Algorithms for Modern Data Models
MS&E 317/CS 263, Spr 2014-15, Stanford University
Instructor: Ashish Goel
Homework 1. Given 4/20/2015, due 4/29/2015, in class.

Collaboration policy: You can discuss problems and solutions as much as you like, but you can not copy any one else's answers, and you can not just memorize someone else's answers. You have to write down your answer in your own notation after understanding and reporoducing every aspect of your answer. You can not ask someone else to verify your written answer. If there is any lack of clarity in these instructions, please check with the instructor, and err on the side of caution.

Non-letter grade students: please do any two problems. If you do more, we will grade any two.

You can do the programming problem in groups of up to 3. Please look in the directory "single_node_shuffle" for details of the problem.

1. The edges of an undirected graph $G = (V, E)$ are given to you. Your goal is to create a sketch for each node so that given any set of nodes S, you can quickly estimate the number of nodes who have an edge to a node in S.

   (a) Write down the Map-Reduce algorithm for this sketch-creation.

   (b) Implement (in whichever high level language you like) the Map and Reduce functions, each as a stand-alone executable program that reads a sequence of records from the standard input and writes its output to the standard output. Use your implementation along with the simple map-reduce runner supplied by the instructor to produce this sketch for the supplied data file. Your mapper should accept the number of hash functions to use in the sketch as a command-line parameter.

   (c) Write a program to read the output of the above problem and repeatedly answer queries, where each query is a list of nodes and the answer is an estimator for the number of nodes adjacent to at least one node in the list.

   (d) Argue that your code is robust to repeated edges, to having an edge (u, v) listed as
   u v
   or
   v u
   or both, and to having nodes that are not adjacent to any edge.

   (e) Make sure the executables run on the Stanford corn machines. Submission instructions will be emailed separately.

2. In the **index representation**, a vector is specified as a list of elements of the form (i, v_i) where $1 \leq i$ refers to a dimension, and v_i is the i-th component of the vector. If a dimension is repeated, then the component of the vector in that dimension is the sum of the values for that dimension in the list. If a dimension is not present in the list, then the value of the vector in that dimension is 0.

   Give an algorithm to efficiently create a sketch $s(a)$ for a vector a given in an index representation, such that given the sketches $s(a)$, $s(b)$ of two vectors a, b, you can quickly estimate

the inner-product of the vectors a and b. Analyze the running time for creating the sketch, the size of the sketch, the time for producing an estimate of $a \cdot b$ given $s(a), s(b)$, and present an error bound for the estimate.

3. You are given a set of N sensors placed at integer positions $1, 2, \ldots N$ on a line. Sensor i observes a value $v(i,t)$ at time t. The sensors can pass messages to their neighbors infinitely fast, but passing and storing messages comes with some cost. Your goal is to create a randomized message-passing algorithm such that at any time t, any node i can draw a uniformly random sample from the values seen by nodes $i-d, \ldots, i$ during the last w time steps without passing any additional messages. Design an efficient algorithm for this problem, and analyze the expected number of messages passed by a node at every time step and the amount of storage space used at each node.

4. (a) Given a stream of values $a_0, a_1, \ldots, a_t$, your goal is to answer queries of the form "What is the median of the last w values?". Design a streaming algorithm for this problem and analyze the memory requirement and the accuracy.

   (b) The stream-sampling algorithm we have seen in this class suffers from the problem that while each query returns a uniformly random sample, different queries don't return independent samples. How would you address this problem in the above context? In other words, how can you tune your algorithm in (a) so that with high probability, you never return an estimate for the median which is outside the $[1/2-\delta, 1/2+\delta]$-quantile? You can assume that $t \leq T$ if you wish and express your algorithm and answers in terms of T.

5. Describe one problem (very briefly, eg. in 3-5 sentences) that you think would be important to discuss in this class.