Lecture IX
RANDOMIZATION AND DERANDOMIZATION

Probabilistic thinking turns out to be uncannily effective for proving the existence of combinatorial objects. Such proofs can often be converted into randomized algorithms. There exist efficient randomized algorithms for problems that are not known to have efficient deterministic solutions. Even when both deterministic as well as randomized algorithms are available for a problem, the randomized algorithm is usually simpler. This fact may be enough to favor the randomized algorithm in practice. Sometimes, the route to deterministic solution is via a randomized one: after a randomized algorithm has been discovered, we may be able to remove the use of randomness. We will illustrate such "derandomization" techniques. For further reference, see Alon, Spencer and Erdös [1].

§1. Introduction to Randomized Algorithms

We use a simple toy problem to illustrate probabilistic thinking in algorithm design.

A *k-coloring* of the edges of a graph $G = (V, E)$ is an assignment $C : E \to \{1, \ldots, k\}$. There are $k^{|E|}$ such colorings. A *random k-coloring* is one that is equal to any of the $k^{|E|}$ colorings with equal probability. Alternatively, a random k-coloring is one that assigns each edge to any of the k colors with probability $1/k$.

Lemma 1. *Let c and n be positive integers. In the random 2-coloring of the edges of the complete graph K_n, the expected number of copies of K_c that are monochromatic is*

$$\binom{n}{c} 2^{1-\binom{c}{2}}$$

Proof. Let X count the number of monochromatic copies of K_c in a random edge 2-coloring of K_n. If V is the vertex set of K_n then

$$X = \sum_U X_U \tag{1}$$

where $U \in \binom{V}{c}$ and X_U is the indicator function of the event that the subgraph of K_n restricted to U is monochromatic. The probability that $X_U = 1$ is the probability that U is monochromatic. Since there are $\binom{c}{2}$ edges in U and we can color it monochromatic in one of two colors (all white or all black), the probability of the event $X_U = 1$ is

$$\Pr\{X_U = 1\} = 2^{1-\binom{c}{2}}.$$

But $\mathtt{E}[X_U] = \Pr\{X_U = 1\}$ since X_U is an indicator function. Since there are $\binom{n}{c}$ choices of U, we obtain $\mathtt{E}[X] = \sum_U \mathtt{E}[X_U] = \binom{n}{2} 2^{1-\binom{c}{2}}$, by linearity of expectation. **Q.E.D.**

Corollary 2. *There exists an edge 2-coloring of K_n such that the number of monochromatic copies of K_c is at most*

$$\binom{n}{c} 2^{1-\binom{c}{2}}$$

¶1. Existence to Construction: Monochromatic Triangles Consider how to find a 2-coloring assured by this corollary. Let the 2 colors be red and blue. Furthermore, suppose $c = 3$ so that we desire a 2-coloring of K_n such that the number of monochromatic triangles is at most $\binom{n}{3}2^{1-\binom{3}{2}} = \binom{n}{3}2^{-2} = \frac{1}{4}\binom{n}{3}$. There is a trivial randomized algorithm if we are willing to settle for a slightly larger number of monochromatic triangles, say $\frac{1}{3}\binom{n}{3}$:

Randomized Coloring Algorithm:
Input: n
Output: a 2-coloring of K_n with $< \frac{1}{3}\binom{n}{3}$ monochromatic triangles
repeat forever:
1. Randomly color the edges of K_n blue or red.
2. Count the number X of monochromatic triangles.
3. If $X < \frac{1}{3}\binom{n}{3}$, return the random coloring.

If the program halts, the random coloring has the desired property. The probability that a random coloring does not have the property is at most 3/4. In proof: if the probability is more than 3/4, then the expected number of monochromatic triangles in a random coloring is more than $\frac{3}{4} \cdot \frac{1}{3}\binom{n}{3}$, contradicting our lemma. (This is really an application of Markov's inequality.) Hence the probability of repeating the loop *at least once* is at most 3/4. If T is the time to do the loop once and $\widetilde{T}$ is the expected time of the algorithm, then

$$\widetilde{T} \leq T + \frac{3}{4}\widetilde{T}$$

which implies $\widetilde{T} \leq 4T$ (how?). But note that

$$T = T(n) = \mathcal{O}(n^3),$$

since there are $\mathcal{O}(n^3)$ triangles to check. Thus the expected running time is $\widetilde{T} = \mathcal{O}(n^3)$. a Las Vegas Algorithm!

Note that our randomized algorithm has unbounded worst-case running time. Nevertheless, the probability that the algorithm halts is 1. Otherwise, if there is a positive probability $\epsilon > 0$ of not halting, then expected running time becomes unbounded ($\geq \epsilon \times \infty$), which is a contradiction. Alternatively, the probability of not halting is at most $(3/4)^\infty = 0$.

Reprise: to appreciate the power and simplicity of the randomized algorithm, suppose we want to solve the same problem *deterministically*. The simplest way to do this is to systematically try all 2-colorings of K_n, and for each, check whether the number of monochromatic triangles is $\leq \frac{1}{3}\binom{n}{3}$. Now, we have no assurance that we can find this rapidly. So the complexity, for all we know, is $\Omega(2^{\binom{n}{2}})$ since there are $2^{\binom{n}{2}}$ 2-colorings.

¶2. A Better Algorithm for Triangles. The previous algorithm clearly generalized to the general case of $c \geq 4$. For the special case of $c = 3$, we can improve the algorithm by using Chebyshev's inequality. Now we need to compute the variance of X. Since X is the sum of X_U's where $U \in \binom{V}{3}$, (see equation (1)), life would be much simpler if the X_U's are pairwise independent. Let $(ijk), (i'j'k') \in \binom{V}{3}$ be two distinct triangles. We claim that X_{ijk} and $X_{i'j'k'}$ are independent. This is clearly true if the triangles have at most one vertex in common. It remains to consider the case where they share two vertices: consider a triangle "twin" (ijk) and (ijk') (figure 1).

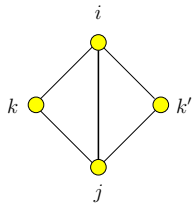

Figure 1: Triangle twin (ijk) and (ijk')

It is easy to see that the probability of (ijk') being monochromatic is not affected by the knowledge that (ijk) is (is not) monochromatic:

$$\Pr\{X_{ijk'} = 1 | X_{ijk} = 1\} = \Pr\{X_{ijk'} = 1\} = 1/4.$$

It follows that the variance X_U's are indeed pairwise independent. CONTINUED...

Exercises

Exercise 1.1: For small values of n, it seems easy to find 2-colorings with fewer than $\frac{1}{4}\binom{n}{3}$ monochromatic triangles. For instance, when $n = 5$, we can color any 5-cycle red and the non-cycle edges blue, then there are no monochromatic triangles (the theorem only gave a bound o 2 monochromatic triangles). Consider the following simple deterministic algorithm to 2-color K_n: pick any T tour of K_n. A tour is an n-cycle that visits every vertex of K_n exactly once and returns to the starting point. Color the n edges in T red, and the rest blue. Prove that this algorithm does not guarantee at most $\frac{1}{4}\binom{n}{3}$ monochrome triangles. For which values of n does this algorithm give fewer than $\frac{1}{4}\binom{n}{3}$ monochrome triangles? ♢

Exercise 1.2: Fixed a G graph with n nodes. Show that a random graph of size $2\log n$ does not occur as an induced subgraph of G. ♢

Exercise 1.3:

(i) What is the role of "3/4" in the first randomized algorithm?

(ii) Give another proof that the probability of halting is 1, by lower bounding the probability of halting at the ith iteration.

(iii) Modify the algorithm into one that has a probability $\varepsilon > 0$ of not finding the desired coloring, and whose worst case running time is $\mathcal{O}_\varepsilon(n^3)$.

(v) Can you improve $T(n)$ to $o(n^3)$? ♢

Exercise 1.4:

(a) Construct a deterministic algorithm to 2-color K_n so that there are at most $2\binom{n/2}{3}$ monochromatic triangles. HINT: use divide and conquer.

(b) Generalize this construction to giving a bound on the number of monochromatic K_c for any constant $c \geq 3$. Compare this bound with the original probabilistic bound. ♢

End Exercises

§2. The Probabilistic Method

§3. Tracking a Random Object

We began with an existence proof of a coloring of K_n that does not have "too many" monochromatic copies of K_c. Then we derived a simple randomized algorithm to find such a coloring. Suppose we now want a deterministic algorithm instead. We use a method of Spencer and Raghavan to convert a randomized algorithm into a deterministic one. Alternatively, the method converts a probabilistic existence proof into a deterministic algorithm. Such a conversion has been termed "derandomization", and is based on conditional probabilities.

For our problem of 2-coloring K_n, the derandomization method is rather simple. We claim that the following deterministic algorithm will compute a 2-coloring with at most $\frac{1}{4}\binom{n}{3}$ monochromatic triangles:

Deterministic Coloring Algorithm:
1. Arbitrarily order the edges $e_1, e_2, \ldots, e_m$, $m = \binom{n}{2}$.
2. Consider each e_i in turn:
   - 2.0. So $e_1, \ldots, e_{i-1}$ had been colored.
   - 2.1. Compute W_i^{red}, defined to be the expected number of monochromatic triangles if e_i is next colored red and the remaining edges are randomly colored. Similarly, compute W_i^{blue}.
   - 2.2. Color e_i red iff $W_{i-1}^{red} \leq W_{i-1}^{blue}$.

¶3. Correctness. We claim that the final coloring has at most $\frac{1}{4}\binom{n}{3}$ monochromatic triangles. In proof, let W_i be the expected number of monochromatic triangles if the remaining edges $e_{i+1}, e_{i+2}, \ldots, e_m$ are randomly colored, conditioned on some given coloring of $e_1, \ldots, e_i$. Initially we have

$$W_0 = \frac{1}{4}\binom{n}{3}.$$

Clearly,

$$W_{i-1} = \frac{W_i^{red} + W_i^{blue}}{2} \geq \min\{W_i^{red}, W_i^{blue}\} = W_i.$$

It follows that $W_0 \geq W_1 \geq \cdots \geq W_m$. But W_m corresponds to a 2-coloring C of K_n and so W_m must count the number of monochromatic triangles of C. Moreover,

$$W_m \leq W_0 \leq \frac{1}{4}\binom{n}{3},$$

as desired.

¶4. Complexity. To see that the above outline can be turned into an effective algorithm, we show that W_i can be computed in polynomial time. This is straightforward: for any triple U of vertices in K_n, let $X_{i,U}$ be the indicator function for the event that U will be monochromatic if the remaining edges $e_{i+1}, \ldots, e_m$ are randomly colored. Then

$$W_i = \sum_U \mathtt{E}[X_{i,U}]$$

where the sum ranges over all U. But $\mathtt{E}[X_{i,U}]$ is just the probability that U will become monochromatic:

$$\mathtt{E}[X_{i,U}] = \begin{cases} 2 \cdot 2^{-3} & \text{if no edge of } U \text{ has been colored,} \\ 2^{-3+i} & \text{if } i = 1,2,3 \text{ edges of } U \text{ has been colored with one color,} \\ 0 & \text{the edges of } U \text{ have been given both colors.} \end{cases}$$

Clearly W_i^{red} and W_i^{blue} can be computed in $\mathcal{O}(n^3)$ time. This leads to an $\mathcal{O}(n^5)$ time algorithm. But it is not hard to see that $\mathcal{O}(n)$ suffices if we already know W_{i-1}, giving an overall $\mathcal{O}(n^3)$ time.

¶5. Framework for deterministic tracking. It is instructive to see the above algorithm in a general framework. Let D be a set of objects and $\chi : D \to \mathbb{R}$ is a real function. For instance, D is the set of 2-colorings of K_n and χ counts the number of monochromatic triangles. In general, think of $\chi(d)$ as computing some "characteristic value" of $d \in D$. Call an object d "good" if $\chi(d) \leq k$ (for some k); and "bad" otherwise. Our problem is to find a good object $d \in D$. First we introduce a sequence *tracking variables* $X_1, \ldots, X_m$ in some probability space $(\Omega, 2^\Omega, \Pr)$. Each X_i is an independent Bernoulli r.v. where $\Pr\{X_i = +1\} = \Pr\{X_i = -1\} = \frac{1}{2}$. We want these variables to be "complete" the sense that for any $\epsilon_1, \ldots, \epsilon_m \in \{\pm 1\}$, the event

$$\{X_1 = \epsilon_1, \ldots, X_m = \epsilon_m\}$$

is an elementary event. For instance, we can simply let $\Omega = \{\pm 1\}^m$ and

$$X_i(\epsilon_1, \epsilon_2, \ldots, \epsilon_m) = \epsilon_i$$

for $(\epsilon_1, \ldots, \epsilon_m) \in \Omega$. We also introduce a random object $g : \Omega \to D$ with the property

$$\mathtt{E}[\chi_g] \leq k \tag{2}$$

where χ_g is the random variable $\chi_g(\omega) = \chi(g(\omega))$, $\omega \in \Omega$. This means that there is *some* sample point ω such that $g(\omega)$ is good. Let $W(\epsilon_1, \ldots, \epsilon_i)$ denote the conditional expectation

$$W(\epsilon_1, \ldots, \epsilon_i) := \mathtt{E}[\chi_g | X_1 = \epsilon_1, \ldots, X_i = \epsilon_i].$$

Write W_i as shorthand for $W(\epsilon_1, \ldots, \epsilon_i)$. Hence, our assumption (2) above amounts to $W_0 \leq k$. Inductively, suppose we have determined $\epsilon_1, \ldots, \epsilon_{i-1}$ so that

$$W_{i-1} = W(\epsilon_1, \ldots, \epsilon_{i-1}) \leq k.$$

Then observe that

$$\begin{aligned} W_{i-1} &= \frac{W(\epsilon_1, \ldots, \epsilon_{i-1}, +1) + W(\epsilon_1, \ldots, \epsilon_{i-1}, -1)}{2} \\ &\geq \min\{W(\epsilon_1, \ldots, \epsilon_{i-1}, +1), W(\epsilon_1, \ldots, \epsilon_{i-1}, -1)\}. \end{aligned}$$

Hence, if we can efficiently compute the value $W(\epsilon_1', \ldots, \epsilon_i')$ for any choice of ϵ_j''s, we may choose ϵ_i so that $W_i \leq W_{i-1}$, thus extending our inductive hypothesis. The hypothesis $W_i \leq k$ implies there is a sample point $\omega \in \{X_1 = \epsilon_1, \ldots, X_i = \epsilon_i\}$ such that $g(\omega)$ is good. In particular, the inequality $W_m \leq k$ implies that $g(\omega)$ is good, where $\omega = (\epsilon_1, \ldots, \epsilon_m)$. Thus, after m steps, we have successfully "tracked" down a good object $g(\omega)$. Note that this is reminiscent of the greedy approach.

The use of the conditional expectations W_i is clearly central to this method. A special case of conditional expectation is when W_i are conditional probabilities. If we cannot efficiently compute the W_i's, some estimates must be used. This will be our next illustration.

Exercises

Exercise 3.1: Generalize the above derandomized algorithm 2-coloring K_n while avoiding too many monochromatic K_c, for any $c \geq 4$. What is the complexity of the algorithm? ♢

Exercise 3.2:

(i) If the edges of K_4 (the complete graph on 4 vertices) are 2-colored so that two edges e, e' have one color and the other 4 edges have a different color, and moreover e, e' have no vertices in common, then we call this coloring of K_4 a "kite". What is the expected number

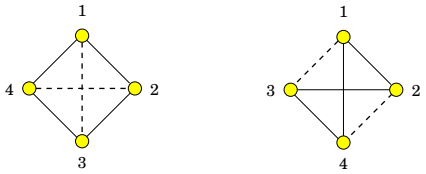

Figure 2: A kite drawn in two ways: edges (1,3) and (2,4) are red, the rest blue.

of kites in a random 2-coloring of the edges of K_n?

(ii) Devise a deterministic tracking algorithm to compute a 2-coloring which achieves *at least* this expected number. but analyze its complexity.

(iii) Forget deterministic tracking, but give a simple $\mathcal{O}(n^2)$ algorithm to do the same. ♢

§4. Maximum Satisfiability

The **MaxSat Problem** is another classic problem for which our tracking framework. In this problem, we are given a set $F = \{C_1, \ldots, C_m\}$ of clauses in the Boolean variables $x_1, \ldots, x_n$, and the problem is to find an assignment $I : x_i \mapsto I(x_i) = b_i (i = 1, \ldots, n)$ of Boolean values such that the number $\#_I(F)$ of clauses in F satisfied by I is maximimzed.

§5. Discrepancy Random Variables

A typical "discrepancy problem" is this: *given real numbers* $a_1, \ldots, a_n$, *choose signs* $\epsilon_1, \ldots, \epsilon_n \in \{\pm 1\}$ *so as to minimize the absolute value of the sum*

$$S = \sum_{i=1}^{n} \epsilon_i a_i.$$

The minimum value of $|S|$ is the *discrepancy* of $(a_1, \ldots, a_n)$. Call[1] a random variable X a *discrepancy r.v.* if the range of X is ± 1; it is *random* if, in addition,

$$\Pr\{X = +1\} = \Pr\{X = -1\} = \frac{1}{2}.$$

The *hyperbolic cosine function* $\cosh(x) = (e^x + e^{-x})/2$ arises naturally in discrepancy random

[1] Clearly, this is just another name for a Bernoulli r.v..

variables. If X_i are random discrepancy r.v.'s, then

$$\begin{aligned}
\mathtt{E}[e^{a_i X_i}] &= \frac{e^{a_i} + e^{-a_i}}{2} \\
&= \cosh(a_i) \\
\mathtt{E}[e^{a_1 X_1 + a_2 X_2}] &= \frac{e^{a_1+a_2} + e^{-a_1-a_2} + e^{a_1-a_2} + e^{-a_1+a_2}}{4} \\
&= \frac{\cosh(a_1+a_2) + \cosh(a_1-a_2)}{2}.
\end{aligned}$$

Using the fact that

$$2\cosh(a_1)\cosh(a_2) = \cosh(a_1+a_2) + \cosh(a_1-a_2),$$

we conclude that $\mathtt{E}[e^{a_1 X_1 + a_2 X_2}] = \cosh(a_1)\cosh(a_2)$. In general, with $S = \sum_{i=1}^n a_i X_i$, we get

$$\mathtt{E}[e^S] = \prod_{i=1}^n \cosh(a_i). \tag{3}$$

A useful inequality in this connection is

$$\cosh(x) \leq e^{x^2/2}, \qquad x \in \mathbb{R}, \tag{4}$$

with equality iff $x = 0$. This can be easily deduced from the standard power series for e^x.

Exercises

Exercise 5.1:

(i) Verify equation (4).

(ii) Show the bound $\Pr\{X_1 + \cdots + X_n > a\} < e^{-a^2/2n}$, where X_i are random discrepancy r.v.'s and $a > 0$. ♢

§6. A Matrix Discrepancy Problem

Raghavan considered a discrepancy problem in which we need to estimate the conditional probabilities. Let $A = (a_{ij})$ be an $n \times n$ input matrix with $|a_{ij}| \leq 1$. Let $\Omega = \{\pm 1\}^n$. Our goal want to find $\overline{\epsilon} = (\epsilon_1, \ldots, \epsilon_n) \in \Omega$, such that for each $i = 1, \ldots, n$,

$$\left| \sum_{j=1}^n \epsilon_j a_{ij} \right| \leq \alpha n$$

where

$$\alpha := \sqrt{\frac{2\ln(2n)}{n}}. \tag{5}$$

This choice of α will fall out from the method, so it is best to treat it as a yet-to-be-chosen constant (n is fixed during this derivation). Using the method of deterministic tracking, we introduce random discrepancy r.v.'s $X_1, \ldots, X_n$ such that $X_i(\overline{\epsilon}) = \epsilon_i$ for all i. Also introduce the r.v.'s

$$S_i = \sum_{j=1}^n X_j a_{ij}, \qquad i = 1, \ldots, n.$$

Suppose the values $\epsilon_1, \ldots, \epsilon_\ell$ have been choosen. Consider the event

$$C^\ell := \{X_1 = \epsilon_1, \ldots, X_\ell = \epsilon_\ell\}$$

and the conditional "bad" event

$$B_i^\ell := \{\, |S_i| > \alpha n \,|\, C^\ell\}.$$

To carry out the deterministic tracking method above, we would like to compute the probability $\Pr(B_i^\ell)$. Unfortunately we do not know how to do this efficiently. We therefore replace $\Pr(B_i^\ell)$ by an easy to compute upper estimate, as follows:

$$\begin{aligned}
\Pr(B_i^\ell) &= \Pr\{|S_i| > \alpha n \;\big|C^\ell\} \\
&= \Pr\{e^{\alpha S_i} > e^{\alpha^2 n} \big|C^\ell\} + \Pr\{e^{-\alpha S_i} > e^{\alpha^2 n} \big|C^\ell\} \\
&\leq e^{-\alpha^2 n}\mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i} \;\big|C^\ell] \quad \text{(Markov inequality)} \\
&= e^{-\alpha^2 n} W_i^\ell,
\end{aligned}$$

where the last equation defines W_i^ℓ. Thus we use W_i^ℓ as surrogate for $\Pr(B_i^\ell)$. We do it because we can easily compute W_i^ℓ as follows:

$$\begin{aligned}
W_i^\ell &= \mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i}|C^\ell] \\
&= \mathtt{E}[\exp\left(\alpha \sum_{j=1}^{\ell} \epsilon_j a_{ij}\right) \exp\left(\alpha \sum_{j=\ell+1}^{n} X_j a_{ij}\right)] + \mathtt{E}[\exp\left(-\alpha \sum_{j=1}^{\ell} \epsilon_j a_{ij}\right) \exp\left(-\alpha \sum_{j=\ell+1}^{n} X_j a_{ij}\right)] \\
&= \exp\left(\alpha \sum_{j=1}^{\ell} \epsilon_j a_{ij}\right) \prod_{j=\ell+1}^{n} \cosh(\alpha a_{ij}) + \exp\left(-\alpha \sum_{j=1}^{\ell} \epsilon_j a_{ij}\right) \prod_{j=\ell+1}^{n} \cosh(\alpha a_{ij}) \\
&= 2\cosh\left(\alpha \sum_{j=1}^{\ell} \epsilon_j a_{ij}\right) \prod_{j=\ell+1}^{n} \cosh(\alpha a_{ij}).
\end{aligned}$$

In particular, for $\ell = 0$,

$$\begin{aligned}
\sum_{i=1}^{n} W_i^0 &= 2\sum_{i=1}^{n}\prod_{j=1}^{n} \cosh(\alpha a_{ij}) \\
&\leq 2\sum_{i=1}^{n}\prod_{j=1}^{n} e^{a_{ij}^2\alpha^2/2} \\
&< 2ne^{n\alpha^2/2},
\end{aligned}$$

where the last inequality is strict since we will assume no row of A is all zero. So the probability that a random choice of $\overline{\epsilon}$ is bad is at most

$$\begin{aligned}
\sum_{i=1}^{n} \Pr(B_i^0) &\leq e^{-n\alpha^2} \sum_{i=1}^{n} W_i^0 \\
&< 2ne^{-n\alpha^2/2}.
\end{aligned}$$

We choose α so that the last expression is equal to 1; this is precisely the α in (5). This proves that there is a sample point $\overline{\epsilon}$ where none of the n bad events B_i^0 occur. The problem now is to track down this sample point. In the usual fashion, we show that if $\epsilon_1, \ldots, \epsilon_\ell$ have been chosen then we can choose $\epsilon_{\ell+1}$ such that

$$\sum_{i=1}^{n} W_i^\ell \geq \sum_{i=1}^{n} W_i^{\ell+1}.$$

This can be done as follows: let C_+^ℓ denote the event $C^\ell \cap \{X_{\ell+1} = +1\}$ and similarly let $C_-^\ell := C^\ell \cap \{X_{\ell+1} = -1\}$. Then

$$\begin{aligned}
\sum_{i=1}^n W_i^\ell &= \sum_{i=1}^n \mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i} | C^\ell] \\
&= \sum_{i=1}^n \frac{\mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i} | C_\ell^+] + \mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i} | C_\ell^-]}{2} \\
&\geq \min\{\sum_{i=1}^n \mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i} | C_\ell^+], \sum_{i=1}^n \mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i} | C_\ell^+]\} \\
&= \sum_{i=1}^n \mathtt{E}[e^{\alpha S_i} + e^{-\alpha S_i} | C^{\ell+1}]
\end{aligned}$$

provided we choose $\epsilon_{\ell+1}$ to make the last equation hold. But the final expression (6) is $\sum_{i=1}^n W_i^{\ell+1}$. This proves $\sum_{i=1}^n W_i^\ell \geq \sum_{i=1}^n W_i^{\ell+1}$. After n steps, we have

$$\sum_{i=1}^n \Pr(B_i^n) \leq e^{-\alpha^2 n} \sum_{i=1}^n W_i^n < 1. \tag{6}$$

But B_i^n is the probability that $|S_i| > \alpha n$, conditioned on the event C^n. As $C^n = \{(\epsilon_1, \ldots, \epsilon_n)\}$ is an elementary event, the probability of any event conditioned on C^n is either 0 or 1. Thus equation (6) implies that $\Pr(B_i^n) = 0$ for all i. Hence C^n is a solution to the discrepancy problem.

We remark that computing W_i is considered easy because the exponential function e^x can be computed relatively efficiently to any desired degree of accuracy (see Exercise)

Exercises

Exercise 6.1: What is the bit-complexity of Raghavan's algorithm? Assume that e^x (for x in any fixed interval $[a, b]$) can be computed to n-bits of relative precision in $\mathcal{O}(M(n) \log n)$ time where $M(n)$ is the complexity of binary number multiplication. The inputs numbers a_{ij} are in floating point notation, *i.e.*, a_{ij} is represented as a pair (e_{ij}, f_{ij}) of binary integers so that

$$a_{ij} = 2^{e_{ij}} f_{ij}$$

and e_{ij}, f_{ij} are at most m-bit numbers. ♢

End Exercises

References

[1] N. Alon, J. H. Spencer, and P. Erdös. *The probabilistic method*. John Wiley and Sons, Inc, 1992.