# CS 591 Randomized Algorithms, HW2

Prof. Jared Saia, University of New Mexico

*Due: October 7th.*
*Note: Some of this homework is adapted from an assignment by Avrim Blum in his 1997 Fall Semester Randomized Algorithms class*

Consider a social network like Friendster or Orkut. Each node in such a network represents a person and there is a link from node $x$ to node $y$ if $y$ is a friend of $x$. One thing we might want to do in such a network is, for any node $x$, get the size of the set of people that $x$ can reach through "friendship" links. In this assignment, we'll work on the following abstract version of this problem. We're given a directed graph $G$, and we want to be able to quickly estimate the size of the set of vertices reachable from each vertex $v$ in $G$. The best exact algorithms for this problem take time $O(\min(mn, n^{2.38}))$ (where $m$ is the number of edges and $n$ is the number of nodes in $G$). In this homework, we'll get a randomized approximation algorithm for this problem which is much faster.

The algorithm is given below ($C$ is a fixed constant parameter which is described in problem 2.). For any vertex $v$, let $r(v)$ be the number of vertices $v$ can reach.

1. We assign to each node in the network $C \ln n$ *id's*. Each id is a random real number chosen independently from the interval $[0, 1]$.
2. For each vertex $v$ in $G$, we find the $C \ln n$ smallest IDs among all nodes reachable from $v$.
3. For each vertex $v$, let $\hat{m}_v$ be the largest of these $C \ln n$ smallest IDs reachable from $v$.
4. If $v$ has no out edges return 1 as our estimate of $r(v)$. Else return $1/\hat{m}_v$ as our estimate of $r(v)$.

**Problems:**



1. Give a deterministic $O(m \log n + n \log^2 n)$ time algorithm for performing step 2 of the algorithm.
2. Prove that for any fixed $\epsilon > 0$, we can choose $C$ depending only on $\epsilon$ so that with probability at least $1 - 1/n$, for all nodes $v$:

$$(1-\epsilon)r(v) \leq 1/\hat{m}_v \leq (1+\epsilon)r(v)$$

   (hint: For a fixed vertex $v$, consider the interval $[0, \frac{1}{(1-\epsilon)r(v)}]$. How many IDs do we expect to fall in that interval? How about in the interval $[0, \frac{1}{(1+\epsilon)r(v)}]$?)
3. Extra Credit: Consider the dynamic version of this problem. Three basic operations can occur: 1) an edge can be added to $G$ (possible with a new node as the sink of the edge), 2) an edge can be deleted from $G$, and 3) a query can be issued for an estimate of $r(v)$ for any node $v$. Describe an algorithm which can handle all of these operations efficiently. In particular, your algorithm should be able to handle reachability queries in time better than $O(m \log n + n \log^2 n)$. To achieve this, you will obviously need to spend more than constant time on operations 1) and 2).
   Challenge: Can you ensure that all three operations take expected time $o(m)$?
4. Problem 4.1
5. Problem 4.8
6. Problem 4.12
7. Problem 4.13