COL758: Advanced Algorithms **Spring 2019**

Lecture 17: March 11

Lecturer: Naveen Garg *Scribe: Kshitij Morodia*

17.1 Data Streaming Algorithms

Streaming algorithms are algorithms for processing sequences of data where the input is received as a stream. A **stream** is a sequence of **tokens** on which we wish to compute some statistics.

The form of stream we will work with will have **m** number of tokens each of which is an integer. So we define data stream as a definite length sequence of integers $t_1, t_2, t_3, ..., t_{\mathbf{m}}$ where $t_i \in \{1, 2, 3, .., \mathbf{n}\}$ where n is the size of the universe from which the tokens are drawn.

Statistics that may be desired to be computed on this stream may consist of finding the most frequent token, k-most frequent tokens or number of distinct tokens in the stream. These would be **trivial** problems if we are not constrained by **memory**. But suppose the amount of space available is $O(\log n + \log m)$. We would consider problems in which the space available to us is thus constrained.

17.2 Heavy Hitters in Data Stream

Our goal is to find the most frequently occurring tokens in a data stream. We will consider problems which require us to find tokens with frequency greater than m/k. Such tokens that would satisfy this condition are known as **heavy hitters**

Token with frequency greater than *m/2*

Consider the problem of finding an element **j** in the stream having a frequency of occurrence $f_{\mathbf{j}} > m/2$.

Algorithm:

Result: Outputs j
initialize: $c \leftarrow 0$;
for *each* $\boldsymbol{i}$ **do**
  if $c = 0$ **then** $j \leftarrow t_i$;
  if $t_i = j$ **then**
    $c \leftarrow c + 1$;
  else
    $c \leftarrow c - 1$;
  end
end

Algorithm 1: m/2 frequent token algorithm

Analysis:

It can be observed that value of j does not change while $c > 0$. The input stream can be divided into segments such that each segment except the last ends with a token having counter $c = 0$. Let the values of j during these segments be $j_1, j_2, j_3, ...$ In the n^{th} segment the value j_n appears exactly 1/2 times among

the tokens in that segment. Therefore if indeed $f_j > m/2$ in the stream, then the last segment would end with $c > 0$ and the value would be j.

If there is no such token value with frequency greater than $m/2$ then j would have an arbitrary input value.

Tokens with frequency greater than *m/k* (*Misra-Gries*)

Now consider a more general problem of finding all the tokens with a frequency greater than m/k. The number of tokens that can have a frequency greater than m/k is at most $k-1$. Hence at most $k-1$ counters will be maintained whereas in the case of $m/2$ we had just 1 counter. The aim of this procedure is to produce frequency estimates ($\widetilde{f_j}$) of all the heavy hitters in the stream which then will be related to actual frequencies f_j.

Algorithm:

The algorithm maintains a set A of upto $k-1$ pairs of tokens with their corresponding counter.

```
Result: Outputs A
initialize: A ← ∅. A is a set of up to k − 1 pairs (j,c_j) ;
for each i do
    if t_i ∈ A then
        c_{t_i} ← c_{t_i} + 1 ;
    else if |A| < k − 1 then
        Add t_i to A ;
        c_{t_i} ← 1 ;
    else
        for each j ∈ A do
            c_{t_i} ← c_{t_i} − 1 ;
            if c_j = 0 then Remove j from A ;
        end
    end
end
```

Algorithm 2: m/k frequent tokens algorithm

Analysis:

Suppose a token $j \in A$ has counter value $\widetilde{f_j}$ (c_j from the algorithm output) while f_j is the frequency of token j in the stream. We seek to establish a relationship between $\widetilde{f_j}$ and f_j. The algorithm can be thought of as maintaining $\widetilde{f_j}$ for all $j \in A$, where $\widetilde{f_j} = 0$ for all $j \notin A$.

Clearly,

$$\widetilde{f_j} \leq f_j$$

because the token counter can at most be incremented as many times as actual occurrence in stream considering the value may be decremented too occasionally.

The $k-1$ storage can be though of as cache. When a token is seen in cache we increment its counter. If the counter was not in cache then we decrement the counters in cache including the counter of this token. So the increment by 1 is made m times while the total decrementation is by at most k each time. Hence the number of time the decrement step can be performed (d) is less than m/k.

$$\implies d \leq \frac{m}{k}$$

From $f_j = \widetilde{f_j} + d$ we get,

$$f_j - \frac{m}{k} \leq \widetilde{f_j} \leq f_j$$

We can see that A consists of at least every token that has a frequency greater than $\frac{m}{k}$. Therefore, every token j which has a frequency more than m/k is in A.

$$\left\{j : f_k > \frac{m}{k}\right\} \subseteq A$$

Now, may consist of some tokens which are not heavy hitters. We can get all the heavy hitters in A with 2 passes through the stream:

- $1^{st} pass$: get all the heavy hitters and some extra elements (A).
- $2^{nd} pass$: Remove all the false positives by counting the actual frequency of all the elements of A.

17.3 Number of Distinct Elements

We want to find the number of distinct elements in the stream. For this we will use the idea of hashing. We will use a hash function which matches values 1 to n to itself $h : [n] \to [n]$. This mapping should be thought of as a *random mapping* where every distinct token is mapped to a corresponding distinct value which is chosen randomly from 1 to n. Example of such a mapping: $h_{a,b}(x) = ax + b \mod n$.

note: A 1-1 mapping cannot be ensured.

Algorithm:

Result: Outputs $2^{z+\frac{1}{2}}$
initialize: $z \leftarrow 0$;
for *each* i **do**
  $z \leftarrow max(zeroes(h(i)), z)$;
end

Algorithm 3: distinct element estimation algorithm

The term $2^{z+\frac{1}{2}}$ is an approximate number of distinct elements in the stream which would only holds for a certain probability. The function zeroes(.) returns the number of trailing zeroes at the end of the binary representation of a number.

17.4 Preliminaries

- X is a *random variable*
- $\mathrm{E}[X] = \sum_i P[X = i] * i$
- $\mathrm{E}[X+Y] = \mathrm{E}[X] + \mathrm{E}[Y]$
- $Var[X] = \mathrm{E}[(X - \mathrm{E}[X])^2]$
- $Var[X+Y] = Var[X] + Var[Y]$ (X and Y independent)

  Proof:

  $Var[X+Y]$
  $= \mathrm{E}[((X+Y)-\mathrm{E}[X+Y])^2]$
  $= \mathrm{E}[((X-\mathrm{E}[X]) + (Y-\mathrm{E}[Y]))^2]$
  $= \mathrm{E}[(X-\mathrm{E}[X])^2+(Y-\mathrm{E}[Y])^2+2(X-\mathrm{E}[X])(Y-\mathrm{E}[Y])]$
  $= \mathrm{E}[(X - \mathrm{E}[X])^2] + \mathrm{E}[(Y - \mathrm{E}[Y])^2] + 2*covariance(X, Y)$
  $= Var[X] + Var[Y]$ (covariance of independent *random variables* is 0)

  hence proved.

- **Markov's Inequality:**

  X is a non-negative random variable such that E[X] exists and is known, then for any fixed positive k;

  $$P[X \geq k] \leq \frac{E[X]}{k}$$

  Proof:

  E[X] = $\sum_{i=0}^{\infty} i * P[X = i]$

  =$\sum_{i=0}^{t-1} i * P[X = i] + \sum_{i=t}^{\infty} i * P[X = i]$

  $\geq \sum_{i=t}^{\infty} i * P[X = i]$

  $\geq t * \sum_{i=t}^{\infty} P[X = i]$

  $= t * P[X \geq t]$

  hence proved.

- **Chebychev's Inequality:**

  Let X be a random variable such that E[X]=μ and Var[X]=σ^2 exists and is known, then for any fixed positive k;

  $$P[|X - \mu| > k] \leq \frac{\sigma^2}{k^2}$$

  Proof:

  X is a random variable

  $\implies$ $(X - \mu)^2$ is a non-negative random variable, with $E[(X - \mu)^2] = \sigma^2$.

  Applying Markov Inequality to this,

  $P[(X - \mu)^2 \geq t^2] \leq \frac{E[(X-\mu)^2]}{t^2}$

  $\implies$ $P[|X - \mu| \geq t] \leq \frac{\sigma^2}{t^2}$

  hence proved.