k = 0; k < columnB; k++) {
        mapKey.set(row + rowColumnSeparator + k);
        mapValue.set(firstMatrix + valueSeparator + column + valueSeparator + val);
        context.write(mapKey, mapValue);
    }
} else {
    for (int i = 0; i < rowA; i++) {
        mapKey.set(i + rowColumnSeparator + column);
        mapValue.set(secondMatrix + valueSeparator + row + valueSeparator + val);
        context.write(mapKey, mapValue);
    }
}</pre>
```

3. Розбір значення в редюсері, множення та сумування для знаходження Rik.

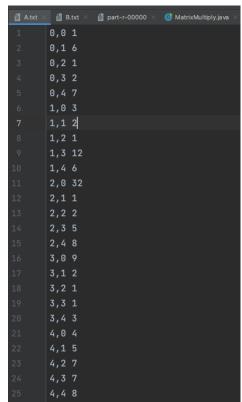
```
int sum = 0;
int[] A = new int[columnA];
int[] B = new int[columnA];

for (Text val : values) {
    String[] MatrixIndexValue = val.toString().split(valueSeparator);
    if (MatrixIndexValue[0].equals(firstMatrix)) {
        A[Integer.parseInt(MatrixIndexValue[1])] = Integer.parseInt(MatrixIndexValue[2]);
    } else
        B[Integer.parseInt(MatrixIndexValue[1])] = Integer.parseInt(MatrixIndexValue[2]);
}

for (int j = 0; j < columnA; j++) {
        sum += A[j] * B[j];
}
context.write(key, new Text(Integer.toString(sum)));
sum = 0;</pre>
```

Приклад виконання програми з вхідними параметрами: ./input/A.txt A-5-5 ./input/B.txt B-5-5 ./output

Приклад файлу з матрицею А:



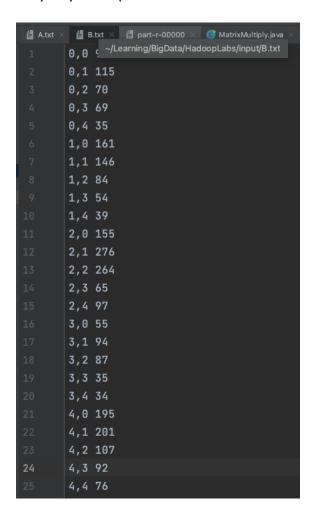
Приклад файлу з матрицею В:

```
0,01
0,15
0,27
0,31
0,42
1,0 3
1,1 5
1,2 7
1,3 9
2,08
2,17
2,2 1
2,33
2,45
3,08
3,15
3,23
3,3 2
4,08
4,1 9
4,22
4,31
4,42
```

Очікуваний результат після множення матриць:

$$\begin{pmatrix}
1 & 6 & 1 & 2 & 7 \\
3 & 2 & 1 & 12 & 6 \\
32 & 1 & 2 & 5 & 8 \\
9 & 2 & 1 & 1 & 3 \\
4 & 5 & 7 & 7 & 8
\end{pmatrix}
\cdot
\begin{pmatrix}
1 & 5 & 7 & 1 & 2 \\
3 & 5 & 7 & 9 & 2 \\
8 & 7 & 1 & 3 & 5 \\
8 & 5 & 3 & 2 & 1 \\
8 & 9 & 2 & 1 & 2
\end{pmatrix}
=
\begin{pmatrix}
99 & 115 & 70 & 69 & 35 \\
161 & 146 & 84 & 54 & 39 \\
155 & 276 & 264 & 65 & 97 \\
55 & 94 & 87 & 35 & 34 \\
195 & 201 & 107 & 92 & 76
\end{pmatrix}$$

## Результуючий файл:



## Приклад логів:

```
22/11/29 01:24:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable 22/11/29 01:24:00 WARN mapred.JobClient: No job jar file set. User classes may not be found. See JobConf(Class) or JobConf#setJar(String).
22/11/29 01:24:00 INFO input.FileInputFormat: Total input paths to process : 2
22/11/29 01:24:00 WARN snappy.LoadSnappy: Snappy native library not loaded
22/11/29 01:24:00 INFO mapred.LocalJobRunner: Waiting for map tasks
22/11/29 01:24:00 INFO mapred.LocalJobRunner: Starting task: attempt_local255255534_0001_m_0000000_0
22/11/29 01:24:00 INFO mapred.MapTask: record buffer = 262144/32768
22/11/29 01:24:00 INFO mapred.MapTask: Starting flush of map output
22/11/29 01:24:00 INFO mapred.Task: Task 'attempt_local255255534_0001_m_000001_0' done.  
22/11/29 01:24:00 INFO mapred.LocalJobRunner: Finishing task: attempt_local255255534_0001_m_000001_0
22/11/29 01:24:01 INFO mapred.JobClient:
                                                                   Reduce input records=250
                                                                   Map input records=50
22/11/29 01:24:01 INFO mapred.JobClient:
                                                                   Reduce shuffle bytes=0
                                                                   Map output records=250
22/11/29 01:24:01 INFO mapred.JobClient:
                                                                   Combine input records=0
22/11/29 01:24:01 INFO mapred.JobClient:
22/11/29 01:24:01 INFO mapred.JobClient:
                                                                   Bytes Written=196
Process finished with exit code 0
```