Randomized Algorithms 2024-5
Lecture 2

A better exponential time algorithm for satisfiabiltiy, streaming, Freivalds Matrix Checking*

Moni Naor

Watch the remaining parts of Ryan O'Donell Lecture 5 on concentration bounds.

1. `https://www.youtube.com/watch?v=cLczU5-CW70`
2. `https://www.youtube.com/watch?v=zz4C-xECIp4`

1 Sat Algorithms

The main algorithms we saw were for 2-SAT and 3-SAT. The second one is due to Uwing Schöning. You can find a thorough description in Chapter 7 of Mitzenmacher and Upfal. The complexities of the algorithms are roughly $O(n^2)$ and $O((4/3)^n)$ respectively.

A famous conjecture, with wide implications, called the **Exponential Time Hypothesis** (ETH), states that 3-SAT cannot be solved in sub-exponential time, i.e., in times less than $(1+\alpha)^n$ for some $\alpha > 0$.

2 Freivalds Algorithm and Randomized QuickSort

Randomized QuickSort is a classical example of an analysis of a a randomized algorithm. The analysis we saw, based on linearity of expectations, can be founds, for instance, in the Chapter 1 of Motwani-Raghavan.

Freivalds presented his matrix multiplication verification algorithms in 1977 (though I did not see the original paper called "Probabilistic Machines Can Use Less Running Time").

Question: Show how to use Freivalds algorithm for verifying "Boolean Matrix multiplication", where addition is logical "or" and multiplication is logical "and".

Freivalds presented the result I mentioned that it is possible for a 2-way finite automata to recognize probabilistically the language $a^n b^n$ in 1981 [2]. Dwork and Stockmeyer showed that any such automata recognizing a non regualr langauge must take exponential time [1].

*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.

3 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 issues 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.

3.1 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 (say a prime $Q > |U|$). The member of the family of functions is called h_r for $x \in GF[Q]$ and defined as $h_r(A) = P_A(r)$. The probability that two sets collide (i.e. $h_r(A) = h_r(B)$, which in turn means that $P_A(r) = P_B(r)$) 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.

We saw a suggestion by one of the students for such a function that was based on min-count sketch. It is not the most efficient, since the he storage. was inverse in the probability of error (this can be amplified)

Here is another suggestion made a few years ago by a student: The Primes 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 Primes 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
  http://timroughgarden.org/w15/l/l1.pdf

References

[1] Cynthia Dwork and Larry Stockmeyer, *A Time Complexity Gap for Two-Way Probabilistic Finite-State Automata*, SIAM Journal on Computing Vol. 19(6), 1990

[2] Rusins Freivalds,*Probabilistic two-way machines*, Mathematical Foundations of Computer Science, 1981.

[3] Jon Kleinberg and Eva Tardos, **Algorithm Design**. Addison Wesley, 2006. The relevant chapter 13.

[4] Dick Lipton's blog, "Rabin Flips a Coin", March 2009
https://rjlipton.wordpress.com/2009/03/01/rabin-flips-a-coin/

[5] Michael Oser Rabin, *Probabilistic algorithms*. In Algorithms and complexity: New Directions and Recent Results, pages 21 - 39. Academic Press, New York.

[6] Rene Schoof, *Four primality testing algorithms*, http://arxiv.org/abs/0801.3840