Matrix Multiplication

1. Overall

   這次的 Map Reduce 是處理矩陣相乘，我的作法分成兩部分，分別是
   - Mapper
   - Reducer

   而以下將會詳細說明
2. Mapper

   MatrixMulMapper

   Input: LongWritable, Text

   Output: Text, Text

	Key		Value	
Input	Type	Format	Type	Format
	LongWritable		Text	M,i,j,M_{ij} or N,j,k,N_{jk}
Output	Type	Format	Type	Format
	Text	Formatted(i,k)	Text	M,j,M_{ij} or N,j,N_{jk} (M,N is a Char)

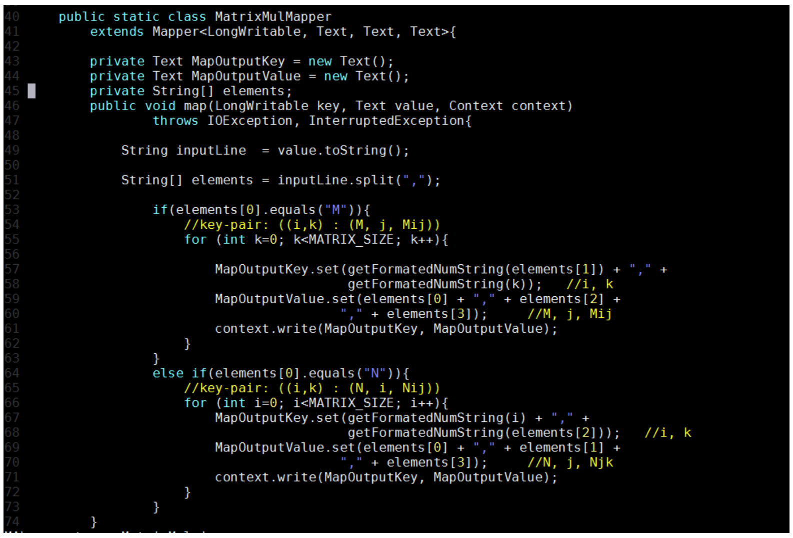
   Because the mapper outputs sorted by key, and we use Text(like String) as key,

   So we need to format the key value to get the right order:

   Format: convert the number to formatted String.

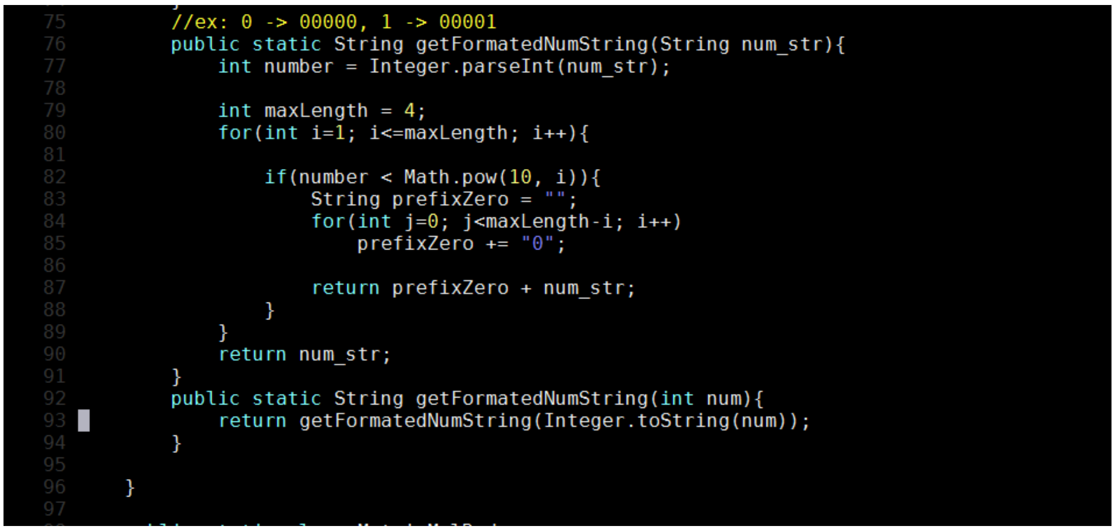
   EX: 1 => “0001”, 200 => “0200”, 8999 => “8999”

   Formatted(100,200)=>”0100,0200”



```java
    public static class MatrixMulMapper
        extends Mapper<LongWritable, Text, Text, Text>{

        private Text MapOutputKey = new Text();
        private Text MapOutputValue = new Text();
        private String[] elements;
        public void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException{

            String inputLine  = value.toString();

            String[] elements = inputLine.split(",");

                if(elements[0].equals("M")){
                    //key-pair: ((i,k) : (M, j, Mij))
                    for (int k=0; k<MATRIX_SIZE; k++){

                        MapOutputKey.set(getFormatedNumString(elements[1]) + "," +
                                         getFormatedNumString(k));   //i, k
                        MapOutputValue.set(elements[0] + "," + elements[2] +
                                        "," + elements[3]);     //M, j, Mij
                        context.write(MapOutputKey, MapOutputValue);
                    }
                }
                else if(elements[0].equals("N")){
                    //key-pair: ((i,k) : (N, i, Nij))
                    for (int i=0; i<MATRIX_SIZE; i++){
                        MapOutputKey.set(getFormatedNumString(i) + "," +
                                         getFormatedNumString(elements[2]));   //i, k
                        MapOutputValue.set(elements[0] + "," + elements[1] +
                                        "," + elements[3]);     //N, j, Njk
                        context.write(MapOutputKey, MapOutputValue);
                    }
                }
        }
```

Format Code:



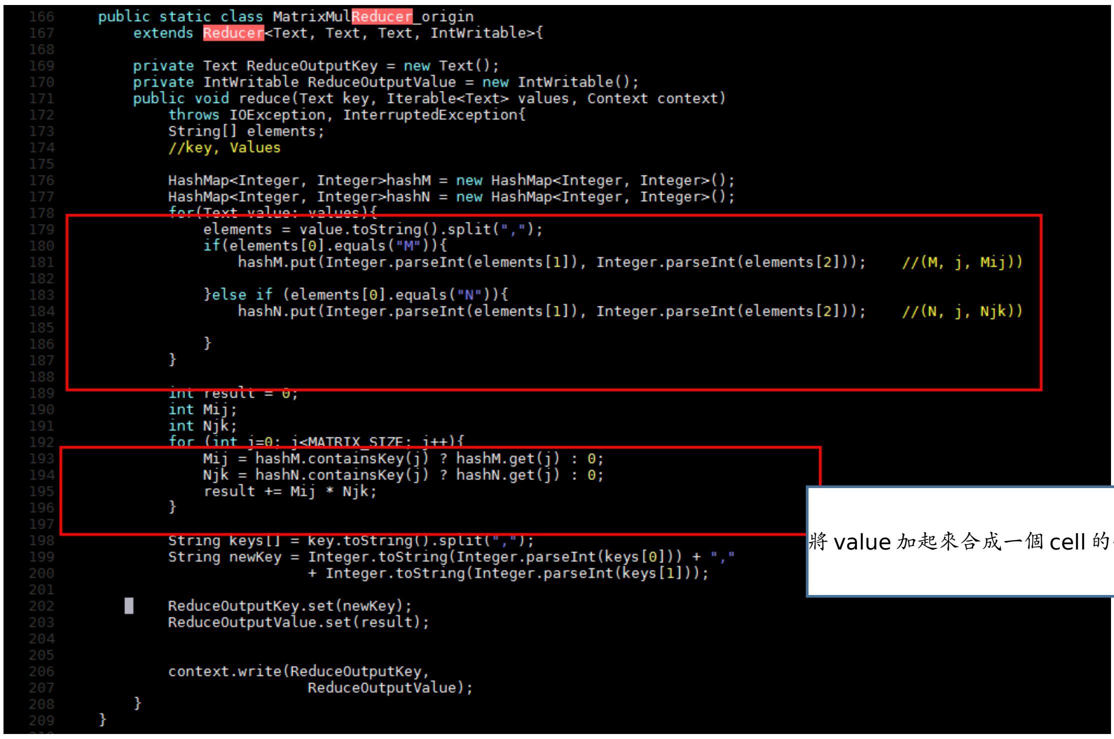
```java
        //ex: 0 -> 00000, 1 -> 00001
        public static String getFormatedNumString(String num_str){
            int number = Integer.parseInt(num_str);

            int maxLength = 4;
            for(int i=1; i<=maxLength; i++){

                if(number < Math.pow(10, i)){
                    String prefixZero = "";
                    for(int j=0; j<maxLength-i; i++)
                        prefixZero += "0";

                    return prefixZero + num_str;
                }
            }
            return num_str;
        }
        public static String getFormatedNumString(int num){
            return getFormatedNumString(Integer.toString(num));
        }

    }
```

In the mapper, when I get the parameter , I split it to the format of output key value pair.
Output Key: the ij of the answer Matrix
Output Value: the info of the cell which in M or N

3. Reducer
   MatrixMulReducer
   Input: Text, Text
   Output: Text, IntWritable

	Key		Value	
Input	Type	Format	Type	Format
	Text	Formatted(i,k)	Text	M,j,M_{ij} or

				N,j,N_{jk} (M,N is a Char)
Output	Type	Format	Type	Format
	Text	i,k	IntWritable	$\Sigma_{j=0\sim500} M_{ij} * N_{jk}$

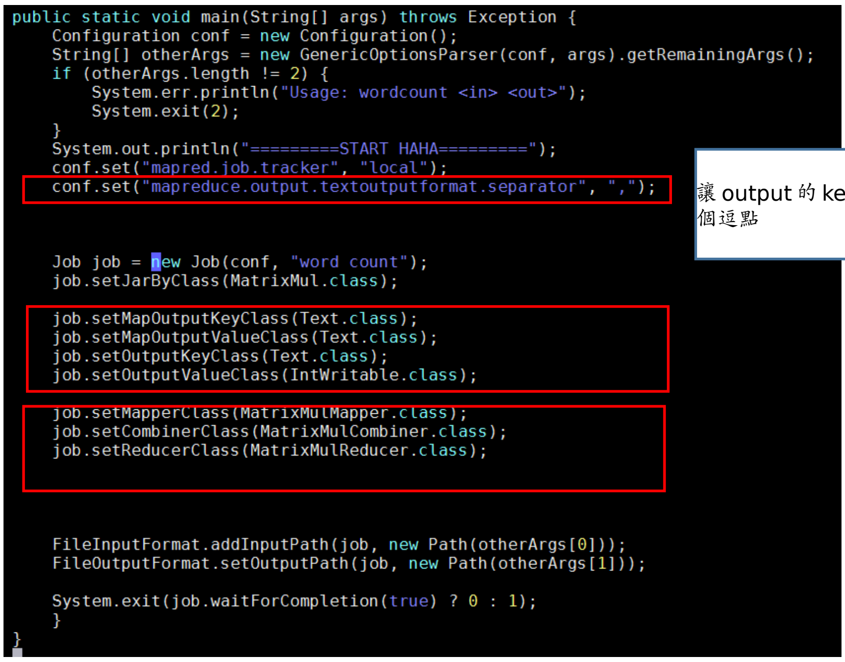


```java
    public static class MatrixMulReducer_origin
        extends Reducer<Text, Text, Text, IntWritable>{

        private Text ReduceOutputKey = new Text();
        private IntWritable ReduceOutputValue = new IntWritable();
        public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException{
            String[] elements;
            //key, Values

            HashMap<Integer, Integer>hashM = new HashMap<Integer, Integer>();
            HashMap<Integer, Integer>hashN = new HashMap<Integer, Integer>();
            for(Text value: values){
                elements = value.toString().split(",");
                if(elements[0].equals("M")){
                    hashM.put(Integer.parseInt(elements[1]), Integer.parseInt(elements[2]));    //(M, j, Mij))

                }else if (elements[0].equals("N")){
                    hashN.put(Integer.parseInt(elements[1]), Integer.parseInt(elements[2]));    //(N, j, Njk))

                }
            }

            int result = 0;
            int Mij;
            int Njk;
            for (int j=0; j<MATRIX_SIZE; j++){
                Mij = hashM.containsKey(j) ? hashM.get(j) : 0;
                Njk = hashN.containsKey(j) ? hashN.get(j) : 0;
                result += Mij * Njk;
            }

            String keys[] = key.toString().split(",");
            String newKey = Integer.toString(Integer.parseInt(keys[0])) + ","
                    + Integer.toString(Integer.parseInt(keys[1]));

            ReduceOutputKey.set(newKey);
            ReduceOutputValue.set(result);


            context.write(ReduceOutputKey,
                    ReduceOutputValue);
        }
    }
```

將 value 加起來合成一個 cell 的值

In the Reducer, I handle the output of the Mapper,
Then because elements in the Iterable<Text> has the same key means they can
Produce one cell of the answer Matrix,
So we can output a value of a cell of the answer Matrix

4. MapReduce Job

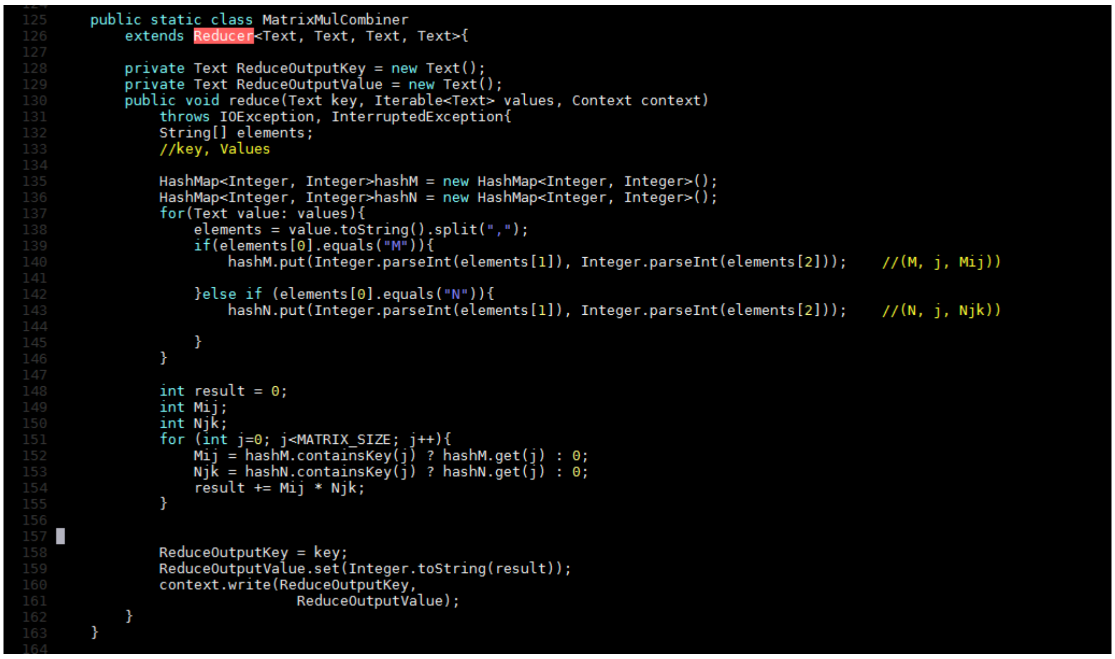


```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: wordcount <in> <out>");
        System.exit(2);
    }
    System.out.println("=========START HAHA=========");
    conf.set("mapred.job.tracker", "local");
    conf.set("mapreduce.output.textoutputformat.separator", ",");


    Job job = new Job(conf, "word count");
    job.setJarByClass(MatrixMul.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(MatrixMulMapper.class);
    job.setCombinerClass(MatrixMulCombiner.class);
    job.setReducerClass(MatrixMulReducer.class);



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

    System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

讓 output 的 key value 中間變隔一個逗點

5. 補充

原本有實作 Combiner，



```
    public static class MatrixMulCombiner
        extends Reducer<Text, Text, Text, Text>{

        private Text ReduceOutputKey = new Text();
        private Text ReduceOutputValue = new Text();
        public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException{
            String[] elements;
            //key, Values

            HashMap<Integer, Integer>hashM = new HashMap<Integer, Integer>();
            HashMap<Integer, Integer>hashN = new HashMap<Integer, Integer>();
            for(Text value: values){
                elements = value.toString().split(",");
                if(elements[0].equals("M")){
                    hashM.put(Integer.parseInt(elements[1]), Integer.parseInt(elements[2]));    //(M, j, Mij))

                }else if (elements[0].equals("N")){
                    hashN.put(Integer.parseInt(elements[1]), Integer.parseInt(elements[2]));    //(N, j, Njk))

                }
            }

            int result = 0;
            int Mij;
            int Njk;
            for (int j=0; j<MATRIX_SIZE; j++){
                Mij = hashM.containsKey(j) ? hashM.get(j) : 0;
                Njk = hashN.containsKey(j) ? hashN.get(j) : 0;
                result += Mij * Njk;
            }


            ReduceOutputKey = key;
            ReduceOutputValue.set(Integer.toString(result));
            context.write(ReduceOutputKey,
                            ReduceOutputValue);
        }
    }
```

而 Reducer 只要把 Combiner output 的值給全部加進來就好了，

不過在小測資(3*3)的效用不大，

而不知道爲什麼在跑大測資(500*500)時，

Output 中全部的值(value)都會歸 0....

故放棄不用。