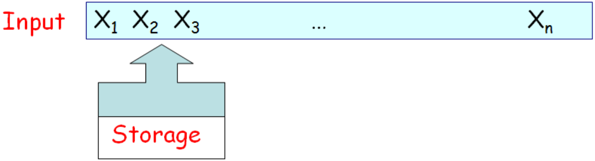
Massive Data Algorithmics

Lecture 13: Streaming Model

Definition

- The input is a sequene $< a_1, a_2, \cdots, a_m >$ recieving one by one (imagine the input is on a tape).
- a_i in an element of the universe $[n]$ where $[n] = \{1, 2, \cdots, n\}$
- Goal is to process the stream using a small amont of space s
- Since m and n are to be thought of as huge, we want to make s much smaller than these
- Specificaly, we want s to be sublinear in both n and m
- The holy grail is to achive $s = O(\log n + \log m)$
- k-passes streaming: allowed to make k passes over the stream



Approximation Algorithms

- It is usually impossible to design exact algorithms in the streaming model
- Let $A(\sigma)$ denote the output of a randomized streaming algorithm A on input σ. Let Φ be the fucntion that A is supposed to compute.
  - We say that the algorithm A (ε, δ)-approximates Φ if we have:

$$Pr(|\frac{A(\sigma)}{\Phi(\sigma)} - 1| > \varepsilon) \leq \delta$$

  - We say that the algorithm A (ε, δ)-additively approximates Φ if we have:

$$Pr(|A(\sigma) - \Phi(\sigma)| > \varepsilon) \leq \delta$$

Frequency Vector

- Let $f_j = |\{i : a_i = j\}|$, # occurrences of j in the stream
- $F = (f_1, f_2, \cdots, f_n)$ is called a frequency vector
- We sometimes would like to compute $\Phi(F)$
- You can imagine that the term a_i of the stream is of form (j, c) which means f_j must be set to $f_j + c$ (in the standard streaming model $c = 1$)
- $||F||_k = (f_1^k + \cdots + f_n^k)^{1/k}$
  - $||F||_0$ is the number of distinct elements in the stream
  - $||F||_1$ is the number of elements in the stream which is m

The Problem

- Majority Problem
  - Input: the stream $\sigma = < a_1, \cdots, a_m >$ where $a_i \in [n]$
  - Output: if $\exists j : f_j > m/2$, output j. Otherwise output null.
- Frequent Problem
  - Input: the stream $\sigma = < a_1, \cdots, a_m >$ where $a_i \in [n]$, and k
  - Output: output set $\{j : f_j > m/k\}$.

Note: we have to report an output upon arrival of a_i for any i

Misra-Gries Algorithm

- A deterministic algorithm to approximate f_j

Initialize : $A \leftarrow$ (empty associative array) ;

Process j:

if $j \in keys(A)$ **then**
    $A[j] \leftarrow A[j] + 1$;
else if $|keys(A)| < k - 1$ **then**
    $A[j] \leftarrow 1$;
else
    foreach $\ell \in keys(A)$ **do**
        $A[\ell] \leftarrow A[\ell] - 1$;
        if $A[\ell] = 0$ **then** remove ℓ from A ;

Output : On query a, if $a \in keys(A)$, then report $\hat{f}_a = A[a]$, else report $\hat{f}_a = 0$

Analysis

- space: $O(k(\log n + \log m))$
- approximation: $f_a - \frac{m}{k} \leq \hat{f}_a \leq f_a$
  - Counter $A[a]$ is incremented only when we process an occurrence of a. So $\hat{f}_a \leq f_a$.
  - Whenever $A[a]$ is decremented (in lines 7 and 8, we pretned that $A[j]$ is incremented from 0 to 1, and then immediately decremented back to 0), we also decrement $k-1$ other counter, corresponding to distinct items in the stream. Then each decrement of $A[a]$ is witnessed by a collection of k distinct items (one of which is a itself) from the stream.
  - Since the stream consists of m items, there can be at most m/k such decrements. So, $f_a - \frac{m}{k} \leq \hat{f}_a$

Example

Assume $k = 3$ and $a = 3$

σ	keys	$A[3]$
$< 2 >$	$\{(2,1)\}$	0
$< 2,1 >$	$\{(2,1),(1,1)\}$	0
$< 2,1,2,2,1 >$	$\{(2,3),(1,2)\}$	0
$< 2,1,2,\underline{2},\underline{1},\underline{3} >$	$\{(2,2),(1,1)\}$	0
$< 2,\underline{1},\underline{2},\underline{2},\underline{1},\underline{3},\underline{3} >$	$\{(2,1)\}$	0
$< 2,\underline{1},\underline{2},\underline{2},\underline{1},\underline{3},\underline{3},3 >$	$\{(2,1),(3,1)\}$	1
$< 2,\underline{1},\underline{2},\underline{2},\underline{1},\underline{3},\underline{3},3,3 >$	$\{(2,1),(3,2)\}$	2
$< \underline{2},\underline{1},\underline{2},\underline{2},\underline{1},\underline{3},\underline{3},3,\underline{3},\underline{1} >$	$\{(3,1)\}$	1
$< \underline{2},\underline{1},\underline{2},\underline{2},\underline{1},\underline{3},\underline{3},3,\underline{3},\underline{1},1,1,1,1 >$	$\{(3,1),(1,4)\}$	1
$< \underline{2},\underline{1},\underline{2},\underline{2},\underline{1},\underline{3},\underline{3},\underline{3},\underline{3},\underline{1},1,1,1,\underline{1},\underline{2} >$	$\{(1,3)\}$	0

A 2-Passes Algorithm for The Frequent Problem

- Each j s.t. $f_j > m/k$ exists in keys(A)
- We can make a second pass over the stream, counting excactly the frequencies f_j for all $j \in \text{keys}(A)$, and then output the desired set of items

References

- **Data Stream Algorithms** (Chapters 0 and 1)
  Lecture notes by A. Chakrabbarti and D. College