MDA\_HW1 report

此次作業主要是參考WordCount這個官方範例的架構去做修改，並且使用1 STEP的MapReduce實作 Martix相乘 M\*N，另外再main function中可以調整矩陣大小，分別為”m” “j” “k”，因為這次input是500X500，所以m j k預設都為500，再由configuration在其他function中取得。

Mapper：

收先取得M矩陣的rows的數量(input file的大小是500)，以及N矩陣的cols的數量(input file的大小是500)，用Configuration這個Hadoop 提供的object去儲存(code裡儲存了M的rows N的rows cols，因為M的cols數一定和N的rows數相同，所以只需用一個變數儲存)， 使用String得split將一行text作分割得到(mapAndreduce)，並且能從第一個字串去判斷是矩陣M還是N，接著M matrix照著書上公式((i,k)(‘M’,j,m\_ij))的key value pair寫入，用一個for loop跑到N的cols數量，由上述公式分別填入Key(mapAndreduce[1]+","+Integer.toString(i)))

Value( mapAndreduce[0]+","+mapAndreduce[2]+","+mapAndreduce[3])

而 N marix的公式則是((i,k),(‘N’,j,n\_jk))的key value pair 寫入，用一個for loop跑到M的rows數量，由上述公式分別填入

Key(Integer.toString(i)+","+mapAndreduce[2]))

Value( mapAndreduce[0]+","+mapAndreduce[1]+","+mapAndreduce[3])

如此一來就完成了Map的部分

Reducer：

由於這裡不須寫出Key的值，所以我在extend class時，第三個參數就給了NullWritable，否則你在context.write時，如果填入null會發生error。我在這裡創了兩個HashMap m與n(方便操作 有大量的build in function以及尋找資料的time complexity 小於 Linked list)用來儲存傳入的value中的相互對應，剛剛的value中有三個數值，分別代表，該pair屬於哪一個matrix，以及用來辨認相乘的代號也就是m\_ij\*n\_jk中的j，最後一個是該點的數值，如果是屬於M矩陣，存入m中，否則則存入n中。

接著處理M-row 與 N-col相乘的運算，這裡需要取得N的rows數量，一樣用Configuration取得，使用for loop跑到N的rows數量，使用HashMap方便的function，先判斷有沒有存在用containsKey()，再來用get()取得該點的數值，result = result + m\_ij\*n\_jk;，直到跑出for loop就能得到key 這個位置上的value