10TH TUTORIAL ON RANDOMIZED ALGORITHMS

Streaming algorithms

- *Input:* sequence of items $a_1, a_2, \ldots, a_m$, where $a_i \in [N]$
- *Goal:* process the stream in *one pass with low memory*, ideally $\text{poly}(\log(N+m))$, and estimate a desired statistic of the stream.

Let f_j be the frequency of $j \in [N]$, i.e., the number of times j appears in the streams.

1. *Streaming warm-up.* Some functions are simple in the streaming model, such as the sum or the average. For the average, try to improve the most straightforward algorithm and in particular, avoid intermediate results of order $m \cdot N$.

2. *Strict majority in constant space.* Design a streaming algorithm using as small space as possible that finds the *majority element* if there is one, i.e., an element j with frequency $f_j > m/2$. If there is no majority element, the output can be arbitrary.

(This is the Boyer–Moore majority vote algorithm from 1981.)

3. *Finding heavy hitters.* Generalize the previous algorithm so that for a parameter k, it returns all elements j with $f_j > m/k$, the so-called "heavy hitters", and possibly some other elements. This can be extended as follows:

a) The algorithm can also return an estimate of f_j for any given $j \in [N]$. What kind of guarantee do we get from it?

b) The algorithm returns all heavy hitters but possibly also elements that may be far from heavy hitters (in extreme cases occurring just once in the stream). How to modify the algorithm so that all of the returned elements are close to being heavy hitters, i.e., having frequency at least $\Omega(m/k)$.

(This is the Misra-Gries algorithm from 1982.)