Randomized Algorithms 2013A
Lecture 5 – Martingales, Closest Pairs, Hash Tables, Existential Proofs and Codes *

Moni Naor

The lecture covered quite varied material, starting from Martingales, moving to a linear time Closest Pairs algorithm, Hash Tables, Probabilistic Existential Proofs, Probabilistic Models for Graphs and Error Correcting Codes

1 Martingales

We reviewed Martingales which were discussed at the end of last lecture. Recall that we are dealing with a sequence of random variables $X_0, X_1, \ldots X_m$ so that for all $0 \leq i < m$ we have

$$E[X_{i+1}|X_i] = X_i \tag{1}$$

The name 'martingale' comes from the betting world. The typical story is that X_i represents the wealth after a sequence of fair bets where the winnings in the ith round are represented by Y_i with $E[Y_i] = 0$. One issue that came up in the discussion is that in the literature one often sees the requirement (1) phrased instead as

$$E[X_{i+1}|X_i, X_{i-1} \ldots X_0] = X_i \tag{2}$$

The question is whether these two requirements are equivalent and whether we can get the nice concentration for both of them. They are not equivalent as the following example shows. Consider a random walk on the integers with a reflecting wall at 0 where the first step chooses whether the walk is in the non negative or non positive integers. That is $X_0 = 0$, the first step is random in $\{-1, 1\}$ and all subsequent steps are random in $\{-1, 1\}$ if $X_i \neq 0$ and reflect to the previous step if $X_i = 0$ (i.e. $X_{i+1} = X_{i-1}$). Now the sequence is not a martingale according to (2), since given $X_i, X_{i-1} \ldots X_0$, if $X_i = 0$ then we know that $X_{i+1} = X_{i-1}$. But requirement (1) holds, since given $X_i = 0$ we have no idea whether we arrived from the positives or negatives.

Under both definitions we have Azuma's inequality:

*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.

Theorem: Let $c = X_0, X_1, \ldots X_m$ be a martingale such that $|X_{i+1} - X_i| \leq 1$ for all $0 \leq i < m$. Then

$$\Pr[|X_m - c| > \lambda\sqrt{m}] < 2e^{-\lambda^2/2}$$

A *Doob Martingale* is one obtained when $X_i = E[f(Z_1, Z_2, \ldots Z_n)|Z_1, Z_2, \ldots Z_i]$ where the Z_i are random variables in some set A and $f : A^n \mapsto R$. This is very convenient when trying to show that the performance of an algorithm is close to its expected value with high probability. One example we saw was the number of '0's in a Bloom filter with truly random hash functions.

Exercise: throwing n balls into n bins at random, what can you say about the expected number of vacant bins and how concentrated is this value around the expectation?

Another example we mentioned and where martingales are useful is the chromatic number $\chi(G)$ (the minimum number of colors needed to color the vertices of the graph G so that no two neighboring nodes receive the same color) of a random graph. Let n be an integer and and $p \in [0, 1]$. Consider $\mathcal{G}_{n,p}$ the distribution on graphs on n nodes where each edges exists with probability p. Let C be the expected value of the chromatic number of $\chi(G)$ a graph chosen from $\mathcal{G}_{n,p}$.

Exercise: show that $\Pr[|\chi(G) - C| > \lambda\sqrt{n-1}] < 2e^{-\lambda^2/2}$.

2 Closest Points in the Plane and a Brief History of Randomized Algorithms

In 1976 Rabin published a very influential paper [10] 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 [11]). The fact that fast algorithms for primality exist enabled the feasibility of the RSA cryptosystem (suggested not long after).

The closest pair algorithm was a Las Vegas type algorithm - 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 [6] . There is a good description of it in Kleinberg and Tardos's book [7]. (you can read a description of Rabin's algorithm, which was also based on constructing a grid, in Lipton's blog [9]) 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. The algorithm yields yet another motivation for having dictionaries with $O(1)$ per operation.

3 Hash Tables

Hash tables is very well studied subject in computer science and one of the more useful practices. Knuth's "The Art of Computer Programming" Volume 3 devotes a lot of space to the various possibilities and the origin of the idea is attributed to a 1956 paper by Arnold Dumey [4]. A major issue is how to resolve collisions and popular suggestions are chaining and Open addressing (or closed hashing). For the latter we need to specify a probing scheme and one of the more popular ones is linear probing which takes advantage of the locality properties of computer memory (in the various levels of hierarchy).

The 'modern' era of investigating hashing can be seen in the work of Carter and Wegman [2] who suggested the idea of thinking of the input as being worst case and the performance is investigated when the hash function is chosen at random from a predefined family (rather than assuming a truly random function).

The simplest way to obtain dictionaries with expected $O(1)$ per operation is to use chained hashing with a table of size $O(n)$. Here, it is enough to choose the hash function from a δ-universal family, for δ which is $O(1/n)$. The expected length of a chain is now $O(1)$ and the length of the chain is what determines the cost of an operation. Note however that there will be long chains. Universality on its own only suffices to guarantee that the expected length of the *longest* chain is $O(\sqrt(n))$. The upper bound follows from considering all potential collisions: there are $\binom{n}{2}$ of them. Each collision occurs with probability $O(1/n)$, so the expected number of collisions is $O(n)$. On the other hand, in a chain of length ℓ there are $\binom{\ell}{2}$ collisions. Therefore the expected length of the longest chain cannot be larger than $O(\sqrt(n))$. For the lower bound see Alon et al. [1].

What happens if the hash function is truly random? Then this is the classical "ball and bins" scenario and here the heaviest bin/chain is likely to contain $\Theta(\log n / \log \log n)$ elements.

Universality on its own also just guarantees *expected* $O(1)$ performance and not, say, high probability $(1 - 1/poly(n))$ amortized $O(1)$ performance. There are several ways to obtain this sort of result. In the next lecture we will explore "Cuckoo Hashing" a method that uses two hash functions h_1 and h_2 and where each element x in the set resides either in location $h_1(x)$ or location $h_2(x)$. This means that lookup requires just two accesses to the memory. Insertion may be more involved and requires relocating elements.

A lecture be Eric Demaine on hashing is available and recommended [3].

4 Probabilistic Proofs of Existence

The probabilistic method for proving the existence of an object with certain properties works by providing a probability space over objects, where the probability that the desired properties hold will be non-zero. It then concludes that at least one of the points in the space must define the desired object. You can find a lot of information on this idea in the book titled "The Probabilistic Method" by Alon and Spencer.

The first example we discussed was Ramsey Graphs. Ramsey Theory studies the conditions for

the appearance of an ordered substructure inside a large enough structure. A simple example is the claim that in any sufficiently large graph there must exist a clique or independent set of size k. The theorem is proved by arguing that $R(k,\ell)$, a bound on the number of nodes above which a graph must contain a clique of size k or an independent set of size ℓ, exists. This is done by induction on $k+\ell$ and then arguing that $R(k,\ell) \leq R(k-1,\ell) + R(k,\ell-1)$. This proof guarantees that $R(k,k) \leq 2^{2k}$ and can be slightly improved by a more careful analysis.

Paul Erdös in 1947 [5] gave an existence proof that there are graphs with no cliques or independent sets larger than $1/2 \log n$, that is he showed that $R(k,k) \leq 2^{k/2}$. His proof was probabilistic, by considering $\mathcal{G}_{n,1/2}$ the distribution on graphs on n nodes where each edges exists with probability $p = 1/2$[1]. Now computing the expected number of cliques of size k is straightforward: $\binom{n}{k} \cdot 2^{-\binom{k}{2}}$ and similarly for independent sets. As long as the sum of these two values is less than 1, there must be a graph on n node and no clique or independent set of size k. This happens around $1/2 \log n$. No constructive proof, that tells us how to build such a graph ('explicit construction'), is known, not even one with much weaker parameters.

One can wonder what exactly do we mean by an 'explicit construction' and there are various answers: one requirement can be that we can list the nodes and edges of the graph in time polynomial in n (the size of the graph). A more stringent requirement would be that given the 'names' of two nodes (i.e. two strings of length $\log n$) it is possible to determine in time polynomial in the representation of the names whether they are neighbors. For instance, in the hypercube it is very easy to decide whether two nodes $x_1, x_2 \in \{0,1\}^{\log n}$ are neighbors: just check that the Hamming distance is 1.

Another point discussed is whether a probabilistic construction may be good enough if it yields a sufficiently high probability of success, in particular if it is possible to check whether the construction satisfies our requirements. This is not the case for Ramsey graphs. We have an algorithm that finds cliques of size $\log n$ in a random graph in $\mathcal{G}_{n,1/2}$: simply by choosing a node and continuing recursively with its neighbors. But no known algorithm can find a larger clique under this distribution (and it has been posed as a challenge by Karp [8] in 1976 in the same conference where Rabin presented his work on randomized algorithms). No algorithm for certifying that a random graph is a Ramsey graph is known. Another problem with probabilistic construction comes when two different entities want to use the constructed object, especially in the context of communication. How do you transmit the object to the other party.

Around the same time of Erdös's work another highly influential application of the probabilistic method appeared, in the work of Claude Shannon on coding for noisy channels [12]. First lets define the entropy of a random variable (for simplicity we will consider discrete ones:

Definition 1. *The* Shannon Entropy *of random variable X obtaining value in a set $\mathcal{A}$ is*

$$H(x) = \sum_{a \in \mathcal{A}} -Pr[a] \log_2 Pr[a]$$

.

There are several interpretations of the Shannon entropy, the most common one is that it is the

[1] He did not use this notation yet, which was developed later by him and Alfred Renyi, but did simple counting.

expected length in bits of encoding X under the best coding scheme. The entropy is maximized for the uniform distribution over $\mathcal{A}$ and is equal to $\log_2 |\mathcal{A}|$ in this case.

We are interested in two parties A and B who want to communicate over a noisy channel. The simplest model for noise is the Binary Symmetric Channel, where each bit is flipped independently with independent p for some $p \in [0, 0.5)$. For $p \in (0.5, 1]$ we may as well deterministically flip each received bit and get to the $p \in [0, 0.5)$ case and if $p = 0.5$ there is nothing we can do in terms of coding. We will denote by $H(p)$ the value $-p \log p - (1-p)\log(1-p)$, which is the entropy of the binary symmetric channel.

Shannon showed that it is possible to encode messages of length n using codewords of length m roughly $n/(1 - H(p))$ so that they will be decoded correctly with exponentially high probability (hence $1-H(p)$ is the channel capacity). The argument is based on choosing a random coding, that is each information word $x \in \{0,1\}^n$ receives a random code word $C(x) \in \{0,1\}^m$. The decoding of a received message $y \in \{0,1\}^m$ is done by finding the x where $E(x)$ is the closest to y. To show that that this works we will use the following inequality:

$$\sum_{i \leq pm} \binom{m}{i} \leq 2^{mH(p)+o(1)} \tag{3}$$

This means that the size (number of points) of the Hamming ball of radius pm around a point in $\{0,1\}^m$ is at most $2^{mH(p)+o(1)}$. Going back to the analysis of the proposed scheme, the bad case of decoding a message x can occur because

- The received word y is too far from from $C(x)$. But this is the probability that in the binomial distribution $B(m,p)$ the result is too far from the expectation which can be handled, for instance, by Chernoff bounds.
- There is a another word x' where $C(x')$ is closer to y than $m(p+\gamma)$ where γ should be chosen with care. But there are less than $2^{mH(p+\gamma)+o(1)}$ points closer to y than $m(p+\gamma)$. For any specific $x' \in \{0,1\}^n$, the probability that it is mapped to such a ball is $2^{-m} \cdot 2^{mH(p+\gamma)+o(1)} = 2^{-m(1-H(p+\gamma)+o(1))}$ and the probability that there is such and x' is at most $2^n \cdot 2^{-m(1-H(p+\gamma)+o(1))}$.

We will finish discussing coding schemes and in particular linear codes next meeting.

References

[1] Noga Alon, Martin Dietzfelbinger, Peter B. Miltersen, Erez Petrank, and Gabor Tardos, *Linear Hashing* Journal of the ACM, Vol. 46(5), 1999. `http://www.brics.dk/RS/97/16/BRICS-RS-97-16.pdf`

[2] J. L. Carter and M. N. Wegman, Universal classes of hash functions, J. Comput. Syst. Sci. 18 (1979) 143–154. `http://www.cs.princeton.edu/courses/archive/fall09/cos521/Handouts/universalclasses.pdf`

[3] Eric Demaine, Lecture 10 in course 6.851: Advanced Data Structures (Spring'12) on Dictionaries, `https://courses.csail.mit.edu/6.851/spring12/lectures/L10.html`

[4] Arnold Isaac Dumey, *Indexing for rapid random-access memory*, Computers and Automation 5 (12), 6–9, 1956.

[5] Paul Erdös, *Some remarks on the theory of graphs*, Bull. Amer. Math. Soc. 53 (4), 1947, pp. 292-294.
http://www.ams.org/journals/bull/1947-53-04/S0002-9904-1947-08785-1/home.html

[6] 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.

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

[8] 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.

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

[10] 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.

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

[12] 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