Randomized Algorithms 2014/5A
Lecture 1 – Min Cut Algorithm, Closest Pairs, (Multi)-Set Equality *

Moni Naor

The lecture introduced randomized algorithms. Why are they interesting? They may solve problems faster than deterministic ones, they may be essential in some settings, especially when we want to go to sublinear algorithms[1], they are essential in distributed algorithms e.g. for breaking symmetry, they yield construction of desirable objects that we do not know how to build explicitly and are essential for cryptography and privacy. Another type of study is to analyze algorithms when assuming some distribution on the input, but our emphasis would be worst case data where the randomness is created independently of it. That is we assume the algorithm or computing device in addition to the inputs gets also a random 'tape' (like the other tapes of the Turing Machine, but this one with truly random symbols). One nice feature that some randomized algorithms have is that they are simple. We demonstrated this in three algorithms in three different scenarios.

Randomized algorithms existed for a long time, from the dawn of computing (for instance the numerical "Monte Carlo Method"[2] from the 1940's or Shannon's work [9], also from that time. The von Neumann 'trick' of turning a biased coin into a fair one is from that period [7].

Later, when people started arguing rigorously about running time the idea of complexity classes of probabilistic machines emerged. We mentioned three such classes RP, $Co-RP$ and BPP. One of the proponents of the power of randomized algorithms in the 1970s was Michael Rabin who published a very influential paper [6] on randomized algorithms. In that paper he gave a randomized algorithm for testing the primality of a number (based on Miller's deterministic test which assumed the 'Extended Riemann Hypothesis') that ran in polynomial time as well as a linear expected time algorithm for finding the closest pair of points from a given set. The primality test was (one directional) 'Monte Carlo' - it would always output 'prime' on a prime input and would output 'non-prime' with high probability on a composite . Since then several randomized algorithms for primality have been discovered as well as a deterministic one (see Schoof [8]). The fact that fast algorithms for primality exist made the RSA cryptosystem (suggested not long after) feasible.

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

[1]For instance, the famed PCP Theorem, that states that every NP statement can be verified using a few queries must use randomness for picking the queries.

[2]Do not confuse with the term "Monte Carlo Algorithm" which is a general name for an algorithm whose running time is deterministic (usually polynomial) but may err.

The Minimum Cut Problem

This demonstrates simplicity in a spectacular way. No need for flows, just pick a random edge and contract! The min-cut algorithm we saw is due to Karger from SODA 1993. There is a faster version with Stein where the repetition is done in a clever way (i.e. not starting from scratch each time), yielding a near $O(n^2)$ algorithm [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 (but needs to be explicitly stores). We required a family of functions H that was incremental in nature, that is for a function $h \in H$:

- Given $h(A)$ and element x it is easy to compute $h(A \cup \{x\}$.
- For any two different multi-sets A and B the probability of 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 larger than the universe from which the elements of A are chosen (say a prime $Q > |U|$). The function is h_x for any $x \in GF[Q]$ and $h(A) = P_A(x)$. The probability that two sets collide is $max\{|A|, |B|\}/Q$. storing h and storing $h(A)$ requires $\log Q$ bits.

Closest Points in the Plane

Rabin's closest pair algorithm was a Las Vegas type algorithm, i.e. it never outputs a wrong result but the run time may take longer than expected. The algorithm we saw in class is much later and is Due to Golin et al. [1]. The analysis was in expectation and was based on the probabiltiy that we will need to rebuild the dictionary from the beginning, which was $2/i$ in the ith phase. There is a good description of it in Kleinberg and Tardos's book [3]. (you can read a description of Rabin's algorithm, which was also based on constructing a grid, in Lipton's blog [5]). The algorithm uses the floor ($\lfloor x \rfloor$) operation to find the square in the grid and hence does not fit the model used by most algorithms in geometry.

To complete the algorithm we need a good hash table to store the non-vacant grid cells. Thus, the algorithm yields yet another motivation for having dictionaries with $O(1)$ per operation. The will be discussed in future lectures

References

[1] Mordecai Golin, Rajeev Raman, Christian Schwarz and Michiel Smid, *Randomized Data Structures For The Dynamic Closest-Pair Problem*, SIAM J. Comput., vol. 26, no. 4, 1998.

[2] David Karger and Clifford Stein, *A new approach to the minimum cut problem.* Journal of the ACM 43 (4): 601, 1996.

[3] Jon Kleinberg and Eva Tardos, **Algorithm Design**. Addison Wesley, 2006. The relevant chapter: `http://www.aw-bc.com/info/kleinberg/assets/downloads/ch13.pdf`

[4] Richard M. Karp, *Probabilistic analysis of some combinatorial search problems.* In Algorithms and Complexity: New Directions and Recent Results, pages 119. Academic Press, New York. Academic Press, 1976.

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

[6] Michael Oser Rabin, *Probabilistic algorithms.* In Algorithms and complexity: New Directions and Recent Results (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, Pa., 1976), pages 21-39. Academic Press, New York.

[7] John von Neumann, *Various techniques used in connection with random digits*, National Bureau of Standards Applied Math Series 12: 36, 1951.

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

[9] Claude Shannon, *A Mathematical Theory of Communication.* Bell System Technical Journal 27 (3): 379-423, (July/October 1948).
`http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf`