CS7960 L8 : Streaming-Counting Distinct Elements

Streaming Algorithms

Stream : $A = <a_1,a_2,\ldots,a_m>$
  
  $a_i \in [n]$ size $\log n$

Compute $f(A)$ in $\text{poly}(\log m, \log n)$ space

-------------------------

Flajolet + Martin '85
Alon, Matias, Szegedy '99

$f_j = |\{a_i \in A \mid a_i = j\}|$

Goal: $F_0 = |\{j \in [n] \mid f_j \geq 0\}|$
  number of distinct elements

$\text{zeros}(p) = \max\{i \mid 2^i \text{ divides } p\}$

####################

Init:
Choose random hash $h : [n] \rightarrow [n]$
\( z := 0 \)

Stream:A
  \[ \text{if } (\text{zeros}(h(ai)) > z) \text{ then } z := \text{zeros}(h(ai)) \]

Output: \( 2^{z+1/2} \)

Let there be \( k \) distinct elements.
  - we don't know answer, but used in analysis

Expect \( 1/k \) distinct elements to have \( \text{zeros}(ai) >= \log k \)
Expect no elements to have \( \text{zeros}(ai) >> \log k \)

Let \( X_{r,j} \) == indicator random variable for \( [\text{zeros}(h(j)) > r] \)
\( Y_r = \sum_{j \text{ s.t. } ai=j} X_{r,j} \)
Let \( t = z \) at end of stream.

\[ Y_r > 0 \iff t \geq r \]
\[ Y_r = 0 \iff t < r \]

\[ E[X_{r,j}] = Pr[\text{zeros}(h(j)) \geq r] = Pr[2^r \text{ divides } h(j)] = 1/2^r \]

\[ E[Y_r] = \sum_{j \text{ s.t. } ai=j} E[X_{r,j}] = k/2^r \]

\[ \text{Var}[Y_r] = \sum_{j \text{ s.t. } ai=j} \text{Var}[X_{r,j}] (= E[(X_{r,j})^2] - E[X_{r,j}]^2) \leq \sum_{j \text{ s.t. } ai=j} E[X_{r,j}]^2 \]
\[ = \sum_{j \text{ s.t. } ai=j} E[X_{r,j}] 
\[ = k/2^r \]

Markov Inequality

\( X \) a rv and \( a > 0 \)
\[ \Pr[|X| \geq a] \leq E[|X|]/a \]
Chebyshev's Inequality:

Y a rv and b>0
Pr[|Y-E[Y]| >= b) <= Var(Y)/b^2

using MI with X = (Y-E[Y])^2 and a=b^2

MI : Pr[Y_r > 0] = Pr[Y_r >= 1] <=
E[Y_r]/1 = k/2^r       (E1)
given r < log k then
CI : Pr[Y_r = 0]   = Pr[|Y_r - E[Y_r]| >= k/2^r] <= Var[Y_r]/(k/2^r)^2 <= 2^r/k

(E2)

Algorithm output: hat{k}
hat{k} = 2^{t+1/2}

Let a == smallest integer s.t. 2^{a+1/2} >= 3k.
Pr[hat{k} > 3d] = Pr[t >= a] = Pr[Y_a > 0] <= k/2^a <= sqrt{2}/3 < 1/2
Let $b$ be the largest integer such that $2^{b+1/2} < k/3$.

$$\Pr[\hat{k} \leq d/3] = \Pr[t \leq b] = \Pr[Y_{b+1} = 0] \leq 2^{b+1}/k \leq \sqrt{2}/3 < 1/2$$

$(\varepsilon=3, \delta=1/2)$-approximation

------------------------
Median Trick
(make delta arbitrary small)

Run $s$ parallel, independent hash functions on the above procedure.
output: $\hat{K} = \{ \hat{k}_1, \hat{k}_2, \ldots, \hat{k}_s \}$
let $\bar{k} = \text{median}[\hat{K}]$

$\bar{k} > 3k$ only if $s/2$ values in $\hat{K} > 3k$.
Each $\leq 3k$ wp $1/2$ -- all independent

$$1/2^{s/2} \leq \delta$$ (where we choose $\delta$)
solve for $s$:
\[ 2^{s/2} \geq 1/\delta \]
\[ s/2 \geq \log(1/\delta) \]
\[ s \geq 2 \log (1/\delta) \]

Similar for lower bound: $\delta \rightarrow \delta/2$

Using $s = 2 \log (2/\delta)$, take median $\bar{k}$ is an $(\varepsilon=3, \delta)$-approximation of # distinct elements.

$O(\log \log n)$ bits to store $t$

$O(\log (1/\delta))$ hash functions

So: $O(\log(1/\delta) \times \log \log n)$ right?

oops, forgot to store hash function:

$O(\log n)$ bits to store hash function

So: $O(\log(1/\delta) \times \log n)$

--------

Better algorithm:
Space: $O(\log m + (1/\epsilon^2) (\log(1/\epsilon) + \log \log m))$
(\epsilon,\delta)-approximation
Hashes to smaller number of bins
Takes average to drive \epsilon down