Assignment 1

CS-UH-2218: Algorithmic Foundations of Data Science

Assignments are to be submitted in groups of at most three. *Upload the solutions on NYU classes as one PDF file for theoretical assignments and separate source code files for each programming assignment. Submit only one copy per group. Clearly mention the participant names on each file you submit.*

Problem 1 (3+3+4 = 10 points)**.** Suppose that we have a large social network G stored in a file. Each line in the file contains a pair (x, y) where x and y are names of people who are 'friends' [1]. Design map-reduce algorithms to produce the following:

(a) A file containing the names of all the people in the network along with the number of friends they have. We thus define the `map` and `reduce` functions as follows.

(b) Given a fixed number k, a file containing all pairs of people (x, y) who have at least k common friends in G. Note that x and y themselves may or may not be friends.

(c) A random sample of the people in the network where each distinct person is included independently with probability $p = 0.01$.

The algorithms you design need not be single-pass. For each pass of your algorithm, clearly define and explain the `map` and `reduce` functions.

Problem 2 (2 + 8 = 10 points)**.** Consider an input sequence of numbers $\langle a_1, a_2, \cdots, a_n\rangle$. We would like to compute $m_i = \min\{a_1, \cdots, a_i\}$ for each $i \in \{1, \cdots, n\}$ using the map-reduce framework. You can assume that the input is a set of key-value pairs $\{(1, a_1), (2, a_2), \cdots, (n, a_n)\}$ and the desired output is the set of key-value pairs $\{(1, m_1), (2, m_2), \cdots, (n, m_n)\}$. Assume that n is known in advance.

(a) Argue that in any one pass algorithm, the maximum reducer size is at least n.

(b) Design a **two pass** Map-Reduce algorithm so that in each pass the reducer size is $O(\sqrt{n})$. *First describe the overall idea and then give precise definitions of the* `map` *and* `reduce` *functions in each pass.*

Problem 3 ($2 \times 5 = 10$ points)**.** Suppose that we have n numbers stored in a file and we would like to find the median[2] of these numbers in a distributed computation framework like Spark using t machines with $O(n/t)$ parallel computation time. Consider the following algorithm for doing this:

- *Step 1:* For some positive number k, pick k of the input numbers so that each input number is picked independently with probability $p = k/n$. Let $x_1 \leq x_2 \leq \cdots \leq x_k$ be the set of numbers picked in sorted order. In addition, let $x_0 = -\infty$ and $x_{k+1} = +\infty$. Broadcast $\langle x_0, x_1 \cdots, x_{k+1}\rangle$ to all the machines.

[1] You may assume that friendship is symmetric and each pair of friends appear together in only one line in the file.

[2] If n is odd, the median is the middle number in sorted order of the numbers. If n is even either of the middle two numbers can be considered the median.

- *Step 2:* For each $i \in \{0, 1, \cdots, n\}$, let G_i be the subset of numbers that lie in the range $[x_i, x_{i+1})$. Compute $n_i = |G_i|$ for each $i \in \{0, 1, \cdots, k\}$.
- *Step 3:* From the n_i's figure out the index j such that the median is contained in group G_j and the rank r of the median in the sorted order of the numbers in group G_j.
- *Step 4:* Collect all elements in G_j on a single machine and compute the median from the above information.

Assuming that the input data is split equally among the t machines,

(a) How would you implement Step 1 so that expected network communication (amount of data sent over the network) is $O(kt)$.

(b) How would you compute the n_i's in Step 2 with only $O(kt)$ network communication. Note that explicitly forming the groups $G_0, \cdots, G_k$ may require communication proportional to n which can be much larger than kt.

(c) How would you compute j and r in Step 3?

(d) How would you implement Step 4? What is the expected network communication?

(e) How would you pick k (as a function of n and t) to minimize the overall expected network communication?

Informal but clear answers in plain English suffice. No need for very detailed explanations. You may make any reasonable assumptions necessary to answer the questions. Please state your assumptions clearly.

Problem 4 (20 points)**.** Download `ml-latest-small.zip` from `https://grouplens.org/datasets/movielens/latest/` and unzip it into a folder. Read the corresponding `README.html` file in the above website to understand the format of the files contained in the folder. The first line in each csv file is the header and the rest of lines contain data. You can remove the first line from each of the csv files to make programming easier.

Write a Spark program that computes and reports the following:

(a) The average number of users a movie is rated by.

(b) For each genre, the average rating of all movies in that genre.

(c) The names of the top three movies (by average user rating) in each genre.

(d) Top ten movie watchers ranked by the number of movies they have rated.

(e) Top ten pairs of users ranked by the number of movies they both have rated.

When considering the ratings for a movie, we only consider the users that have rated the movie. For instance, to compute the average rating for a movie we add all the ratings of the movie and divide by the number of users that have rated the movie.

Your program should assume that the csv files are located in the same directory as your program. In addition to the code you submit, write a ***short report*** *explaining what you did for each task and what output you obtained.*