Randomized Algorithms 2020-1
Lecture 4

Streaming Algorithms *

Moni Naor

1 Streaming Algorithms

Suppose you want to compute a function on a stream of data but do not have enough memory to store it. We will consider single pass algorithm, that is once the data has passed there is no further access to it. We would like as little extra storage as possible.

Several issue come up: Which functions are computable? At what accuracy can they be computed? There is a rich literature one the subject with many interesting algorithms and lower bounds.

For most tasks, if they are doable at all, then randomness is essential. One notable 'counter-example' we saw was the Boyer Moore algorithm for finding a a majority element, provided such an element exists.

2 Multiset equality

The problem we addressed can be viewed as a 'streaming' one. We have two multi-sets A and B and they are given in an arbitrary order. Once an element is given it cannot be accessed again (unless it is explicitly stored) and our goal is to have a low memory algorithm. We required a family of functions H that was incremental in nature, in the sense that for a function $h \in H$:

- Given $h, h(A)$ and an element x it is easy to compute $h(A \cup \{x\})$.
- For any two different multi-sets A and B the probability over the choice of $h \in H$ that $h(A) = h(B)$ is small.
- The description of h is short and the output of h is small.

The function we saw was based on treating the set A as defining a polynomial $P_A(x) = \Pi_{a \in A}(x - a)$ over a finite field whose size is larger than the universe from which the elements of A are chosen

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. In the interest of brevity, most references and credits were omitted.

(say a prime $Q > |U|$). The member of the family of functions is called h_x for $x \in GF[Q]$ and defined as $h_x(A) = P_A(x)$. The probability that two sets collide (i.e. $h_x(A) = h_x(B)$, which in turn means that $P_A(x) = P_B(x)$) is $\max\{|A|, |B|\}/Q$, since this is the maximum number of points that two polynomials whose degree is at most $\max\{|A|, |B|\}$ can agree without being identical.

Storing h_x and storing $h(A)$ as it is computed requires just $O(\log Q)$ bits, so the resulting algorithm never needs to store anything close size to the original sets.

The Petrushka proposal: let $f : N \mapsto N$ be an ordering of the primes, i.e. $f(i)$ returns the ith prime. Now an alternative to the definition of a polynomial P_A define an integer $N_A = \Pi_{a \in A} f(a)$. Claim: for all mutli-sets A and B, if $A \neq B$, then $N_A \neq N_B$. Of course one cannot hope to store N_A explicitly. Instead, just as evaluating $P_A(x)$ at point y can be done on-the-fly, it is possible to compute N_A mod Q. The hash family now is h_Q where Q is a random prime chosen from a certain size.

Question: Analyze the Petrushka method. Suggest the appropriate domain from which to chose Q.

Reading and Watching assignment:

- Watch the lecture by David Woodruff on "Adversarially Robust Streaming Algorithms" `https://www.youtube.com/watch?v=9qP3JCWNgnc`
- Tim Roughgarden's Notes on streaming and communication complexity. Read Sections 1-5 `http://timroughgarden.org/w15/l/l1.pdf`

Old questions we did not finish discussing and to which we will return next week:

Vague Question: Can you argue that taking the majority is the best way to amplify the probability of success of a BPP Algorithm? Or is there some other function, e.g. majority of majorities, that is better?

Question Prove that if $A \cdot B \neq C$, then Freivald's algorithm for the verification of matrix multiplication says 'no' with probability at least $1 - 1/q$ if the finite field is $GF[q]$ and the random vector is chosen appropriately.