**MapReduce Exercises**

Mining of Massive Datasets book (exercises of section 2.3)

**Exercise 2.3.1**

Design map-reduce algorithms to take a very large file of integers and produce as output:

**(a) The largest integer.**

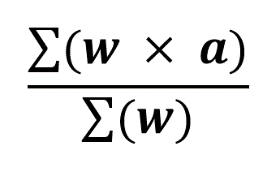
Map function: It takes input and returns (key= 1, value= max(value)) for each entry.

Reduce function: Map Function send all inputs with key=1 thus Reduce Function need to only return Max value with key=1.

**(b) The average of all the integers.**

Map function: It takes input and return (key=1, value= (w,a)), where w is weight and a is average.

Reduce function: Calculate weighted average on value list where the key is fixed:

[](https://github.com/E008001/Minnig-of-massive-datasets-Exercises/blob/master/f1.png)

**(c) The same set of integers, but with each integer appearing only once.**

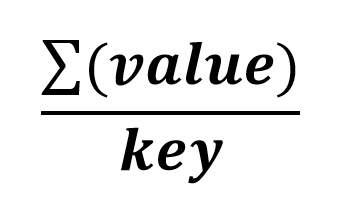
Map function: It takes input and returns (key= input, value= input) for each entry.

Reduce function: Key list is output.

**(d) The count of the number of distinct integers in the input.**

Map function: It takes input and returns (key= input, value= input) for each entry.

Reduce function: Number of distinct integers:

[](https://github.com/E008001/Minnig-of-massive-datasets-Exercises/blob/master/f2.png)

# Exercise 2.3.1 :

**Design MapReduce algorithms to take a very large file of integers and produce as output:  
(a) The largest integer.  
(b) The average of all the integers.  
(c) The same set of integers, but with each integer appearing only once.  
(d) The count of the number of distinct integers in the input.**

**solution:**  
(a)  
Map function : take inputs and return ( value = max(inputs value) , key = 1 ).  
Reduce function : return the max of key=1.

(b)  
Map function : take inputs and return ( inputs value , 1 ).  
Reduce function : take a list value of maptask with key=1 and return (average [list of value] , 1 ).

(c)  
Map function : take inputs and return ( input , input ).  
Reduce function : the list of keys is the desired output.

(d)  
Map function : take inputs and return (input , 1 ).  
Reduce function : the number of distinct integers is a count of the value with key=1

# Exercise 2.3.2 :

**Our formulation of matrix-vector multiplication assumed that the matrix M was square. Generalize the algorithm to the case where M is an r-by-c matrix for some number of rows r and columns c.**

**solution :**

assume that

* v must have c components
* v will fit into memory

Map function : take all matrix values as input and return ( MijVj , c).  
Reduce function : Each key corresponds to a component of the vector v whose value is just the sum of the values associated with each key.

# Exercise 2.3.4 :

**Selection can also be performed on bags. Give a MapReduce implementation that produces the proper number of copies of each tuple t that passes the selection condition. That is, produce key-value pairs from which the correct result of the selection can be obtained easily from the values.**

**solution:**  
Map function : return (t,1) for each t.  
Reduce function : take (t,[1,...,1]) and return (t,sum[1,...,1])

# Exercise 2.3.5 :

**The relational-algebra operation R(A,B) ⊲⊳ B<C S(C,D) produces all tuples (a, b, c, d) such that tuple (a, b) is in relation R, tuple (c, d) is in S, and b < c. Give a MapReduce implementation of this operation, assuming R and S are sets.**

**solution :**  
Map function : take ((a,b),1) and ((c,d),1) and return ((b,(R,a)),1) and ((c,(S,d)),1).  
Reduce function : take values of Maptask and for each pair that b<c return(a,b,c,d).

sajede

## Exercises for Section 2.3

### ****Exercise 2.3.1****

#### i:

#### map: For each tuple t, emit a key/value pair (t, t)

#### Reduce: Apply max to the list of all key

Emit the key/value pair (x,x), where x = max ([t] t in key value)

#### ii:

#### map: For each tuple t, emit a key/value pair (t, t)

#### Reduce: Apply Averag to the list of all key

Emit the key/value pair (x,x), where x = Averag([t] t in key value)

### ****Exercise 2.3.5****:

#### map: For a tuple (a,b) in R emit a key/value pair (b, (‘R’,a))

#### For a tuple (c,d) in S, emit a key/value pair(c, (‘R’,d))

#### Reduce: for each (b, (‘R’,a)) and (c, (‘R’,d)) if key b< key c then emit a key/value pair ((a,b),(c,d)) Otherwise, emit a key/value pair (t, NULL)

# Exerciese for Section 2.3 (Mining of Massive Datasets)

## Exercise 2.3.1 :

Design map-reduce algorithms to take a very large file of integers and produce as output:   
(a) The largest integer.   
(b) The average of all the integers.   
(c) The same set of integers, but with each integer appearing only once.   
(d) The count of the number of distinct integers in the input.

### solution (a)

Map : for each inteager i in the file , emit key-value pair(m,i)  
Reduce:input of this task is pair thet this key is m ,and associated value is lis[i1,...,in].that each ij is value of pair of output of map task.output of reduce task is pair(m,max{i1,...,in})  
notice that max is a associative and commutative function,we can produce only one pair (m,max{i1,...,in}) in map task.

### solution (b)

Map : for each inteager i in the file , emit key-value pair(m,i)  
Reduce:input of this task is pair thet this key is m,and associated value is list[i1,...,in] that each ij is value of pair of output of map task.output of reduce task is pair(1,averge{i1,..,in})

### solution (c)

Map : for each inteager i in the file , emit key-value pair(i,i)  
Reduce:input of this task is pair thet this key is i,and associated value is lis[i,..,i].that each i is value of pair of output of map task. output of reduce task is pair(i,i)   
it produces exactly one pair (i,i) for this key i.  
The second method: Map: for each inteager i in the file,emite key-value pair(i,1)  
Reduce: turn the value list into 1.  
note the result is obtain from the keys of the output.

### solution (d)

we sould use two map-reduse   
1: Map : for each inteager i in the file , emit key-value pair(i,i)  
Reduce:input of this task is pair thet this key is i,and associated value is lis[i,..,i].that each i is value of pair of output of map task. output of reduce task is pair(i,i)   
it produces exactly one pair (i,i) for this key i.   
2: Map : for each inteager i in the file , emit key-value pair(1,1)   
Reduce : input of this task is pair thet this key is 1 ,and associated value is lis[1,..,1].that each 1 is value of pair of output of map task.output of reduce task is pair(1,sum{1,...,1}).output of reduce task is number of distinct integers in the input.

## Exercise 2.3.2 :

Our formulation of matrix-vector multiplication assumed that the matrix M was square. Generalize the algorithm to the case where M is an r-by-c matrix for some number of rows r and columns c.

### solution

The matrix M and the vector v each will be stored in a file of the DFS. We assume that the row-column coordinates of each matrix element will be discoverable, either from its position in the file, or because it is stored with explicit coordinates, as a triple (i, j,mij). We also assume the position of element vj in the vector v will be discoverable in the analogous way.   
The Map Function: Each Map task will take the entire vector v and a chunk of the matrix M. From each matrix element mij it produces the key-value pair (i,mijvj). Thus, all terms of the sum that make up the component xi of the matrix-vector product will get the same key.   
The Reduce Function: A Reduce task has simply to sum all the values associated with a given key i. The result will be a pair (i, xi)  
If the Vector v Cannot Fit in Main Memory   
we can divide the matrix into vertical stripes of equal width and divide the vector into an equal number of horizontal stripes, of the same height. Our goal is to use enough stripes so that the portion of the vector in one stripe can fit conveniently into main memory at a compute node.  
The ith stripe of the matrix multiplies only components from the ith stripe of the vector. Thus, we can divide the matrix into one file for each stripe, and do the same for the vector. Each Map task is assigned a chunk from one of the stripes of the matrix and gets the entire corresponding stripe of the vector. The Map and Reduce tasks can then act exactly as was described above for the case where Map tasks get the entire vector.

## ! Exercise 2.3.3 :

In the form of relational algebra implemented in SQL, relations are not sets, but bags; that is, tuples are allowed to appear more than once. There are extended definitions of union, intersection, and difference for bags, which we shall define below. Write map-reduce algorithms for computing the following operations on bags R and S:   
(a) Bag Union, defined to be the bag of tuples in which tuple t appears the sum of the numbers of times it appears in R and S.  
(b) Bag Intersection, defined to be the bag of tuples in which tuple t appears the minimum of the numbers of times it appears in R and S.  
(c) Bag Difference, defined to be the bag of tuples in which the number of times a tuple t appears is equal to the number of times it appears in R minus the number of times it appears in S. A tuple that appears more times in S than in R does not appear in the difference.

### solution (a)

Map: for each tuple t in R and S emit key-value pair(t,t)   
Reduce: input: (t,t) output : (t,t)

### solution (b)

1: Map: for each tuple t in R emit key-value pair((t,R),1) and for each tuple t in S emit key-value pair((t,S),1)   
Reduce : input:((t,R),[1,..,1])or((t,S),[1,..,1]) output: ((t,R),sum[1,..,1]=m)or((t,S),sum[1,..,1]=n)  
2: Map:input: ((t,R),m) or ((t,S),n) output:(t,m)or(t,n)   
Reduce: input:(t,[m,n]) output:(t,min(m,n)=d)   
3: Map: input: (t,d) output:(t,d)   
Reduce: input:(t,d) output:produce d tuple(t,t).

### solution (c)

1: Map: for each tuple t in R emit key-value pair((t,R),1) and for each tuple t in S emit key-value pair((t,S),1)   
Reduce : input:((t,R),[1,..,1])or((t,S),[1,..,1]) output: ((t,R),sum[1,..,1]=m)or((t,S),sum[1,..,1]=n)  
2: Map:input: ((t,R),m) or ((t,S),n) output:(t,m)or(t,n)   
Reduce: input:(t,[m,n]) output:(t,m-n=d)   
3: Map: input: (t,d) output:(t,d)   
Reduce: input : (t,d) output : produce d tuple(t,t).(if d=<0,prodce nothing)

## ! Exercise 2.3.4:

Selection can also be performed on bags. Give a map-reduce implementation that produces the proper number of copies of each tuple t that passes the selection condition. That is, produce key-value pairs from which the correct result of the selection can be obtained easily from the values.

### solution

Map: for each tuple t stisfy C output:(t,1)  
Reduce: input:(t,[1,..,1]) output: (t,sum(1,..,1))

## Exercise 2.3.5 :

The relational-algebra operation R(A,B) ⊲⊳ B<C S(C,D) produces all tuples (a, b, c, d) such that tuple (a, b) is in relation R, tuple (c, d) is in S, and b < c. Give a map-reduce implementation of this operation, assuming R and S are sets.

### solution

Map: For each tuple (a, b) of R, produce the key-value pair (b, (R, a)). For each tuple (c,d) of S, produce the key-value pair (c,(S,d))   
Reduce: if b<c output: ((b,c),[(a1,b,c,d1),(a2,b,c,d2),..]) that is, (b,c) associated with the list of tuples that can be formed from an R-tuple and an S-tuple with condition b<c