24.03.2025 Week 6 exercises: MapReduce

Exercise 1:

You are given a symmetric social network (like Facebook) where a is a friend of b implies that b is also a friend of a. The input is a dataset D (sharded) containing such pairs of identifiers (a, b)—no assumption is made regarding the order between a and b. Pairs appear exactly once and are not repeated. Find the last names of those users whose first name is "Kanye" and who have at least 300 friends. You can chain MapReduces if you want (but only if you must, and even then, the shortest possible chain).

You don't need to write code—pseudocode is fine as long as it is understandable. Your pseudocode may assume the presence of appropriate primitives (e.g., "`firstname(user_id)`", etc.). The Map function takes as input a tuple `(key = a, value = b)`.

```
procedure MAP(a: User, b: User)
    if firstname(a) is "Kanye" then
        emit(a, b)
    if firstname(b) is "Kanye" then
        emit(b, a)
procedure REDUCE(u: User, friends: List[User])
    if length(friends) $\geq$ 300 then
        emit(lastname(u))
```

Exercise 2:

For an asymmetrical social network, you are given a dataset D where lines consist of (a, b) which means user a follows user b. Write a MapReduce program (Map and Reduce separately) that outputs the list of all users U who satisfy the following three conditions simultaneously: i) user U has at least 2 million followers, and ii) U follows fewer than 20 other users, and iii) all the users that U follows, also follow U back.

```
procedure MAP(a: User, b: User)
    emit($a, \langle b, 1\rangle$)                          ▷ a follows b
    emit($b, \langle a, 0\rangle$)                          ▷ b is followed by a
procedure REDUCE(u: User, list: List[$\langle$User, Int$\rangle$])
    follows ← $\emptyset$
    $count_0$ ← 0
    $count_1$ ← 0
    for all pair in list do              ▷ we assume pair has key and value attributes
        if pair.value = 0 then
            $count_0 \leftarrow count_0 + 1$                ▷ count of followers
        if pair.value = 1 then
            $count_1 \leftarrow count_1 + 1$                ▷ count of follows
            follows ← follows $\cup$ pair.key              ▷ set of follows
    if $count_0 \geq 2M$ and $count_1 < 20$ then
        for all user in follows do
            if $\langle$user, 0$\rangle \notin$ list then
                return
        emit(u)
```

Exercise 3 (Final Exam 2021):

Matrix multiplication is a fundamental operation in machine learning. Assume we now want to perform matrix multiplication over two extremely large matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

Design an $O(1)$-round Map-Reduce program for computing a matrix-matrix product $\boldsymbol{M} = \boldsymbol{AB}$, where $\boldsymbol{A} \in \mathbb{R}^{mxn}$ is an m x n matrix and $\boldsymbol{B} \in \mathbb{R}^{nxm}$.

Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are represented through mn pairs, and each pair corresponds to $(A, i, j, \boldsymbol{A}[i, j])$ or $(B, i, j, \boldsymbol{B}[i, j])$ (the first three entries are string while the last one is a floating point). The Map function reads each available pair as the input.

Recall how to perform matrix multiplication:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix}$$

```
1: procedure MAP($\langle X, i, j, X_{i,j} \rangle$: $\langle$String, String, String, Float$\rangle$)    ▷ Matrix element
2:     if $X$ = "A" then
3:         for $k$ in {0, 1, 2, ..., m-1} do
4:             emit($\langle i, k \rangle, \langle X, j, X_{ij} \rangle$)    ▷ Computation of $M[i, k]$ needs $A[i, j]$
5:     if $X$ = "B" then
6:         for $k$ in {0, 1, 2, ..., m-1} do
7:             emit($\langle k, j \rangle, \langle X, i, X_{ij} \rangle$)    ▷ Computation of $M[k, j]$ needs $B[i, j]$
8: procedure REDUCE($\langle i, j \rangle$: $\langle$String, String$\rangle$, list: List[$\langle$String, String, Float$\rangle$])
9:     Avec ← ∅
10:    Bvec ← ∅
11:    $M_{i,j} \leftarrow 0$
12:    for all $\langle X, k, X_k \rangle$ in list do    ▷ Separate out matrix elements
13:        if $X$ = "A" then
14:            Avec ← Avec ∪ $\langle k, X_k \rangle$
15:        if $X$ = "B" then
16:            Bvec ← Bvec ∪ $\langle k, X_k \rangle$
17:    sort(Avec)
18:    sort(Bvec)
19:    for $k$ in {0, 1, 2, ..., n-1} do
20:        $M_{i,j} \leftarrow M_{i,j}$ + Avec[$k$] ∗ Bvec[$k$]    ▷ Multiply corresponding elements
21:    emit($\langle M, i, j, M_{i,j} \rangle$)
```

Exercise 4 ([10 points] Final Exam 2022 – multiple choice: Only 1 correct answer. No negative marking.):

1) For each of the following questions, compute the total communication cost between the mappers and the reducers, i.e., the total number of (key, value) pairs outputted by the mappers. Use the optimal algorithm that runs in one MapReduce step for each of the tasks. Also, assume that there is no combiner.

(a) [2 points] Word count for a dataset comprising W total words with d distinct words. The mappers receive a single word as input.

( ) d

( ) W

The algorithm will emit one key-value pair per word.

( ) $\frac{W}{d}$

( ) dW

( ) $2W$

```
procedure MAP(word: String)
    emit(word, 1)
procedure REDUCE(word: String, counts: List[Integer])
    $total \leftarrow 0$
    for all $count$ in counts do
        $total \leftarrow total + count$
    emit(word, total)
```

(b) [3 points] Matrix multiplication of two matrices of size $m \times n$ and $n \times p$. The mappers read input tuples in the form `<Matrix identifier, row index, column index, value>`.

( ) mp

( ) $n(m+p)$

( ) $2n(m+p)$

( ) mnp

( ) $2mnp$

The final matrix will be $m \times p$. Each element (i, j) will require n elements of the first matrix and n elements of the second matrix.

```
procedure MAP(MatrixID: String, i: Integer, j: Integer, value: Float)
    if MatrixID = "A" then
        for $k \leftarrow 1$ to $p$ do
            emit($(i, k), (A, j, value)$)
    else
        for $k \leftarrow 1$ to $m$ do
            emit($(k, j), (B, i, value)$)
```

(c1) [2 points] Computing the INNER JOIN of two relations defined as follows: relation `R1(X, Y)` with four tuples {(5,21), (7,16), (15,3), (3,21)} and relation `R2(Y, Z)` with {(3,1), (4,8), (21,28)}. The mappers read input tuples in the form `<Relation identifier, column-1 value, column-2 value>`.

Recall: The database INNER JOIN operation of two relations outputs concatenated tuples from R1 and R2 having equal values on the common column.

( ) 12

( ) 7

The comparison has to be done in the reducer. The mapper will emit all the rows of both the relations with a tag.

( ) 4

( ) 3

( ) 2

(c2) [1 point] How many output tuples are produced by the reducer(s) in question (c1) ?

( ) 12

( ) 7

( ) 4

( ) 3

(5, 21, 28), (15, 3, 1), (3, 21, 28)

( ) 2

```
procedure Map(RelationID: String, X: Integer, Y: Integer)
    if RelationID = "R1" then
        emit(Y, (RelationID, X))
    else
        emit(X, (RelationID, Y))
procedure Reduce(Key: Integer, Values: List[Tuple])
    R1_list ← []
    R2_list ← []
    for all (RelationID, Value) in Values do
        if RelationID = "R1" then
            append R1_list with Value
        else
            append R2_list with Value
    for all v1 in R1_list do
        for all v2 in R2_list do
            emit((v1,Key,v2))
```

(d) [2 points] Difference of two sets X and Y containing a total of x and y elements respectively. The mappers read input tuples in the form `<Set identifier, Value>`.

Recall: The difference of two sets X and Y is a set that contains those elements of X that are NOT in Y.

( ) $x + y$

All the values have to be emitted once from both the sets.

( ) $x - y$

( ) x

( ) y

( ) xy

```
procedure MAP(SetId: String, Value: Integer)
    emit(Value, SetId)
procedure REDUCE(Value: Integer, SetIds: List[String])
    if "X" in SetIds and "Y" not in SetIds then
        emit(Value)
```