Probability in computing

Péter Gács

Computer Science Department
Boston University

Fall 17

Course structure: see the syllabus on
www.cs.bu.edu/~gacs/courses/cs537.

Event space, probability space

Event space: a pair $(\Omega, \mathcal{A})$ where Ω is a set. Elements $\omega \in \Omega$ are called outcomes. Further, $\mathcal{A}$ is a set of subsets of Ω called events.

Example 1.1 If Ω is a countable set then we will always assume that $\mathcal{A} = 2^{\Omega}$, that is every subset of Ω is an event.

In general, we assume that $\mathcal{A}$ is an algebra: $\emptyset$, Ω are in $\mathcal{A}$ and if E, F are in $\mathcal{A}$ then so is $E \setminus F$ (and then of course $E \cup F$ and $E \cap F$).

Example 1.2 $\Omega = \mathbb{R}$ and $\mathcal{A}$ is the set of all unions of a finite set of intervals (closed, open, half-closed).

Remark: normally it is also required that $\mathcal{A}$ is closed with respect to countable union, but we do not need this in the course.

Given an event space $(\Omega, \mathcal{A})$ and a function $\mathsf{P} : \mathcal{A} \to [0, 1]$, we call P a probability measure if $\mathsf{P}(\Omega) = 1$ and for disjoint events E, F we have $\mathsf{P}(E \cup F) = \mathsf{P}(E) + \mathsf{P}(F)$. In this case, the triple

$$(\Omega, \mathcal{A}, \mathsf{P})$$

is called a probability space.

Examples 1.3

- Ω is a finite set, $\mathcal{A} = 2^{\Omega}$ and and $\mathsf{P}(E) = |E|/|\Omega|$.
- $\Omega = \mathbb{R}^2$, (plane). Let $\mathcal{A}$ be the smallest algebra containing all the rectangles. Let $f : \mathbb{R}^2 \to \mathbb{R}$ be an integrable function with $\int_{\mathbb{R}^2} f(x)dx = 1$. For all elements $E \in \mathcal{A}$ we defined $\mathsf{P}(E) = \int_E f(x)dx$. In this case we say that $f(x)$ is the density function of the probability measure P.

Conditional probability

Let $(\Omega, \mathcal{A}, \mathsf{P})$ be a probability space and E, F events. If $\mathsf{P}(E) > 0$ then $\mathsf{P}(F \mid E) = \mathsf{P}(E \cap F)/\mathsf{P}(E)$ is called the conditional probability of event F with respect to E.
If $E_1, \ldots, E_n$ are disjoint events with $\bigcup_i E_i = \Omega$ then we will call the set $\{E_1, \ldots, E_n\}$ a partition. If $\{E_1, \ldots, E_n\}$ is a partition and F is an event then the following relations will be often used. The last one is called the law of total probability.

$$\begin{aligned} F &= (F \cap E_1) \cup \cdots \cup (F \cap E_n), \\ \mathsf{P}(F) &= \mathsf{P}(F \cap E_1) + \cdots + \mathsf{P}(F \cap E_n), \\ &= \mathsf{P}(F \mid E_1)\,\mathsf{P}(E_1) + \cdots + \mathsf{P}(F \mid E_n)\,\mathsf{P}(E_n). \end{aligned}$$

Example 1.4 Let X be a number that is prime with probability 0.1, pseudo-prime with probability 0.2 and other with probability 0.7. We perform a prime test which is always correct when the number is prime, gives wrong answer with probability 0.02 if the number is pseuodprime and with probability 0.01 if the number is other. What is the probability of wrong answer?

$$
\begin{aligned}
&\mathsf{P}(\text{wrong}) \\
&= \mathsf{P}(\text{wrong} \mid \text{prime})\,\mathsf{P}(\text{prime}) \\
&+ \mathsf{P}(\text{wrong} \mid \text{ps-prime})\,\mathsf{P}(\text{ps-prime}) \\
&+ \mathsf{P}(\text{wrong} \mid \text{other})\,\mathsf{P}(\text{other}) \\
&= 0.1 \cdot 0 + 0.2 \cdot 0.02 + 0.7 \cdot 0.01.
\end{aligned}
$$

Independence

Events $A_1, \ldots, A_n$ are independent when for every possible subsequence $1 \leq i_1 < \cdots < i_k \leq n$ we have

$$\mathsf{P}(A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}) = \mathsf{P}(A_{i_1})\,\mathsf{P}(A_{i_2}) \cdots \mathsf{P}(A_{i_k}).$$

In particular, A, B are independent if $\mathsf{P}(A \cap B) = \mathsf{P}(A)\,\mathsf{P}(B)$. If $\mathsf{P}(B) > 0$ this can be written as

$$\mathsf{P}(A \mid B) = \mathsf{P}(A).$$

Note that pairwise independence is weaker.

Example 1.5 Toss an unbiased coin twice, let A say that the first toss is head, B that the second toss is head and C that both are heads or both are tails. These three events are only pairwise independent.

Random variables

For a probability space $(\Omega, \mathcal{A}, \mathsf{P})$, a random variable is a function $X : \Omega \to \mathbb{R}$ with the property that each set of the form $\{\, \omega : X(\omega) < a \,\}$ and $\{\, \omega : X(\omega) > a \,\}$ is an event. We will write

$$\mathsf{P}\{\, X < a \,\} = \mathsf{P}\{\, \omega : X(\omega) < a \,\}.$$

If X is a random variable then the function

$$F(a) = \mathsf{P}\{\, X < a \,\}$$

is called the distribution function of X.

Example 1.6 Let A be an event, then we define the random variable 1_A called the indicator variable of the event A:

$$1_A(\omega) = \begin{cases} 1 & \text{if } \omega \in A, \\ 0 & \text{otherwise.} \end{cases}$$

So $1_A = 1$ if the event A occurs and 0 otherwise. Distribution function: $F_A = F_{1_A}$:

$$F_A(x) \begin{cases} 0 & \text{if } x \leq 0, \\ 1 - \mathsf{P}(A) & \text{if } x \leq 1, \\ 1 & \text{otherwise.} \end{cases} .$$

If two random variables X, Y are on the same probability space then they have a joint distribution: for any nice subset E of the plane $\mathbb{R}^2$, the set

$$\{\,\omega : (X(\omega), Y(\omega)) \in E\,\}$$

is an event, so we may ask the probability $\mathsf{P}\{\,(X, Y) \in E\,\}$.

Example 1.7 X is my arrival time to class, Y is the last arrival time of the a student. What is $\mathsf{P}\{\,X - Y < 10 \text{ minutes}\,\}$?

Let $X_1, \ldots, X_n$ be random variables on the same probability space. We say that they are independent if for all sequences $a_1, \ldots, a_n \in \mathbb{R}$ the events

$$\{\,\omega : X_1(\omega) < a_1\,\}, \ldots, \{\,\omega : X_n(\omega) < a_n\,\}$$

are independent.

Example 1.8 We toss a die twice. The outcomes ω are $(1, 1), (1, 2), \ldots, (6, 6)$ with probabilities $1/36$ for each. Let X_1 be value of the outcome of the first toss, and X_2 the value of the second toss. Then X_1, X_2 are independent. On the other hand, X_1 is not independent of $X_1 + X_2$.

Note: if $X_1, \ldots, X_n$ are independent and f_i are any functions then $f_1(X_1), \ldots, f_n(X_n)$ are also independent. Also, if X, Y, Z are independent and $f(x), g(y, z)$ are functions then $f(X), g(Y, Z)$ are independent.

Binomial random variable

We repeat some experiment n times. Each time it succeeds with probability p and fails with probability $1 - p$. Let $X_{n,p}$ be the random variable counting the number of successes. We have

$$\mathsf{P}\left\{ X_{n,p} = k \right\} = \binom{n}{k} p^k (1-p)^{n-k}.$$

This is called the binomial random variable.
If we make $p = \lambda / n$ and and $n \to \infty$ then this converges to

$$e^{-\lambda} \frac{\lambda^k}{k!},$$

which is $\mathsf{P}\{ Y_\lambda = k \}$ for the so-called Poisson random variable.

Expected value

Suppose that X is a random variable that takes on a discrete set of values $a_1, a_2, \ldots$ with probabilities $p_1, p_2, \ldots$. Then its expected value is defined as

$$\mathsf{E}\, X = p_1 a_1 + p_2 a_2 + \cdots .$$

Does not always exist.

Examples 1.9

1. Let $\mathsf{P}\{X = 1\} = p$, $\mathsf{P}\{X = 0\} = 1 - p$. Then $\mathsf{E}\, X = p$.
2. Toss a die a single time: the expected value is

$$1 \cdot (1/6) + \cdots + 6 \cdot (1/6).$$

For a probability space $\Omega = \mathbb{R}^d$ with a density function $p(x)$ and a random variable $X(\omega)$ we define $\mathsf{E}\, X = \int X(\omega) p(\omega) d\omega$.

The expected value is additive:

$$\mathsf{E}(\alpha X + \beta Y) = \alpha \,\mathsf{E}\, X + \beta \,\mathsf{E}\, Y,$$

even if X, Y are not independent.

Example 1.10 We toss a die twice, and win the sum of the points if the two tosses are equal. Let $X_i = 2i$ if both tosses are equal to i. Then $\mathsf{E}\, X_i = (1/36)(2i)$. Our expected win is

$$\mathsf{E}(X_1 + \cdots + X_6) = \mathsf{E}\, X_1 + \cdots + \mathsf{E}\, X_6 = (1/36) \cdot 2 \cdot (1 + \cdots + 6).$$

Examples 1.11

1. The binomial variable with parameters n, p has expected value np, since it is the sum of n variables with expectation p.
2. Correspondingly, the Poisson variable with parameter λ has expected value λ.
3. Consider a fair game of tossing a coin repeatedly and betting on head each time. Play until you win, doubling your bet in every step: in step n it is 2^n.
   Expected win is 1.
   Expected maximum loss before winning is ∞.

Conditional expectation, multiplication

- The conditional expectation $\mathsf{E}\{X \mid A\}$ of a random variable X with respect to some event A is defined using the conditional probabilities with respect to A. There is a law of total expectation—derive it from the law of total probability above! See the example of geometric variable below.
- If $X_1, \ldots, X_n$ are independent random variables then the expected value is also multiplicative:

$$\mathsf{E}\, X_1 \cdots X_n = \mathsf{E}\, X_1 \cdots \mathsf{E}\, X_n.$$

Example: for independent events A, B,

$$\mathsf{E}\, 1_A 1_B = \mathsf{P}(A \cap B) = \mathsf{P}(A)\,\mathsf{P}(B) = \mathsf{E}\, 1_A\, \mathsf{E}\, 1_B.$$

- Converse: if $\mathsf{E} f_1(X_1) \cdots f_n(X_n) = \mathsf{E} f_1(X_1) \cdots \mathsf{E} f_n(X_n)$ for all functions $f_1, \ldots, f_n$ then $X_1, \ldots, X_n$ is independent.

Markov inequality

An important tool for estimating probabilities with the help of expectations.

Theorem 1.12 Let X be a nonnegative random variable, $\lambda > 0$, with $\mathsf{E}\, X = \mu$, then

$$\mathsf{P}\{ X > \lambda \mu \} < \frac{1}{\lambda}.$$

The proof is simple. Another form of the inequality:
$\mathsf{P}\{ X > \lambda \} < \mu/\lambda$.

Corollary 1.13 If X has finite expected value then its tail probability $\mathsf{P}\{ X > \lambda \}$ decreases as $O(\lambda^{-1})$.

In other words, if the tail decreases slower, say as $\lambda^{-1/2}$ ("heavy tail"), then it has no finite expected value

Geometric distribution, restart

Suppose that some attempt has probability α of succeeding. You keep repeating the attempts until the first success. The number of necessary repetitions is a random variable X, with

$$\mathsf{P}\{ X = n \} = \alpha(1-\alpha)^{n-1}.$$

Let Y_k be the variable that counts which experiment succeeds first, but starting only from the kth one, so its distribution is defined by

$$\mathsf{P}\{ Y_k > n \} = \mathsf{P}\{ X > k+n \mid X > k \} = \frac{(1-\alpha)^{n+k}}{(1-\alpha)^k} = (1-\alpha)^n.$$

So Y_k is also a geometric random variable with the same parameter α. We call this the memoryless property, of geometric random variables.

Expectation of the geometric

Let X be a geometric random variable with parameter α.

$$\mu = \mathsf{E}\, X = \sum_{k=1}^{\infty} k \cdot \alpha(1-\alpha)^{k-1} = 1 \cdot \alpha + 2 \cdot \alpha(1-\alpha) + \cdots .$$

Let Y_1 be the variable introduced above, then by the memoryless property $\mathsf{E}\, Y_1 = \mathsf{E}\, X = \mu$. Let S be the event $X \leq 1$, then $\mathsf{E}\{X \mid \neg S\} = 1 + \mathsf{E}\, Y_1 = 1 + \mu$. By the law of total expectation (similar to the law of total probability):

$$\mu = \mathsf{P}(S)\, \mathsf{E}\{X \mid S\} + (1 - \mathsf{P}(S))\, \mathsf{E}\{X \mid \neg S\} = \alpha + (1-\alpha)(1+\mu).$$

Solving the equation gives $\mu = 1/\alpha$. So, if the success probability of each attempt is $1/10$, we can expect to succeed on the tenth attempt (on average).

Sum formula for expectation

Another expression for expected value, useful and easy to check: Let X be a random variable taking positive integer values. Then

$$\mathsf{E}\, X = \mathsf{P}\{ X \geq 1 \} + \mathsf{P}\{ X \geq 2 \} + \mathsf{P}\{ X \geq 3 \} + \cdots .$$

We could have used this also to compute the expectation of the geometric variable. The corresponding formula for the continuous case (works also in the discrete case):

$$\mathsf{E}\, X = \int_0^\infty \mathsf{P}\{ X \geq z \}\, \mathrm{d}z.$$

Inner product

- If $\boldsymbol{u} = (u_1, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, \ldots, v_n)$ are vectors then their inner product is $u_1 v_1 + \cdots + u_n v_n$. It is symmetric, bilinear, and has the property that $\langle \boldsymbol{u}, \boldsymbol{u} \rangle = 0$ implies $\boldsymbol{u} = \boldsymbol{0}$.
- Random variables over a probability space can be viewed as vectors: there is an operation of linear combination $\lambda X + \mu Y$. Example subspace: those random variables X with $\mathsf{E}\, X = 0$. We introduce an inner product:

$$\langle X, Y \rangle = \mathsf{E}\, XY.$$

it is bilinear, and $\langle X, X \rangle \geq 0$ with $\langle X, X \rangle = 0$ iff $\mathsf{P}\{ X = 0 \} = 1$.

Example 1.14 Let 1_A, 1_B be the indicator variables of events A, B.

$$\mathsf{E}\, 1_A 1_B = \mathsf{P}(A \cap B).$$

- If $\boldsymbol{u} = (u_1, \ldots, u_n)$ is a vector then its length is defined as $|\boldsymbol{u}| = \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle} = (u_1^2 + \cdots + u_n^2)^{1/2}$ (Pythagorean Theorem). The inner product can then be expessed as

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = |\boldsymbol{u}| \cdot |\boldsymbol{v}| \cdot \cos\theta \tag{1.1}$$

  where θ is the angle between $\boldsymbol{u}$ and $\boldsymbol{v}$.
- Two vectors $\boldsymbol{u}, \boldsymbol{v}$ are called orthogonal (perpendicular) if $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$.

The cosine of an angle is ≤ 1, so (1.1) implies

$$(\langle \boldsymbol{u}, \boldsymbol{v} \rangle)^2 \leq \langle \boldsymbol{u}, \boldsymbol{u} \rangle \langle \boldsymbol{v}, \boldsymbol{v} \rangle.$$

This is called the Cauchy-Schwartz inequality. For random variables, it says:

Theorem 1.15 (Cauchy-Schwartz)

$$(\mathsf{E}\, XY)^2 \leq \mathsf{E}\, X^2\, \mathsf{E}\, Y^2.$$

Covariance

For X, Y with $\mathsf{E}\,X = \mu$, $\mathsf{E}\,Y = \nu$ we define the covariance

$$\mathrm{Cov}(X, Y) = \langle X - \mu, Y - \nu \rangle = \mathsf{E}(X - \mu)(Y - \nu).$$

X, Y are uncorrelated if their covariance is 0, in other words if $X - \mu$ and $Y - \nu$ are orthogonal.

- If X, Y are independent then they are uncorrelated. Indeed, let $X' = X - \mu$, $Y' = Y - \nu$. Then $\mathsf{E}\,X' = \mathsf{E}\,Y' = 0$, and X', Y' are independent. therefore $\mathsf{E}(X'Y') = \mathsf{E}\,X'\,\mathsf{E}\,Y' = 0$.
- The converse does not hold in general: uncorrelatedness is just one equation, independence is many equations. But if $f(X)$, $g(Y)$ are uncorrelated for all f, g then X, Y are independent.

Example 1.16

$$\begin{aligned} \mathsf{P}(X = Y = 0) &= 1/3, \\ \mathsf{P}(X = Y = -1) = \mathsf{P}(X = Y = 1) &= \mathsf{P}(X = -1, Y = 1) \\ &= \mathsf{P}(X = 1, Y = -1) = 1/6. \end{aligned}$$

These variables are uncorrelated but not independent.

On the other hand, if X, Y only take 2 values and are uncorrelated, then they are independent.
Let $X = 1$ with probability p and 0 with probability $(1 - p)$. Let $Y = 1$ with probability q and 0 with probability $(1 - q)$. Indeed,

$$\mathsf{E}(X - p)(Y - q) = \mathsf{E}\,XY - p\,\mathsf{E}\,Y - q\,\mathsf{E}\,X + pq = \mathsf{E}\,XY - pq.$$

If this is 0 then $pq = \mathsf{E}\,XY = \mathsf{P}\{X = 1 \wedge Y = 1\}$, showing that X, Y are independent.

Variance

If X is a random variable $\mu = \mathsf{E}\, X$, with $X' = X - \mu$ then we define

$$\mathsf{Var}\, X = \mathrm{Cov}(X, X) = \mathsf{E}(X - \mu)^2.$$

Equivalent expression: $\mathsf{Var}\, X = \mathsf{E}\, X^2 - \mu^2$.
Since $\mathsf{Var}(\lambda X) = \lambda^2\, \mathsf{Var}(X)$, we introduce the quantity $\sqrt{\mathsf{Var}\, X}$ and call it the standard deviation σ_X.

Theorem 1.17 $\mathsf{Var}(X+Y) = \mathsf{Var}\,X + \mathsf{Var}\,Y + 2\mathrm{Cov}(X, Y)$.

Thus if $\mathrm{Cov}(X, Y) = 0$ then $\mathsf{Var}(X+Y) = \mathsf{Var}\,X + \mathsf{Var}\,Y$. In particular, this is true if X, Y are independent.
More generally, if $X_1, \ldots, X_n$ are pairwise uncorrelated then

$$\mathsf{Var}(X_1 + \cdots + X_n) = \mathsf{Var}\,X_1 + \cdots + \mathsf{Var}\,X_n.$$

Application: let $X_1, \ldots, X_n$ be uncorrelated, identically distributed, with $\sigma^2 = \mathsf{Var}\,X_i$, $S_n = X_1 + \cdots + X_n$. Then

$$\mathsf{Var}\,S_n = n\sigma^2.$$

Note that $\mathsf{Var}\,X$ is not a linear function of X, it is not even homogenous: the standard deviation is homogenous. But, the standard deviation of S_n grows only as $\sqrt{n}$.

Chebyshev inequality

Let $\mu = \mathsf{E}\, X$, $\sigma^2 = \mathsf{Var}\, X$. The new inequality comes from the Markov inequality applied to the variable $(X - \mu)^2$. It says, for $\lambda > 0$:

$$\mathsf{P}\{\, |X - \mu| > \lambda\sigma \,\} < \frac{1}{\lambda^2}.$$

- Another form of the inequality: $\mathsf{P}\{\, |X - \mu| > \lambda \,\} < \sigma^2/\lambda^2$.
- Markov's inequality implies that if X has a finite expected value then by its tail $\mathsf{P}\{\, X > \lambda \,\}$ decreases at least as $O(1/\lambda)$.
- Chebyshev's inequality says that if X also has a finite variance then we can say more: the tail decreases at least as fast as $O(1/\lambda^2)$.

Variance of the geometric

For a geometric variable with parameter α, a trick similar to the one used for the expectation computes

$$\mathsf{Var}\, X = 1/\alpha^2 - 1/\alpha$$

Let $S_n = X_1 + \cdots + X_n$.

Theorem 1.18 (Law of large numbers) For each n, let $X_1, \ldots, X_n$ be pairwise uncorrelated, with $\mathsf{E}\, X_i = \mu, \mathsf{Var}\, X_i = \sigma^2$. Then for all $\varepsilon > 0$ we have

$$\lim_{n\to\infty} \mathsf{P}\{\, |S_n/n - \mu| > \varepsilon \,\} = 0.$$

Indeed, by Chebyshev's inequality for $\lambda > 0$:

$$\mathsf{P}\left\{\, |S_n - n\mu| > \lambda\sigma\sqrt{n} \,\right\} < 1/\lambda^2,$$
$$\mathsf{P}\left\{\, |S_n/n - \mu| > \lambda\sigma/\sqrt{n} \,\right\} < 1/\lambda^2,$$
$$\mathsf{P}\{\, |S_n/n - \mu| > \varepsilon \,\} < \frac{\sigma^2}{n\varepsilon^2}$$

where we chose $\lambda = \varepsilon\sqrt{n}/\sigma$.

Thus, for large λ the value S_n is mostly in the interval $n\mu \pm \lambda\sigma\sqrt{n}$.

Example 1.19 Let $X_i = 1$ with probability p and 0 otherwise: when a biased coin falls head up. Then $\mathsf{E}\, X_i = p$, $\mathsf{Var}\, X_i = p(1-p)$. Further S_n is the number of heads in n tosses. The inequality says that with large probability this number is within

$$np \pm \lambda\sqrt{np(1-p)}.$$

This interprets probability in terms of relative frequency.

Central limit theorem

Can we make the interval tighter then $\pm c\sqrt{n}$ in the law of large numbers? No, this is shown by the theorem below. A random variable X is called standard if $\mathsf{E}\, X = 0$, $\mathsf{Var}\, X = 1$. If $\mathsf{E}\, X = \mu$ and $\mathsf{Var}\, X = \sigma^2$ then $X' = (X - \mu)/\sigma$ is standard, it is called the standardized version of X.

Let $X_1, \ldots, X_n$ be independent, identically distributed with $\mathsf{E}\, X_i = \mu$, $\mathsf{Var}\, X_i = \sigma^2$, $S_n = X_1 + \cdots + X_n$. We are interested in the distribution of the standardized version $(S_n - \mu n)/\sigma n$ of S_n.

Let $\phi(x) = (2\pi)^{-1/2} e^{-x^2/2}$ be the density function of the so-called standard Gaussian distribution, and let $\Phi(x) = \int_{-\infty}^{x} \phi(y)dy$.

Theorem 1.20 (Central limit theorem) We have

$$\lim_{n\to\infty} \mathsf{P}\left\{ \frac{S_n - \mu n}{\sigma\sqrt{n}} < a \right\} = \Phi(a).$$

It is remarkable that this limit distribution, the Gaussian distribution, does not depend on the distribution of X_i. The proof of this theorem (under the condition that $\mathsf{E}\,|X_i|^3 < \infty$) is given in separate notes.

Note 1 The theorem shows that just as it is unlikely for $|S_n - \mu n|$ to be much larger than $\sqrt{n}$, it is also unlikely for it to be much smaller! If somebody shows us a 0-1 sequence $X_1, \ldots, X_n$ where the number of 0's and 1's is much closer to each other than $\sqrt{n}$, then we should doubt that it came from coin-tossing.

Exponential convergence

Can we improve the $O(1/n)$ bound on the tail probability in the law of large numbers? Yes, there are theorems of the sort

$$\mathsf{P}\{\,|S_n/n - \mu| > \varepsilon\,\} < e^{-nf(\varepsilon)},$$

that is the probablity of S_n/n to be outside some constant-size interval around μ should be exponentiall small. Such theorems are sometimes so-called "Chernoff bound", "Hoeffding bound", or "large deviation theorem". For these theorems, mutual independence is essential (identical distribution of the X_i id not). For one such theorem, assume $0 \le X_i \le 1$ (if this is not the case, rescale), $\mu_i = \mathsf{E}\, X_i$, $\mu = \frac{1}{n}\sum_i \mu_i$.

Theorem 1.21 ("Chernoff" bound)

$$\mathsf{P}\{\,S_n - \mu n > \varepsilon n\,\} < e^{-2\varepsilon^2 n},$$

$$\mathsf{P}\{\,\mu n - S_n > \varepsilon n\,\} < e^{-2\varepsilon^2 n}.$$

Transformation to another range Suppose we want $-1 \le X_i \le 1$. Then we can apply the bound to $X'_i = (X_i + 1)/2$. that is $X_i = 2X'_i - 1$. If $S_n > \mathsf{E}\, S_n + \varepsilon n$ then $2S'_n - n > 2\,\mathsf{E}\, S'_n - n + \varepsilon n$, that is $S'_n > \mathsf{E}\, S'_n + \varepsilon/2$. By the Chernoff bound this has probability bound

$$e^{-2(\varepsilon/2)^2 n} = e^{-\varepsilon^2 n/2}. \tag{1.2}$$

Mutual independence is essential For pairwise independent variables the $O(1/n)$ tail bound is best possible. Indeed, in a homework we saw an example of n such variables X_i with $\mathsf{P}\{X_i = 1\} = \mathsf{P}\{X_i = 0\} = 1/2$, and $\mathsf{P}\{X_1 = \cdots = X_n = 0\} = \frac{1}{n+1}$. This gives $\mathsf{E}\, S_n = n/2$ but $\mathsf{P}\{S_n = 0\} = \frac{1}{n+1}$.

The theorem and its proof uses the following functions.

$$H(\lambda) = -\lambda \log \lambda - (1-\lambda)\log(1-\lambda) \geq 0$$

is called the entropy of the distribution $(\lambda, 1-\lambda)$, and

$$H_\mu(\lambda) = \overline{H}_\mu(\lambda)/\ln 2 = \lambda \log \frac{\mu}{\lambda} + (1-\lambda)\log \frac{1-\mu}{1-\lambda}.$$

is called its relative entropy with respect to $(\mu, 1-\mu)$.
Note that $H_{1/2}(\lambda) = H(\lambda) - 1$. When log is replaced with ln, we denote

$$\overline{H}(\lambda) = H(\lambda)\ln 2 = -\lambda \ln \lambda - (1-\lambda)\ln(1-\lambda),$$

and so on. Concavity of the logarithm function proves

Theorem 1.22 If $\lambda \neq \mu$ then $H_\mu(\lambda) < 0$.

We will derive the "Chernoff" bound from the following theorem.

Theorem 1.23 Let $\lambda - \mu = \varepsilon$, then

$$\lambda > \mu \Rightarrow \mathsf{P}\{ S_n/n > \lambda \} \leq e^{n\overline{H}_\mu(\lambda)} \leq e^{-2n\varepsilon^2}, \tag{1.3}$$

$$\lambda < \mu \Rightarrow \mathsf{P}\{ S_n/n < \lambda \} \leq e^{n\overline{H}_\mu(\lambda)} \leq e^{-2n\varepsilon^2}, \tag{1.4}$$

For the proof, fix a positive number b, and compute, using independence:

$$\mathsf{E}\, b^{S_n} = \mathsf{E}\, b^{X_1} \cdots \mathsf{E}\, b^{X_n}.$$

For $\mu < \lambda < 1$, choose $b > 1$, then $S_n > \lambda n \Leftrightarrow b^{S_n} > b^{\lambda n}$. For $0 < \lambda < \mu$, choose $b < 1$, then $S_n < \lambda n \Leftrightarrow b^{S_n} > b^{\lambda n}$. In both cases, by Markov's inequality,

$$\mathsf{P}\left\{ b^{S_n} > b^{\lambda n} \right\} < b^{-\lambda n}\, \mathsf{E}\, b^{S_n} = b^{-\lambda n} \prod_{i=1}^{n} \mathsf{E}\, b^{X_i}.$$

Simplify, using that b^x, a convex function of x, is below the chord in $0 \le x \le 1$:

$$b^x \le 1 + x(b-1).$$

Taking expectation: $\mathsf{E}\, b^{X_i} \le 1 + \mu_i(b-1)$.

By the inequality of arithmetic and geometric mean:

$$\prod_i (1 + \mu_i(b-1)) \leq \left(\frac{1}{n} \sum_i (1 + \mu_i(b-1)) \right)^n$$
$$= (1 + \mu(b-1))^n.$$

Multiplying with $b^{-\lambda}$, our bound is $(g(b))^n$ where

$$g(b) = b^{-\lambda}(1 + \mu(b-1)) = (1-\mu)b^{-\lambda} + \mu b^{1-\lambda},$$

a convex function. Keeping μ fixed, choose b to minimize this by differentiation:

$$b^* = \frac{\lambda(1-\mu)}{(1-\lambda)\mu},$$

which is indeed < 1 if $\lambda < \mu$ and > 1 if $\lambda > \mu$.

Substitution gives

$$g(b^*) = \left(\frac{\mu}{\lambda}\right)^{\lambda}\left(\frac{1-\mu}{1-\lambda}\right)^{1-\lambda} \leq \lambda\frac{\mu}{\lambda} + (1-\lambda)\frac{1-\mu}{1-\lambda} = 1,$$

where we used the arithmetic-geometric inequality again. This inequality is strict unless $\lambda = \mu$. We can write $g(b^*) = e^{\overline{H}_\mu(\lambda)}$. The inequality $\overline{H}_\mu(\lambda) < -2\varepsilon^2$ is is proved (for example in a paper by Hoeffding) by analyzing this (concave) function of λ around $\lambda = \mu$ or possibly even choosing a more convenient (though not optimal) value for b.

For $\lambda > \mu$ we have

$$\overline{H}_\mu(\lambda) = \lambda \ln \frac{\mu}{\lambda} + (1-\lambda) \ln \frac{1-\mu}{1-\lambda} < \lambda \ln \frac{\mu}{\lambda} + \lambda .$$

Indeed, ignore the $1-\mu$ and use the inequality $(1-\lambda) \ln \frac{1}{1-\lambda} \le \lambda$ (homework). This results in the upper bound

$$\mathsf{P}\{ S_n/n > \lambda \} < \left(\frac{e\mu}{\lambda} \right)^{\lambda n}, \tag{1.5}$$

which is useful only if $\lambda > e\mu$.

Let $X_i = 1$ with probability $1/2$ and 0 otherwise, then $\mu = 1/2$. Let $\lambda < 1/2$. Then (1.4) implies

$$2^{-n} \sum_{k \le \lambda n} \binom{n}{k} = \mathsf{P}\{ S_n < \lambda n \} \le 2^{nH_\mu(\lambda)},$$

$$\sum_{k \le \lambda n} \binom{n}{k} \le 2^{nH(\lambda)},$$

a very useful inequality. For small λ, we can again simplify using $(1-\lambda) \ln \frac{1}{1-\lambda} \le \lambda$:

$$\sum_{k \le \lambda n} \binom{n}{k} \le \left(\frac{e}{\lambda}\right)^{n\lambda}, \qquad \sum_{k \le m} \binom{n}{k} \le \left(\frac{ne}{m}\right)^{m}. \tag{1.6}$$

A more general version of "Chernoff's" bound uses a quantity

$$v(a, b) = (b - a)^2/4,$$

Lemma 1.24 For a random variable X with $a \le X \le b$ we have $\mathsf{Var}(X) \le (b-a)^2/4$. This bound is achieved for $\mathsf{P}(X = a) = \mathsf{P}(X = b) = 1/2$.

The proof is an exercise. For independent random variables X_i with $a_i \le X_i \le b_i$, let $v_n = \sum_{i=1}^n v(a_i, b_i)$. Then $\mathsf{Var}(S_n) \le v_n$ (equal when $X_i = a_i$ or b_i with probability $1/2$ each).

Theorem 1.25 (Hoeffding) For all $\lambda > 0$:

$$\mathsf{P}\left\{ S_n - n\mu \ge \lambda \sqrt{v_n} \right\} \le e^{-\lambda^2/2}.$$

Alternatively, $\mathsf{P}\{ S_n - n\mu \ge \lambda \} \le e^{-\lambda^2/2v_n}$.

Let us apply the alternative form to the case $a_i = 0$, $b_i = 1$, then $v_n = n/4$. Choosing $\lambda = \varepsilon n$:

$$\mathsf{P}\{ S_n - n\mu > \varepsilon n \} \leq e^{-\frac{\varepsilon^2 n^2}{n/2}} = e^{-2\varepsilon^2 n}.$$

Application to the set balancing problem

We have sets $A_1, \ldots, A_m \subseteq U = \{1, 2, \ldots, n\}$. We want to paint the elements of U red or blue in such a way that in each set A_i, the number of blue and red elements balances out as much as possible. That is if B is the set of blue elements then we want the (maximum) discrepancy

$$\max_{i=1}^{m} ||A_i \cap B| - |A_i \setminus B||$$

to be minimal. Other language: we are looking for a sequence $\boldsymbol{b} = (b_1, \ldots, b_n)$, $b_i = \pm 1$, for which

$$\max_i |\sum_{j \in A_i} b_j|$$

is minimal.

Let us see how good a vector $\boldsymbol{b}$ we can get by random choice. Let $\beta_1, \ldots, \beta_n$ be independent, with
$\mathsf{P}\left\{ \beta_j = 1 \right\} = \mathsf{P}\left\{ \beta_j = -1 \right\} = 1/2$.
Each $\sum_{j \in A_i} \beta_j$ is a sum of independent random variables. Let us fix a parameter t, to be chosen later. Two cases.

- $|A_i| \le t$. Then $|\sum_{j \in A_i} \beta_j| \le t$.
- $|A_i| > t$. By the Chernoff bound (1.2), since $\mathsf{E}\, \beta_j = 0$, with $k = |A_i|$:

$$\mathsf{P}\left\{ |\sum_{j \in A_i} \beta_j| > t \right\} \le 2e^{-\frac{1}{2}(t/k)^2 k} = 2e^{-\frac{1}{2}t^2/k} \le 2e^{-t^2/(2n)}.$$

Choose $t = \sqrt{4n \ln m}$, then this is $2/m^2$. So with probability $1 - 2/m$, all m sets A_i have discrepancy $\le t$.

Can one do better?

Yes, but not by just random choice. Spencer achieves a discrepancy of

$$O\left(\sqrt{n \ln(m/n)}\right).$$

(See the Alon-Spencer book.) For $m = n$ this is $O(\sqrt{n})$ while the random choice gives $O(\sqrt{n \ln n})$.

A homework will show that $O(\sqrt{n})$ for $m = O(n)$ cannot be improved.

Generating a random variable

Suppose some random number generator provides a random variable U distributed uniformly over the interval $[0, 1]$. We can compute from this a random variable X with any given cumulative distribution function $F(a)$. First, we define the inverse function $G(y) = F^{-1}(y)$. Some care is needed since $F(x)$ is not strictly monotonic in general (but is left-continuous):

$$G(y) = \sup\{\, x : F(x) \le y \,\}.$$

(Check that $G(y)$ is right-continuous: $G(y) = \lim_{y' \searrow y} G(y')$.)

Claim 2.1 The variable $X = G(U)$ has the distribution $F(x)$.

Indeed, $G(U) < x \Leftrightarrow \forall x' \big(F(x') \le U \Rightarrow x' < x \big) \Leftrightarrow F(x) > U$.
On the other hand, $\mathsf{P}\{\, F(x) > U \,\} = F(x)$.

Examples 2.2

- Let $x_1 < \cdots < x_n$ be some values, and let X be a random variable with $\mathsf{P}\{ X = x_i \} = p_i$ where $\sum_i p_i = 1$. For $i = 0, 1, \ldots, n$ let $q_i = \sum_{j \le i} p_j$ (so $q_0 = 0$, $q_n = 1$). The distribution function is

$$\mathsf{P}\{ X < a \} = F_X(a) = \sum_{i : x_i < a} p_i = \max_{i : x_i < a} q_i,$$

$$G_X(u) = \sup\{ a : F_X(a) \le u \} = \max\{ x_i : q_{i-1} \le u \},$$

  so outputting x_i if $q_{i-1} \le U < q_i$ we get the distribution of X.

- Let Y be an exponential random variable: $\mathsf{P}\{ Y < a \} = F_Y(a) = 1 - e^{-\beta a}$. To invert this function, let $u = 1 - e^{-\beta a}$, and express a via u: $a = -\beta^{-1} \ln(1 - u)$, so outputting $G_Y(U) = -\beta^{-1} \ln(1 - U)$ we get the distribution of Y.

Randomization or average case

Algorithms can be analyzed probabilistically from several points of view. First distinction:

1. The algorithm is deterministic, but we analyze it on random inputs. This approach 1 is less frequently used, since we rarely have reliable information about the distribution of our inputs. (Levin's theory of problems that are hard on average adresses general questions of this type.)
2. We introduce randomness during computation, but the input is fixed. Most practical uses of randomness belong to category 2, randomization.

We will see later a connection (Yao's theorem) between approaches 1 and 2 from the point of view of worst-case analysis. (It is, of course, also possible to randomize and analyze on random inputs.)

Example 3.1 (Quicksort) The deterministic quicksort algorithm has a quadratic worst-case performance. Its average-case performance is good, but there is no reason to assume that the permutation we have to sort is "random". In practice, getting a reversely ordered file is more likely than getting a completely disordered one.
On the other hand, a randomized version provides us the same benefits as a random input would (see below).

Example 3.2 (Simplex algorithm) It has been known for some time, that the simplex algorithm of linear programming performs well on random inputs. But it is much less clear, how to turn this observation into a well-performing randomized version of the simplex method.

Quicksort

Given an array $A = (a_1, \ldots, a_n)$, we want to sort it using comparisons. The recursive algorithm

$$\text{Quicksort}(i, k, A).$$

sorts elements $a_i, \ldots, a_j$ if $i \leq k$. It uses a subroutine called

$$\text{Partition}(i, j, k, A), \qquad i < k, \quad i \leq j \leq k.$$

This will take pivot element a_j and rearrange $a_i, \ldots, a_k$ comparing them with a_j, putting all elements less than a_j before it and all elements greater after it.
Now Quicksort(i, k, A) works as follows. It does nothing if $i \geq k$, else it chooses some element $i \leq j \leq k$ and returns $(A', j') = \text{Partition}(i, j, k, A)$ where j' is the new position of a_j and A' is the new order. Then it applies Quicksort$(i, j' - 1, A')$ and Quicksort$(j' + 1, k, A')$.

Let the sorted order be $z_1 < z_2 < \cdots < z_n$. If $i < j$ then let

$$Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}.$$

Let the random variable C_{ij} be defined to be 1 if z_i and z_j will be compared sometime during the sort, and 0 otherwise.
Every comparison happens during some partition, with the pivot element. Let π_{ij} be the first (random) pivot element entering Z_{ij}. A little thinking shows:

Lemma 4.1 We have $C_{ij} = 1$ if and only if $\pi_{ij} \in \{z_i, z_j\}$. Also, for every $x \in Z_{ij}$, we have

$$\mathsf{P}\left\{\pi_{ij} = x\right\} = \frac{1}{j - i + 1}.$$

It follows that $\mathsf{P}\left\{ C_{ij} = 1 \right\} = \mathsf{E}\, C_{ij} = \frac{2}{j-i+1}$. The expected number of comparisons is

$$\sum_{1 \le i < j \le n} \mathsf{E}\, C_{ij} = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} \le 2(n-1)\left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right).$$

From analysis we know that the harmonic function $H(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \ln n + O(1)$. Hence the average complexity is $\le 2n \ln n = O(n \log n)$.

Is this all we want to know?

No, an estimate on the expected value may still allow very large fluctuations. By Markov's inequality we know only that if N is the number of comparisons then, say, $\mathsf{P}\{ N > 20n \ln n \} < 1/20$. To know more about the concentration of N around its expected value, we may need to know more about, say, $\mathsf{Var}\, N$. We do not have exponential convergence like in Chernoff bound, but it is better than polynomial (McDiarmid-Hayward 96).

Polynomial time

- An algorithm $A(x)$ has time complexity $t(n)$ if there is a constant c such that for all inputs x, $A(x)$, terminates in time $c \cdot t(|x|)$, where $|x|$ is the length of x.
- Some function function $f(x)$ over strings x is in complexity class $\mathrm{DTIME}(t(n))$ if it has an algorithm of time complexity $t(n)$ computing it.
- The class P of polynomial-time computable functions is defined as $P = \bigcup_k \mathrm{DTIME}(n^k)$.
- A set L of strings (also called a language) is in the same complexity class as its indicator function $1_L(x)$.

Example 5.1

- The usual algorithm of multiplying two integers has polynomial time complexity.
- The school algorithm of deciding whether an integer is composite (has a nontrivial divisor) is not polynomial.

Witnesses

The set balancing problem is an example from an important class. We are given some input data (in this case, the family of sets $\mathcal{A} = \{A_1, \ldots, A_n\}$), a number t, and a requirement for a certain set B, namely $\max_i ||A_i \cap B| - |A_i \setminus B|| \leq t$. We are looking for a set B satisfying this requirement. When such a set B is found, it can be called a witness, or certificate showing for that the requirement can be satisfied.

Matrix product testing

Here is another example.

- Given $n \times n$ integer matrices $\boldsymbol{A}$, $\boldsymbol{B}$, there is no known algorithm to compute $\boldsymbol{AB}$ in $O(n^2)$ algebraic operations. Even if we are given a matrix $\boldsymbol{C}$, there is no known deterministic algorithm to test the equality $\boldsymbol{AB} = \boldsymbol{C}$.
- Idea: $\boldsymbol{AB} = \boldsymbol{C}$ if and only if for all vectors $\boldsymbol{x}$ we have $\boldsymbol{A}(\boldsymbol{Bx}) = \boldsymbol{Cx}$. This can be tested in time $O(n^2)$. So if $\boldsymbol{ABx} \neq \boldsymbol{C}$ then $\boldsymbol{x}$ is a witness for the statement $\boldsymbol{AB} \neq \boldsymbol{C}$.
  The following algorithm will find a witness, if one exists, with probability $\geq 1/2$. Repeating it k times will reduce the probability of not finding a witness to 2^{-k}.

Simple randomized algorithm Choose a random vector $\boldsymbol{x}$ with 0-1 entries.
Compute $\boldsymbol{c} = \boldsymbol{Cx}$, $\boldsymbol{b} = \boldsymbol{Bx}$, $\boldsymbol{c}' = \boldsymbol{Ab}$. If $\boldsymbol{c} = \boldsymbol{c}'$, accept, else reject.

This algorithm takes $O(n^2)$ operations.
If $\boldsymbol{C} = \boldsymbol{AB}$ then it always accepts. Else, it accepts with probability $\leq 1/2$. Indeed, let $\boldsymbol{D} = \boldsymbol{C} - \boldsymbol{AB}$, and assume $d_{ij} \neq 0$. We have

$$(\boldsymbol{Dx})_i = d_{ij}x_j + \sum_{k \neq j} d_{ik}x_k.$$

Let x_k for $k \neq j$ be chosen arbitrarily, this fixes the sum term on the right-hand side. Now x_j is still chosen independently and uniformly over $\{0, 1\}$, and one of the two choices makes the result $\neq 0$.

Given two functions $f(x)$, $g(x)$, is it true that $f(x) = g(x)$ for all input values x?
The functions may be given by a formula, or by a complicated program.

Example 5.2 The matrix product checking example of above can be seen as checking

$$\boldsymbol{A}(\boldsymbol{B}\boldsymbol{x}) = \boldsymbol{C}\boldsymbol{x}$$

for all $\boldsymbol{x}$.

As in the example, if the identity does not hold we may hope that there will be many witnesses $\boldsymbol{x}$.

An example where mathematics can be used is when $f(x), g(x)$ are polynomials. The following fact is well-known from elementary algebra.

Proposition 5.3 A degree d polynomial of one variable has at most d roots.

So, if we find $P(r) = 0$ on a random r, this can only happen if r hits one of the d roots.

But, what is this good for? Checking whether a given polynomial is 0 is trivial: just check all coefficients. In the interesting applications, the polynomial has many variables, and is given only by computing instructions.

Examples 5.4

1. The polynomial is given by a formula or aritmetical circuit, using multiplications and additions, like $(x_1 + 2x_2)(x_2 - 3x_3) - (x_2 + 5x_1 - 3)(4x_3 - x_2^2 - 2x_1)$.
2. It is given as $\det(\boldsymbol{A}_1 x_1 + \cdots + \boldsymbol{A}_k x_k + \boldsymbol{A}_{k+1})$ where the $\boldsymbol{A}_i$ are $n \times n$ integer matrices. Has potentially exponential size in n, but for each fixed value of $(x_1, \ldots, x_k)$ there is an efficient algorithm of computing the determinant: Gaussian elimination. (The algorithm uses divisions, too, so its polynomial complexity is not trivial—as rounding of fractions is not allowed.)

We need to estimate the probability of hitting a root in a multivariate polynomial.

Lemma 5.5 (Schwartz) Let $p(x_1, \ldots, x_m)$ be a nonzero polynomial, with each variable having degree at most d. If $r_1, \ldots, r_m$ are selected randomly from $\{1, \ldots, f\}$ then the probability that $p(r_1, \ldots, r_m) = 0$ is at most md/f.

Proof. Induction on m. Let $p(x_1, \ldots, x_m) = p_0 + x_1 p_1 + \cdots + x_1^d p_d$, where at least one of the p_i, say p_j, is not 0. Let $q(x_1) = p(x_1, r_2, \ldots, r_m)$. Two cases:

$$\begin{cases} p_j(r_2, \ldots, r_m) = 0 & \text{with probability } \leq (m-1)d/f, \\ q(r_1) = 0 & \text{with probability } \leq d/f. \end{cases}$$

Total is $\leq md/f$. □

Let $G = (V, G)$ be a bipartite graph, with the edges between sets A and B, $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$. Assign to each edge $a_i b_j$ a variable x_{ij}. Matrix $\boldsymbol{M}$:

$$m_{ij} = \begin{cases} x_{ij} & \text{if } a_i b_j \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 5.6 (König) There is a complete matching in G if and only if $\det(M)$ is not identically 0.

Proof. Consider a term in the expansion of the determinant:

$$\pm m_{1\pi(1)} m_{2\pi(2)} \cdots m_{n\pi(n)},$$

where π is a permutation of the numbers $1, \ldots, n$. For this not to be 0, we need that a_i and $b_{\pi(i)}$ be connected for all i; in other words, that $\{a_1 b_{\pi(1)}, \ldots, a_n b_{\pi(n)}\}$ be a complete matching in G. In this way, if there is no complete matching in G then the determinant is identically 0. If there are complete matchings in G then to each one of them a nonzero expansion term corresponds.
These terms do not cancel each other (any two of these monomials contain at least two different variables), the determinant is not identically 0. □

We found a polynomial-time randomized algorithm for the matching problem in bipartite graphs.
What's the point? We mentioned it before that there are also a polynomial-time deterministic algorithms for this problem (for example via maximum flows). But there is a theoretical advantage, since determinant-computation can be parallelized: performed on a parallel machine in $\log^c n$ steps.

Definition 5.7 A language L is in $\mathrm{R}(t(n))$ if there is a randomized algorithm A working in time $O(t(n))$ such that for all $x \in \Sigma^*$

- if $x \notin L$ then $A(x)$ rejects.
- if $x \in L$ then $A(x)$ accepts with probability $\geq 1/2$.

Let $\mathrm{RP} = \bigcup_k \mathrm{R}(n^k)$.

If we want $1 - 2^{-k}$ in place of $1/2$, we can repeat k times; this does not change the definition of RP.

Example? We have not seen an interesting example of an RP language yet, since matrix product testing is also in P.

Such an example will be prime testing, though now it is also known that primes are in P.

Contrast RP with NP

$L \in$ RP if there is a k and a (deterministic) algorithm $A(x, r)$ running in time n^k with $x \in \Sigma^n$, $r \in \{0, 1\}^{n^k}$ such that

- if $x \notin L$ then $A(x, r)$ rejects for all r.
- if $x \in L$ then $A(x, r)$ accepts for at least half of all values of r.

On the other hand $L \in$ NP if there is a k and a (deterministic) algorithm $A(x, r)$ running in time n^k with $x \in \Sigma^n$, $r \in \{0, 1\}^{n^k}$ such that

- if $x \notin L$ then $A(x, r)$ rejects for all r.
- if $x \in L$ then $A(x, r)$ accepts for at least one value of r.

The algorithm $A(x, r)$ in the NP definition is called the verifier algorithm, the values of r for which it accepts are called witnesses, or certificates. Thus, RP $\subseteq$ NP.

An NP language L is also in RP if it has a verifier algorithm with the property that if x has a witness then it has many (at least half of all potential ones).

Example 5.8 The word has is important in the above remark. The language COMPOSITE is in NP, since a verifier predicate is

$$B(x, r) \Leftrightarrow r|x \wedge r \notin \{1, x\}.$$

For this verifier, however, it is not true that if there is one witness there are many. For example, if $x = p^2$ for a prime p then p is the only witness.
Nevertheless, COMPOSITE $\in$ RP. But this is shown with the help of a randomized prime/compositeness test algorithm (see later), that is with a different verifier $B'(x, r)$ for the same language COMPOSITE.

Two-sided error

It is actually more natural to consider a randomized complexity class with two-sided error.

Definition 5.9 A language L is in $\mathrm{BP}(t(n))$ if there is a randomized polynomial-time algorithm A working within time $O(t(n))$ such that for all $x \in \Sigma^*$

- if $x \in L$ then $A(x)$ rejects with probability $< 1/3$.
- if $x \notin L$ then $A(x)$ accepts with probability $< 1/3$.

Let $\mathrm{BPP} = \bigcup_k \mathrm{BP}(n^k)$.

Example? I do not recall a simple natural example of BPP.

But since RP is not closed under complementation, if $L_1, L_2 \in \mathrm{RP}$ then about $L_1 \setminus L_2$ we can only say that it is in BPP.

Theorem 5.10 The definition of BPP does not change if we replace $1/3$ with $1/2 - \varepsilon$ for a fixed $\varepsilon > 0$, and also not if we replace it with 2^{-n^k} for any $k > 0$.

To get from error probability $1/2 - \varepsilon$ to error probability 2^{-n^k}, use repetition const $\cdot\, n^k$ times, majority voting and the Chernoff bound.

Why not 1/2? The definition of BPP does not work with $1/2$ in place of $1/3$: in that case we get a (probably) much larger class closely related to #P (see later). But we could use any $1/2 - \varepsilon$ for some constant ε.

Randomized classes

What questions to ask about a randomized algorithm? Clearly, with randomization we give up (almost always) some certainty, but what sort?

1. If an algorithm always solves the problem exactly (like Quicksort), it be called a Las Vegas algorithm (for no particular reason) as opposed to Monte-Carlo algorithm for the case when the result may be wrong. In the Las Vegas case, we want to know, how fast (in various statistical measures)?
2. Otherwise we will generally assume a fixed time bound and are interested in the probability of correct solution, or other statistical goodness of the solution.

No error

Let Σ be a finite alphabet. We will consider languages, sets $L \subseteq \Sigma^*$ of strings with letters in Σ.

Definition 5.11 For a language $L \subseteq \Sigma^*$ we say that $L \in \mathrm{ZP}(t(n))$ if there is a Las Vegas algorithm $A(x)$ working in time $t(n)$ deciding $x \in L$ in expected time $O(t(n))$. Let $\mathrm{ZPP} = \bigcup_k \mathrm{ZP}(n^k)$.

Example? It is not easy to show a nontrivial example of a ZPP language. Adleman and Huang have shown that prime testing is in ZPP, but now it is also known that it is in P.

The proof of the following theorem is a good opportunity to practice our notions.

Theorem 5.12 The following properties are equivalent for a language L:

i. $L \in \mathrm{ZPP}$;
ii. $L \in \mathrm{RP} \cap \text{co-RP}$;
iii. There is a randomized polynomial-time algorithm that accepts or rejects, correctly, or returns the answer "I give up", with probability $\leq 2/3$.

Proof. It is obvious that (i) implies (ii) (Markov inequality).
To see that (ii) implies (iii), submit x to a randomized algorithm that accepts L in polynomial time and also to one that accepts $\Sigma^* \setminus L$ in polynomial time. If they give opposite answers then the answer of the first machine is correct. If they give identical answers then we "give up". In this case, one of them made an error and this has probability at most $1/2 < 2/3$.
To see that (iii) implies (i) modify the algorithm A given in (iii) in such a way that instead of the answer "I give up", it restarts. If on input x, the number of steps of $A(x)$ is τ and the probability of giving it up is p then on this same input, the expected number of steps of the modified machine is

$$\sum_{t=1}^{\infty} p^{t-1}(1-p)t\tau = \frac{\tau}{1-p} \leq 3\tau.$$

□

Corollary 5.13 If a problem has a Las Vegas algorithm $A(x)$ with running time $T(x)$ with $\mathsf{E}\, T(x) \leq p(n)$ for a polynomial $p(n)$, $(n = |x|)$, then it has another one, $A'(x)$, with a constant $\rho < 1$ and random running time $U(x)$, with the property

$$\mathsf{P}\left\{ \frac{U(x)}{p(n)} > t \right\} < \rho^t.$$

In other words, $U(x)$ has very small probability of being large; much better than what comes from the Markov inequality for T. (Then, of course, not just $\mathsf{E}\, U$ is polynomial but also for example $\mathsf{E}\, U^2$.)

Relation to P/poly

A computational model similar to P is the set of languages recognizable by a polynomial-size Boolean circuit. It can also be characterized in terms of Turing machines.

Definition 5.14 A language L is in $\text{Time}(t(n))/f(n)$ if there is Turing machine M such that on inputs x, a of size n, $f(n)$, $M(x, a)$ halts in time $O(t(n))$, and for each n there is a fixed string a_n such that $x \in L$ iff $M(x, a_n)$ accepts.
We define $\text{P/poly} = \bigcup_{k,l} \text{Time}(n^k)/n^l = \bigcup_k \text{Time}(n^k)/n^k$.

The string a_n is the advice string for n; it is the same for all inputs x of length n.

Theorem 5.15 A language is in P/poly if and only if it can be computed by a logic circuit of polynomial size.

The idea of the proof is that for each n the advice string a_n describes the logic circuit that decides $x \in L$ for each n. (The strict proof needs to elaborate the idea in both directions.)

The more interesting theorem is:

Theorem 5.16 (Adleman) BPP $\subseteq$ P/poly.

Proof. Let $L \in \mathrm{BPP}$, $|x| \in \Sigma^n$, with $|\Sigma| > 1$. A simple idea: the good coin tosses deciding $x \in L$ could play the role of the advice string a_n. But this is too simple, since the good coin tosses may be different for each x. Let M be a probabilistic Turing machine working in time n^k that decides L with error probability $< |\Sigma|^{-n}$. Write $M(x)$ as $M'(x, r)$, where M' is a deterministic Turing machine with input $x \in \Sigma^n$ and auxiliary input $r \in \{0, 1\}^{n^k}$ which represents the coin tosses. Let

$$E(x) = \{\, r : M(x, r) \text{ decides incorrectly whether } x \in L \,\},$$

then $\mathsf{P}(E(x)) < |\Sigma|^{-n}$. Let $E_n = \bigcup_{x \in \Sigma^n} E(x)$, then $\mathsf{P}(E_n) < 1$, therefore there is a string $a_n \in \{0, 1\}^{n^k} \setminus E_n$. Then $M(x, a_n)$ decides $x \in L$ always correctly, so a_n can be taken as the advice string. □

Adleman's theorem shows that the class P/poly encompasses all functions computable in polynomial time and using randomization. Can we get even more by using both polynomial circuits and randomization?

The answer is, no. There is a natural definition of randomized logic circuits, namely circuits with some extra inputs for random bits. The reasoning of Adleman's theorem generalizes to this case, showing anything that can be computed by a polynomial-size randomized circuit can be also computed by a (somewhat larger, but still polynomial-size) circuit without randomization.

A zero-sum two-person game is played between players R and C (stands for "row and column" and defined by an $m \times n$ matrix $\boldsymbol{A}$. We say that if player R chooses a pure strategy $i \in \{1, \ldots, m\}$ and player C chooses pure strategy $j \in \{1, \ldots, n\}$ then there is payoff: player C pays amount a_{ij} to player R.

Example 6.1 $m = n = 2$, pure strategies $\{1, 2\}$ are called "attack left", "attack right" for player R and "defend left", "defend right" for player C. The matrix is

$$\boldsymbol{A} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Mixed strategy: a probability distribution over pure strategies.
$\boldsymbol{p} = (p_1, \ldots, p_m)$ for player R and $\boldsymbol{q} = (q_1, \ldots, q_m)$ for player C.
Expected payoff:

$$\sum_{ij} a_{ij} p_i q_j = \boldsymbol{p}^T \boldsymbol{A} \boldsymbol{q}.$$

If player R knows the mixed strategy $\boldsymbol{q}$ of player C, he will want to achieve

$$\max_{\boldsymbol{p}} \sum_i p_i \sum_j a_{ij} q_j = \max_i \sum_j a_{ij} q_j$$

since the maximum is always achieved by some pure strategy.

Player C wants to minimize this and can indeed achieve

$$\min_{\boldsymbol{q}} \max_{i} \sum_{j} a_{ij} q_j .$$

This can be rewritten as a linear program:

$$\begin{array}{lrll} \text{minimize} & t & & \\ \text{subject to} & t \geq \sum_j a_{ij} q_j, & & i = 1, \ldots, m \\ & q_j \geq & 0, & j = 1, \ldots, n \\ & \sum_j q_j = & 1. & \end{array}$$

It is straightforward to check that (as for any real function of $\boldsymbol{p}$, $\boldsymbol{q}$):

$$\max_{\boldsymbol{p}} \min_{\boldsymbol{q}} \boldsymbol{p}^T \boldsymbol{A} \boldsymbol{q} \leq \min_{\boldsymbol{q}} \max_{\boldsymbol{p}} \boldsymbol{p}^T \boldsymbol{A} \boldsymbol{q}.$$

But in our case, we will have equality!

The theorem below directly follows from the duality theorem of linear programming:

Theorem 6.2 (von Neumann) We have

$$\min_{\boldsymbol{q}} \max_{i} \sum_{j} a_{ij} q_j = \min_{\boldsymbol{q}} \max_{\boldsymbol{p}} \sum_{j} a_{ij} p_i q_j$$

$$= \max_{\boldsymbol{p}} \min_{\boldsymbol{q}} \sum_{j} a_{ij} p_i q_j = \max_{\boldsymbol{p}} \min_{j} \sum_{j} a_{ij} p_i .$$

Geometrically, this says that as a function of the mixed strategies $\boldsymbol{p}$, $\boldsymbol{q}$, the function $\boldsymbol{p}^T \boldsymbol{A} \boldsymbol{q}$ has a global saddle point, which is the maximum (in $\boldsymbol{p}$) of minima (in $\boldsymbol{q}$) as well as the minimum of maxima (minimization and maximization is both times by the same variables).

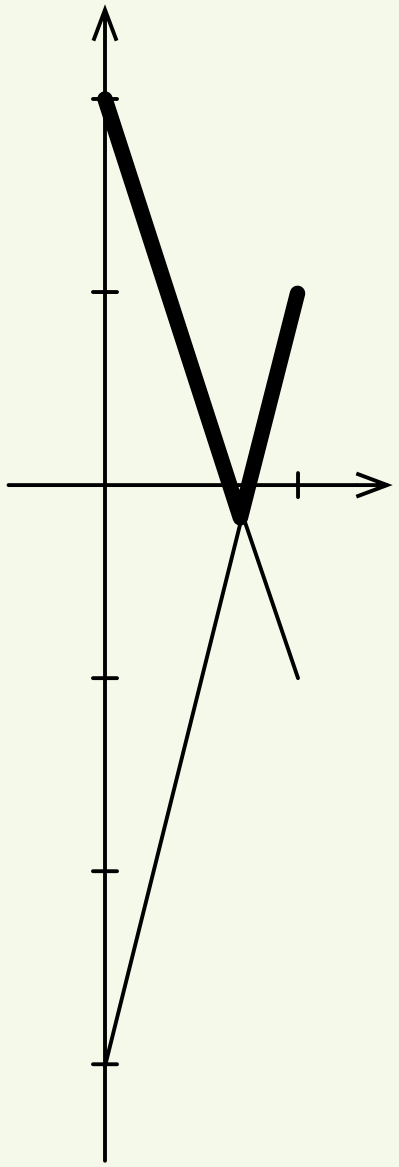
For the matrix $\boldsymbol{A} = \left(\begin{smallmatrix} -1 & 2 \\ 1 & -3 \end{smallmatrix}\right)$, writing $\boldsymbol{p}^T = (p, 1-p)$: the min-max is

$$\min_q \max_i \sum_j a_{ij} q_j = \min_q \max\{-q + 2(1-q), q - 3(1-q)\}$$

$$= \min_q \max\{2 - 3q, -3 + 4q\}.$$

The height of the line $2 - 3q$ runs from 2 to -1, and for the line $-3 + 4q$ from -3 to 1. Let $q^* = 5/7$ be the point where they meet. For $q < q^*$, the maximum is for $i = 1$ and decreasing; for $q > q^*$ it is for $i = 2$ and increasing. So the min-max is $2 - 3q^* = -1/7$. The value $\max_p \min\{-p + (1-p), 2p - 3(1-p)\}$ will be the same (compute it!).

$\max\{2 - 3q, -3 + 4q\}$



Application to algorithms

Yao's theorem will connect the average time of some deterministic algorithms (on varying inputs) with that of a randomized algorithms (on fixed input). Consider a finite number m of possible inputs x_i. and a finite number n of possible computations (algorithms) C_j. Let the row player R choose an input x_i, and the column player C choose an algorithm C_j. The payoff is some aspect

$$a_{ij} = L(C_j(x_i))$$

of the computation $C_j(x_i)$. For definiteness, let it be the computation time.
A distribution $\boldsymbol{p}$ on inputs is a random input. A distribution $\boldsymbol{q}$ on algorithms a randomized algorithm.

Let us apply the minimax theorem:

Theorem 6.3 (Yao) We have

$$\max_{\boldsymbol{p}} \min_{j} \sum_{i} L(C_j(x_i))p_i = \min_{\boldsymbol{q}} \max_{i} \sum_{j} L(C_j(x_i))q_j.$$

Thus the time we get by choosing an input distribution that maximizes the smallest possible expected time of all (deterministic) algorithms (in our collection), is the same as the by choosing the randomized algorithm that minimizes the worst expected time on all possible (deterministic) inputs.
In practice, frequently only the easy, weak form of the theorem is used, namely that for each $\boldsymbol{p}$, $\boldsymbol{q}$ we have

$$\min_{j} \sum_{i} L(C_j(x_i))p_i \leq \max_{i} \sum_{j} L(C_j(x_i))q_j.$$

Game trees or tree formulas

An arbitrary Boolean function $f(x_1, \ldots, x_n)$ can be computed by a formula using only NOR operations. Such a formula, can be represented as a tree:

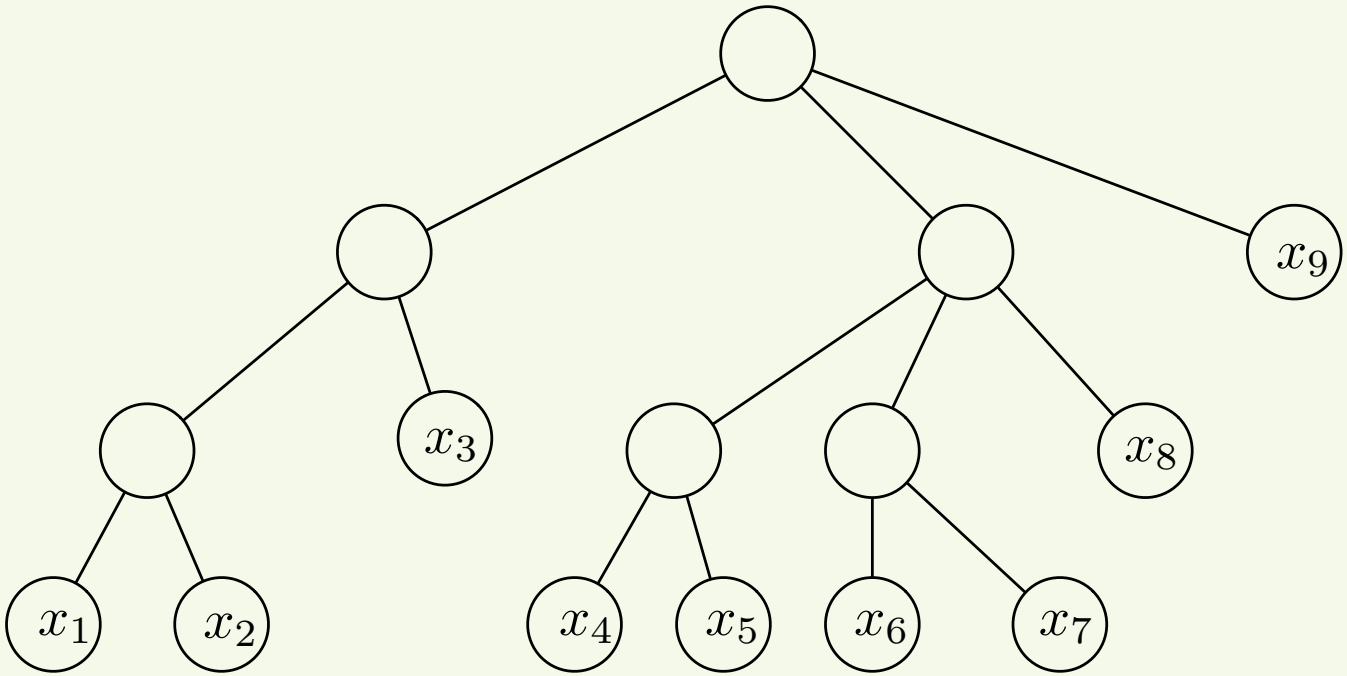


Figure: A game tree

- Evaluating the expression can be seen as finding the value of a two-person game of strategy. Each node is a possible state of the game. Player A has the move at the top, player B has the move at the children of the top, and so on.
  A player wins at a node if the other player loses at all of its children. $f(x_1, \ldots, x_n) = 1$ if player A wins at the top.
- We are interested now in the query complexity: How many inputs need to be evaluated to compute the result? For simplicity, assume that the formula is a full binary tree, so $n = 2^k$ for a height k.
- It is easy to see that each correct algorithm will have to evaluate all variables in the worst case. But a randomized algorithm may do better.

A randomized algorithm

A recursive randomized algorithm $A(h)$ for binary trees of height h:

- Choose a child of the root uniformly at random, and evaluate it according to $A(h-1)$.
- If it returns 1 then return 0; otherwise evaluate also the other child.

Let $\alpha_0(h)$ be the expected number of probes if the value is 0, and $\alpha_1(h)$ if it is 1.

$$\alpha_1(h) \le 2\alpha_0(h-1),$$
$$\alpha_0(h) \le \alpha_1(h-1) + \alpha_0(h-1)/2.$$

Hence $\alpha_0(h) \le 2\alpha_0(h-2) + \alpha_0(h-1)/2$. This is a linear recursion; standard methods show that its solutions grow like c^h where $c = \frac{1+\sqrt{33}}{4}$ is the positive solution of the equation

$$x^2 = 2 + x/2.$$

Is this the best one can do?

Yes, but we will only (half-)prove a weaker lower bound here.

Ⓐ We want a lower bound on the expected value on arbitrary randomized algorithms, on the worst inputs.

Ⓑ Instead we will find some (clever) input distribution and lowerbound the expected value of all deterministic algorithms on it.

- The easy part of Yao's Theorem says that solving (B) gives a lower bound for (A), too. The hard part says that if we are really clever we get the best possible lower bound.
- We only need the easy part now, but still need to be clever!

For a non-optimal lower bound, we choose a distribution P in which each $x_i = 1$ with probability p, independently (the optimal distribution is not independent).
Let $p = \frac{1}{2}(3 - \sqrt{5})$, then it satisfies

$$(1 - p)^2 = p,$$

and $\mathsf{P}\left\{ Y_j = 1 \right\} = p$ also for each internal node Y_j of the NOR tree.

We need to find a lower bound on all deterministic evaluation algorithms. This is still too many algorithms.

Definition 6.4 Call an algorithm depth-first if for each subtree T, if after evaluating some variables in it, the value Y_T is not found, then the next evaluated variable is also taken from T.

The following lemma helps simplifying the situation.

Lemma 6.5 The smallest expected value over distribution P will (also) be achieved by a depth-first algorithm.

For this lemma, it is important that the tree is the full binary tree. Still, the proof is not easy, we will skip it here.

Let us give a lower bound using the lemma. If $W(k)$ is the expected value for a tree of height k, then

$$\begin{aligned} W(k) &= W(k-1) + (1-p)W(k-1) = (2-p)W(k-1) \\ &= (2-p)^k = 2^{k\log(2-p)} = n^{\beta} \end{aligned}$$

where $\beta = \log(2-p) = \log\left(\frac{\sqrt{5}+1}{2}\right) < \log\sqrt{3} = \alpha$.
So our lower bound does not meet the upper bound. It can be matched by a more clever, non-independent distribution.

Packet routing

- Given a network: a directed graph $G = (V, E)$ that is strongly connected. Edges are called links.
- From each point i a message v_i needs to get to its destination $d(i)$. For simplicity, assume that

$$i \mapsto d(i)$$

  is a permutation.
- Discrete timesteps. In each step, each link can forward only one message, the others are queued up in the queue of that link.
- The issue: the congestion delays caused by this.
- For packet i let L_i be the total time to reach its destination. We are interested in minimizing $L = \max_i L_i$.

- Trivial lower bound in the worst case, $L \geq$ the diameter of G. How much worse can congestion make it?
- We want routing algorithm, that is oblivious: the path ρ_i it chooses from source i to destination $d(i)$ does not depend on $d(j)$ for $j \neq i$.

Theorem 7.1 (Nontrivial) On a network of size N in which each vertex has outdegre $\leq d$, for every deterministic oblivious algorithm there is a choice of destinations giving $L = \Omega(\sqrt{N}/d)$.

When the diameter of G is much smaller than $\sqrt{N}$ (not the case of a 2-dimensional grid) then we hope that a randomized algorithm improves on this.
Such an example will now be examined.

- The hypercube network $G = (V, E)$ is popular for connecting many parallel processors.
  V = the set of all $N = 2^n$ binary strings of length. An edge connects two binary strings (in both directions) if they differ only in one bit.
- Degree of nodes: n. Diameter: n.
- There are examples of graphs with constant degree and logarithmic diameter (expanders), but the hypercube is easy to analyze.

Example 7.2 (A simple oblivious algorithm)

- Bit fixing: keep changing the first bit in which you still differ from the destination.
- A simple adversarial permutation causing this to take $\Omega(\sqrt{N}/n)$ steps: flip the bits around the middle of the string.

Wild idea

Two phases.

1. Send every packet v_i to a random intermediate destination $\sigma(i)$. (The map $i \to \sigma(i)$ need not be one-to-one.)
2. Send each v_i from $\sigma(i)$ to $d(i)$.
3. Use the bit-fixing algorithm for each phase.

Bit fixing has a convenient property: if two routes depart from each other, they will never meet again.

Theorem 7.3

a. This algorithm achieves $\mathsf{E}\, L(i) \le 2n$ for each i (thus, delay n).

b. It also gives $\mathsf{P}\{ L > 14n \} \le 1/N$.

Lemma 7.4 Let packet v follow a sequence of edges $\rho = (e_1, \ldots, e_k)$. Let S be the set of packets other than v passing through at least one of the e_j. Then the delay of v on ρ is at most $|S|$.

Proof. Let $\rho = (a_0, a_1, \ldots, a_k)$, $e_i = (a_{i-1}, a_i)$. Draw a 2 dimensional grid (see figure). For each packet, v, if it dwells in point a_i at time t, we show it at coordinate $(i, t - i)$. So, as long as it is on ρ in each step it either moves right (when not delayed) or moves up (when delayed). Now each horizontal line below the top one ends in a packet leaving ρ. □

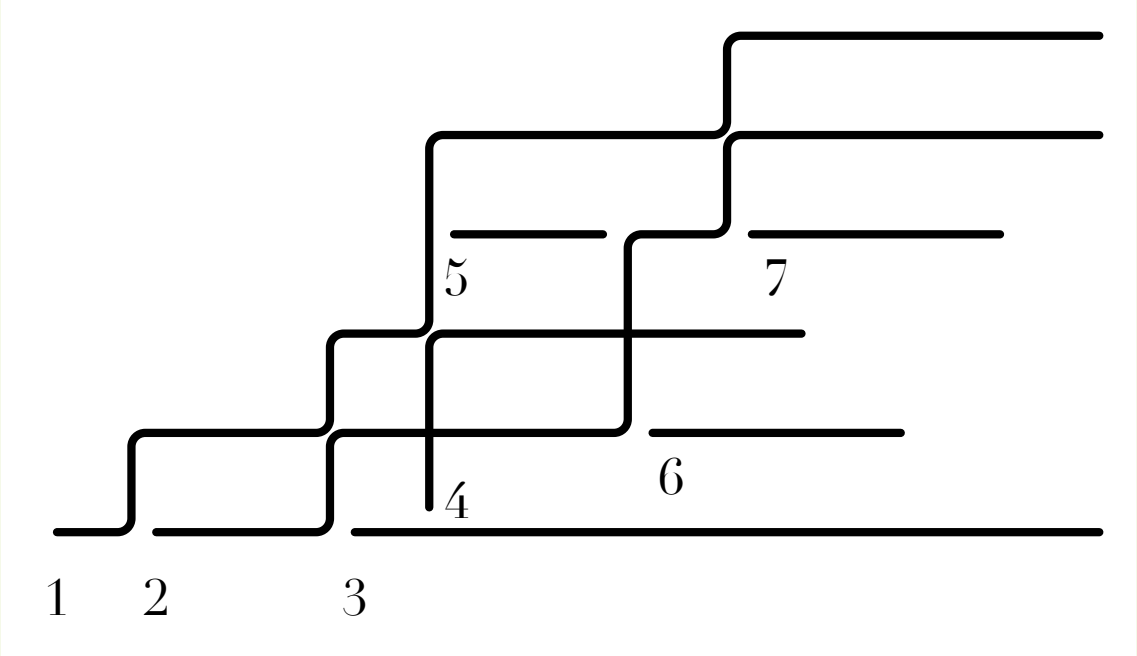


Figure: The structure of delays along a route. Going right: proceeding, going up: waiting. Messages are numbered. They can enter and leave the route, but not return (property of bit-fixing).

- ρ_i = the path from i to $\sigma(i)$.
  $\mathsf{E}\,|\rho_i| = n/2$ (the expected number of bits in which i and $\sigma(i)$ differ).
- Let $H_{ij} = 1$ if ρ_i and ρ_j share an edge.
  $T(e)$ = the number of paths crossing edge e.
- By symmetry $\mathsf{E}\,T(e)$ does not depend on e, let us estimate it via the sum:

$$\sum_e T(e) = \sum_i |\rho_i|,$$

$$Nn\,\mathsf{E}\,T(e) = \sum_{e'} \mathsf{E}\,T(e') = \sum_i \mathsf{E}\,|\rho_i| = Nn/2,$$

$$\mathsf{E}\,T(e) = 1/2.$$

Let $T'(e) \le T(e)$ the number of paths $\rho_2, \ldots, \rho_N$ crossing edge e. Since the paths are chosen independently, for any value r of ρ_1:

$$\mathsf{E}\{T'(e) \mid \rho_1 = r\} = \mathsf{E}\, T'(e) \le \mathsf{E}\, T(e) = 1/2.$$

By the lemma, the total delay of, say, v_1 is at most

$$\sum_{j \neq 1} H_{1j} < \sum_{e \in \rho_1} T'(e),$$

$$\mathsf{E}\{\sum_{j \neq 1} H_{1j} \mid \rho_1 = r\} < \sum_{e \in r} \mathsf{E}\, T'(e) \le \sum_{e \in r} \mathsf{E}\, T(e) = |r|/2 \le n/2.$$

This bounds the expected delay on $1 \to \sigma(1)$ by $n/2$. We saw that the expected path length, $|\rho_1|$, is $n/2$, so the total expected time of $1 \to \sigma(1)$ is $\le n$. It follows that $\mathsf{E}\, L(1) \le 2n$, proving part (a) of the theorem.

For the proof of part (b) notice that $H_{1j}, j = 2, \ldots, N$ are independent random variables in $[0, 1]$, and

$$\mathsf{E} \sum_{j \neq 1} H_{1j} \leq \sum_{e} \mathsf{E}\, T'(e) \leq n/2.$$

A Chernoff bound calculation (see the book) gives that

$$\mathsf{P}\left\{ \sum_{j \neq 1} H_{1j} > 14n \right\} < 1/N^2,$$

then it follows by the union bound that

$$\mathsf{P}\left\{ \max_{i} \sum_{j \neq i} H_{ij} > 14n \right\} < 1/N.$$

Local search

(Following hints by Shanghua Teng.)
Let $G = (V, E)$ be an undirected graph, and $f : V \to \mathbb{R}$. Function f has a local minimum at some $v \in V$ if $f(v) \leq f(u)$ for all neighbors u of v in G. We are looking for a local minimum, while trying to minimize is the number of queries. (Seems easier than global minimum.) Let us add the condition $f(v) \leq f(u_0)$ for some fixed starting point u_0.
Consider the special case where $V = I \times J \times K$ is a lattice slab, where I, J, K are of the form, $I = \{i_0, i_0 + 1, \ldots, i_1\}$ for integers $i_0 \leq i_1$. Two points $(x_1, y_1, z_1), (x_2, y_2, z_2)$ in V are neighbors if

$$|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| \leq 1.$$

In other words, each point can only have neighbors along the coordinate axes at a distance 1.
Let $n = \max\{|I|, |J|, |K|\}$.

Theorem 8.1 There is an algorithm to find the minimum in time $O(n^2)$.

Proof. Without loss of of generality, assume $n = |I|$. Let $m = \lfloor (i_0 + i_1)/2 \rfloor$. $I_1 = \{i_0, \ldots, m\}$, $I_2 = \{m+1, \ldots, i_1\}$, $U = \{m\} \times J \times K$. Then $V = V_1 \cup V_2$ where $V_j = I_j \times J \times K$.
Without loss of generality, assume $u_0 \in V_1$.
Let $f(u) = \min_{x \in U} f(x)$, for some $u \in U$ (not just local minimum).
Let $u' \in V_2$ be the right neighbor of u along dimension 1. Cases:

- $f(u) > f(u_0)$. Search recursively in V_1.
- $f(u) \leq f(u_0)$. Cases:
  - $f(u) > f(u')$. Search recursively in V_2 with u' in the role of u_0.
  - $f(u') \geq f(u)$. Search recursively in V_1 with u in the role of u_0.

In three recursion steps, $n \to n/2$, using $< 3n^2 + 6$ queries. Hence termination in $O(n^2)$ queries. □

Theorem 8.2 No deterministic algorithm can do better than $O(n^2)$ in the worst case.

Here is an adversary to show it, for the cube V where $I = J = K = \{1, \ldots, n\}$. For a set $S \subseteq V$ let

$$\partial(S)$$

be the outside boundary of S in G, and $d_G(x, y)$ the distance function in G.
The adversary will decide some answers in advance in such a way that no decided point forms a local minimum, and the undecided points form a connected component H of G. When an undecided point u will be asked this may break up H into several (at most 4) components H_i. If not, she answers with $f(u)$ smaller than in any neighbors. If so, let H_1 be the largest of these components, and $\Delta = \partial(\bigcup_{i>1} H_i)$, $m = \min_{x \in \Delta \setminus \{u\}} f(x)$. Let D be the maximum diameter of H_i, $i > 1$.

The adversary answers $f(u) = m - 1 - D$, and for each $x \in H_i$, $i > 1$ decides

$$f(x) = f(u) + d_{H_i}(x, u).$$

This creates no local minimum: indeed, the values descend in H_i towards u, and u itself has an undecided neighbor in H_1. The lower bound rests on the following lemma.

Lemma 8.3 There is a time when $\frac{1}{6}|V| \le |H| \le \frac{5}{6}|V|$.

Proof. Let us look at the last step when $|H| > \frac{5}{6}|V|$. Then after this step we have

$$5|H_1| \ge 4|H_1| + 1 \ge |H| > \frac{5}{6}|V|,$$

$$|H_1| > \frac{1}{6}|V|.$$

□

Lemma 8.4 If for $H \subseteq V$ we have $\frac{1}{6}|V| \leq |H| \leq \frac{5}{6}|V|$ then $\partial(H) = \Omega(n^2)$.

Proof. Exercise, see also Example 12.18. □

The last two lemmas imply that that there is a time in the process when $\partial(H) = \Omega(n^2)$. Since all points of $\partial(H)$ have been queried, the number of queries is indeed $\Omega(n^2)$.
The result can be generalized to dimension d: there is a deterministic algorithm with $O(n^{d-1})$ queries, and each deterministic algorithm requires $\Omega(n^{d-1})$ queries.

The following result is completely independent of the structure of the graph. Let $|N| = |V|$, and let Δ be the maximum degree.

Theorem 8.5 (Aldous 83) There is a randomized algorithm finding a local minimum in random time T with $\mathsf{E}\, T = O(\sqrt{N\Delta})$. Moreover, we have

$$\mathsf{P}\left\{ T/\sqrt{N\Delta} > k + 1 \right\} < e^{-k}.$$

- In particular, for the d-cube the expected time is $O(\sqrt{d}n^{d/2})$.
- Special case: $n = 2$, so we are searching on the vertices of a d-dimensional cube (Hamming cube). The upper bound gives $\sqrt{d}2^{d/2}$.
- The adversary argument for the deterministic algorithm gives a lower bound for the Hamming cube. $2^{d(1-\varepsilon)}$ (via a lemma similar to 8.4).
- The bound of the theorem is optimal for randomized algorithms. The lower bound proof for the randomized case is difficult; for the Hamming cube, see Aldous 83.

The algorithm:

1. Choose a set S of points randomly, let $s = |S|$. Find $f(u) = \min_{x \in S} f(x)$.
2. Follow a sequence $u = u_0, u_1, \ldots, u_\tau$ where u_{i+1} is a neighbor of u_i with $f(u_{i+1}) < f(u_i)$, as long as you can. When you cannot continue, you found a local minimum u_τ. Total number of queries $\sigma \leq s + \Delta\tau$.

Let ρ be the rank of $f(u)$ in a sorted $\{f(x) : v \in V\}$. For any $k > 0$,

$$\begin{aligned} \mathsf{P}\{\sigma > s + k\Delta N/s\} &\leq \mathsf{P}\{\tau > kN/s\} \leq \mathsf{P}\{\rho > kN/s\} \\ &\leq \left(\frac{N - kN/s}{N}\right)^s = \left(1 - \frac{k}{s}\right)^s \leq e^{-k}. \end{aligned}$$

Choose $s = \sqrt{\Delta N}$, then $\mathsf{P}\left\{\sigma > (k+1)\sqrt{\Delta N}\right\} < e^{-k}$. So the expected number of queries is $O(\sqrt{\Delta N})$, and the distribution has an exponentially bounded tail.

- A popular optimization method, simulated annealing is similar: search for the optimum using random steps, where the size of the steps used is gradually decreased.
- Our example is extreme.
  - First stage: random steps with no size constraint.
  - Second stage: deterministic steps of minimum size.

  The precise formulation of simulated annealing uses Markov processes (see later).

Fingerprinting

Recall the equality check $\boldsymbol{AB} = \boldsymbol{C}$ for matrices by using a random vector $\boldsymbol{x}$. Another view of the same idea: imagine that matrices $\boldsymbol{A}, \boldsymbol{B}$ are held in one location (see by player Alice) and the matrix $\boldsymbol{C}$ elsewhere (say by player Bob). Task: check the equality $\boldsymbol{AB} = \boldsymbol{C}$ with a minimal amount of communication. Randomized algorithm: Bob chooses a random $\boldsymbol{x}$, sends the fingerprint $\boldsymbol{Cx}$ (along with $\boldsymbol{x}$ itself) to Alice, who then checks $\boldsymbol{A}(\boldsymbol{Bx}) = \boldsymbol{Cx}$. This communicates only $O(n)$ bits instead of $O(n^2)$.
(In the original application we counted the operations performed: $O(n^2)$ in place of n^3.)

Equality checking

Alice and Bob hold bit strings a and b of length n, and want to check $a = b$ without having to to communicate all of a.

- There is a theorem showing that any deterministic protocol needs at least n bits communication in the worst case (see the Lovász notes).
- Randomized algorithm: treat a, b as numbers. Choose a random prime number p from some set $\{p_1, \ldots, p_k\}$ of primes and send p and $a \bmod p$. Of course, Bob checks whether $a \bmod p = b \bmod p$. Error only $a \neq b$ but $p \mid b - a$. If a, b have n bits, then $|b - a| \leq 2^n$ has at most n prime divisors, the probability of error is bounded by n/k. For error ε choose $k = n/\varepsilon$.

We need to choose from among at least k primes. Let $\pi(n)$ be the number of primes up to n. The theorem below follows from the "great" prime number theorem, but has a much simpler proof (see the Appendix of the Lovász notes).

Theorem 9.1 (Chebyshev) For large n we have $\pi(n) \geq 0.75n/\ln n$.

Algorithm: choose $p \leq 1.5k \ln k$. Apply a prime test. If p is not a prime, repeat. Else send p and the fingerprint $a \bmod p$ ($< 2 \log n$ bits, instead of $2n$).

Bonus: From the bit string $\boldsymbol{a} = a_1 a_2 \cdots a_n$, the fingerprint $\boldsymbol{a} \bmod p$ can be computed with very little cost, using the following loop:

$s \leftarrow 0$.

for $i = 1$ to n:

$\quad s \leftarrow 2s + a_i \bmod p$.

Given: universe U, and a typically much smaller subset $S \subseteq U$, say $|S| < m$. We want an easily computable function $h_S : U \to \{0, \ldots, m-1\}$ in such a way that h is (nearly) 1-1 on S. The set N will be called the set of buckets, or bins.

Example 10.1 (Application) Decide questions of the form $x \in S$ fast, or create tables for functions f defined on S.

We may want some additional features: say, easy modification of h_S in case some elements are added to S or deleted.
In a data structures course you have learned several methods, for example via self-balancing trees. The method based on "hashing" wins in most applications by its irresistible simplicity.

Look at functions $h : U \to B$ independently of S. The event $h(x) = h(y)$ for $x, y \in S$ is called a collision. We say then that h is a "hash function" for S if the number of collisions is small on S. In this case there will be collision resolution methods to create an efficient h_S. We say h is perfect for S if it has no collisions on S. Each function h behaves very badly on some sets S.
Assume $m = |B|$, $n = |U|$, and $|S| \leq s$ for all S.

Theorem 10.2 If $n \geq s \cdot m$ then for every function $h : U \to B$ there is a set S mapped by h into a single bucket.

Proof. We have $\sum_{y \in B} |f^{-1}(y)| = n$,

$$\frac{1}{m} \sum_{y \in B} |f^{-1}(y)| = \frac{n}{m} \geq s,$$

so there is a $y_0 \in B$ with $|f^{-1}(y_0)| \geq s$. Let $S = f^{-1}(y_0)$. □

For some finite set H, we can view the function $h : U \times H \to B$ as a randomized hash function, or a family of hash functions in the sense that for each fixed $r \in H$ (which can be chosen randomly) the function $h(\cdot, r)$ maps from U to B.

Definition 10.3 The family $h(\cdot, \cdot)$ is 2-universal if for all $x \neq y \in U$ and a random R we have

$$\mathsf{P}\{ h(x, R) = h(y, R) \} \leq \frac{1}{|B|}. \tag{10.1}$$

Equality here if $h(x, R)$ and $h(y, R)$ are independent for all pairs $x \neq y$.

Example 10.4 A universal hash function in which in the equation (10.1) we don't always have equality: let $U = B$, $h(x, r) = x$ for all r. Then there are no collisions at all, so their probability is 0. Of course, in this example hashing is unnecessary.

Example 10.5 (Complete independence) $H = B^U$, and $h_1(x, (r_1, \ldots, r_{|U|})) = r_x$. Here, all the values $h(x, r)$ for different x are independent.
This is useless, we want r to be small enough for storage and computation.

An example universal hash function

We assume that our table size $m = |B|$ is a prime number.
For an integer "dimension" $d > 0$ assume $n \leq m^d$, and break up the key x into a sequence

$$x = (x_1, x_2, \ldots, x_d), \quad 0 \leq x_i < m.$$

(If x is a bit string, break it into segments of size $\log m$.) Fix random coefficients $0 \leq r_i < m$, $i = 1, \ldots, d$: the number of possible random inputs is $|H| = m^d$.

$$h(x, r) = r_1 x_1 + \cdots + r_d x_d \bmod m.$$

We use the notation $a \equiv b \pmod{m}$ for $a \bmod m = b \bmod m$. This is the same as requiring $m|(a-b)$.

Fact 10.6 Let p be a prime, $d \not\equiv 0 \pmod{p}$ and $ad \equiv bd \pmod{p}$ then $a \equiv b \pmod{p}$.

Indeed, by the fundamental theorem of arithmetic, if a prime number divides a product, it must divide one of its factors. Here, p divides $(a-b)d$. It does not divide d, so it divides $a-b$.

Let us show that $h(x, r)$ is universal: in fact, the values $h(x, R)$ for different x are pairwise independent.
Assume $(x_1, \ldots, x_d) \neq (y_1, \ldots, y_d)$. We show that $\mathsf{P}\{ h(x, r) = h(y, r) \} \leq 1/m$ (actually, $= 1/m$). There is an i with $x_i \neq y_i$, we might as well assume $x_1 \neq y_1$. If $h(x, r) = h(y, r)$ then

$$0 \equiv h(x, r) - h(y, r) \equiv r_1(x_1 - y_1) + A \pmod{m},$$
$$A \equiv r_1(y_1 - x_1) \pmod{m},$$

where A only depends on the random numbers $r_2, \ldots, r_d$. No matter how we fix $r_2, \ldots, r_d$, there are m equally likely ways to choose r_1. According to the Fact above, only one of these choices gives $r_1(y_1 - x_1) \equiv A \pmod{m}$, so the probability of this happening (conditionally on fixing $r_2, \ldots, r_d$) is $1/m$. Since this probability is the same under all conditions, it is equal to $1/m$.

Counting problems: the class # P

Definition 11.1 Function f is in $\#P$ if it can be written as $|W(x)|$ where $W(x)$ is the set of witnesses for a predicate $V(x, y)$ computable in time polynomial in $|x|$:

$$W(x) = \{ y : V(x, y) = 1 \}.$$

Note that $W(x)$ (if nonempty) consists of elements y of length $\leq p(|x|)$ for some polynomial.

- Some $\#P$-problems are all obviously NP-hard: the ones coming from an NP-complete $V(x, y)$.
- There are reductions among these problems, and there are $\#P$-complete ones, for example the number of bipartite matchings (even though this one is not coming from an NP-complete $V(x, y)$).

How to aproximate a $\#P$ function?

- If y is from the range $R(x) = \Sigma^{p(n)}$, for $n = |x|$, then repeated independent tests will work only if the probability of success $\frac{f(x)}{|R(x)|}$ is not tiny.
- More formally: let $X_1, \ldots, X_N$ be i.i.d. random variables, $\mathsf{Var}\, X_i = \sigma^2$, $\mathsf{E}\, X_i = \mu$. By Chebyshev's inequality

$$\mathsf{P}\left\{ \left| \sum_i X_i/N - \mu \right| > t \right\} \leq (\sigma/t)^2 N^{-1}, \text{ so}$$

$$\mathsf{P}\left\{ \left| \sum_i X_i/N - \mu \right| > \mu/2 \right\} \leq (2\sigma/\mu)^2 N^{-1}.$$

Converges too slowly to 0 if σ/μ is large.

Example 11.2 $X_i = 1$ with probability p and 0 otherwise. Then $\sigma^2 = p(1-p)$, our bound is $\frac{4(1-p)}{pN}$, and we need $N > \frac{1}{p}$ if p is small. Estimating an exponentially small probability this way would require an exponentially large number of samples.

A general idea for approximate counting

Find sets $U_i = U_i(x) \subseteq R(x)$, $i = 1, 2, \ldots, m$ where

- We know $|U_1(x)|$.
- $U_m(x) = \{ y : V(x, y) = 1 \}$.
- The quotients $\frac{|U_i|}{|U_{i-1}|}$ are approximable by sampling. Say if $U_{i-1} \cap U_i = \emptyset$ then sampling from $U_{i-1} \cup U_i$ helps estimate $\frac{|U_i|}{|U_i|+|U_{i-1}|}$ and hence $\frac{|U_i|}{|U_{i-1}|}$.
- To sample from $U_{i-1} \cup U_i$, sometimes the Markov chain Monte-Carlo method helps, see below.

Example 11.3 (see also under the Metropolis algorithm, later)
$U_k(G)$ is the set of matchings μ of size k of a bipartite graph G (see the book). Step of a random walk: for $\mu \in U_{k-1} \cup U_k$, take a random edge e and:

- If $\mu \in U_k$, delete e from μ if possible.
- If $\mu \in U_{k-1}$, and at least one end of e is unmatched, add e to μ. If another edge intersects, delete it from μ.

What can we hope from a randomized algorithm?

Definition 11.4 An algorithm $A(x)$ is a FPRAS (fully polynomial randomized approximation scheme) for computing $f(x)$ if it is polynomial in $|x|$ and $1/\varepsilon$, and

$$\mathsf{P}\{\,|A(x) - f(x)| > \varepsilon f(x)\,\} \leq 1/3.$$

This definition combines randomization and approximation.
We will return to the question of changing $1/3$ to arbitrary constant $\delta < 1/2$.

FPRAS for DNF satisfaction

Try to find the number of satisfying assignments of a disjunctive normal form

$$f(\boldsymbol{x}) = C_1(\boldsymbol{x}) \vee \cdots \vee C_m(\boldsymbol{x}),$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$. Here each $C_i(\boldsymbol{x})$ is a conjunction of some variables x_k or their negations. Let $S_i = \{\, \boldsymbol{x} : C_i(\boldsymbol{x}) = 1) \,\}$. Then it is easy to compute $|S_i|$. For example, if $C_i(\boldsymbol{x}) = x_1 \wedge \neg x_3 \wedge \neg x_4$ then $|S_i| = 2^{n-3}$.

We want to compute $|S|$ where $S = S_1 \cup \cdots \cup S_m$. This would be easy if the S_i were disjoint, but they are not. But we know something about the intersections. For each $\boldsymbol{x}$ we can compute the cover multiplicity $c(\boldsymbol{x}) = |\{\, i : \boldsymbol{x} \in S_i \,\}|$. Indeed, $c(\boldsymbol{x}) = |\{\, i : C_i(\boldsymbol{x}) = 1 \,\}|$.

More generally, estimate the size of a set

$$S = S_1 \cup \cdots \cup S_m,$$

assuming that

- We can generate uniformly the elements of S_i for each i (this is true for $\{ x : C_i(x) = 1 \}$).
- We can compute in polynomial time $|S_i|$.
- For each element x, we can compute in polynomial time the cover multiplicity $c(x) = |\{ i : x \in S_i \}|$.

- Let $M = |S_1| + \cdots + |S_m|$. Pick $I \in \{1, \ldots, m\}$ such that $\mathsf{P}\{ I = i \} = |S_i|/M$, then pick an element $X \in S_I$ uniformly. Then

$$\mathsf{P}\{ X = x \} = \sum_{S_i \ni x} \mathsf{P}\{ I = i \} \mathsf{P}\{ X = x \mid I = i \} = \sum_{S_i \ni x} \frac{|S_i|}{M} \frac{1}{|S_i|} = \frac{c(x)}{M}.$$

- Let $Y = \frac{M}{c(X)}$, then

$$\mathsf{E}\, Y = \sum_{x \in S} \frac{M}{c(x)} \mathsf{P}\{ X = x \} = |S|.$$

- $\mathsf{Var}\, Y \leq M^2 \leq m^2 |S|^2$, hence $\frac{\sqrt{\mathsf{Var}\, Y}}{\mathsf{E}\, Y} \leq m$, a polynomial bound. So if we sample Y, repeatedly, Chebyshev's inequality allows to approximate $\mathsf{E}\, Y = |S|$ fast: we have a FPRAS.

Recall that an algorithm $A(x)$ is a FPRAS if it is polynomial in $|x|$ and $1/\varepsilon$, and

$$\mathsf{P}\{\,|A(x) - f(x)| > \varepsilon f(x)\,\} \leq 1/2.$$

Theorem 11.5 The definition of a FPRAS does not change if we replace $1/3$ on the right-hand side with any constant $\delta \leq 1/2$.

The idea of the proof is repetition, but we cannot apply the law of large numbers to $A(x)$, not knowing its expected value. We only know that it is in the interval $f(x) \pm \varepsilon f(x)$ with probability $3/4$.

Let our algorithm $B(x)$ repeat N times the algorithm $A(x)$. Let $Z_1, \ldots, Z_N$ be the values of $A(x)$ in the repetitions. Then $B(x)$ outputs the median M of $Z_1, \ldots, Z_N$.
To analyze, let $U_i = 1$ if $Z_i > f(x) + \varepsilon f(x)$ and 0 otherwise. Then $\mathsf{P}\{ U_i = 1 \} \leq 1/3$, while

$$\mathsf{P}\{ M > f(x) + \varepsilon f(x) \} \leq \mathsf{P}\left\{ \sum_i U_i \geq N/2 \right\} \leq e^{-N/18}$$

from the Chernoff bound. A choice $N = O(\log \frac{1}{\delta})$ makes this smaller than $\delta/2$.
We find $\mathsf{P}\{ M < f(x) - \varepsilon f(x) \} < \delta/2$ similarly.

Let $X_1, X_2, \ldots$ be a sequence of random variables with values in a discrete state space $V = \{v_1, v_2, \ldots\}$. The sequence is called a Markov chain if for all t, for all $s_1, \ldots, s_{t+1}$ we have

$$\mathsf{P}\{X_{t+1} = s_{t+1} \mid X_1 = s_1, \ldots, X_t = s_t\} = \mathsf{P}\{X_{t+1} = s_{t+1} \mid X_t = s_t\}.$$

In words, X_{t+1} depends on $X_1, \ldots, X_{t-1}$ only through X_t.

Example 12.1 Let $Y_1, Y_2, \ldots$ be a sequence of independent random variables, $S_t = Y_1 + \cdots + Y_t$. Then the sequence $S_1, S_2, \ldots$ forms a Markov chain.

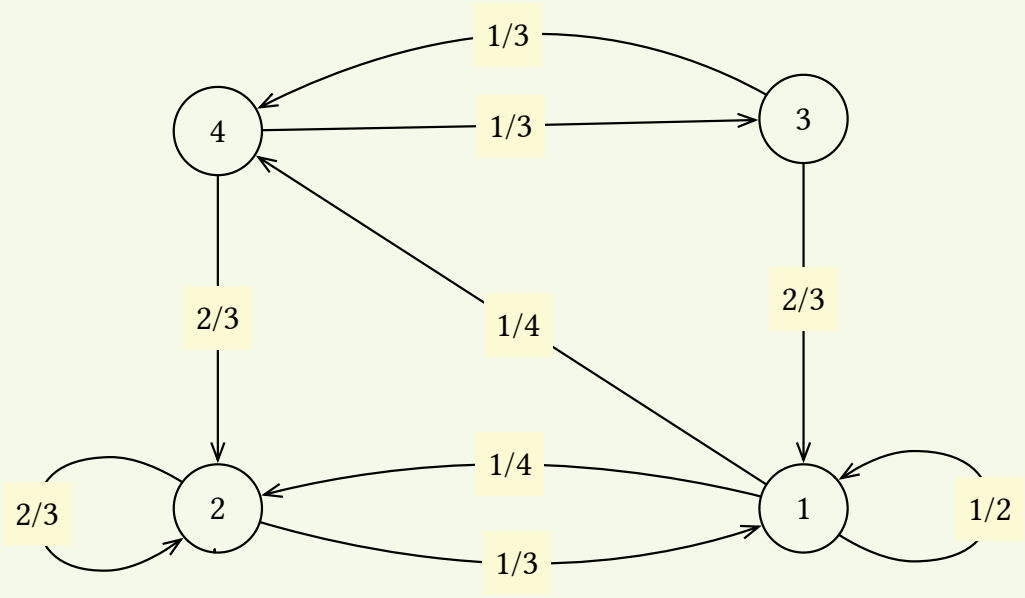
If for each i, j the value $p_{ij} = \mathsf{P}\{X_{t+1} = i \mid X_t = j\}$ is independent of t then the Markov chain is called homogenous. The matrix (p_{ij}) is called its transition matrix.

From now on, we will only consider homogenous Markov chains, in a finite state space.

View a Markov transition matrix as a directed graph $G = (V, E, \boldsymbol{P})$ on the state space V. There is an edge $u \to v$ if $p_{uv} > 0$; label the edge with p_{uv}.

Example 12.2

$$\boldsymbol{P} = \begin{pmatrix} 1/2 & 1/4 & 0 & 1/4 \\ 1/3 & 2/3 & 0 & 0 \\ 2/3 & 0 & 0 & 1/3 \\ 0 & 2/3 & 1/3 & 0 \end{pmatrix}.$$



- The Markov condition says that X_t has enough information about the system to determine (the probability distribution of) its future behavior $X_{t+1}, X_{t+2}, \ldots$: the past influences the future only through the present.
- The condition may fail if we omit some information: If $X_0, X_1, \ldots$ is a Markov process and f a function then the process defined by $Y_t = f(X_t)$ is typically not Markov. It is called a hidden Markov process.

Example 12.3 In Example 12.2, let $f(s) = 1$ if $s \in \{1, 2\}$, and 2 if $s \in \{3, 4\}$. Now $\mathsf{P}\{f(X_{i+1}) = 2 \mid f(X_1) = f_1, \ldots, f(X_{i-1}) = f_{i-1}, f(X_i) = 1\}$ typically depends on $f_1, \ldots, f_{i-1}$. Indeed: $\mathsf{P}\{f(X_{i+1}) = 2 \mid X_i = 1\} = 1/4$ and $\mathsf{P}\{f(X_{i+1}) = 2 \mid X_i = 2\} = 0$, but $f(s) = 1$ does not tell us whether $s = 1$ or 2.

- If we only know the values $Y_t = f(X_t)$ then it is a challenge to infer from them some (not unique) underlying Markov process X_t. This is a typical machine learning task.

If the vector $\boldsymbol{q}(t)$ describes the distribution of X_t, that is $q_i(t) = \mathsf{P}\{X_t = i\}$, then

$$\boldsymbol{q}(t+1) = \boldsymbol{q}(t)\boldsymbol{P}.$$

Since the process is homogenous the numbers

$$p_{ij}(t) = \mathsf{P}\{X_{t+k} = j \mid X_k = i\}$$

do not depend on k, and are elements of the matrix $\boldsymbol{P}^t$, that is the t-step transition matrix is the tth power of the transition matrix $\boldsymbol{P}$. An u-v walk of length t on the directed graph of states is a sequence of states $W = (v_0, v_1, \ldots, v_t)$ where $u = v_0$, $v = v_t$, and $p_{v_i,v_{i+1}} > 0$. We don't call it a "path" since it may intersect itself (many times). Then

$$p_{uv}(t) = \sum_W p_{v_0v_1}p_{v_1v_2}\cdots p_{v_{t-1}v_t}$$

where W runs through all u-v-walks W of length t.

Let if $X_1, X_2, \ldots$ be our Markov chain. The matrix-vector multiplication $\boldsymbol{Pf}$ also has probability meaning. If $\boldsymbol{f} = (f(1), \ldots, f(n))^T$ then let

$$\boldsymbol{g} = \boldsymbol{Pf}, \quad \boldsymbol{g} = (g(1), \ldots, g(n))^T.$$

Then

$$g(i) = \sum_j p_{ij} f(j) = \mathsf{E}\{f(X_{t+1}) \mid X_t = i\},$$

so $\boldsymbol{Pf}$ shows the conditional expectations of $f(X_{t+1})$, when looking at time t.

Convergence

We are interested in the limiting behavior of Markov chains.
A distribution $\boldsymbol{q}$ is stationary, or invariant, or an equilibrium if $\boldsymbol{q} = \boldsymbol{q}\boldsymbol{P}$.

Does every Markov chain have an invariant distribution? No.

Example: Let $S_t = X_1 + \cdots + X_t$ where $X_i > 0$ are integer i.i.d. random variables. Then S_t is a Markov chain but $\lim_{t\to\infty} \mathsf{P}\{ S_t = n \} = 0$ for all n.

This does not happen if the set of states is finite.

Examples 12.4

1. A stochastic matrix $\boldsymbol{P}$ is called doubly stochastic if we not only have $\sum_j p_{ij} = 1$ for all i but also $\sum_i p_{ij} = 1$ for all j. It is easy to see that then the uniform distribution is invariant.
2. If a stochastic matrix $\boldsymbol{P}$ is symmetric then it is doubly stochastic. As a simplest example, let

$$\boldsymbol{P} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}.$$

Theorem 12.5 Every homogenous finite Markov chain has a stationary distribution.

Proof sketch. Let $\boldsymbol{q}(0)$ be arbitrary distribution, $\boldsymbol{q}(t) = \boldsymbol{q}\boldsymbol{P}^t$, and let

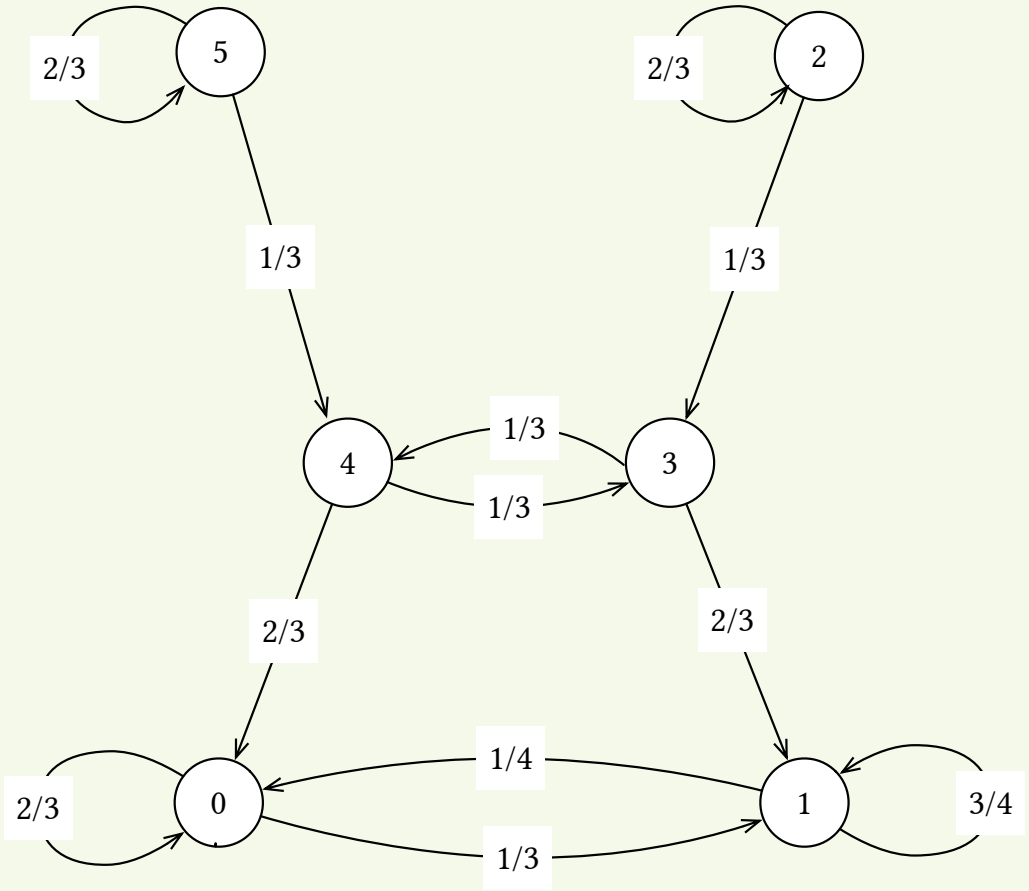
$$\overline{\boldsymbol{q}}(t) = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{q}(i) = \sum_{i=1}^{t} \boldsymbol{q}(0)\boldsymbol{P}^i.$$

The sequence $\overline{\boldsymbol{q}(t)}$ may not converge, but we can select a convergent subsequence $\overline{\boldsymbol{q}(t_k)}$; let its limit be $\boldsymbol{\pi}$.
Exercise: show that $\boldsymbol{\pi}$ is stationary. □

- Can a Markov chain have more than one invariant distribution?
  Yes. Example: Let $\boldsymbol{P} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Then every distribution $\boldsymbol{q}$ over the two states is invariant with respect to $\boldsymbol{P}$.
  This example Markov chain is pathological since its graph is not connected.
- A state v is called transient when $q_v = 0$ for every invariant distribution $\boldsymbol{q}$. Else it is called persistent. (This terminology is different from the one in Motwani-Raghavan.)

Consider the strongly connected components of the graph G of a finite Markov chain. From now on, we will just call them components.

- A Markov chain is called irreducible if the graph consists of a single component.
- The (strongly connected) components (of every directed graph G) form a (new) acyclic graph G'. (View the edges of this acyclic graph as going downward.)
- A component is called final (minimal) if no edge leaves it—so when it is at the bottom of the graph G'.



The Markov chain above has components {2}, {5}, {3, 4}, {0, 1}. Only the component {0, 1} is final.

Theorem 12.6 Let the matrix $\boldsymbol{P}$ belong to a Markov chain.

- **a** A state v is persistent if and only if it is in a final component.
- **b** There is a unique invariant distribution $\boldsymbol{\pi}$ if and only if there is only a single final component. In this case for each initial state $\boldsymbol{q}(0)$ we have $\overline{\boldsymbol{q}(t)} = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{q}(i) \to \boldsymbol{\pi}$.

The proof is not hard.

- If there is a unique invariant distribution, does $\boldsymbol{q}(t)$ always converge to it?

  No. Example: $\boldsymbol{P} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. If $\boldsymbol{q}(0) = (q_1, q_2)$ then $\boldsymbol{q}(1) = (q_2, q_1)$, $\boldsymbol{q}(2) = (q_1, q_1)$, and so on.
  There is only one invariant distribution here, $\boldsymbol{\pi} = (1/2, 1/2)$, but $\boldsymbol{q}(t)$ does not converge to it (unless $\boldsymbol{q}(0) = \boldsymbol{\pi}$).

The following condition eliminates the problem:
A finite irreducible Markov transition matrix $\boldsymbol{P}$ is called aperiodic if there is a t with $\boldsymbol{P}^t > 0$. (There is a number of equivalent characterizations: see any probability book.)

Theorem 12.7 Let $\boldsymbol{P}$ be a the matrix of an irreducible Markov chain with stationary distribution $\boldsymbol{\pi}$. We have $\boldsymbol{q}(t) \to \boldsymbol{\pi}$ for all initial distributions $\boldsymbol{q}(0)$ if and only if $\boldsymbol{P}$ is aperiodic.

You might see some parts of the proof in homework.

Random walk on a graph

From a directed graph $G = (V, E)$ define transition probabilities as follows: from each point choose each outgoing edge with equal probability. If $\boldsymbol{A}$ is the adjacency matrix and $\boldsymbol{D}$ the diagonal matrix formed from the outdegrees d_i then $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$.

Connected undirected graph G: replace each edge with two directed edges. Let

$$\operatorname{vol} G = \sum_i d_i, \;\; q_i = d_i/(\operatorname{vol} G).$$

Check: the distribution $\boldsymbol{q}$ is stationary, moreover, since $\boldsymbol{DP} = \boldsymbol{A}$ is a symmetric matrix, then $q_i p_{ij} = q_j p_{ji}$, that is

$$\mathsf{P}\{X_t = i, X_{t+1} = j\} = \mathsf{P}\{X_t = j, X_{t+1} = i\}. \qquad (12.1)$$

- Processes with (12.1) are called reversible: the paths have the same statistics forward in time as backward. With $\boldsymbol{Q}$ the diagonal matrix for $\boldsymbol{q}$, this says that $\boldsymbol{QP}$ is symmetric.
- In the undirected graph random walk also each edge has the same probability of passing:

$$q_i p_{ij} = 1/(\operatorname{vol} G).$$

Speed of convergence

How fast do we approach equilibrium?

- Can be very exponentially slow for directed graphs. Example:

$$A = \{a_0, \ldots, a_n\},\ B = \{b_0, \ldots, b_n\},\ V = A \cup B,$$
$$\begin{aligned} E = \{&(a_0, a_1), (a_1, a_2), \ldots, (a_n, b_0),\\ &(b_0, b_1), (b_1, b_2), \ldots, (b_n, a_0),\\ &(a_1, a_0), (a_2, a_0), \ldots, (a_n, a_0),\\ &(b_1, b_0), (b_2, b_0), \ldots, (b_n, b_0)\}. \end{aligned}$$

  For the equilibrium $\boldsymbol{\pi}$ we have $\pi(A) = \pi(B) = 1/2$. But there a $c > 1$ (compute one!) such that $\mathsf{P}\{ X_i \in B \mid X_1 = a_0 \} < 1/4$ for all $i < c^n$. Similarly, if the walk starts from a_0 then it will take exponential expected time to reach B.

- For undirected graphs, the convergence is much faster: we develop the theory below.

- For vectors $\boldsymbol{x} = (x_1, \ldots, x_n)^T, \boldsymbol{y} = (y_1, \ldots, y_n)^T$, let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y} = \sum_i x_i y_i$. For vector $\boldsymbol{x}$ we denote by $\|\boldsymbol{x}\| = \|\boldsymbol{x}\|_2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle^{1/2} = (\sum_i |x_i|^2)^{1/2}$ its length, also called the L_2-norm.
- But for any $\boldsymbol{q} = (q_1, \ldots, q_n)$ where $q_i > 0$ we could define a new inner product. Let $\boldsymbol{Q}$ be the diagonal matrix with elements q_i on the diagonal. Define $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{Q}} = \sum_i q_i x_i y_i$. Let us denote $\|\boldsymbol{x}\|_{\boldsymbol{Q}} = \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\boldsymbol{Q}}^{1/2}$. then

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{Q}} = \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{y}.$$

  This new inner product is also bilinear. It also obeys the Cauchy-Schwartz inequality:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{Q}} \leq \|\boldsymbol{x}\|_{\boldsymbol{Q}} \cdot \|\boldsymbol{y}\|_{\boldsymbol{Q}},$$

  and therefore among all vectors $\boldsymbol{y}$ with $\|\boldsymbol{y}\|_{\boldsymbol{Q}} = 1$, the expression $\langle \boldsymbol{y}, \boldsymbol{x} \rangle_{\boldsymbol{Q}}$ is maximized by $\boldsymbol{y} = \boldsymbol{x} / \|\boldsymbol{x}\|_{\boldsymbol{Q}}$.

The L_1-norm is defined as $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$. The Cauchy-Schwartz inequality gives

$$\|\boldsymbol{x}\|_1 = \sum_i 1 \cdot |x_i| \leq (\sum_{i=1}^{n} 1^2)^{1/2} (\sum_i x_i^2)^{1/2} = \sqrt{n}\|\boldsymbol{x}\|.$$

Just as we call vectors $\boldsymbol{x}, \boldsymbol{y}$ orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$, we can call them $\boldsymbol{Q}$-orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{Q}} = 0$. An $n \times n$ matrix $\boldsymbol{M}$ is $\boldsymbol{Q}$-symmetric if for all $\boldsymbol{x}, \boldsymbol{y}$

$$\langle \boldsymbol{x}, \boldsymbol{M}\boldsymbol{y} \rangle_{\boldsymbol{Q}} = \langle \boldsymbol{M}\boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{Q}}.$$

When $\boldsymbol{q} = (1, \ldots, 1)$ then this is the ordinary notion of symmetry. Generalizing to $\boldsymbol{Q}$-inner products a basic theorem of linear algebra:

Theorem 12.8 Let $\boldsymbol{M}$ be a $\boldsymbol{Q}$-symmetric matrix. There is a basis of $\boldsymbol{Q}$-orthonormal vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ that are eigenvectors of $\boldsymbol{M}$ with some real eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. For an arbitrary vector $\boldsymbol{v} = \sum_i c_i \boldsymbol{u}_i$ we have

$$\boldsymbol{M}\boldsymbol{v} = \sum_i \lambda_i c_i \boldsymbol{u}_i.$$

Matrices with all nonnegative elements have some special properties used in Markov chain theory.

Theorem 12.9 (Perron-Frobenius)

- **a** If $\boldsymbol{M}$ is any matrix with nonnegative elements (not necessarily symmetric) then it has a nonnegative eigenvalue λ_1 with eigenvector $\boldsymbol{u}_1 \geq 0$, and $\lambda_1 \geq |\lambda_i|$ for all $i > 1$.
- **b** If also $\boldsymbol{M}$ is aperiodic (that is $\boldsymbol{M}^t > 0$ for some t), then $\boldsymbol{u}_1 > 0$, and $\lambda_1 > |\lambda_i|$ for $i > 1$.

Let $\boldsymbol{M}$ be a $\boldsymbol{Q}$-symmetric matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$.

- Suppose $\lambda_1 = 1$, and denote $\delta = 1 - \max_{i>1} |\lambda_i|$, then for $\boldsymbol{x} = c_1\boldsymbol{u}_1 + \cdots + c_n\boldsymbol{u}_n$ we have

$$\left\|\boldsymbol{M}^t\boldsymbol{x} - c_1\boldsymbol{u}_1\right\|_{\boldsymbol{Q}} = \left(\sum_{i=2}^{n} |\lambda_i|^t q_i c_i^2\right)^{1/2} \tag{12.2}$$

$$\leq (1-\delta)^t \|\boldsymbol{x}\|_{\boldsymbol{Q}} \leq e^{-\delta t}\|x\|_{\boldsymbol{Q}}, \tag{12.3}$$

  that is $\boldsymbol{M}^t\boldsymbol{x} \to c_1\boldsymbol{u}_1$ with speed $e^{-\delta t}$. The speed of the convergence depends on δ, often called the eigenvalue gap.

- (For nonsymmetric matrix $\boldsymbol{M}$ there is a similar, but more complex estimate.)

For a Markov chain $\boldsymbol{P}$, with reversible distribution $\boldsymbol{q}$, the matrix $\boldsymbol{P}$ is $\boldsymbol{Q}$-symmetric. Indeed,

$$\begin{aligned}\langle \boldsymbol{x}, \boldsymbol{P}\boldsymbol{y}\rangle_{\boldsymbol{Q}} &= \sum_i x_i q_i \sum_j p_{ij} y_j \\ &= \sum_{ij} x_i p_{ij} q_j y_j = \sum_{ij} = \sum_{ij} x_i q_j p_{ji} y_j \\ &= \sum_j y_j q_j \sum_i p_{ji} x_i = \langle \boldsymbol{P}\boldsymbol{x}, \boldsymbol{y}\rangle_{\boldsymbol{Q}}.\end{aligned}$$

So we can apply the theorem about the eigenvectors to the matrix $\boldsymbol{P}$ with the $\boldsymbol{Q}$-inner product, giving us eigenvalues λ_i.

Theorem 12.10 We have $|\lambda_i| \leq 1$.

Proof. Let λ be an eigenvalue of $\boldsymbol{P}$ with eigenvector $\boldsymbol{x}$. Assume without loss of generality that $|x_1| = \max_i |x_i|$. Then

$$\lambda x_1 = \sum_j p_{ij} x_j,$$

$$|\lambda||x_1| \leq \sum_j p_{ij}|x_j| \leq |x_1|.$$

□

- Since $\boldsymbol{P}\mathbf{1} = \mathbf{1}$, the vector $\mathbf{1}$ is an eigenvector of $\boldsymbol{P}$ with with eigenvalue 1. By Theorem 12.10 no other eigenvalues have larger absolute value.
- Assume that $\boldsymbol{P}$ is nonperiodic, and therefore $\lambda_1 > |\lambda_i|$ for $i > 1$. Let $\boldsymbol{u}_i$ be the orthonormal basis of Theorem 12.8.
- Recalling that $\boldsymbol{QP}$ is symmetric,

$$\boldsymbol{QP}\boldsymbol{u}_i = \lambda_i \boldsymbol{Q}\boldsymbol{u}_i,$$
$$\boldsymbol{u}_i^T \boldsymbol{QP} = \lambda_i \boldsymbol{u}_i^T \boldsymbol{Q}.$$

  So $\boldsymbol{v}_i = \boldsymbol{u}_i^T \boldsymbol{Q}$ are the left eigenvectors of $\boldsymbol{P}$ with eigenvalues λ_i. In particular, $\boldsymbol{v}_1^T = \mathbf{1}^T \boldsymbol{Q} = \boldsymbol{q}$ has eigenvalue 1.
- The vectors $\boldsymbol{v}_i = \boldsymbol{Q}\boldsymbol{u}_i$ are orthonormal with respect to the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{Q}^{-1}}$.

- Let $\boldsymbol{r} \geq 0$ be a distribution over the states, so a row vector with $\sum_i r_i = 1$ that is $\boldsymbol{r}\mathbf{1} = 1$, and express $\boldsymbol{r} = \sum_i c_i \boldsymbol{v}_i^T$. Computing the first coordinate, using orthonormality of $\boldsymbol{v}_i$ with $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{Q}^{-1}}$:

$$c_1 = \boldsymbol{r}\boldsymbol{Q}^{-1}\boldsymbol{v}_1 = \boldsymbol{r}\mathbf{1} = 1.$$

- By (12.2)

$$\boldsymbol{r}\boldsymbol{P}^t \to c_1 \boldsymbol{v}_1^T = \boldsymbol{u}_1^T = \boldsymbol{q}.$$

Convergence speed again: $\left\| \boldsymbol{r}\boldsymbol{P}^t - \boldsymbol{q} \right\|_{\boldsymbol{Q}^{-1}} = O((1-\delta)^t)$.

Eigenvalue gap for lazy walks

With no negative eigenvalues, $\delta = 1 - \lambda_2$. This can be achieved by creating a lazy version of any graph random walk.
Let $\boldsymbol{P}$ be a random walk on a graph G. Obtain G' by adding self-loops of probability $1/2$ to each point of G: then $\boldsymbol{P}' = (\boldsymbol{I} + \boldsymbol{P})/2$. The eigenvalues $\lambda_i' = (\lambda_i + 1)/2$ of $\boldsymbol{P}'$ are all nonnegative.
In what follows the notation $i \sim j$ means $\{i, j\} \in E$.

- Recall $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$. The eigenvalues $1 - \lambda_i$ belong to $\boldsymbol{I} - \boldsymbol{P}$, with the same eigenvectors $\boldsymbol{u}_i$. The following, Laplacian matrix has the same eigenvalues:
  $\boldsymbol{L} = \boldsymbol{D}^{1/2}(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{D}^{-1/2} = \boldsymbol{D}^{-1/2}(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{D}^{-1/2}$. Its eigenvectors are $\boldsymbol{D}^{1/2}\boldsymbol{u}_i$. Since $\boldsymbol{L}$ is symmetric, they are mutually orthogonal. For $\boldsymbol{y} = \boldsymbol{D}^{-1/2}\boldsymbol{x}$,

$$\begin{aligned}\langle \boldsymbol{x}, \boldsymbol{L}\boldsymbol{x}\rangle &= \langle \boldsymbol{y}, (\boldsymbol{D} - \boldsymbol{A})\boldsymbol{y}\rangle \\ &= \sum_i \left(d_i y_i^2 - \sum_{j:i\sim j} y_i y_j \right) = \sum_{i<j:i\sim j} (y_i - y_j)^2.\end{aligned}$$

- Linear algebra: the second lowest eigenvalue of $\boldsymbol{L}$ is

$$1 - \lambda_2 = \min_{\substack{\langle \boldsymbol{x}, \boldsymbol{D}^{1/2}\mathbf{1}\rangle = 0 \\ \|\boldsymbol{x}\| = 1}} \langle \boldsymbol{x}, \boldsymbol{L}\boldsymbol{x}\rangle = \min_{\sum_i d_i y_i = 0} \frac{\sum_{i<j:i\sim j}(y_i - y_j)^2}{\sum_i d_i y_i^2}. \quad (12.4)$$

The expression (12.4) implies

Theorem 12.11 $1 - \lambda_2 \geq \frac{1}{(\operatorname{diam} G)(\operatorname{vol} G)}$.

Proof. Multiplying each y_i by the same constant does not change $\frac{\sum_{i<j:i\sim j}(y_i-y_j)^2}{\sum_i d_i y_i^2}$. Let us do this to get $\sum_i d_i y_i^2 = \sum_i d_i = \operatorname{vol} G$. Let $y_u = \max_i y_i$, $y_v = \min_i y_i$, then now $y_u - y_v \geq 1$. Take a shortest path $u = v_0, v_1, \ldots, v_k = v$ from u to v. Then

$$\sum_{i<j:i\sim j} (y_i - y_j)^2 \geq \sum_{j=1}^{k} (y_{v_j} - y_{v_{j-1}})^2 \geq (\sum_{j=1}^{k} y_{v_j} - y_{v_{j-1}})^2/k$$
$$= (y_u - y_v)^2/k \geq 1/k \geq 1/\operatorname{diam} G.$$

The inequality bringing in $1/k$ used Cauchy-Schwartz:

$$k \cdot \sum_i x_i^2 = \|\mathbf{1}\|^2 \|\boldsymbol{x}\|^2 \geq \langle \mathbf{1}, \boldsymbol{x} \rangle^2 = (\sum_i x_i)^2.$$

□

Speed of convergence

The number of steps needed to get within distance ε of the equilibrium can be measured by $1/\delta$:

$$e^{-\delta t} < \varepsilon,$$

if $t > \frac{\log(1/\varepsilon)}{\delta}$. A spectrum of possibilities:

- Directed graphs: sometimes exponential-time convergence.
- Undirected graphs, no negative eigenvalue: polynomial-time convergence. Indeed, by Theorem 12.11 $1/\delta < (\operatorname{diam} G)(\operatorname{vol} G) = O(|V|^3)$.
- Sampling, approximate counting an exponential-size subset V of an exponential-size universe U. We need logarithmic (in $|U|$) convergence, hence we need $1/\delta = O((\log |U|)^{-c})$ for some $c > 0$.
- Expanders (see later): constant-time convergence, constant δ.

Sampling via Markov chains (MC Monte-Carlo)

- Sometimes a distribution $\boldsymbol{q}$ is given (it may even be uniform) over a set U. To approximate the probabilities of certain interesting sets, we need to sample from $\boldsymbol{q}$.
- As a special case, $\boldsymbol{q}$ is uniformly distributed on some $V \subset U$. The set V is exponentially large, but it is only an exponentially small part of some easy-to-sample universe U. So we cannot just pick a random element of U repeatedly until we hit V.
- Idea: set up some Markov chain $X_1, X_2, \ldots$ in U, with invariant distribution $\boldsymbol{q}$, starting from some arbitrary element $X_1 = v_1$. (If $\boldsymbol{q}$ is supported by a set V then let $v_1 \in V$.) Simulate X_i. If the distribution of X_n converges fast to $\boldsymbol{q}$ then soon we can stop, and take X_n as an approximate sample.

Example 12.12 (Ising model) Call two pairs (i, j), (i', j') are neighbors and write $(i, j) \sim (i', j')$ if $|i - i'| + |j - j'| = 1$. View a matrix $\boldsymbol{\sigma}$ with elements in $\{-1, 1\}$ like a "ferromagnet" with individual atomic magnets pointing up or down. The energy $U(\boldsymbol{\sigma})$ is defined as

$$-\sum_{u \sim v} \sigma_u \sigma_v.$$

Let $\beta > 0$ be a fixed parameter (the "inverse temperature"). We define $Z_\beta = \sum_{\boldsymbol{\sigma}} e^{-\beta U(\boldsymbol{\sigma})}$, and the probability

$$q_{\boldsymbol{\sigma}}(\beta) = e^{-\beta U(\boldsymbol{\sigma})}/Z(\beta)$$

(the so-called Boltzmann distribution). Many versions of this example are of central interest in physics.

An appropriately designed Markov chain will converge to $\boldsymbol{q}$ and allow sampling from it.

Designing a reversible Markov chain

Suppose that a graph $G = (V, E)$ is given, along with a distribution $\boldsymbol{q}$ over V. Define transition probabilities p_{uv} along the edges such that

$$q_u p_{uv} = q_v p_{vu}, \tag{12.5}$$

so the process is reversible (and thus $\boldsymbol{q}$ is invariant). Define p_{uu} to make sure $\sum_v p_{uv} = 1$. Relation (12.5) remains true if we multiply p_{uv} by c_{uv} where $c_{uv} = c_{vu}$. Let $\boldsymbol{C} = (c_{uv})$ be an arbitrary symmetric stochastic matrix (say $c_{uv} = 1/d$ for $u \neq v$ and maximum degree d). The following formula, called the Metropolis algorithm, is a popular choice:

$$p_{uv} = c_{uv} \min\left(1, \frac{q_v}{q_u}\right) \text{ for } v \neq u.$$

Check that this is reversible!

Example 12.13 (Subgraph) Let $G = (V, E)$ where $V = \{1, \ldots, n\} \times \{1, \ldots, n\}$, $(i, j) \sim (i', j')$ if $|i - i'| + |j - j'| = 1$. Let $\boldsymbol{q}$ be the uniform distribution. The transitions for the Metropolis algorithm give

$$p_{uv} = \frac{1}{4} \min\left(1, \frac{q_v}{q_u}\right) \text{ for } v \neq u.$$

In words: try to move in each of the 4 directions (north, south, east, west) with probability 1/4. If the move would take you outside the square, stay in place.
This is not quite the the same as the random walk on the graph, it gives larger stationary probability to the border points.

Example 12.14 (Metropolis algorithm for the Ising model)

For states $\boldsymbol{\sigma}, \boldsymbol{\sigma}'$ of the Ising model say $\boldsymbol{\sigma} \sim \boldsymbol{\sigma}'$ if they differ only in a single position $u = (i, j)$, that is $\sigma_u \neq \sigma'_u$. In this case

$$\frac{q_{\boldsymbol{\sigma}'}}{q_{\boldsymbol{\sigma}}} = e^{\beta(U(\boldsymbol{\sigma}) - U(\boldsymbol{\sigma}'))}$$

depends only the values of σ_v for neighbors v of lattice point u. Such Markov processes are called probabilistic cellular automata.

Example 12.15 (Counting matchings) Let M_k be the set of matchings of size k in a bipartite graph G. Under certain conditions one can estimate M_k for each k. For this, one will estimate

$$\frac{|M_k|}{|M_{k-1} \cup M_k|}$$

for each k starting from $k = 1$.

Define a graph whose points are matchings in $M_{k-1} \cup M_k$. Let m, m' be two matchings $m, m' \in M_k \cup M_{k-1}$. We say that they are neighbors if there is an $e = (u, v) \in V^2$ such that one of the following holds:

Addition $e \in E$, $m \in M_{k-1}$, $m' \in M_k$, $m' = m \cup \{e\}$.

Deletion $e \in E$, $m' \in M_{k-1}$, $m \in M_k$, $m = m' \cup \{e\}$.

Rotation $m, m' \in M_{k-1}$ and there is a w with $(u, w) \in m$ and $(v, w) \in m'$.

The Markov chain is defined by the Metropolis algorithm: choose a random (u, v), apply an operation determined by (u, v) if it is possible, else do nothing.
Irreducibility is easy to establish; the difficulty is estimating the eigenvalue gap.

Conductance

The following quantities may help estimating the eigenvalue gap. For any $S \subset V$ let $q(S) = \sum_{i \in S} q_i$, and $\delta(S)$ the set of edges between S and $V \setminus S$. By $q(\delta(S)) = \sum_{(i,j) \in \delta(S)} q_i p_{ij}$ we measure the flow through the edges of $\delta(S)$. The conductance

$$\Phi' = \min_{q(S) \leq 1/2} \frac{q(\delta(S))}{q(S)}.$$

lowerbounds the relative probability flow out of a small part S of the graph. A related quantity (sometimes this is called conductance):

$$\Phi(S) = \frac{q(\delta(S))}{q(S)q(V \setminus S)}, \qquad \Phi = \min_S \Phi(S).$$

Clearly $\Phi' \leq \Phi \leq 2\Phi'$.

If the graph is regular, then $\operatorname{vol} G = dn$, $q(S) = |S|/n$,

$$\Phi(S) = \frac{|\delta(S)|}{d|S| \cdot |V \setminus S|/n}, \qquad \Phi' = \min_{|S| \le n/2} \frac{|\delta(S)|}{d|S|}.$$

Theorem 12.16 We have $\Phi^2/8 \le 1 - \lambda_2 \le \Phi$.

The upper bound follows easily by choosing y_i to be constant on S and on $V \setminus S$. The lower bound is harder, we will take it on faith.

The following method sometimes helps.
Suppose that for each $i \neq j$ we can construct a flow f_{ij} of value $q_i q_j$ from i to j, in such a way that the sum of all these flows on any edge (i, j) is at most $c q_i p_{ij}$.

Theorem 12.17 If the above flow has been defined then $\Phi \geq \frac{1}{c}$.

Proof. Let $\Phi = \Phi(S)$. The total amount of flow from S to $V \setminus S$ is $q(S)q(V \setminus S)$. Summing these up on the edges of the cut $\delta(S)$, we get

$$q(S)q(V \setminus S) \leq q(\delta(S))c, \qquad \frac{1}{c} \leq \frac{q(\delta(S))}{q(S)q(V \setminus S)} = \Phi(S).$$

□

Flow method for undirected graph random walks

Scaling up, from i to j we send a flow $d_i d_j$. On each edge, the sum of flows passing through should be $\le c \cdot \operatorname{vol} G$.

Example 12.18 Let G be the $n \times n \times n$ lattice cube, the degrees are ≤ 6, $\operatorname{vol} G = 6n^3(1 - o(1))$. From point (i, j, k) to point (i', j', k') we send a flow of size ≤ 36 along the path that first goes from i to i', then from j to j', then from k to k'. Each $i - i'$ edge carries a flow sum of at most $n \cdot 36n^3 = 36n^4$. Only $1/n$ of these continues after the turn to the $j - j'$ edges, but the flow can accumulate again, keeping the bound $36n^4$. The same happens on the $k - k'$ edges, so the bound on the sum of flows is $36n^4 \le (6 + o(1))n \cdot \operatorname{vol} G$. By Theorem 12.17, $\Phi \ge 1/7n$ when n is large. This also proves Lemma 8.4.

Your book illustrates this technique on the example of the Markov chain on matchings.

Expanders

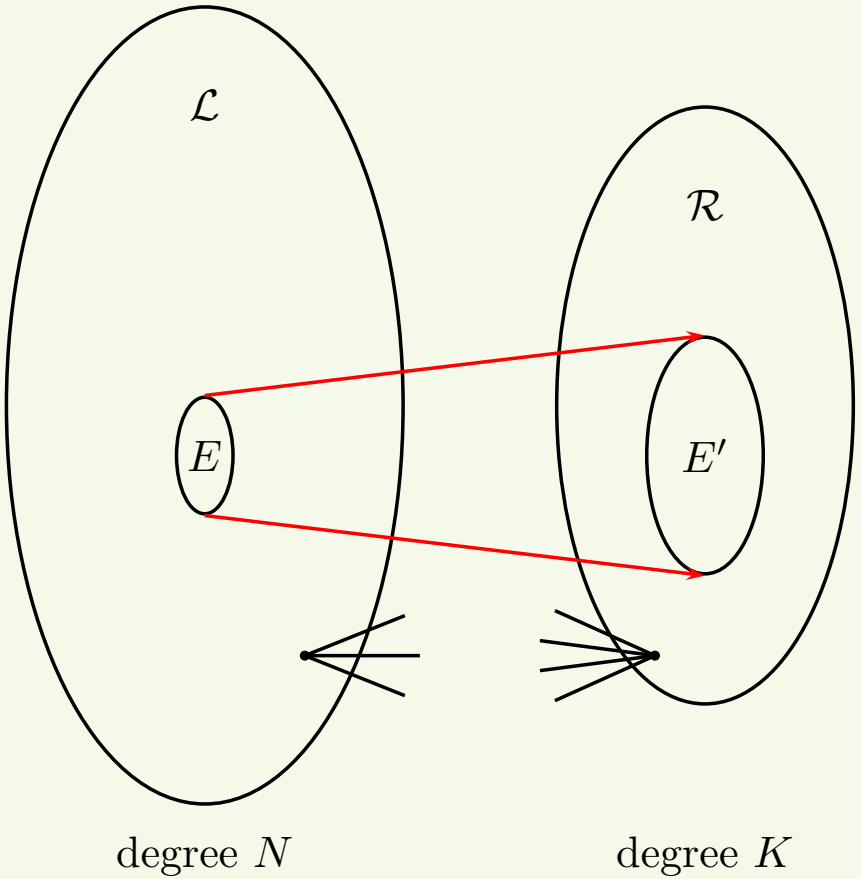
- An "expander" is essentially a graph whose eigenvalue gap is lowerbounded by some constant. We generally look at a family of expanders, with infinitely many possible sizes, all sharing the same eigenvalue gap lower bound.
- Restrict attention to bipartite, regular graphs, but allow multigraphs: edges have integer multiplicity.

The eigenvalue gap measures, in a way, the degree of connectivity, as seen from the formula

$$\Phi' = \min_{|S| \leq n/2} \frac{|\delta(S)|}{d|S|}.$$

seen earlier. This says that for any set S of size $\leq n/2$, at least a fraction Φ' of all edges starting from points of S must go to $V \setminus S$.

- Bipartite graph: left and right parts. For a subset S of the left set of $G = (V, E)$, let $\mathrm{Nb}(S)$ be the set of neighbors of S: points connected by an edge to some element of S.
- The graph expands S by a factor λ if $|\mathrm{Nb}(S)| \geq \lambda |S|$.
- For $d, \alpha, \lambda > 0$, a d-regular bipartite multigraph G is a (d, α, λ, n)-expander if it expands every subset S of size $\leq \alpha n$ of the left set by a factor λ.


$\mathcal{L}$
$\mathcal{R}$
E
E'
degree N
degree K

Random expanders

- Are there expanders with constant d, α, λ and arbitrarily large n? The proof illustrates the probabilistic method.
- We will choose a random bipartite multigraph of degree d and show that it is expander with positive probability. Choosing α small (constant) will bring λ arbitrarily close to d (you cannot hope better).

- Start with dn left nodes $u_1, \ldots, u_{dn}$ and dn right nodes $v_1, \ldots, v_{dn}$. Choose a random complete matching among these. Call the resulting graph M.
- Obtain G as follows: collapse each group of d left nodes into a single node: $u_1, \ldots, u_d$ into one node, $u_{d+1}, \ldots, u_{2d}$ into another node, and so on. Similarly collapse each group of d right nodes. Edges are inherited from the ancestors. The process may give multiple edges: a multigraph B. Two nodes of M are called cluster neighbors if they are collapsed to the same node of B.

Theorem 13.1 For each $f > 0$, if α is sufficiently small then this process gives a $(d, \alpha, d(1-f), n)$-expander with positive probability for all sufficiently large n.

In what follows we will prove this.

Let S be a set of size αn in the left set of G. Estimate the probability that it has too few neighbors. Assign edges to the nodes of S in some fixed order of the preimage of S in M. Call a node of the right set of M occupied if it has a cluster neighbor already reached by an earlier edge. Let random variable X_i be 1 if the ith edge goes to an occupied node and 0 otherwise. There are

$$dn - i + 1 \geq dn - d\alpha n = dn(1 - \alpha)$$

choices for the ith edge, at most $d^2|S|$ of these are occupied. Therefore

$$\mathsf{P}\{ X_i = 1 \mid X_1, \ldots, X_{i-1} \} \leq \frac{d^2|S|}{dn(1-\alpha)} = \frac{d\alpha}{1-\alpha} =: p.$$

By Theorem 1.23 (“Chernoff bound”) and (1.5):

$$\mathsf{P}\left\{\sum_{i=1}^{d\alpha n} X_i \geq fd\alpha n\right\} \leq e^{d\alpha n \overline{H}_p(f)} \leq \left(\frac{ep}{f}\right)^{fd\alpha n}.$$

The number of different neighbors of S is $d\alpha n - \sum_i X_i$, hence

$$\mathsf{P}\{\,\mathrm{Nb}(S) \leq d\alpha n(1-f)\,\} \leq \left(\frac{ep}{f}\right)^{fd\alpha n} = \left(\frac{ed\alpha}{f(1-\alpha)}\right)^{fd\alpha n}.$$

Multiply with the number of sets S of size $\leq \alpha n$ as estimated in (1.6):

$$\sum_{i\leq\alpha n}\binom{n}{i} \leq \left(\frac{e}{\alpha}\right)^{\alpha n},$$

$$\left(\frac{e}{\alpha}\right)^{\alpha n}\left(\frac{ed\alpha}{f(1-\alpha)}\right)^{fd\alpha n} = \left(\frac{e^2 d}{f(1-\alpha)}\left(\frac{e\alpha d}{f(1-\alpha)}\right)^{fd-1}\right)^{\alpha n}.$$

The base is < 1 if $f > 1/d$ and α is sufficiently small.

The above proof is an existence proof.

- Not only does not it help us compute an expander efficiently, but even if we are handed one, does not give any effective way of checking it.
- However, if we just want a large eigenvalue gap, that can be computed effectively.
- In many theoretical applications, we need efficiently computable expanders: say, of size 2^n in which the neighbors of each point are listed in time polynomial in n. Such constructions exist. In this course, we will not see them, since the proofs are generally complicated. But we will see an application.

BPP amplification

Let $L \in$ BPP, with a randomized polynomial algorithm $A(x, r)$, that for each x decides whether it is in L, and fails only with probability $\leq 1/100$. Assume $|r| = |x| = n$. How to decrease the error probability to, say, 2^{-k}? We learned to repeat independently some number of times and to take the majority. In some cases, we must be parsimonious with the number of random bits used. Two interesting possibilities:

- Use independent repetitions $r_1, \ldots, r_m$. The Chernoff bound shows that $O(nk)$ random bits suffice, and the number of operations still polynomial in n, k.
- Use pairwise independent random strings of the form $r_i = ai + b \pmod{p}$: then $O(n + k)$ random bits suffice, but the number of operations becomes exponential in k.

With constructive expanders, we will only use $n + O(k)$ random bits and still a polynomial number of operations.

Algorithm 1 (Call it $B(x, r)$.) Let $N = 2^n$. For some constants d, δ, take a d-regular undirected graph $G_N = (V_N, E_N)$ for $V_N = \{1, 2, \ldots, N\}$. With self-loops, we also achieve that λ_2 is the second largest eigenvalue. Assume $\delta \leq 1 - \lambda_2$, so G is an "expander". The graph must be given in such a way that for each point i, the list of its d neighbors is computable in time polynomial in n.

Choose a random $r_0 = R(0) \in V_N$ uniformly. Let $R(0), R(1), R(2), \ldots$ be a random walk over G_N. Given $R(i-1)$, we need only a new random binary string r_i of size $\lceil \log d \rceil$ to find $R(i+1)$, since we only need to choose between the outgoing edges. Let β be such that $\lambda_2^\beta \leq 0.1$. Compute $A(x, R(i\beta))$ for each $i = 1, \ldots, m$ and take the majority.

We will achieve $\mathsf{P}\{ B(x, r) \text{ is wrong} \} \leq 2^{-k}$ with $m = O(k)$.

Let $X_1, X_2, \ldots \in \{0, 1\}$ be independent variables with $\mathsf{P}\{ X_i = 1 \} \leq \sigma$, and let $i_1 < i_2 < \cdots < i_k$. Then $\mathsf{P}\left\{ X_{i_1} = 1, \ldots, X_{i_k} = 1 \right\} \leq \sigma^{-k}$. We will develop a similar estimate for our Markov chain.

Let $\boldsymbol{P}$ be transition matrix obtained by making β steps of the random walk on our expander. Then $\lambda_2(\boldsymbol{P}) \leq 0.1$. The equilibrium distribution $\boldsymbol{\pi}$ is still the uniform distribution.

For a matrix $\boldsymbol{A}$ let $\|\boldsymbol{A}\| = \|\boldsymbol{A}\|_2 = \max_{\boldsymbol{x}\neq\boldsymbol{0}} \|\boldsymbol{x}\boldsymbol{A}\|/\|\boldsymbol{x}\|$. It is easy to see $\|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.
Let $\boldsymbol{A}$ be a symmetric matrix with eigenvalues λ_i, then it is easy to see

$$\|\boldsymbol{A}\| = \max_i |\lambda_i|.$$

Let $X_i = R(i\beta)$, then X_i is a random walk with matrix $\boldsymbol{P}$ and initial distribution $\boldsymbol{q}$. For a set $S \in V$, let $\boldsymbol{D}_S$ be the diagonal matrix with 1's in positions $i \in S$ and 0's elsewhere. We have

$$\begin{aligned}\mathsf{P}\{X_1 \in S_1, \ldots, X_m \in S_m\} &= \left\|\boldsymbol{q}\boldsymbol{P}\boldsymbol{D}_{S_1}\cdots\boldsymbol{P}\boldsymbol{D}_{S_m}\right\|_1 \\ &\leq \sqrt{N}\left\|\boldsymbol{q}\boldsymbol{P}\boldsymbol{D}_{S_1}\cdots\boldsymbol{P}\boldsymbol{D}_{S_m}\right\|.\end{aligned}$$

Though our final interest is in the norm $\|\cdot\|_1$, due to the better properties of the norm $\|\cdot\| = \|\cdot\|_2$ we will go through estimating $\left\|\boldsymbol{P}\boldsymbol{D}_{S_1}\cdots\boldsymbol{P}\boldsymbol{D}_{S_m}\right\| \leq \left\|\boldsymbol{P}\boldsymbol{D}_{S_1}\right\|\cdots\left\|\boldsymbol{P}\boldsymbol{D}_{S_m}\right\|$.

Lemma 13.2 Let S be a set of states with $\boldsymbol{\pi}(S) \le \sigma$. We have

$$\|\boldsymbol{P}\boldsymbol{D}_S\| \le \min(1, \sigma^{1/2} + \lambda_2).$$

Proof. Let $\boldsymbol{\pi}, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$ be an orthonormal basis of row eigenvectors of $\boldsymbol{P}$. For any vector $\boldsymbol{q} = c_1\boldsymbol{\pi} + \sum_i c_i\boldsymbol{u}_i =: \boldsymbol{u} + \boldsymbol{v}$ we have

$$\begin{aligned}
\boldsymbol{q}\boldsymbol{P} &= \boldsymbol{u} + \sum_{i>1} \lambda_i c_i \boldsymbol{u}_i =: \boldsymbol{u} + \boldsymbol{v}', \\
\|\boldsymbol{v}'\| &\le \lambda_2 \|\boldsymbol{v}\|, \\
\|\boldsymbol{q}\boldsymbol{P}\boldsymbol{D}_S\| &\le \|\boldsymbol{q}\boldsymbol{P}\| \le \|\boldsymbol{q}\|, \\
\|\boldsymbol{q}\boldsymbol{P}\boldsymbol{D}_S\| &\le \|\boldsymbol{u}\boldsymbol{D}_S\| + \|\boldsymbol{v}'\boldsymbol{D}_S\| \\
&\le \sigma^{1/2}\|\boldsymbol{u}\| + \lambda_2\|\boldsymbol{v}\| \le (\sigma^{1/2} + \lambda_2)\|\boldsymbol{q}\|.
\end{aligned}$$

□

Let $S = \{ r : A(x, r) \text{ is wrong } \}$, and assume $\boldsymbol{\pi}(S) \leq 0.01$. Using $\lambda_2 \leq 0.1$ and lemma 13.2 gives $\|\boldsymbol{P}\boldsymbol{D}_S\| \leq 0.1 + 0.1 = 1/5$.
Let $S_1, S_2, \ldots, S_m$ be a sequence of sets where each S_i is either S or $V_N \setminus S$ and there are w occurrences of S. Then we obtain

$$\begin{aligned}
\|\boldsymbol{\pi}\| &= N^{-1/2}, \\
\left\|\boldsymbol{P}\boldsymbol{D}_{S_1} \cdots \boldsymbol{P}\boldsymbol{D}_{S_m}\right\| &\leq (1/5)^w 1^{m-w} \leq (1/5)^w, \\
\left\|\boldsymbol{\pi}\boldsymbol{P}\boldsymbol{D}_{S_1} \cdots \boldsymbol{P}\boldsymbol{D}_{S_m}\right\| &\leq \|\boldsymbol{\pi}\|\left\|\boldsymbol{P}\boldsymbol{D}_{S_1} \cdots \boldsymbol{P}\boldsymbol{D}_{S_m}\right\| \leq N^{-1/2}(1/5)^w, \\
\left\|\boldsymbol{\pi}\boldsymbol{P}\boldsymbol{D}_{S_1} \cdots \boldsymbol{P}\boldsymbol{D}_{S_m}\right\|_1 &\leq N^{1/2}N^{-1/2}(1/5)^w = (1/5)^w.
\end{aligned}$$

A wrong majority decision corresponds to a sequence $S_1, \ldots, S_m$ with at least $w \geq m/2$ occurrences of S. There are at most 2^m different ways to choose this sequence, therefore the probability of a wrong decision is at most $2^m(1/5)^{m/2} = (4/5)^{m/2}$.

Note $\|\boldsymbol{\pi}\| = N^{-1/2}$ was important. Starting from a single point instead of the uniform distribution does not work.

What is probability?

What is probability? The question is often answered referring to relative frequency in repeated independent experiments.

- The classical law of large numbers offers an "internal" justification of this interpretation.
- The Markov chain applications go beyond this interpretation. The BPP amplification does not repeat experiments independently, and still achieves a very small probability of error.
- Markov-chain Monte-Carlo works on a space of exponential size, but takes a sample in a polynomial number of steps that has nearly the desired distribution. Typically each sample will be different, making a frequentist interpretation again dubious.

Randomness

- Probability theory started approximately with Pascal (mid 17th century).
- Its current mathematical framework was introduced by Kolmogorov (1931). This framework does not address the question: what is a random string?
- This question was raised maybe first by Laplace (early 19th century), then by von Mises (early 20th century). Von Mises proposed the convergence of relative frequency on subsequences generated by some "rule" as a criterion. But he could not formalize the notion of a rule.
- Church (1947) defined "rule" as a recursive rule. But the frequency test proved insufficient, as shown by Ville.
- The current theory is based on Kolmogorov's introduction of desciption complexity (1965) and later work by Martin-Löf and Levin.

The paradox giving rise to the notion of randomness is this. Suppose your friend gives you a 0-1 sequence $x_1x_2\cdots x_{100}$ of length 100, and told that it is the result of a series of coin tosses he made the day before. If the sequence is $010101\cdots01$ then you will not believe your friend. He may challenge you for your reasons, however, since each sequence of length 100 has the same probability 2^{-100}.

You may say, "yes, but this sequence is created by a rule", but what is a rule? There is an infite number of possible rules.

Laplace's guessed that there are only few sequences obeying a simple rule, but without saying what a simple rule is. Kolmogorov formalized the notion of the simplicity needed here.

Description complexity

Fix some alphabet $\Sigma \supset \{0, 1\}$. Let T be a Turing machine with

- input tape with alphabet $\{0, 1, *\}$, where $*$ is the "blank symbol",
- output tape with alphabet $\Sigma \cup \{*\}$.
- work tape with some alphabet containing $*$.

We define the partial function $T : \{0, 1\}^* \to \Sigma^*$ as follows. To compute $T(p)$ we write string p onto the input tape of T, leave the other tapes blank. We start T. If T halts and the output tape contains a single nonblank string x at the beginning, then $T(p) = x$. Otherwise $T(p)$ is not defined.

We view the machine T as an interpreter of descriptions. Thus, we way that the binary string p describes the output $T(p)$. (We use binary descriptions for having a common base of comparison.) Let

$$K_T(x) = \min_{T(p)=x} |p|.$$

We say that $K_T(x)$ is the description complexity, or Kolmogorov complexity, of string x on machine T. This notion is, of course, machine-dependent. For every string x there is an interpreter T_x such that $K_{T_x}(x) = 0$. But interestingly, the machine-dependence is quite moderate.

Theorem 14.1 (Invariance) There is an interpreter U that is optimal in the following sense. For every other interpreter T there is a constant c_T such that for all strings $x \in \Sigma^*$ we have

$$K_U(x) \leq K_T(x) + c_T.$$

The theorem shows that no other interpreter T can have much shorter descriptions than the optimal interpreter U, since the difference will be bounded by a constant (dependent on T).

Proof. There is an interpreter (an appropriate universal Turing machine) U such that for all interpreters T there is a string q_T such that for all strings $p \in \{0, 1\}^*$ we have

$$T(p) = U(q_T p).$$

Here, q_T contains an encoded description of the machine T (its transition table). Now for all strings x and all machines T we have

$$K_U(x) \leq K_T(x) + |q_T|.$$

□

Fix an optimal interpreter U and write

$$K(x) = K_U(x).$$

We will use the notation $f(n) \stackrel{+}{<} g(n)$ to mean $f(n) \leq g(n) + O(1)$. The notion $f(n) \stackrel{+}{=} g(n)$ is defined similarly.

The function $K(x)$ has several synonymous names, some of them suggest other interpreations.

- Description complexity.
- Minimal compression size.
- Amount of individual information.
- Algorithmic information.
- Algorithmic entropy.

Theorem 14.2 For a binary string x of length n we have $K(x) \stackrel{+}{<} n$.

Proof. Define a machine T that simply outputs its input. Then apply the invariance theorem. □

The following theorem shows that this bound is sharp, in a statistical sense.

Theorem 14.3 Let X be a uniformly chosen random binary string of length n. Then

$$\mathsf{P}\{\, K(X) < n - k \,\} < 2^{-k}.$$

Proof. There are at most 2^i descriptions of length i, so at most $1 + 2 + 4 + \cdots + 2^{n-k-1} < 2^{n-k}$ strings with complexity $< n - k$. □

Example 14.4 Note that if the string x is of the form $0101\cdots01$ ($n/2$ times) then

$$K(x) \stackrel{+}{<} \log n.$$

Since as we have seen, the probability is very small of getting any string of such low-complexity, we have a sort of justification for our suspicion about its coin-tossing pedigree. In fact, a good argument can be made (in the more advanced versions of this theory) to view $n - K(x)$ as a measure of the non-randomness of string x.

Example 14.5 If x is a binary string of length n such that $\sum_i x_i = k$ then with $p = k/n$ we have

$$K(x) \stackrel{+}{<} \log \binom{n}{k} + O(\log n) = nH(p) + O(\log n),$$

where $H(p) = -p \log p - (1-p)\log(1-p)$. (This indicates a relation to information.)
Indeed, given n and k (by strings of length $\log n$) we can enumerate all binary strings containing k 1's, and describe x by the rank of x in this enumeration.

The following theorem limits the usefulness of the function $K(x)$.

Theorem 14.6 The function $K(x)$ is not computable.

Proof. (By contradiction.) Assume that $K(x)$ is computable. For each k let $f(k)$ be the smallest x (lexicographically) with $K(x) > k$, and let T be a Turing machine computing $f(k)$ from a binary representation of k. We have

$$K(f(k)) \leq K_T(f(k)) + c_T \leq \log k + 1 + c_T.$$

But by definition we have $k < K(f(k))$, leading to a contradiction for large k. □

This proof formalizes the paradox: "the smallest number defineable with fewer than 100 characters".

Despite its non-computability, the notion of description complexity and the definitions of randomness and information built on it have proved of great value in clarifying some important issues of randomness, information and prediction.
In randomized computations and in cryptographical applications we frequently work with strings that are not random, only pseudo-random; again, in order to understand the difference, it is useful to know what randomness means.

Frequently, a randomized algorithm can help us to a deterministic one. We will illustrate this on the set balancing problem, which we recall.
Given an $n \times m$ matrix $\boldsymbol{A}$ with 0-1 entries. We are looking for a vector $\boldsymbol{b}$ with $1, -1$ entries for which

$$\|\boldsymbol{A}\boldsymbol{b}\|_\infty$$

is minimal.

A good vector can be found by random choice. Let $\beta_1, \ldots, \beta_n$ be independent, with $\mathsf{P}\left\{ \beta_j = 1 \right\} = \mathsf{P}\left\{ \beta_j = -1 \right\} = 1/2$, and let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$. Let $\boldsymbol{a}_i^T$ the ith row, let $\mathcal{E}_i$ be the event $|\boldsymbol{a}_i^T \boldsymbol{\beta}| > 8\sqrt{m \ln n}$. We showed

$$\mathsf{P}\left\{ \mathcal{E}_i \right\} \leq 2/n^2.$$

So with probability $1 - 2/n$, all n rows have discrepancy $\leq 8\sqrt{m \ln n}$. Let the random variable X_i be 1 if $\mathcal{E}_i$ holds and 0 otherwise. Let $Y = \sum_i X_i$. Our theorem shows $\mathsf{E}\, Y \leq 2/n$.

Consider a tree with all possible values $b_1 \cdots b_n$ of the sequence $\beta_1 \cdots \beta_n$ at the leaves, and the inner nodes at level k steps down from the top labeled by values of $b_1 \cdots b_k$. If you think into it you will see that for each $b_1 \cdots b_k$, the value

$$f(b_1 \cdots b_k) = \mathsf{E}\{\, Y \mid \beta_1 = b_1, \ldots, \beta_k = b_k \,\}$$

is computable in polynomial time, and

$$f(b_1 \cdots b_k) = (f(b_1 \cdots b_k 0) + f(b_1 \cdots b_k 1))/2.$$

Therefore the following algorithm works for computing a sequence $b_1 \cdots b_n$ recursively: For $k = 0, \ldots, n-1$, compute the b_k such that

$$f(b_1 \cdots b_{k-1} b_k) \leq f(b_1 \cdots b_{k-1}(1 - b_k)).$$

With this sequence we will have $\|\boldsymbol{A}\boldsymbol{b}\|_\infty \leq 8\sqrt{m \ln n}$. Indeed, we have

$$\mathsf{E}\{\, Y \mid \beta_1 = b_1, \ldots, \beta_n = b_n \,\} \leq 2/n < 1.$$

But this is a deterministic integer value: if its expectation is < 1 then it is 0: in other words, none of the events $\mathcal{E}_i$ happens.

Two-point sampling

- The following example is from the area of parallel computing. We have had such an example earlier, since the randomized algorithm to decide whether a graph has a matching adds value only in the parallel computing model.
- We will not give a formal definition of a parallel computer. Different versions exist (EREW, CEREW, etc.). The class NC of functions computable in polylog time and with a polynomial number of processors is insensitive to these differences.

Given a graph $G = (E, V)$, we want to find a maximal independent set (an independent set not contained in any other independent set). (Finding a maximum independent set is NP-complete.) It is easy to find one in polynomial time, but the greedy algorithm of finding one is sequential. The parallel algorithm of Luby given below puts the problem into NC.

Plan:

1. Find a randomized parallel algorithm (putting the problem in RNC, randomized NC).
2. Derandomize it.

We will follow the scheme only approximately.

Definition 15.1 For a subset $S \subseteq V$ of the graph G, let $E(S)$ be the set of edges of G with at least one end in S.
Let $\Gamma(v)$ be the set of neighbors of point v, $d_v = |\Gamma(v)|$ its degree, and $\Gamma S = \bigcup_{v \in S} \Gamma(v)$.

Theorem 15.2 There is a constant c and a randomized parallel algorithm to find an independent set $S(r) \subseteq V$ with
$\mathsf{E}\,|E(\Gamma S(r))| \geq c|E|$.

We can repeat this algorithm on $V_1 = V \setminus \Gamma S(r)$. Iterating similarly a logarithmic number of times gives a maximal idependent set.

Algorithm 2 We construct the set S as follows.

1. (in parallel) Put each point $v \in V$ with probability $\frac{1}{2d_v}$ into a set S_1. Assume first that the choice is made independently for each point.
2. (in parallel) If an edge has both ends in S_1 then delete the lower-degree end from S_1 (break ties arbitrarily), resulting in the desired set S.

The following probabilistic analysis lowerbounds the expected value $\mathsf{E}\,|E(\Gamma S)|$.

Definition 15.3 A point is called good if it has at least $d_v/3$ neighbors with degree $\leq d_v$; otherwise it is bad. An edge is good, if at least one of its endpoints is good.

The following lemma allows us to concentrate on good edges:

Lemma 15.4 At least of half of all edges are good.

Proof. Let us direct the edges from lower to higher degree, and let d_v^+, d_v^- denote the in- and outdegrees. Let V_B, V_G be the bad and good points, $e(X, Y)$ be number of edges between sets X, Y.
In a bad point $d_v \le 3(d_v^+ - d_v^-)$. Hence the sum of degrees of bad points is $\le 3\times$ (the flow from bad points to good points):

$$\begin{aligned} 2e(V_B, V_B) + e(V_B, V_G) + e(V_B, V_G) &\le 3(e(V_B, V_G) - e(V_G, V_B)) \\ &\le 3(e(V_B, V_G) + e(V_G, V_B)), \\ e(V_B, V_B) &\le e(V_B, V_G) + e(V_G, V_B). \end{aligned}$$

□

Let $c_1 = 1 - e^{-1/6}$.

Lemma 15.5 If v is a good point with positive degree then it is in ΓS_1 with probability $\geq c_1$.

Lemma 15.6 Every point $v \in S_1$ will also be in S with probability $\geq \frac{1}{2}$.

Lemma 15.4 says that at least half of the edges are good. Lemma 15.5 implies that a good edge will get an endpoint into ΓS_1 with probability $\geq c_1$. Lemma 15.6 implies that this endpoint will be also in ΓS with probability $\geq \frac{1}{2}$. Multiplying the results lowerbounds the expected value:

$$\mathsf{E}\,|E(\Gamma S)| \geq \tfrac{1}{2} \cdot c_1 \cdot \tfrac{1}{2} = c_1/4.$$

The theorem follows with $c = c_1/4$.

Proof of Lemma 15.5. We make at least $d_v/3$ independent attempts with probability $\geq \frac{1}{2d_v}$, so the the probability of not succeeding even once is $\leq (1 - \frac{1}{2d_v})^{d_v/3} \leq e^{-1/6}$. □

Proof of Lemma 15.6. A point v will only be deleted if a neighbor of degree $\geq d_v$ has been selected. For each such neighbor this happens with probability $\leq \frac{1}{2d_v}$. Apply the union bound to the at most d_v such neighbors. □

Derandomization

- We used n independent random variables $X_v(r)$ where $X_v(r) = 1$ if point v is selected, and 0 otherwise, and $\mathsf{P}\{ X(r)_v = 1 \} = \frac{1}{2d_v}$. The only lemma in which we used independence was Lemma 15.5.
- In a homework problem we will see that with a different constant c_1 the lemma will also hold when only pairwise independence is used.
- Two-point sampling creates p independent random variables $Y_v(r)$ ranging over $\{0, \ldots, p-1\}$, using only $2 \log p$ independent random bits. Using an appropriate p (say $p = O(n^2)$ this can be used to generate pairwise independent random variables $X_v(r)$, with $\mathsf{P}\{ X_v(r) = 1 \} \approx \frac{1}{2d_v}$.
- We still have $\mathsf{E}\,|E(\Gamma S(r))| \geq c|E|$, so there is a particular choice r_0 of these bits giving $|E(\Gamma S(r_0))| \geq c|E|$. Since now $|r| = O(\log n)$, we can search for r_0 in polynomial time.