CS4225/CS5425 BIG DATA SYSTEMS FOR DATA SCIENCE

Tutorial 1: MapReduce

|  |  |
| --- | --- |
| **1.** | Suppose our input data to a map-reduce operation consists of integer values (the keys are not important). The map function takes an integer *i* and produces the list of pairs (*p*,*i*) such that *p* is a prime divisor of *i*. For example, map(12) = [(2,12), (3,12)].  The reduce function is addition. That is, reduce(*p*, [*i*1, *i*2, ...,*ik*]) is (*p*,*i*1+*i*2+...+*ik*).  Given that the input is the list [15, 21, 24, 30, 49], identify all the pairs in the output. |
| **2.** | We want to use map-reduce to compute the result of matrix-vector multiplication of the following matrix and vector:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | | 5 | 6 | 7 | 8 | | 9 | 10 | 11 | 12 | | 13 | 14 | 15 | 16 | | |  | | --- | | 1 | | 2 | | 3 | | 4 | |   Design a suitable map and reduce function, and identify all the pairs in the output.  (Note: your Map function only needs to take the matrix elements as input; the vector can be treated as fixed. Hint: For each matrix element, think about what computation needs to be done on it for computing matrix multiplication (for the “Map”). Then think about which of the resulting outputs need to be aggregated, to help you decide what the intermediate keys should be.) |
|  |  |
|  |  |

3. Consider a simple example: we have a large dataset where input keys are strings and input values are integers, and we wish to compute the mean of all integers associated with the same key (rounded down). A real-world example might be a large user log from a popular website, where keys represent user ids and values represent some measure of activity such as elapsed time for a particular session. A program Tommy has implemented the problem on MapReduce. He has written a few versions with the pseudo code shown in Figures 1—4.

a) Initially, Tommy has finished an implementation with Version 1 (Figure 1). He finds that the implementation can have correct results, but the performance is very slow. Why?

b) Tommy wants to improve the performance using combiner. He comes out the second implementation (Version 2 in Figure 2). He finds that he can seldom get the reasonable results. Why?

c) After careful design, Tommy finally develops an efficient and correct implementation (Version 3 in Figure 3). Analyze the correctness of the combiner and efficiency of the algorithm (i.e., why it is more efficient than Version 1).

d) Tommy analyzes the efficiency of Version 3, and comes out an even more efficient implementation (Version 4 in Figure 4). Why is Version 4 even more efficient than Version 3?

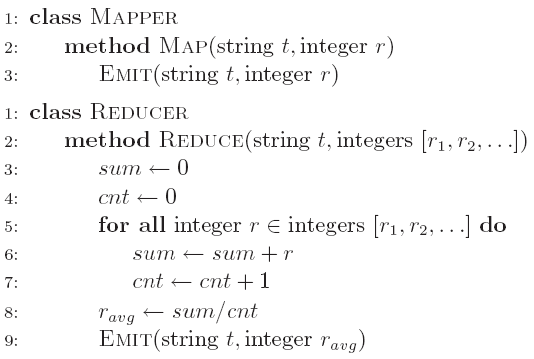


Figure 1. Computing the average: Version 1

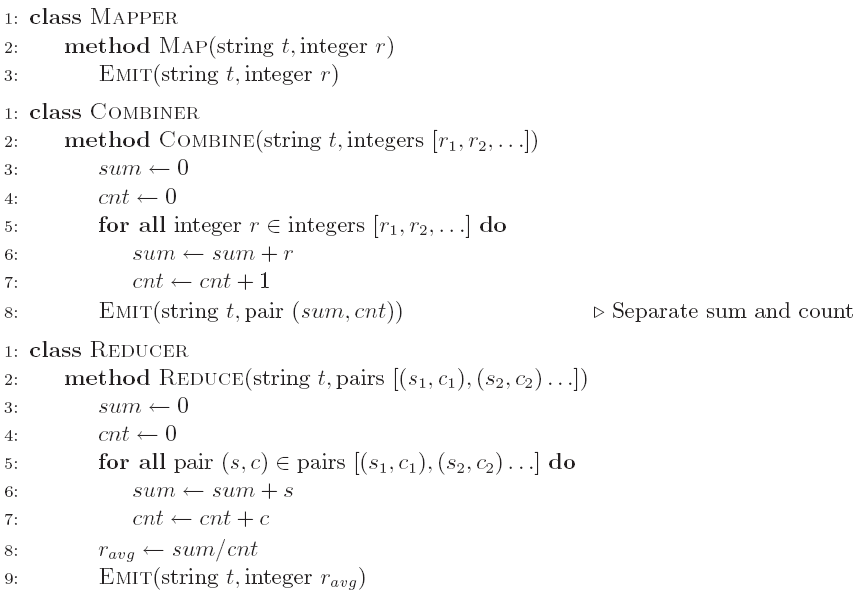


Figure 2. Computing the average: Version 2

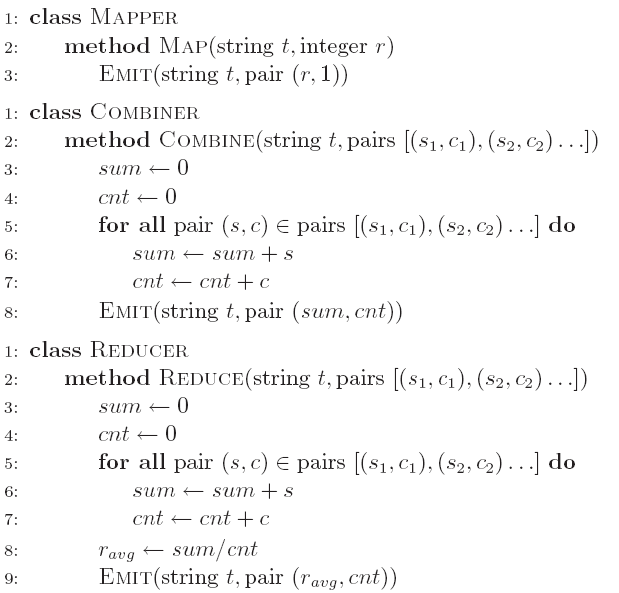


Figure 3. Computing the average: Version 3

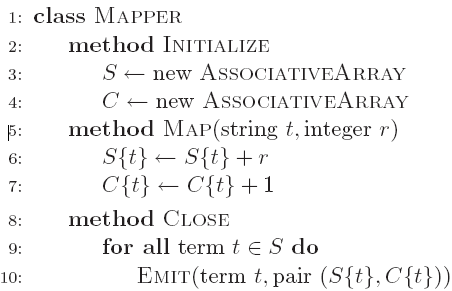


Figure 4. Computing the average: Version 4