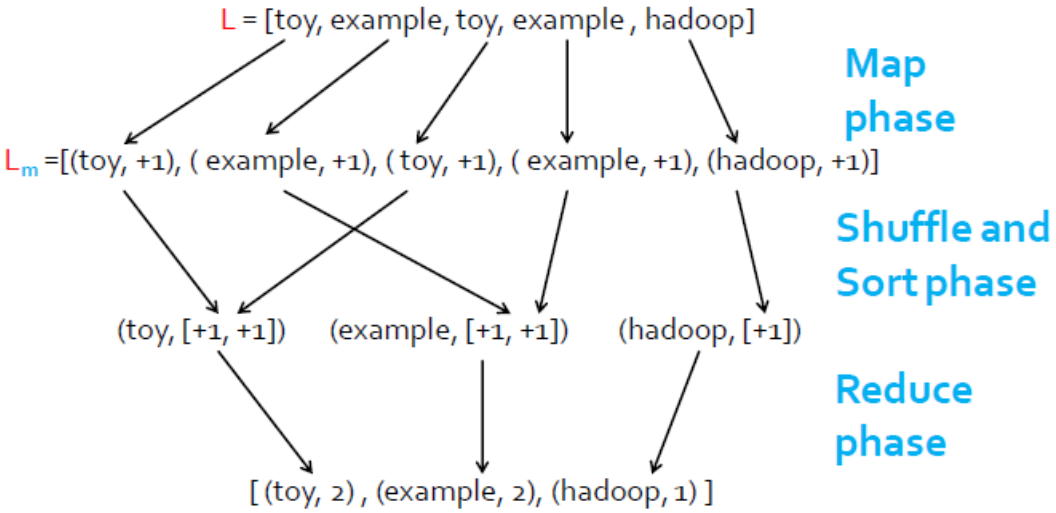
CORRECTION TD N°2

Algorithmique avec MapReduce

Exercice 1 : Comptage de mots

//Input file: a textual document with one word per line
//The map function is invoked over each word of the input file
map(key, value):
// key: document name; value: text of the document
for each word w in value:
emit(w, 1)
reduce(key, values):
// key: a word; values: a list of integers over counts
result = 0
for each count v in values:
result += v
emit(key, result)

Un exemple



Exercice 2 : Multiplication de deux matrices

Algorithm 1 Matrix Vector Multiplication on MapReduce

```
1: function MAP(< i, j, a_ij >)
2:     Emit(i, a_ij v[j])
3: end function
4: function REDUCE(key,values)
5:     ret ← 0
6:     for val ∈ values do
7:         ret ← ret + val
8:     end for
9:     Emit(key, ret)
10: end function
```

Algorithm 1: The Map Function

1 **for** *each element* m_{ij} *of* M **do**
2 produce $(key, value)$ pairs as $((i,k),(M,j,m_{ij}))$, for $k = 1,2,3,..$ up to the number of columns of N
3 **for** *each element* n_{jk} *of* N **do**
4 produce $(key, value)$ pairs as $((i,k),(N,j,n_{jk}))$, for $i = 1,2,3,...$ up to the number of rows of M
5 **return** *Set of (key, value) pairs that each key,* (i,k)*, has a list with values* (M,j,m_{ij}) *and* (N,j,n_{jk}) *for all possible values of* j

Algorithm 2: The Reduce Function

1 **for** *each key (i,k)* **do**
2 sort values begin with M by j in $list_M$
3 sort values begin with N by j in $list_N$
4 multiply m_{ij} and n_{jk} for j_{th} value of each list
5 sum up $m_{ij} * n_{jk}$
6 **return** $(i,k), \sum_{j=1} m_{ij} * n_{jk}$

Exercice 3 : ventes dans un magasin

/ / La fonction map lit l'agrégat order et en extrait les données que l'on souhaite
/ / agréger à savoir la liste des quantités et des prix des produits vendus .
map (int orderID, Order order)
List intermediateKeyValueList ;

```
for each item in order
intermediateKey := item.getItemName();
value := item.getPriceAndQuantity( );
intermediateKeyValueList.append (intermediateKey , value ) ;
return {intermediateKeyValueList} ;

/ / La fonction reduce consomme une liste de valeurs intermédiaires associées à une
/ / même clé, c-à-d à un même article, et calcule la somme des prix et des quantités
// de cette liste intermédiaire
reduce (int intermediateKey, List intermediateListOfValues)
totalPrice := 0 ;
totalQuantity := 0 ;
for each value in intermediateListOfValues
totalPrice := totalPrice + value . getPrice( ) ;
totalNumber := totalNumber + value . getQuantity ( ) ;
return {totalNumber , totalQuantity } ;
```