HW3 Randomized algorithms & Random structures

MPRI 1.24 Wed. Dec. 9, 2015 - **Due on Wed. Dec. 16, 2015**

You are asked to complete the exercise marked with a [★] and to send me your solutions at:
nicolas.schabanel@cnrs.fr
(or drop it in my mail box at the 4th floor of Sophie Germain) on **Wed. Dec. 16, 2015**.

■ **Exercise 1 (A streaming algorithm for counting the number of distinct values).** [★]
We are given a *stream* of numbers $x_1, \ldots, x_n \in [m]$ and we want to compute the number of distinct values in the stream: $F_0(x) = \#\{x_i : i \in [n]\}$. (Note that if $f_a(x) = \#\{i : x_i = a\}$, we can express $F_0(x) = \sum_{a=0}^{m-1} (f_a(x))^0$, as the zero-th moment of the frequencies of each element of $[m]$ in the stream). Let us denote by $S_x = \{x_i : i \in [n]\}$ the set of the values in the stream x. Note that $F_0(x) = \#S_x$. (We may drop the x when the context is clear.)

The *streaming* constraint is that the algorithm will see every x_i only once as it reads the stream from left to right and we want to minimize the memory needed by the algorithm to accomplish this task. One can show that any deterministic algorithm that approximates the value of F_0 within 10% requires at least $\Omega(n)$ bits of memory. Here, we will design a randomized algorithm that accomplish this task using only $O(\log n + \log m)$ bits of memory.

We start with an hypothetical algorithm using uniform real random numbers and a hypothetical family of hash functions and then see how to turn it into an effective algorithm.

Assume that we are given a random function $h : [m] \to (0, 1]$, i.e. such that for every $x \in [m]$, $h(x)$ is a (fixed) independent uniform random real in $(0, 1]$. The algorithm proceeds as follows: when reading the stream, record in memory the minimum value μ so far of the $h(x_i)$s, and output $1/\mu - 1$ at the end.

▶ ***Question* 1.1)** *Show that* $\Pr\{\mu \geqslant t\} = (1-t)^{F_0}$.

▶ ***Question* 1.2)** *Show that* $\mathbb{E}[\mu] = \frac{1}{F_0+1}$.

However, the following fact seems to imply that the algorithm is wrong.

▶ ***Question* 1.3)** *Show that* $\mathbb{E}[1/\mu] = \infty$.

But, fortunately:

▶ ***Question* 1.4)** *Compute* $\mathbb{V}\mathrm{ar}(\mu)$ *and show that* $\mathbb{V}\mathrm{ar}(\mu) \leqslant \mathbb{E}[\mu]^2$.

▶ ***Question* 1.5)** *Design and analyze a* (ε, δ)*-estimator for* F_0*. Still, what is the expected value of its output? Is there a paradox here?*
▷ Hint. *First, design an* (ε, δ)*-estimator for* μ*.*

Unfortunately, such a random function h requires storing m reals in memory. The key to reduce the memory needed is to relax the independence of the hash value to pairwise independence only. In the following, we will approximate the minimum of the hash keys by recording only the position of their first non-zero bit in their binary writing. We proceed as follows.

Let $\ell = \lceil \log_2 m \rceil$ such that $2^{\ell-1} < m \leqslant 2^\ell$ and consider the field with 2^ℓ elements $\mathbb{F}_{2^\ell}$. We identify $\mathbb{F}_{2^\ell}$ through canonical bijections to the set of bit-vectors $\{0,1\}^\ell$ and to the set

of integers $\{0, \dots, 2^\ell - 1\}$ written in binary. For every pair $(a, b) \in \mathbb{F}_{2^\ell}^2$, consider the hash function $h_{ab} : \mathbb{F}_{2^\ell} \to \mathbb{F}_{2^\ell}$ defined as $h_{ab}(y) = a + b \cdot y$. For every $y \in \mathbb{F}(2^\ell) \equiv \{0,1\}^\ell$, we denote by $\rho(y) = \max\{j \in [\ell] : y_1 = \cdots = y_j = 0\}$ the largest index j such that the first j bits of y, seen as a bit-vector, are all zero. Let us now consider the following streaming algorithm:

Algorithm 2 Streaming algorithm for F_0

Let $\ell = \lceil \log_2 m \rceil$, we identify each element $x_i \in [m]$ of the stream with its corresponding element in $\mathbb{F}_{2^\ell}$.
Pick uniformly and independently two random elements $a, b \in \mathbb{F}_{2^\ell}$.
Compute $R = \max_{i=1..n} \rho(h_{ab}(x_i))$.
return 2^R.

► ***Question* 1.6)** *Show that for all $c \in \mathbb{F}_{2^\ell}$ and $r \in \{0, \dots, \ell\}$, $\Pr_{a,b}\{\rho(h_{ab}(c)) \geqslant r\} = \dfrac{1}{2^r}$.*
▷ Hint. *Show that $h_{ab}(c)$ is uniform in $\mathbb{F}_{2^\ell}$.*

Let W_c^r the indicator random variable for the event $\rho(h_{ab}(c)) \geqslant r$. Let $Z_r = \sum_{c \in S_x} W_c^r$, be the number of the values in the stream whose r first bits of their hash key are all zero.

► ***Question* 1.7)** *Show that $\mathbb{E}[Z_r] = F_0/2^r$.*

► ***Question* 1.8)** *Show that the random values $h_{ab}(0), \dots, h_{ab}(2^\ell - 1)$ are uniform and pairwise independent.*
▷ Hint. *Show that if $c \neq d$, then for all $\gamma, \delta \in \mathbb{F}_{2^\ell}$, $\Pr_{a,b}\{(h_{ab}(c), h_{ab}(d)) = (\gamma, \delta)\} = \frac{1}{\#\mathbb{F}_{2^\ell}^2}$.*

► ***Question* 1.9)** *Show that $\mathbb{V}\mathrm{ar}(Z_r) = \dfrac{F_0}{2^r}\left(1 - \dfrac{1}{2^r}\right) < \mathbb{E}[Z_r]$.*

Fix some $\eta > 1$.

► ***Question* 1.10)** *Show that $\Pr\{Z_r > 0\} < \frac{1}{\eta}$ for all $r \in \{0, \dots, \ell\}$ such that $2^r > \eta F_0$.*
▷ Hint. *Z_r is an integer and use Markov's inequality.*

► ***Question* 1.11)** *Show that $\Pr\{Z_r = 0\} < \frac{1}{\eta}$ for all $r \in \{0, \dots, \ell\}$ such that $2^r < F_0/\eta$.*
▷ Hint. *Z_r is an integer and apply Chebyshev's inequality.*

► ***Question* 1.12)** *Conclude that for all $\eta > 2$, $\Pr\{2^R \in [F_0/\eta, \eta F_0]\} > 1 - \frac{2}{\eta}$. The algorithm outputs thus a η-approximation of F_0 with probability at least $1 - 2/\eta$ for all $\eta > 2$. How many bits of memory does it require?*

We have thus obtained a $(\varepsilon, 2/(1+\varepsilon))$-estimator for F_0 using $O(\log m)$ bits of memory **for $\varepsilon > 1$**. Getting a (ε, δ)-estimator for F_0 in $O_{\varepsilon,\delta}(\log m + \log n)$ bits of memory for arbitrarily small $\varepsilon, \delta > 0$ requires a lot more work...

■ **Exercise 2 (Generating function for the Galton-Watson population total number).** Let $\mathcal{Z}$ be the random variable for the total number of nodes in a Galton-Watson branching process for which the extinction probability is 1: $\mathcal{Z} = \sum_{i,n} Z_i^{(n)}$, where $Z_i^{(n)} \sim Z$ denotes the number of children of the ith node on level n. Let $g_{\mathcal{Z}}$ be its generating function.

► ***Question* 2.1)** *Show that $g_{\mathcal{Z}}(s) = s g_Z(g_{\mathcal{Z}}(s))$.*

■ **Exercise 3 (Branching processes in continuous time (★)).** Recall that an *exponential random variable* X with parameter $\lambda > 0$ is defined by: $(\forall x \geqslant 0)\ \Pr\{X \geqslant x\} = e^{-\lambda x}$.

Consider the following process:

- At time 0, $Z_0 = 1$ (the root of the process). By convention, this node is born at time 0.
- When a node i is born, its lifetime is exponentially distributed with parameter μ: if it is born at time t, it dies at time $t + U_i$, where U_i is exponentially distributed with parameter μ.
- An alive node i can give birth to children. Its children birthdates are generated according to an exponential distribution with parameter λ: if a node is born at time t, its first child is generated at time $t + V_i^1$ (if it is not dead before), its second child at time $t + V_i^1 + V_i^2$, and so on where V_i^j is exponentially distributed with parameter λ.
- All the lifetimes (U_i) and birth intervals (V_i^j) form a mutually independent family of random variables.

► ***Question* 3.1)** *Show that the exponential distribution is* memoryless*, i.e. if X is exponentially distributed with parameter λ, then* $(\forall t, u \geqslant 0)$ $\mathsf{Pr}\{X \geqslant t+u \mid X \geqslant t\} = \mathsf{Pr}\{X \geqslant u\}$.

Let X_1 and X_2 be two independent exponentially distributed random variables with respective parameters λ and μ.

► ***Question* 3.2)** *Show that* $\min(X_1, X_2)$ *is also exponentially distributed. What is its parameter?*

► ***Question* 3.3)** *What is the probability that* $\min(X_1, X_2) = X_1$*?*

We are back to the branching process.

► ***Question* 3.4)** *What is the law of the number of children for each node?*

► ***Question* 3.5)** *What is the probability of extinction of this process?*