HW1 Report: Matrix Multiplaication

- **Mapper: extend Mapper<LongWritable, Text, Text, Text>**

在這邊有兩個變數要先設定好，分別是 M 矩陣的列數和 N 矩陣行數，在程式中，我用 m 和 p 表示。
接著是設計 key 和 value。
對於每個 M 矩陣元素(M_{ij})，key 是 i,k (k = 0 … p-1)，value 則是”M, j, val”。
對於每個 N 矩陣元素(N_{ij})，key 則為 k,j (j = 0… m-1)。
由於最後一行有空行，所以在這邊前面 M 和 N 都要判斷。

為何要如此設計呢?
矩陣相乘中，以 M 矩陣第一列為例，他分別要和所有 N 矩陣中的所有行做相乘並相加，所以我必須要對 M_i 生成所有 N 行數相關的 pair。N 同理，對於每一行，都要和 M 矩陣的每列做 pair。但是在這邊要注意，這邊要反過來，有要義起運算的元素才會有一樣的 pair，到 Reduce 步驟時，就可以讓這些擁有相同 key 的元素們聚集在一起了。

但當然，也不能忘記紀錄矩陣來源，這邊用”M, j, val”儲存資訊，這樣一來到下一階段時，就可以輕鬆的用 split 把”,”分開。

- **Reducer: extend Reducer<Text, Text, Text, NullWritable>**

在這邊我生出兩個 Hashmap 資料結構 mapM 和 mapN，然後把 value 放進去。
在這邊，shuffle 完擁有相同 key 的已經 group 在一起了，所以接下來是要把被分到同群的所有 value 們做運算。
矩陣運算中 M_{ik} x N_{kj}，所以上一階段我們有儲存”j”在 value 中，現在我們可以使用他來當作我們 Hashmap 的 key。如此一來就可以把在 mapM, mapN 擁有相同 key 的東西取出來並做相乘，然後再用一個最後的變數把所有可能相加，這就會是最終的答案。

在這邊還有值得一提的點，因為輸出格式特殊(全部用”,”分開)，所以我用 NullWritable，在 context.write 時只有寫 key，並在 key 中包含 value 然後用”,”隔開以達到助教要求。