Finding Frequent Elements

We are given a sequence of elements $x = (x_1, \ldots, x_N)$, where each $x_i$ is an element of some alphabet $\Sigma$ with $n$ symbols. We can think of $\Sigma$ as $\{1, 2, \ldots, n\}$. Think of the sequence as being revealed to us, one by one. As a motivating example, $x$ is the set of IP addresses seen by a particular router on a given day.

We are also given a real number $\Theta$ such that $0 < \Theta \leq 1$. We are interested in the elements $a \in \Sigma$ that occur more than $\Theta N$ times.
in $x$.

Let us introduce some notation for this. For $a \in \Sigma$, let $f_x(a)$ denote the number of occurrences of $a$ in $x$.

We wish to determine

$$I(x, \theta) = \{ a \in \Sigma : f_x(a) > \theta \}$$

In the router example, we wish to determine frequently accessed IP addresses. With $\theta = 0.001$, for example, we want the IP addresses that are accessed more than $\frac{1}{1000}$ of the time.
Now to what makes the problem interesting. Both N and n are extremely large and our algorithm cannot afford \( O(N) \) or \( O(n) \) storage. This rules out the solution where we have an array of size n and keep track of the no. of occurrences of each element. It also rules out having a hash table that keeps track of no. of occurrences of only the elements that occur at least once, there may be \( \Omega(N) \) such elements.
We assume, however, that we can afford storage of about \( \frac{1}{\Theta} \).

Notice that \( |I(x, \Theta)| < \frac{1}{\Theta} \).

Our algorithm will not determine \( I(x, \Theta) \) exactly, but a superset \( K \) of at most \( \frac{1}{\Theta} \) elements.

Let \( t \) be the smallest integer strictly greater than \( \frac{1}{\Theta} \). Note \( t = \lfloor \frac{1}{\Theta} \rfloor + 1 \).

The idea of the algorithm is simple but clever. It keeps track of a set \( K \) of at most \( \frac{1}{\Theta} \) elements along with their counts. If \( K \) grows to \( t \), it decrements the
counts of all elements in $K$, and deletes all elements whose count becomes 0.

$x[i]...x[N]$ is the input sequence.

$K \leftarrow \emptyset$

For $i \leftarrow 1$ to $N$ do

* if $x[i] \in K$, count[$x[i]$] $\leftarrow$ count[$x[i]$] + 1

else

* insert $x[i]$ in $K$

* count[$x[i]$] $\leftarrow$ 1

* if $|K| = t$ then

* for all $a \in K$ do

* count[$a$] $\leftarrow$ count[$a$] - 1

* if count[$a$] = 0, delete $a$ from $K$

Output $K$. 
Example: $0 = 0.5$

$19:1$
$19:1$  $27:1$
$32:1$
$32:2$
$41:1$
$32:1$
$19:1$
$32:1$
$32:1$  $19:1$
$32:1$
$32:2$
$32:3$  $19:1$
$32:4$  $19:1$
$32:5$  $19:1$
$32:6$
$32:7$
$32:8$
$32:8$  $81:1$
Claim  At the end of algorithm,

$I(x, 0) \leq K$.

Proof: We argue that if $a \notin K$ at the end, then $f_x(a) < \emptyset N$,

thus $a \notin I(x, 0)$.

Consider an $a$ such that $a \notin K$ at the end. Each occurrence of $a$ was "eliminated" together with $t-1$

occurrences of other symbols.

No. of occurrences eliminated along

with $a$ is therefore $f_x(a) \times t$.

We must have $f_x(a) \times t \leq N$

$\implies f_x(a) \times \frac{1}{\emptyset} < N$

$\implies f_x(a) < \emptyset N$. 
Note: This material is not from the textbook. It is taken from the paper:

'A Simple Algorithm for Finding Frequent Elements in Streams and Bags'
by Kaup, Shenker, and Papadimitriou.