

## Lecture 4: Point queries, heavy hitters

Lecturer: *Przemysław Uznański*Scribe: *Michał Jaworski*

## 1 Finding frequent elements

### 1.1 Majority elements (Boyer Moore 1981)

**Definition 1.** An element of a multiset  $X$ ,  $|X| = n$  is called a majority element if it occurs more than  $\frac{n}{2}$  times.

We derive algorithm for finding majority element in 2-passes over the input. An useful fact:

**Lemma 2.** If  $t$  is the majority element of  $X$ , and for some  $x, y \in X$  we have  $x \neq y$ , then  $t$  is the majority element of  $X \setminus \{x, y\}$ .

We can derive an algorithm utilizing this fact (this describes first pass, which returns a candidate for majority element).

```
s = anything
c = 0
for x in X:
    if x == s:
        c++
    else if c == 0:
        x = s
        c = 1
    else:
        c--
return s
```

If  $X$  contains a majority element it will be returned, otherwise we get random garbage. Second pass is just to verify if candidate is actually a majority element.

### 1.2 $\frac{1}{k}$ -heavy elements

**Definition 3.** An element of a multiset  $X$ ,  $|X| = n$ , is called  $\frac{1}{k}$ -heavy if it occurs at least  $\frac{n}{k}$  times.

**Lemma 4.** If  $t$  is an  $\frac{1}{k}$ -heavy element of  $X$ , and  $x_1, \dots, x_k \in X$  are pairwise distinct, then  $t$  is a  $\frac{1}{k}$ -heavy element of  $X \setminus \{x_1, \dots, x_k\}$ .

We can again derive an algorithm:

- Summary of a stream: a multiset  $S$  (represented for example as a collection of pairs  $(x_i, c_i)$ )
- Invariant:  $\#distinct(S) \leq k$

- Insert: add the element to  $S$ , then run the pruning step
- Pruning: while  $\#distinct(S) > k$ : remove from  $S$  one copy of each element in  $S$

At the end the summary will contain all heavy hitters but it may also contain other elements - we need a second pass to check. This algorithm is deterministic and the summaries are mergeable.

## 2 $L_p$ point queries, $L_p$ heavy hitters

Assume turnstile streaming model, we maintain vector  $x$  under updates. Choose a norm  $L_p$ .

- **Point query**: for a given  $i$  return  $x_i \pm \varepsilon|x|_p$
- **Heavy hitters**:  $HH_\varepsilon(x) = \{i : |x_i| \geq \varepsilon|x|_p\}$ . Output  $L$  such that:
  1.  $HH_\varepsilon(x) \subseteq L \subseteq HH_{\varepsilon'}(x)$  for some  $\varepsilon' < \varepsilon$ , or
  2.  $HH_\varepsilon(x) \subseteq L$  and  $|L| = O(\frac{1}{\varepsilon^p})$  (intuition:  $|HH_\varepsilon(x)| \leq \frac{1}{\varepsilon^p}$ ).

**Observation 5.** *Roughly, if we can solve one of these problems, then we can also solve the other (they reduce to each other).*

## 3 CountMin (Cormode, Muthukrishnan 2004)

### 3.1 Point queries

Assume  $\forall_i x_i \geq 0$ . Let  $h : [n] \rightarrow [t]$  be a 2-wise independent hashing function for  $t = \frac{2}{\varepsilon}$ . Maintain an array  $Z[1..t]$  such that

$$Z[j] = \sum_{i:h(i)=j} x_i$$

- Update( $i, \Delta$ ):  $Z[h(i)] \leftarrow Z[h(i)] + \Delta$
- Query( $i$ ): output  $x'_i = Z[h(i)]$

Properties of queries:

1.  $x'_i \geq x_i$
2.  $\mathbb{E}[x'_i - x_i] = \sum_{j \neq i} \Pr[h(j) = h(i)] \cdot x_j \leq \frac{1}{t}|x|_1 = \frac{\varepsilon}{2}|x|_1$

From Markov's inequality:

$$\Pr[x'_i - x_i \geq \varepsilon|x|_1] \leq \frac{1}{2}$$

so we get an  $L_1$  guarantee.

We can now derive the actual algorithm: we repeat the process independently  $r = \log(\delta^{-1})$  times, using independent hashing functions  $h_1, \dots, h_r : [n] \rightarrow [t]$ . We take the minimum of query results as the answer:

- Update( $i, \Delta$ ):  $\forall_{j \in [r]} Z[j][h_j(i)] \leftarrow Z[j][h_j(i)] + \Delta$
- Query( $i$ ): output  $x'_i = \min_j Z[j][h_j(i)]$

We now get:

$$P(x'_i - x_i \geq \varepsilon |x|_1) \leq \left(\frac{1}{2}\right)^r = \delta$$

Recall we assumed  $\forall_i x_i \geq 0$ . For the general case replace minimum with median, and take  $t = \frac{3}{\varepsilon}$ . Finally, space complexity:  $O\left(\frac{\log \delta^{-1}}{\varepsilon}\right)$  words and time complexity:  $O(\log \delta^{-1})$  per update/query.

## 3.2 Heavy hitters

### 3.2.1 Generic transformation

This method only works in the semi-turnstile model.

- Maintain  $|x|$  from sketch, keep heavy hitters in a priority queue
- On update run a point query, if  $x_i$  is a heavy hitter - insert it into the queue or update its weight
- Whenever the element with lowest weight gets below the  $\varepsilon|x|$  threshold, remove it from the queue

### 3.2.2 Our case

We construct a binary search tree with  $O(\log n)$  levels and  $n$  nodes in the lowest level. Each node represents the sum of values from an interval of the form  $[a2^b + 1, (a+1)2^b]$ . Each level is implemented as a separate CountMin structure. We perform queries recursively descending only into nodes where the output is at least  $\varepsilon|x|_1$ . Since there are only  $\frac{1}{\varepsilon}$  such nodes at each level, we can apply the union bound  $\delta' = \frac{\delta}{2^{\log n}}$ .

## 4 CountSketch (Charikar, Chen, Farach-Colton 2002)

Let  $h_1, \dots, h_r : [n] \rightarrow [t]$  and  $s_1, \dots, s_r : [n] \rightarrow \{-1, 1\}$  be 2-wise independent hashing functions.

- Update( $i, \Delta$ ):  $\forall_j \in [r] Z[j][h_j(i)] \leftarrow Z[j][h_j(i)] + s_j(i) \cdot \Delta$
- Query( $i$ ): output  $x'_i = \text{median}_j Z[j][h_j(i)]$

For a fixed  $j$ :

$$\begin{aligned} \mathbb{E}[(x_i - Z[j][h_j(i)])^2] &= \mathbb{E} \left[ \left( \sum_{k \neq i} \Pr[h_j(k) = h_j(i)] x_k s_j(k) \right)^2 \right] = \sum_{k \neq i} \frac{1}{t} x_k^2 \leq \frac{1}{t} |x|_2^2 \\ \Pr \left[ (x_i - Z[j][h_j(i)])^2 > \frac{3}{t} |x|_2^2 \right] &< \frac{1}{3} \\ \Pr \left[ |x_i - Z[j][h_j(i)]| > \sqrt{\frac{3}{t}} |x|_2 \right] &< \frac{1}{3} \end{aligned}$$

For  $t = O\left(\frac{1}{\varepsilon^2}\right)$  we get

$$\Pr [|x_i - Z[j][h_j(i)]| > \varepsilon |x|_2] < \frac{1}{3}$$

so we get an  $L_2$  norm guarantee. We then use the median with  $r = O(\log \delta^{-1})$  to get  $1 - \delta$  correctness.