

# Weekplan: Streaming III

Philip Bille

Inge Li Gørtz

Eva Rotenberg

## References and Reading

[1] Amit Chakrabarti: *Data Stream Algorithms* 2011 (revised 2015) Chapter 4, section 4.1, 4.2, 4.5.

[2] Moses Charikar. Lecture notes on the CountMin sketch.

We recommend reading the specified chapters and sections of [1] and [2] in detail.

## Exercises

**1 Range queries with a single CountMin sketch** Explain how we can use a single CountMin sketch to solve range queries. What is the query time and how large is the error?

**2 Range queries** In this exercise we will use the tree of dyadic intervals to give a sketch for range queries of the form:

- $\text{range}(a, b)$ : Return an estimate for the number of elements with value between  $a$  and  $b$  (both included).

We maintain a separate CountMin sketch for each level in the tree, where for level  $j$  the  $j$ th CountMin sketch treats two elements that fall into the same interval in level  $j$  as the same element. For all intervals  $i$  in the tree, let  $C(i)$  denote the value that the appropriate CountMin sketch returns for  $i$ .

**2.1** Show that any interval in the range from 1 to  $n$  can be expressed as the disjoint union  $I$  of at most  $2 \lg n$  dyadic intervals. *Hint*: Use the tree.

**2.2** Explain how to use  $I$  to return an estimate for the range query  $\text{range}(a, b)$ .

**2.3** We want to have an additive error of at most  $\epsilon m$  as in the original CountMin sketch. Let  $f_{[a,b]}$  be the total frequency of the elements in the range  $[a, b]$ . Show that if we set  $w = \frac{4 \lg n}{\epsilon}$  in all the CountMin sketches then the expected error is  $\epsilon m$ , that is  $\mathbb{E}[\sum_{i \in I} C(i)] \leq f_{[a,b]} + \epsilon m$ .

**2.4** What is the total space and query time? (we still use  $d = \lg \frac{1}{\delta}$ ).

**3 Combine** Given two streams  $S_1$  and  $S_2$  and the CountMin sketches  $\text{CM}_1$  and  $\text{CM}_2$  of the two streams, explain how to combine the two sketches to get a sketch for the stream  $S_2 \cup S_1$ .

**4 Idea behind CountSketch** Consider the following simple algorithm for finding estimates of all  $f_i$ . Let  $s$  be a pairwise independent hash function from  $[n]$  to  $\{-1, 1\}$  and let  $C$  be a counter. While processing the stream, each time we encounter an item  $q_i$ , update the counter  $c = c + s[q_i]$ . When answering a query we return  $\hat{f}_i = c \cdot s[q_i]$ .

**4.1** Compute the expected value of  $s[j]$  for  $j \in [n]$ .

**4.2** Show that  $\hat{f}_i = f_i + \sum_{j \neq i} f_j \cdot s[j] \cdot s[i]$ .

**4.3** For two independent random variables  $X$  and  $Y$  we have:  $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$ . Use this and linearity of expectation to show that  $\mathbb{E}[\hat{f}_i] = f_i$ .

The variance of this algorithm is very large and many elements have estimates that are wrong by more than the variance. In the CountSketch algorithm there are  $t$  hash functions  $h_j : [n] \rightarrow [b]$  and  $t$  hash functions  $s_j : [n] \rightarrow \{-1, 1\}$  and a  $t \times b$  array of counters. When an element  $q$  arrives the data structure is update by setting  $h_j([q]) = h_j([q]) + s_j[q]$  for all  $i \in [t]$ . To estimate the frequency of element  $q$  return  $\text{median}_j \{h_j[q] \cdot s_j[q]\}$ .

**Graph Streaming** In graph streaming, the stream consists of a sequence of edges  $(u_1, v_1), (u_2, v_2), \dots$ . In general, these edges can be additions (appearing) or subtractions (disappearing), however, in the following exercises, we will focus on the case where edges are only appearing. We will also assume that the graph is *undirected* and *unweighted*. The goal is to use  $O(n \log^{O(1)} n)$  space.

**5 Graph Streaming 0: Warmup** [ $w$ ] How much space in terms of  $n$  do you need in the worst case to describe a graph with  $n$  nodes?

**6 Graph Streaming I: Connectivity** Design a streaming algorithm that counts the number of connected components. Analyse the space consumption, the update time, and the query time of your algorithm.

**7 Graph Streaming II: Shortest Paths** The following algorithm produces a graph  $H$  that we can use to compute an estimate of the shortest path between any pair of vertices in the unweighted graph  $G$  as its edges arrive in a stream. Let  $d_G(u, v)$  and  $d_H(u, v)$  be the distance between the nodes  $u$  and  $v$  in  $G$  and  $H$ , respectively.

---

**Algorithm 1:** SHORTESTPATH(stream  $S$ )

---

```

 $H \leftarrow \emptyset$ 
for  $(u, v)$  in stream  $S$  do
  | if  $d_H(u, v) > \alpha$  then
  | |  $H \leftarrow H \cup (u, v)$ 
end
return  $H$ 

```

---

**7.1** How close are the shortest path distances in  $H$  to the true shortest path distances? Give upper and lower bounds for  $d_H(u, v)$  in terms of  $d_G(u, v)$ .

**7.2** The *girth* of a graph is the length of the shortest cycle in the graph. Show that  $H$  has girth at least  $\alpha + 2$ .

**7.3** Let  $m$  be the number of edges in  $H$  and let  $d = 2m/n$  be the average degree. Consider the graph  $H'$  obtained by iteratively removing all nodes with degree less than  $d/2$  from  $H$ . Show that  $H'$  has at least one vertex.

**7.4** The graph  $H'$  has minimum degree at least  $d/2$  and girth at least  $\alpha + 2$ . Let  $\ell = \lfloor \frac{\alpha+1}{2} \rfloor$ . Show that for any vertex  $v$  in  $H'$  the subgraph of  $H'$  induced by the set of all vertices at distance at most  $\ell$  from  $v$  is a tree.

**7.5** Use the minimum degree of  $H'$  to show that if we start a BFS in any node  $v$  in  $H'$  then we get a tree of depth at least  $\ell$  and give a lower bound on the number of nodes in the tree.

**7.6** Show that  $m \leq n + n^{1+1/\ell}$ .

**7.7** Analyze the space usage of the algorithm in terms of  $n$  and  $\alpha$ .