MCMD L8 : Streaming | Heavy Hitters = Approximate Counts

Streaming Algorithms

Stream : A = ⟨a₁,a₂,...,aₘ⟩
ai in [n] size log n
Compute f(A) in poly(log m, log n) space

Let fⱼ = |{aᵢ in A | aᵢ = j}|

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

MAJORITY: if some fⱼ > m/2, output j
else, output NULL

one-pass requires Ω(min{m,n}) space

Simpler:
FP-MAJORITY: if some fⱼ > m/2, output j
else, output anything

How good w/ O(log m + log n) (one counter c + one location l)?

...#

c = 0, l = X
for (aᵢ \in A)
  if (aᵢ = l) c += 1
  else c -= 1
  if (c <= 0) c = 1, l = aᵢ
return l

Analysis: if fⱼ > m/2, then
  if (l != j) then c decremented at most < m/2 times, but c > m/2
  if (l == j) can be decremented < m/2, but is incremented > m/2
if fⱼ < m/2 for all j, then any answer ok.

----- another view of analysis ------

Let fⱼ > m/2, and k = m - fⱼ.
After s steps, let gₛ = unseen elements of index j
  let kₛ = unseen elements != index j
  let cₛ = c if l!=j, and -c if l==j
Claim: gₛ > c+kₛ
  base case (s=0, or even s=1) easily true.
Inductively 4 cases:
\( a_i = l = j : (g_s \text{ decremented, } c \text{ decremented}) \)
\( a_i = l != j: (c \text{ incremented, } k_s \text{ decremented}) \)
\( a_i != l != j: (c \text{ decremented, } k_s \text{ decremented}) \)
\( a_i != l = j : (k_s \text{ decremented, maybe } c \text{ incremented}) \)

Since at the end \( g_s = k_s = 0 \), then
\[
0 > c + 0, \text{ implies } c < 0, \text{ and } l==j.
\]

FREQUENT: for \( k \), output the set \( \{j : f_j > m/k\} \)
also hard.

k-FREQUENCY-ESTIMATION: Build data structure \( S \).
For any \( j \) in \([n]\), \( \hat{f}_j = S(j) \) s.t.
\[
f_j - m/k <= \hat{f}_j <= f_j
\]
aka eps-approximate phi-HEAVY-HITTERS:
\[
\text{Return all } f_j \text{ s.t. } f_j > \phi \\
\text{Return no } f_j \text{ s.t. } f_j < \phi - \epsilon \cdot m \\
\text{(any } f_j \text{ s.t. } \phi - \epsilon \cdot m < f_j < \phi \text{ is ok)}
\]

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

Misra-Gries Algorithm [Misra-Gries '82]

Solves k-FREQUENCY-ESTIMATION in \( O(k(\log m + \log n)) \) space.

Let \( C \) be array of \( k \) counters \( C[1], C[2], \ldots, C[k] \)
Let \( L \) be array of \( k \) locations \( L[1], L[2], \ldots, L[k] \)

Set all \( C = 0 \)
Set all \( L = X \)

for (\( a_i \) in \( A \))
    if (\( a_i \) in \( L \)) <at index \( j >
        \[ C[j] += 1 \]
    else  <\( a_i \) !in \( L >
        if (|L| < k)
            \[ C[j] = 1 \]
            \[ L[j] = a_i \]
        else
            \[ C[j] -= 1 \text{ forall } j \in [k] \]
    for (\( j \) in \( [k] \))
        if (\( C[j] <= 0 \)) set \( L[j] = X \)
On query q in [n] 
    if (q in L {L[j]=q}) return \( \hat{f}_q = C[j] \) 
    else return \( \hat{f}_q = 0 \)

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

Analysis

A counter \( C[j] \) representing \( L[j] = q \) is only incremented if \( a_i = q \)

\( \hat{f}_q \leq f_q \)

If a counter \( C[j] \) representing \( L[j] = q \) is decremented, then \( k-1 \) other counters are also decremented. 
This happens at most \( \frac{m}{k} \) times. 
A counter \( C[j] \) representing \( L[j] = q \) is decremented at most \( \frac{m}{k} \) times.

\( f_q - \frac{m}{k} \leq \hat{f}_q \)

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

How do we get an additive \( \epsilon \)-approximate FREQUENCY-ESTIMATION ?
i.e. return \( \hat{f}_q \) s.t. 
\( |f_q - \hat{f}_q| \leq \epsilon m \)

Set \( k = \frac{2}{\epsilon} \), return \( C[j] + \frac{(m/k)}{2} \)

Space \( O((1/\epsilon) \log m + \log n) \)

Also:
\( \epsilon \)-approximate \( \phi \)-HEAVY-HITTERS for any \( \phi > m \epsilon \) in 
space \( O((1/\epsilon) \log m + \log n) \)

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

Can solve \( k \)-FREQUENT optimally in two passes w/ \( O(k \log n + \log m) \) space. 
Run M-G algorithm w/ \( k \) counters. 
For each stored location, make second pass and count exactly.