

## Lecture 20

Instructor: Madhu Sudan

Scribe: Santhoshini Velusamy

## Administrative Notes

- Sign up for presentations.
- Check in next week.

## Topics covered

- Streaming Algorithms and their limits
  - Description of the model
  - Example algorithms
  - Lower bounds on the space complexity of streaming algorithms

## Streaming setting

### Model

Consider an input stream  $x_1, x_2, \dots, x_m$ , where  $x_i \in [n]$  for every  $i$ . The goal is to compute a function on the stream,  $f(x_1, x_2, \dots, x_m) \in \mathcal{T}$  using as less space as possible. Typically, we are interested in algorithms that require  $\text{poly}(\log n, \log m)$  space and there are no constraints on the runtime of the algorithm except that it must be finite.

More formally, the algorithm is a function  $A : \{0, 1\}^s \times [n] \rightarrow \{0, 1\}^s$  and the output is a function  $O : \{0, 1\}^s \rightarrow \mathcal{T}$ . We can treat  $\sigma_0 \in \{0, 1\}^s$  as a seed fed to the algorithm. Define  $\sigma_i \triangleq A(\sigma_{i-1}, x_i)$  for  $i \geq 1$ . We want

$$\Pr_{\sigma_0} [O(\sigma_m) = f(x_1, x_2, \dots, x_m)] \geq 1 - \epsilon.$$

In the next subsection, we describe an example problem in the streaming setting and outline some algorithms and proofs of lower bounds.

### Frequency moments

For  $i \in [n]$ ,  $f_i \triangleq f_i(x_1, \dots, x_m) = \# \{j | x_j = i\}$ . The  $k^{\text{th}}$  moment of  $(f_1, f_2, \dots, f_n)$  is defined as  $F_k \triangleq \sum_{i=1}^n f_i^k$ . The zeroth moment is defined as

$$F_0 = \lim_{k \rightarrow 0} F_k = \# \{i | f_i \neq 0\}.$$

### Brief history

1. It is easy to see that  $F_1 = m$ . Every item is counted in exactly one of the  $f_i$ 's.
2. (1985[1]) Flajolet-Martin: An algorithm to approximate  $F_0$  with  $S = \text{poly}(\log(n, m))$ .
3. (1998[2]) Alon-Matias-Szegedy: An algorithm to approximate  $F_2$  in space  $\text{poly}(\log(n, m))$ . Algorithms to approximate  $F_k$  in space  $n^{1-\frac{1}{k}}$ .

4. (2005) [BJKS] Set-disjointness via Information Complexity

- (a) Approximating  $F_k$  requires at least space  $n^{1-\frac{2}{k}}$ .
- (b) Even multi-pass  $F_k$  requires at least space  $n^{1-\frac{3}{k}}$ .

**Flajolet-Martin algorithm**

**Algorithm 1**

Let  $h : [n] \rightarrow [0, 1]$  be a uniform hash function. Define  $h_{\min}(x_1, x_2, \dots, x_m) \triangleq \min_j \{h(x_j)\}$ . The algorithm outputs  $\frac{1}{h_{\min}}$ . The claim is  $\mathbb{E} \left[ \frac{1}{h_{\min}} \right] = F_0$ . Intuitively, the elements that appear in the stream divide the space uniformly and hence, on expectation the algorithm outputs  $F_0$ .

**Algorithm 2**

We take a hash function  $h : [n] \rightarrow \mathbb{R}^{\geq 0}$ , where for every  $i$ ,  $h(i) \sim \exp(1)$  (Exponential random variable with mean 1). Similar to Algorithm 1, we compute  $h_{\min}(x_1, x_2, \dots, x_m) = \min_j \{h(x_j)\} \sim \exp\left(\frac{1}{F_0}\right)$ . The algorithm outputs  $\frac{1}{h_{\min}}$ . This gives a  $(1 \pm \epsilon)$  approximation algorithm to compute  $F_0$  using space  $\mathcal{O}\left(\frac{1}{\epsilon^2}\right) \text{poly}(\log(m, n))$ .

A space complexity lower bound of  $\Omega\left(\frac{1}{\epsilon^2}\right)$  can be proved for any  $(1 \pm \epsilon)$  approximation algorithm to compute  $F_0$ , using Gap Hamming distance.

Exercise: Prove all the claims above!

**Alon-Matias-Szegedy algorithm**

We will first describe their algorithm to compute  $F_2$ . Consider a uniformly random hash function  $h : [n] \rightarrow \{+1, -1\}$  (Note: It takes  $n$  bits to remember this hash function. We can reduce the space to  $\text{poly}(\log n)$  by using pseudorandom hash functions). Compute  $v = \sum_{j=1}^m h(x_j)$ . Output  $v^2$ .

*Claim.*  $\mathbb{E}_h [v^2] = F_2$

*Proof.* We have,

$$\begin{aligned} \mathbb{E}_h [v^2] &= \mathbb{E}_h \left[ \sum_{j=1}^m h(x_j) \sum_{l=1}^m h(x_l) \right] \\ &= \sum_{j,l} \mathbb{E} [h(x_j)h(x_l)] \\ &= \sum_i \sum_{j,l} \mathbb{E} [1_{x_j=x_l=i}], \end{aligned}$$

where  $1_{x_j=x_l=i}$  is the indicator random variable for the event that  $x_j = x_l = i$  and last equality follows from the fact that for  $x_j = x_l$ ,  $\mathbb{E} [h(x_j)h(x_l)] = 1$  and for  $x_j \neq x_l$ ,  $\mathbb{E} [h(x_j)h(x_l)] = 0$ .  $\sum_i \sum_{j,l} \mathbb{E} [1_{x_j=x_l=i}]$  is exactly equal to  $\sum_i f_i^2 = F_2$ . □

We will now describe their algorithm to compute  $F_k$  for  $k \geq 3$  that uses space  $\mathcal{O}\left(n^{1-\frac{1}{k}}\right)$ . Pick  $j \in \{1, 2, \dots, m\}$  uniformly at random. Let  $a = x_j$  and  $a$  appears  $r$  times later. Output  $r^k - (r - 1)^k$ .

*Claim.*  $\mathbb{E}_j [r^k - (r - 1)^k] = F_k$  and variance is  $n^{1-\frac{1}{k}}$ .

*Proof.* Exercise! □

The variance can be further reduced to 1 by repeating the algorithm  $N = \lceil n^{1-\frac{1}{k}} \rceil$  times and taking the mean. The mean doesn't change but the variance decreases by a factor of  $\frac{1}{N}$ . Thus, we get an algorithm that uses  $\mathcal{O}\left(n^{1-\frac{1}{k}}\right)$  space. Note: If  $m$  is not known beforehand, we can slightly change the algorithm and run it in parallel for different values of  $m$  or potentially use "reservoir sampling" to sample a uniform  $j$ .

In the following subsection, using the known communication complexity of a variant of the set disjointness problem, we prove worst case lower bounds on the space complexity of any streaming algorithm that allows a single pass or multiple passes over the data and computes  $F_k$ .

## $t$ -party communication complexity of $t$ -set disjointness

### $t$ -set disjointness

- Inputs: YES and NO instances. The task is to distinguish between the two instances.  $S_i \subseteq [n]$  for all  $i$ .
  - YES:  $(S_1, S_2, \dots, S_t)$  such that
    - \*  $\exists i \in S_1 \cap S_2 \cdots \cap S_t$ ,
    - \*  $\forall i' \neq i, \#\{j | S_j \ni i'\} \leq 1$ .
  - NO:  $(S_1, S_2, \dots, S_t)$  such that  $\forall i, \#\{j | S_j \ni i\} \leq 1$ .

### $t$ -party communication model

There are  $t$  players  $p_1, p_2, \dots, p_t$  with inputs  $x_1, x_2, \dots, x_t$  respectively and the goal is to compute a function  $f(x_1, x_2, \dots, x_t)$ . Communication is allowed only from the  $i^{\text{th}}$  player to  $(i+1)^{\text{st}}$  player. The last player  $p_t$  outputs the function value.

In the model with  $r$  rounds, player  $p_t$  is allowed to send a message to player  $p_1$  and at most  $r$  rounds of communication is allowed before  $p_t$  outputs the function value.

Let us consider the  $t$ -set disjointness problem in the  $t$ -party communication model. The input given to the  $i^{\text{th}}$  player is  $x_i = S_i$ .

**Fact 1.** [3, Theorem 7.1] [4] One-way communication complexity of  $t$ -set disjointness is at least  $\Omega\left(\frac{n}{t}\right)$ .

**Fact 2.** [3, Theorem 7.1], [4] Communication complexity of  $t$ -set disjointness is at least  $\Omega\left(\frac{n}{t \log t}\right)$ .

We will now demonstrate how to use the above communication complexity lower bounds to prove worst case lower bounds on the space complexity of any streaming algorithm that computes  $F_k$  for  $k \geq 2$ . Specifically, we will show how a streaming algorithm that computes  $F_k$  can be used to solve the  $t$ -party  $t$ -set disjointness problem.

Let there be a one-pass streaming algorithm that computes  $F_k$  using space  $c$ . Starting from player  $p_1$ , we can assume that each player  $p_{i+1}$  can receive the previous state of the streaming algorithm from player  $p_i$  and run the algorithm starting from this state on the set  $S_{i+1}$  (assume that the elements in the set are streamed one by one) and sends the final state to player  $p_{i+2}$ . The overall communication is upper bounded by  $ct$ . In the NO instance, we have  $F_k \leq n$ . This is because every element in  $[n]$  occurs at most once in the stream. In the YES instance, we have  $F_k \geq t^k$  as there exists an element  $i \in [n]$  that occurs in every set  $S_j$  and hence appears  $t$  times in the stream. By choosing  $t = (2n)^{\frac{1}{k}}$ , we can distinguish the YES and the NO instances. Using Fact 1, we know that  $ct \geq \Omega\left(\frac{n}{t}\right)$  and hence  $c \geq \Omega\left(\frac{n}{t^2}\right)$ . Substituting  $t = (2n)^{\frac{1}{k}}$ , we get  $c \geq \Omega\left(n^{1-\frac{2}{k}}\right)$ .

Similarly, by using Fact 2, we can prove that any multi-pass streaming algorithm that computes  $F_k$  requires space at least  $\Omega\left(n^{1-\frac{3}{k}}\right)$ .

## References

- [1] Philippe Flajolet and G. Nigel Martin. 1985. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.* 31, 2 (September 1985), 182-209. DOI=[http://dx.doi.org/10.1016/0022-0000\(85\)90041-8](http://dx.doi.org/10.1016/0022-0000(85)90041-8)
- [2] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing (STOC '96)*. ACM, New York, NY, USA, 20-29. DOI: <https://doi.org/10.1145/237814.237823>
- [3] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.
- [4] Chakrabarti, A & Khot, S & Sun, Xiaodong. (2003). Near-optimal lower bounds on the multi-party communication complexity of set disjointness. 107- 117. 10.1109/CCC.2003.1214414.