Computer Science 70: Extra Notes

Fall 2017

Sinho Chewi

Contents

Extra Note 1

Cauchy Induction

1.1 Introduction

Cauchy induction is an interesting technique which demonstrates the flexibility of the principle of mathematical induction. First, we prove a statement $P(k)$ is true for all powers of two.

$$P(k) \implies P(2k) \tag{1.1}$$

Next, we prove that the statement $P(k)$ implies the *previous* statement.

$$P(k) \implies P(k-1) \tag{1.2}$$

(1.1) jumps forward by powers of two $(1, 2, 4, 8, 16, \dots)$, and then (1.2) goes *backwards* in order to fill in the gaps. Hence, Cauchy induction is also known as **forward-backward induction**. Together with the base case, (1.1) and (1.2) are sufficient to prove a statement $P(n)$ for all $n \in \mathbb{N}$.

1.2 Arithmetic Mean vs. Geometric Mean

We will examine the most famous application of Cauchy induction: proving that the arithmetic mean is larger than the geometric mean. First, some definitions:

Definition 1.1. The **arithmetic mean** of two numbers $x, y \in \mathbb{R}$ is $(x+y)/2$. More generally, if $x_1, \dots, x_n$ are real numbers, their arithmetic mean is

$$\mathrm{AM}(x_1, \dots, x_n) = \frac{x_1 + \cdots + x_n}{n} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{1.3}$$

Definition 1.2. The **geometric mean** of two non-negative reals x and y is $\sqrt{xy}$. More

generally, if $x_1, \ldots, x_n$ are non-negative real numbers, their geometric mean is

$$\mathrm{GM}(x_1, \ldots, x_n) = (x_1 \cdots x_n)^{1/n} = \left(\prod_{i=1}^{n} x_i \right)^{1/n}. \tag{1.4}$$

We will prove that $\mathrm{AM}(x_1, \ldots, x_n) \geq \mathrm{GM}(x_1, \ldots, x_n)$, where each x_i is a non-negative real. It is helpful to prove the case of $n = 2$ first.

Lemma 1.3. *Let x, y be non-negative real numbers. Then*

$$\frac{x+y}{2} \geq \sqrt{xy}. \tag{1.5}$$

Proof. Observe that

$$x - 2\sqrt{xy} + y = (\sqrt{x} - \sqrt{y})^2 \geq 0.$$

Rearranging the LHS yields $(x+y)/2 \geq \sqrt{xy}$. □

Proposition 1.4 (Cauchy AM-GM)**.** *Let $x_1, \ldots, x_n$ be non-negative real numbers. Then* $\mathrm{AM}(x_1, \ldots, x_n) \geq \mathrm{GM}(x_1, \ldots, x_n)$.

Proof. If $n = 0$, there is nothing to prove. For the base case, let $n = 1$. Then $\mathrm{AM}(x_1) = x_1 = \mathrm{GM}(x_1)$.

Suppose that we have $\mathrm{AM}(x_1, \ldots, x_k) \geq \mathrm{GM}(x_1, \ldots x_k)$ for $k \in \mathbb{N}$ and any non-negative real numbers $x_1, \ldots, x_k$. Then, if we have non-negative real numbers $x_1, \ldots, x_{2k}$,

$$\begin{aligned}
\mathrm{AM}(x_1, \ldots, x_{2k}) &= \frac{1}{2k} \sum_{i=1}^{2k} x_i = \frac{1}{2} \left(\frac{1}{k} \sum_{i=1}^{k} x_i + \frac{1}{k} \sum_{i=k+1}^{2k} x_i \right) \\
&\geq \frac{1}{2} \left[\left(\prod_{i=1}^{k} x_i \right)^{1/k} + \left(\prod_{i=k+1}^{2k} x_i \right)^{1/k} \right] \geq \left[\left(\prod_{i=1}^{k} x_i \right)^{1/k} \left(\prod_{i=k+1}^{2k} x_i \right)^{1/k} \right]^{1/2} \\
&= \left(\prod_{i=1}^{2k} x_i \right)^{1/(2k)} = \mathrm{GM}(x_1, \ldots, x_{2k}).
\end{aligned}$$

where the second-to-last line is a consequence of (1.5). We have established (1.1).

Suppose that we have $\mathrm{AM}(x_1, \ldots, x_k) \geq \mathrm{GM}(x_1, \ldots, x_k)$ for $k \in \mathbb{N}$, $k \geq 2$, for any non-negative real numbers $x_1, \ldots, x_k$. If $x_1, \ldots, x_{k-1}$ are non-negative real numbers, let

$z = \mathrm{AM}(x_1, \ldots, x_{k-1})$. Then

$$\begin{aligned}\mathrm{AM}(x_1, \ldots, x_{k-1}, z) &= \frac{1}{k}\left(\sum_{i=1}^{k-1} x_i + z\right) = \frac{1}{k}\left(\sum_{i=1}^{k-1} x_i + \frac{1}{k-1}\sum_{i=1}^{k-1} x_i\right) \\ &= \frac{1}{k}\left(1 + \frac{1}{k-1}\right)\sum_{i=1}^{k-1} x_i = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i = \mathrm{AM}(x_1, \ldots, x_{k-1}).\end{aligned}$$

The reason for this should be intuitively clear: z is the average of $x_1, \ldots, x_{k-1}$, and so including z does not change the arithmetic mean. Also,

$$\begin{aligned}\mathrm{AM}(x_1, \ldots, x_{k-1}) = \mathrm{AM}(x_1, \ldots, x_{k-1}, z) &\geq \left(z\prod_{i=1}^{k-1} x_i\right)^{1/k} = \left(\prod_{i=1}^{k-1} x_i\right)^{1/k} z^{1/k} \\ &= \mathrm{GM}(x_1, \ldots, x_{k-1})^{(k-1)/k}\left(\mathrm{AM}(x_1, \ldots, x_{k-1})\right)^{1/k}, \\ \left(\mathrm{AM}(x_1, \ldots, x_{k-1})\right)^{(k-1)/k} &\geq \left(\mathrm{GM}(x_1, \ldots, x_{k-1})\right)^{(k-1)/k}.\end{aligned}$$

We can see the last inequality implies $\mathrm{AM}(x_1, \ldots, x_{k-1}) \geq \mathrm{GM}(x_1, \ldots, x_{k-1})$. We have established (1.2), so we have established the claim for all $n \in \mathbb{N}$ by Cauchy induction. □

Extra Note 2

Top Trading Cycles (TTC) Algorithm

To go along with our discussion of stable marriage, we introduce the **Top Trading Cycles (TTC)** algorithm which solves another matching problem.

2.1 Problem Statement

Consider the following problem: we have n people and n objects, where each person starts off with one of the objects. Each individual has a preference ranking over the n objects. In a typical formulation of the problem, the objects are houses and we seek to reallocate the houses among the people in the best possible manner.

Of course, no one would participate in the reallocation if we did not guarantee that, for each person, the object that he/she receives is at least as good as the object that he/she initially owned. Our goal is to find an allocation which "cannot be improved", that is, no group of people would all be left better off if they decided to redistribute their objects amongst themselves. It is an ambitious goal, but we will see a clever approach for solving the problem.

2.2 Top Trading Cycles Algorithm

The algorithm proceeds as follows: on each day, each person will point at the person who currently owns the object that he/she would most like to possess (the highest object on his/her preference list). Then, we look for **cycles**. A cycle is a string of people, each person pointing to the next, such that the last person points back at the first person. We will discuss cycles more formally in the context of graphs, but the idea is intuitively clear: if we find a cycle amongst the people, then we can make *everyone* in the cycle better off by redistributing their objects: each person gives his/her object to whoever is pointing at him/her. Since each person was pointing at the person with the object he/she most prefers, this redistribution ensures that each person in the cycle obtains his/her favorite item!

How do we know that such a cycle always exists? Easy: just pick any person and follow his/her finger to the next person. Then, follow *this* person's finger to yet another person, and we keep going, following fingers until we reach a person we have already seen before.

Once we reach a person that we have seen before, we have identified a cycle! We only need to ensure that the procedure described above eventually terminates, that is, we need to ensure that we are not following fingers until the end of time. However, since there are only n people, it is clear that we can only see at most n people before we see a person we have already seen before. (The fact that we have a finite number of people is essential for this argument, but that seems like a fairly reasonable assumption.)

Note that it is possible for a person to point to himself/herself; thankfully this is not a problem for the algorithm. We simply consider that person to be a cycle of one person.

Now we have an algorithm: instruct each person to point to his/her most preferred object, find all of the resulting cycles, and redistribute the objects within each cycle. Each person who was part of a cycle is now as happy as he/she could be, so we remove everyone in a cycle from the game. Now, we recurse! By the argument given above, we are guaranteed to find a cycle at each step of the algorithm, and whenever we find a cycle, we remove at least one person from the game. Hence, the game ends in at most n days.

2.3 Optimality & Correctness

Most of the work in showing that the algorithm is optimal was described above, but we should check a few more facts.

First, the algorithm never assigns any person an object which he/she finds *worse* than the object he/she initially possessed. This is true, for the following reason: once every object that the individual prefers over his/her original object is taken by someone else, then the individual will point to himself/herself, producing a self-cycle. This will remove the individual from the game, and the the individual will be left with his/her original object. Therefore, it is impossible for anyone to do worse than his/her original object.

Second, is it possible for two cycles to overlap? The answer is no, and the reason is that each person is only ever pointing to one other person.

Third, have we fully proved that no group of individuals can, at the end of the algorithm, redistribute their objects and improve their welfare? Here, the answer is *no*, perhaps we would need to use induction on the steps of the algorithm to be fully confident of the optimality. However, I'll leave this for you to verify.

Extra Note 3

Chinese Remainder Theorem

3.1 Introduction

We have seen that prime numbers play a central role in the study of modular arithmetic. In particular, the finite field with p elements (denoted $\mathrm{GF}(p)$), where p is a prime, enjoys several nice properties which we recall here:

1. Every non-zero element has a multiplicative inverse.
2. As a consequence of 1, for given $a, b \in \mathrm{GF}(p)$, where a is non-zero, the equation $ax \equiv b \pmod{p}$ for non-zero a has a unique solution given by $x \equiv a^{-1}b \pmod{p}$.
3. Fermat's Little Theorem holds: for non-zero $a \in \mathrm{GF}(p)$, $a^{p-1} \equiv 1 \pmod{p}$. In particular, Fermat's Little Theorem is the basis for the correctness of the RSA public-key cryptosystem.

The properties above do not hold modulo m, when the positive integer m is not prime. [1] Therefore, it is of interest to develop a systematic method for reducing problems modulo m to problems modulo p, where p is *prime*.

The **Chinese Remainder Theorem (CRT)** is the systematic method we seek. Specifically, the CRT gives an explicit procedure for constructing the unique solution to the system of linear congruences

$$\begin{aligned} x_1 &\equiv b_1 \pmod{m_1} \\ &\vdots \\ x_n &\equiv b_n \pmod{m_n} \end{aligned}$$

where n is any positive integer, $b_1, \ldots, b_n$ are given integers, and the moduli $m_1, \ldots, m_n$ are positive integers which are assumed to be *pairwise coprime*, that is, no two of the moduli have any common factors. Here, the solution is unique modulo $m_1 \cdots m_n$.

[1] Fermat's Little Theorem, when suitably generalized, *does* have an analogue modulo m.

Let us first give an example to illustrate our claim that the CRT allows us to reduce problems modulo m to problems modulo p for a prime p.

Example 3.1. Let p and q be primes, and let $m := pq$. Consider the equation $x^2 \equiv -1 \pmod{m}$, that is, we are interested in finding the square roots of -1 modulo m. Note that if $x^2 \equiv -1 \pmod{m}$, then $m \mid x^2 + 1$, and so $p \mid x^2 + 1$. Likewise, $q \mid x^2 + 1$. Thus, $x^2 + 1$ must satisfy the following system of linear congruences:

$$x^2 + 1 \equiv 0 \pmod{p} \tag{3.1}$$
$$x^2 + 1 \equiv 0 \pmod{q} \tag{3.2}$$

The CRT tells us that the solutions to $x^2 \equiv -1 \pmod{m}$ are precisely the x which solve the system of linear congruences

$$x \equiv x_1 \pmod{p}$$
$$x \equiv x_2 \pmod{q}$$

where x_1 is a solution to (3.1) and x_2 is a solution to (3.2). Thus we see that a question posed modulo m is equivalent to first solving the equations (3.1) and (3.2) (which is done modulo primes), and then using the CRT to combine the solutions of (3.1) and (3.2) into solutions to the original question modulo m.

3.2 Solving Modular Equations

Consider the following system of linear congruences:

$$x \equiv 3 \pmod{5}$$
$$x \equiv 7 \pmod{9}$$

We claim that the unique solution modulo 45 is 43. You can verify that 43 satisfies the two equations above, and uniqueness can be checked (rather cumbersomely) by listing all of the numbers from 0 to 43 and checking which ones satisfy the above equations.

Here is a procedure for arriving at the answer 43. The first equation implies

$$x \in \{3, 8, 13, 18, 23, 28, 33, 38, 43\} \pmod{45},$$

and out of the possibilities listed above, the value 43 also satisfies the second equation. This direct approach tends to work well for smaller systems of equations and is often *faster* than applying the CRT construction. The advantage of the CRT is that

- the CRT generalizes easily to larger systems of equations, and
- it guarantees the existence and uniqueness of a solution.

Moreover, we will later see that there is a more insightful perspective of the CRT rather than simply a computational tool for solving equations. For now, let us prove the CRT.

Theorem 3.2 (Chinese Remainder Theorem)**.** *Let n be a positive integer and $a_1, \ldots, a_n$ be given integers. For positive integers $m_1, \ldots, m_n$ which are pairwise coprime, the system of linear congruences*

$$\begin{aligned} x &\equiv a_1 \pmod{m_1}, \\ &\vdots \\ x &\equiv a_n \pmod{m_n}, \end{aligned}$$

has a unique solution modulo $M := m_1 \cdots m_n$, and moreover, the solution can be explicitly computed.

Proof. Existence. The idea is to define y_i, for each $i = 1, \ldots, n$, so that:

$$y_i \bmod m_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

We achieve this with

$$y_i := \frac{M}{m_i} \cdot \left(\left(\frac{M}{m_i} \right)^{-1} \bmod m_i \right).$$

For any $j = 1, \ldots, n$, $j \neq i$, y_i is a multiple of m_j, so $y_i \bmod m_j = 0$. On the other hand, by the definition of the multiplicative inverse, $y_i \bmod m_i = 1$ (the existence of the multiplicative inverse here requires the pairwise coprime assumption).

Our solution is $x = a_1 y_1 + \cdots + a_n y_n$. By construction, for each $i = 1, \ldots, n$, $y_i \equiv 1 \pmod{m_i}$, and for $j \neq i$, $y_j \equiv 0 \pmod{m_i}$, and thus $x \equiv a_i \pmod{m_i}$. Hence, x is a solution to the system of linear congruences.

Uniqueness. Suppose x and y are two solutions to the system of linear congruences. Since $x \equiv y \equiv a_i \pmod{m_i}$, we see that $x - y \equiv 0 \pmod{m_i}$, so $x - y$ is divisible by m_i, for each $i = 1, \ldots, n$. The moduli are pairwise coprime, so $x - y$ is divisible by the product $m_1 \cdots m_n$, which means that $x \equiv y \pmod{m_1 \cdots m_n}$. In other words, the solution is unique up to a multiple of $m_1 \cdots m_n$. □

We construct the CRT solution to the system of linear congruences given above. We have

$$\begin{aligned} y_1 &= 9 \cdot (9^{-1} \bmod 5) = 9 \cdot 4 = 36, \\ y_2 &= 5 \cdot (5^{-1} \bmod 9) = 5 \cdot 2 = 10, \end{aligned}$$

and thus

$$x = 3 \cdot 36 + 7 \cdot 10 = 178 \equiv 43 \pmod{45}.$$

Remark: Example 3.1 is slightly misleading. In order to fully reduce a question modulo m into a question over the finite field $\mathrm{GF}(p)$, one needs more tools than the CRT alone.

In general, one prime factorizes $m = p_1^{\alpha_1} \cdots p_n^{\alpha_n}$ for primes $p_1, \ldots, p_n$ and positive integer powers $\alpha_1, \ldots, \alpha_n$. Then, solving an equation modulo m is equivalent to first solving the equation modulo a prime power p^α. The technique of transferring solutions modulo p to solutions modulo p^α is known as "lifting" and will not be discussed further here.

3.3 Isomorphism

We now give a deeper understanding of the CRT. Consider the system of equations

$$x \equiv a \pmod 5 \tag{3.3}$$
$$x \equiv b \pmod 9 \tag{3.4}$$

where now we think of varying (a, b) in $\{0, \ldots, 4\} \times \{0, \ldots, 8\}$. Earlier, we saw that when $(a, b) = (3, 7)$, the solution modulo 45 is uniquely given by $x = 43$. Given any other choice of (a, b), we can again apply the CRT to find the unique solution x modulo 45. In fact, we can define a function

$$f : \{0, \ldots, 4\} \times \{0, \ldots, 8\} \to \{0, \ldots, 44\}$$
$$\text{given by } f(a, b) = x, \text{ where } x \text{ is the unique solution to (3.3) and (3.4).}$$

The CRT tells us that f is one-to-one and onto, that is, f is a bijection. We can explicitly write down the inverse function:

$$f^{-1} : \{0, \ldots, 44\} \to \{0, \ldots, 4\} \times \{0, \ldots, 8\}$$
$$\text{given by } f^{-1}(x) = (x \bmod 5, x \bmod 9).$$

In fact, f is an *isomorphism*, a term from algebra which means that the two structures $\{0, \ldots, 4\} \times \{0, \ldots, 8\}$ and $\{0, \ldots, 44\}$ are *essentially the same* with respect to addition and multiplication. We already have an understanding of how to add and multiply elements in $\{0, \ldots, 44\}$, namely, we add and multiply them modulo 45. To pursue this idea further, we must define the operations of addition and multiplication on the set $\{0, \ldots, 4\} \times \{0, \ldots, 8\}$.

Let (a_1, b_1), (a_2, b_2) be two elements of $\{0, \ldots, 4\} \times \{0, \ldots, 8\}$. We define addition and multiplication componentwise:

$$(a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2),$$
$$(a_1, b_1)(a_2, b_2) = (a_1 a_2, b_1 b_2).$$

The special property of the function f defined above is that it preserves the operations of addition and multiplication, that is:

$$f\big((a_1, b_1) + (a_2, b_2)\big) = f(a_1, b_1) + f(a_2, b_2), \tag{3.5}$$
$$f\big((a_1, b_1)(a_2, b_2)\big) = f(a_1, b_1) f(a_2, b_2). \tag{3.6}$$

Notice that on the left, the expression $(a_1, b_1) + (a_2, b_2)$ is the operation of addition in the set $\{0, \ldots, 4\} \times \{0, \ldots, 8\}$, while on the right, the expression $f(a_1, b_1) + f(a_2, b_2)$ is the operation

of addition in the set $\{0, \ldots, 44\}$. This is the precise sense in which we say that f preseves the operations of addition and multiplication between the two sets.

To see that f is an isomorphism, it is enough to understand why f^{-1} is an isomorphism. However, the statement that f^{-1} is an isomorphism is nothing more than the familiar facts

$$(x_1 + x_2) \bmod m = (x_1 \bmod m) + (x_2 \bmod m)$$
$$(x_1 x_2) \bmod m = (x_1 \bmod m)(x_2 \bmod m)$$

for any positive integer modulus m. (Recall that the two facts above are pivotal for carrying out fast computations in modular arithmetic.)

We will now build a table of the values of f.

	0	1	2	3	4	5	6	7	8
0	0	10	20	30	40	5	15	25	35
1	36	**1**	**11**	21	**31**	**41**	6	**16**	**26**
2	27	**37**	**2**	12	**22**	**32**	42	**7**	**17**
3	18	**28**	**38**	3	**13**	**23**	33	**43**	**8**
4	9	**19**	**29**	39	**4**	**14**	24	**34**	**44**

Additional insights can be gleaned from the table.

- The table can be constructed by writing 0 in the upper left corner, and then writing successive numbers diagonally to the right, wrapping around the rows and columns on the table when necessary.
- The bolded numbers are the numbers which are relatively prime to 45. Note that the bolded numbers appear precisely in the rows which are relatively prime to 5 and the columns which are relatively prime to 9. This is the statement that if $f(a, b) = x$, then x has an inverse modulo 45 if and only if a has an inverse modulo 5 and b has an inverse modulo 9.

In fact, the structures $\{0, \ldots, 4\} \times \{0, \ldots, 8\}$ (with the operations of addition and multiplication defined above), as well as the set $\{0, \ldots, 45\}$ (with addition and multiplication taken modulo 45) are examples of structures called *rings*. A simple way to describe a ring is a structure on which you can perform addition and multiplication.

In this language, f is known as a *ring isomorphism*. The existence of a ring isomorphism means that any fact about the structure $\{0, \ldots, 4\} \times \{0, \ldots, 8\}$ which is phrased solely in terms of addition and multiplication (e.g., the fact that 2 has an inverse modulo 5 and 4 has an inverse modulo 9) can be transferred over to a corresponding fact in $\{0, \ldots, 44\}$ (e.g., 22 has an inverse modulo 45). The algebra perspective therefore provides a unified way of understanding how structures behave. If you are interested in learning more, consider taking Mathematics 113.

Extra Note 4

Hard Problems & Public Key Cryptosystems

4.1 Introduction

This week's extra note is a special edition, written by **Siqi Liu**!

4.2 Discrete Logarithms

In class, we learned that the security of RSA relies on the assumption that factoring is hard. More specifically, given the public key N and e, it's hard to find N's prime factors p and q. On the other hand, given p and q we can compute N efficiently. The efficency of multiplication and the hardness of the inverse process of factoring ensure that the private key is hidden, even after the public key is revealed.

Besides multiplication-factoring, some other numerical operations are also considered to be easy to compute but hard to invert. A noticeable example is the modular exponentiation-discrete logarithm problem. Let's now formally define the two operations.

Definition 4.1. Let p be a prime number, and g be an element in the set $\{1, 2, \ldots, p-1\}$. **Modular exponentiation** is defined as following:

$$\mathrm{EXP}_{g,p}(x) := g^x \bmod p.$$

The inverse function is

$$\mathrm{DL}_{g,p}(y) := x, \qquad \text{where } g^x \equiv y \pmod{p} \quad \text{and} \quad x \in \{1, 2, \ldots, p-1\}.$$

This inverse function is called the **discrete logarithm** problem.

Similar to factoring, the discrete logarithm problem is believed to be very hard to solve.

The difficulty of solving this problem yields the **ElGamal public key cryptosystem**. To illustrate how this system works, let's go back to the story of Alice and Bob. How can Alice send a message m to Bob through the ElGamal cryptosystem?

1. First, Bob generates a large prime p and randomly chooses two arbitrary numbers $g, x \in \{1, 2, \ldots, p-1\}$.

2. Then, Bob calculates $y \equiv g^x \pmod{p}$. He uses x as his private key and reveals (p, g, y) to the public. Notice that because it's difficult to solve discrete logarithm problem, x is hidden from the public, even when they know p, g, and y.

3. Next, Alice wants to send the message m to Bob. She first picks an arbitrary key $k \in \{1, 2, \ldots, p-1\}$, and generates $r \equiv g^k \pmod{p}$. She encrypts m by calculating $c \equiv y^k m \pmod{p}$. Finally, she sends (r, c) to Bob.

4. Bob decrypts the message m by calculating

$$cr^{-x} \equiv y^k m (g^k)^{-x} \equiv (g^x)^k m g^{-kx} \equiv m \pmod{p}.$$

The advantage of public key cryptosystems is that they don't require Alice and Bob to exchange a shared private key in advance through some secure channel. In many settings, the transmission of a secret key is vulnerable to a third party's attack. However, public key cryptosystems cleverly avoid this problem.

Extra Note 5

Eisenstein's Criterion

5.1 Introduction

We will illustrate the connection between polynomials and modular arithmetic by proving **Eisenstein's criterion**, which gives sufficient conditions for a polynomial to be irreducible. In order to prove Eisenstein's criterion, we will first arrive at **Gauss's lemma**, which essentially states that irreducibility over the integers is the same as irreducibility over the rationals. First, a few definitions:

Definition 5.1. The following are **polynomial rings**:

- $\mathbb{Z}[x]$ is the set of polynomials with integer coefficients.
- $\mathbb{Q}[x]$ is the set of polynomials with rational coefficients.
- $\mathbb{F}_p[x]$ is the set of polynomials with coefficients that are integers modulo p.

Definition 5.2. A polynomial $p(x)$ is **reducible** in $\mathbb{Z}[x]$ (or in $\mathbb{Q}[x]$, or in $\mathbb{F}_p[x]$) if there exist non-constant polynomials $a(x)$, $b(x)$ in $\mathbb{Z}[x]$ (or in $\mathbb{Q}[x]$, or in $\mathbb{F}_p[x]$ respectively) such that $p(x) = a(x)b(x)$. A polynomial is **irreducible** if it is not reducible.

5.2 Gauss's Lemma

Suppose that we can factor a polynomial $p(x) \in \mathbb{Z}[x]$ over the rationals. Under what circumstances can we also factor the polynomial over the integers? As an example, take $p(x) = 2x^2 + 3x + 1$. We can factor $p(x)$ over the rationals: $p(x) = (x + 1/2)(2x + 2)$. In this case, we can also factor $p(x)$ over the integers as $p(x) = (2x+1)(x+1)$ by moving a constant factor of 2 from one factor to the other. Gauss's Lemma is essentially the statement that this rearrangement is always possible.

Lemma 5.3 (Gauss's Lemma)**.** *If $p(x) \in \mathbb{Z}[x]$ is reducible in $\mathbb{Q}[x]$, then it is also reducible in $\mathbb{Z}[x]$.*

Proof. Suppose that $p(x) = a(x)b(x)$, where $a(x), b(x) \in \mathbb{Q}[x]$. Then, the coefficients of $a(x)$ and $b(x)$ are fractions, so let us multiply through the equation by the common denominator d:

$$dp(x) = a'(x)b'(x), \tag{5.1}$$

where $a'(x), b'(x) \in \mathbb{Z}[x]$. If $d = 1$, then we're done; otherwise, suppose that the prime factorization of d is $p_1 \cdots p_n$. Take the equation modulo p_1:

$$0 \equiv a'(x)b'(x) \pmod{p_1} \tag{5.2}$$

(5.2) can only be satisfied if $a'(x) \equiv 0 \pmod{p_1}$ or $b'(x) \equiv 0 \pmod{p_1}$. (This is true only because p_1 is prime. Can you see why?) Assume that $a'(x) \equiv 0 \pmod{p_1}$: then every coefficient of $a'(x)$ is divisible by p_1. Divide both sides of (5.1) by p_1 to obtain

$$d'p(x) = a''(x)b'(x),$$

where $d' = d/p_1 \in \mathbb{Z}$ and $a''(x) = a(x)/p_1 \in \mathbb{Z}[x]$. We can continue this procedure inductively, dividing by each of the prime factors of d, until $d = 1$. This proves our claim. □

5.3 Eisenstein's Criterion

We are ready to prove Eisenstein's Criterion.

Proposition 5.4 (Eisenstein's Criterion)**.** *Let p be prime and let*

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0 \in \mathbb{Z}[x].$$

Suppose that p divides $a_0, \ldots, a_{n-1}$, but p^2 does not divide a_0. Then $f(x)$ is irreducible in $\mathbb{Z}[x]$ (and by 5.3, in $\mathbb{Q}[x]$ as well).

Proof. Suppose that $f(x) = a(x)b(x)$. Take the equation modulo p:

$$x^n \equiv a(x)b(x) \pmod{p}$$

(since $a_0, \ldots, a_{n-1}$ are divisible by p). The constant term of the polynomial on the LHS is $0 \pmod{p}$, so the constant term of the RHS must also be $0 \pmod{p}$. Since p is prime, then the constant terms of both $a(x)$ and $b(x)$ are $0 \pmod{p}$. However, if the constant terms of $a(x)$ and $b(x)$ are both divisible by p, then this contradicts the fact that a_0 is not divisible by p^2. □

Let's take a look at a few applications of Eisenstein's Criterion.

Example 5.5. If a is any integer which is divisible by a prime p, but not by p^2, then $x^n - a$ is irreducible in $\mathbb{Q}[x]$. In particular, $x^n - p$ is irreducible, i.e. $x^n - p$ has no roots in $\mathbb{Q}$, i.e. the nth roots of p are irrational.

Example 5.6. Let p be prime and consider the **cyclotomic polynomial**

$$\Phi_p(x) = \frac{x^p - 1}{x - 1} = x^{p-1} + x^{p-2} + \cdots + x + 1.$$

We cannot apply Eisenstein's Criterion to $\Phi_p(x)$ directly. However, we can apply Eisenstein's Criterion to $\Phi_p(x + 1)$:

$$\Phi_p(x+1) = \frac{(x+1)^p - 1}{x} = x^{p-1} + px^{p-2} + \cdots + \frac{p(p-1)}{2}x + p,$$

where we have used the Binomial Theorem to expand $(x+1)^p$. All of the coefficients are divisible by p, and the constant term is not divisible by p^2, so $\Phi_p(x+1)$ is irreducible over $\mathbb{Q}[x]$. This also shows that our original polynomial, $\Phi_p(x)$, is irreducible over $\mathbb{Q}[x]$ (if $\Phi_p(x)$ could be factored over $\mathbb{Q}[x]$, we could replace x by $x + 1$ to obtain a factorization for $\Phi_p(x + 1)$ in $\mathbb{Q}[x]$, which is impossible).

5.4 Further Reading

All of the results proven here can be extended to a considerably more general setting in the field of study known as ring theory. The primary reference for this extra note was *Abstract Algebra* by Dummit and Foote.

Extra Note 6

Algebraic Coding Theory

Previously, we have discussed parity bits in modular arithmetic and Reed-Solomon codes using polynomial error correction. Here, we will introduce basic ideas from algebraic coding theory in order to delve a little deeper into error correction schemes.

6.1 Codewords

First, we should introduce the basic setup. Throughout the note, we will solely work within $\mathbb{Z}_2 = \{0, 1\}$, that is, the integers modulo 2 (the idea is that we will be concerned only with binary strings). Suppose we start with a length-n binary string $\mathbf{x} \in \mathbb{Z}_2^n$. We will make use of an **encoding function**

$$E : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$$

and a **decoding function**

$$D : \mathbb{Z}_2^m \to \mathbb{Z}_2^n.$$

We "encode" our message into **codewords**: $\mathbf{y}$ is a codeword if it belongs to the range of E. Let $\mathcal{C}$ denote the set of codewords.

What properties do we require of our encoding function and our decoding function? First, we want distinct messages to be encoded in distinct codewords, or else we would lose our original message. Therefore, we require that E is one-to-one. Similarly, we want to require $D(E(\mathbf{x})) = \mathbf{x}$, which says that we can decode any message that we encoded. This automatically tells us that D is onto. Typically, however, we will want more: we will want $D(\mathbf{y}) = \mathbf{x}$ for all $\mathbf{y}$ which are "close" enough to $E(\mathbf{x})$; this is what allows us to recover from errors. We will make this more precise shortly.

You may assume that $m > n$ (we are allowing ourselves to use additional bits in order to recover from errors).

6.2 Hamming Distance

The **Hamming distance** of binary strings $\mathbf{x}$ and $\mathbf{y}$ is the number of bits by which $\mathbf{x}$ and $\mathbf{y}$ differ. We will denote the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ by $d(\mathbf{x}, \mathbf{y})$.

First, we verify that the Hamming distance satisfies the necessary properties for any notion of "distance", that is:

Proposition 6.1. *For all $\mathbf{x}, \mathbf{y}$, the following hold:*

1. *Non-Negativity: $d(\mathbf{x}, \mathbf{y}) \geq 0$, with $d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$.*
2. *Symmetry: $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$.*
3. *Triangle Inequality: $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.*

Proof.

1. The Hamming distance is non-negative and $d(\mathbf{x}, \mathbf{x}) = 0$ by definition. If $d(\mathbf{x}, \mathbf{y}) = 0$, then $\mathbf{x}$ and $\mathbf{y}$ do not differ in any bits, so they must be the same.
2. Symmetry is also clear from the definition.
3. It takes $d(\mathbf{x}, \mathbf{y})$ bit flips to change $\mathbf{x}$ to $\mathbf{y}$, and it takes $d(\mathbf{y}, \mathbf{z})$ bit flips to change $\mathbf{y}$ to $\mathbf{z}$, so flipping $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$ bits is enough to change $\mathbf{x}$ to $\mathbf{z}$. Since the $d(\mathbf{x}, \mathbf{z})$ is the least number of bits to change $\mathbf{x}$ to $\mathbf{z}$, we have $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

□

The reason why we care about the Hamming distance is because it quantifies the quality of our encoding function:

Proposition 6.2. *Suppose that $\min_{\mathbf{x},\mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}} d(\mathbf{x}, \mathbf{y}) = 2k + 1$. Then, we can detect up to $2k$ errors and recover from up to k errors.*

Proof. To prove the first claim, we start with a codeword $\mathbf{z}$. Since $2k+1$ is the minimum distance between two codewords, if we introduce at most $2k$ errors, then the resulting string $\mathbf{z}'$ is not a codeword, so we can detect the error.

To prove the second claim, suppose that our codeword $\mathbf{z}$ is corrupted to the string $\mathbf{z}'$ with at most k errors. Then, $d(\mathbf{z}, \mathbf{z}') \leq k$. For any other codeword $\mathbf{x}$, one has

$$2k + 1 \leq d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{z}') + d(\mathbf{z}, \mathbf{z}').$$

Since $d(\mathbf{z}, \mathbf{z}') \leq k$, one has $d(\mathbf{x}, \mathbf{z}') \geq k + 1$, which says that $\mathbf{z}'$ is closer to $\mathbf{z}$ than to any other codeword $\mathbf{x}$; hence, we can recover the original codeword $\mathbf{z}$ by looking for the codeword which is closest to $\mathbf{z}'$. □

6.3 Group Codes

The Hamming distance provides some guarantees regarding error recovery, but it requires computing all of the pairwise distances between the codewords. Furthermore, how does one construct a good encoding function? We can simplify the discussion if the codewords form a **group**, which in this case means that if $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, then $\mathbf{x} + \mathbf{y} \in \mathcal{C}$.

Define the **weight** of a codeword to be $w(\mathbf{x}) = d(\mathbf{x}, \mathbf{0})$, the number of non-zero characters in $\mathbf{x}$. Then, we observe that $\mathbf{x} + \mathbf{y}$ is 0 exactly in the positions where $\mathbf{x}$ and $\mathbf{y}$ are the same, and 1 otherwise; hence, $w(\mathbf{x} + \mathbf{y})$ is the number of positions by which $\mathbf{x}$ and $\mathbf{y}$ differ, or in other words, $w(\mathbf{x} + \mathbf{y}) = d(\mathbf{x}, \mathbf{y})$.

Proposition 6.3. *For a group code,*

$$\min_{\substack{\mathbf{x},\mathbf{y}\in\mathcal{C} \\ \mathbf{x}\neq\mathbf{y}}} d(\mathbf{x}, \mathbf{y}) = \min_{\substack{\mathbf{x}\in\mathcal{C} \\ \mathbf{x}\neq\mathbf{0}}} w(\mathbf{x}).$$

Proof. We note that if $\mathbf{x} \neq \mathbf{y}$, then $\mathbf{x} + \mathbf{y} \neq \mathbf{0}$; and since $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} + \mathbf{y})$, the two sides of the equation above are the same. Note that the proof works because $\mathbf{x} + \mathbf{y} \in \mathcal{C}$, due to our assumption that $\mathcal{C}$ is a group code. □

Although we have a simple way of computing the minimum distance between codewords, now we have an additional restrictive assumption that for $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, $\mathbf{x} + \mathbf{y} \in \mathcal{C}$. One way of satisfying this condition is to consider the **null space** $\mathcal{N}(\mathbf{A})$ of a $m \times n$ matrix $\mathbf{A}$, that is,

$$\mathbf{x} \in \mathcal{N}(\mathbf{A}) \iff \mathbf{A}\mathbf{x} = \mathbf{0}.$$

We can quickly verify that $\mathcal{C} = \mathcal{N}(\mathbf{A})$ is a group code: for any $\mathbf{x}, \mathbf{y} \in \mathcal{N}(\mathbf{A})$, we have

$$\mathbf{A}(\mathbf{x} + \mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y} = \mathbf{0} + \mathbf{0} = \mathbf{0},$$

so $\mathbf{x} + \mathbf{y} \in \mathcal{C}$. We call these codes **linear codes** since they make use of a linear map (a matrix).

6.4 Further Reading

Of course, there is much more we could have said about algebraic coding theory: how do we construct the pair of encoding and decoding functions, what conditions are necessary and sufficient to ensure that we can recover from a fixed number of errors, how can we decode efficiently? The primary reference for this note was *Abstract Algebra: Theory and Applications* by Jusdon and Beezer.

Extra Note 7

Cantor-Schröder-Bernstein Theorem

In order to prove that a set is countable, we try to exhibit a bijection with the natural numbers. How about if we want to show that a set is uncountably infinite? One way to accomplish this is to exhibit a bijection with the real numbers, but this is often quite difficult. For example, can you construct a bijection between the intervals $(0, 1]$ and $[0, 1]$? It's not so easy to prove directly, but the **Cantor-Schröder-Bernstein theorem** is an important result that says: given two injections, one in each direction, then a bijection exists. In order words, we need only find injections between the two sets, and then we're done! The proof is conceptually challenging, but requires no knowledge outside of the course.

The proof is rather long, but that is only because I have tried to describe the proof very carefully. Once you understand the idea, the proof is rather straightforward and the details are clear. Now, if I haven't scared you off, let's begin.

7.1 Constructing a Bijection

Theorem 7.1 (Cantor-Schröder-Bernstein Theorem)**.** *If $f : X \to Y$ and $g : Y \to X$ are injections (one-to-one), then there exists a bijection $\varphi : X \to Y$.*

Proof. The proof is constructive, but by the end, you may end up wondering whether we have really constructed anything at all! We start with a fixed point $x \in X$, and we attempt to apply the function g^{-1}, i.e. we try to compute $g^{-1}(x)$. This is only possible if x lies in the range of g, but let us accept this for now. The point $g^{-1}(x)$ lies in the set Y, and now we can try to compute $f^{-1}(g^{-1}(x))$. Again, this may or may not be possible, depending on whether $g^{-1}(x)$ lies in the range of f, but we try anyway. If we succeed, then we can try to compute $g^{-1}(f^{-1}(g^{-1}(x)))$ and so forth. Now, for every $x \in X$, we apply this procedure, and we see that there are three possible cases:

1. This procedure terminates at some point, because we are unable to apply g^{-1}.
2. This procedure terminates at some point, because we are unable to apply f^{-1}.

3. This procedure never terminates.

Since these cases are exhaustive, we see that we can assign every $x \in X$ to one of three sets, depending on which of the above conditions it satisfies. Define:

1. Define X_X to be the set of $x \in X$ which fall into the first case above.
2. Define X_Y to be the set of $x \in X$ which fall into the second case above.
3. Define X_∞ to be the set of $x \in X$ which fall into the third case above.

We can see that X is the disjoint union of the three sets above, i.e. $X = X_X \cup X_Y \cup X_\infty$. Now, we apply the same trick to partition Y as well. Explicitly, fix some $y \in Y$ and attempt to apply f^{-1} to x; if we succeed then we try to apply g^{-1} to $f^{-1}(y)$, and if we succeed we attempt to apply f^{-1} to $g^{-1}(f^{-1}(y))$, etc. Again, we split Y into three cases:

1. Define Y_X to be the set of $y \in Y$ for which the procedure terminates, because we cannot apply g^{-1} anymore.
2. Define Y_Y to be the set of $y \in Y$ for which the procedure terminates, because we cannot apply f^{-1} anymore.
3. Define Y_∞ to be the set of $y \in Y$ for which the procedure never terminates.

Now, we can write Y as a disjoint union as well: $Y = Y_X \cup Y_Y \cup Y_\infty$. Now, our plan is to construct the bijection φ in the following way:

$$\varphi(x) = \begin{cases} f(x), & x \in X_X \\ g^{-1}(x), & x \in X_Y \\ f(x), & x \in X_\infty \end{cases}$$

The bijection above is actually composed of three different bijections:

$$X_X \xrightarrow{f} Y_X$$
$$X_Y \xrightarrow{g^{-1}} Y_Y$$
$$X_\infty \xrightarrow{f} Y_\infty$$

Our task now is to verify the above bijections. First, a bit of terminology to simplify the explanation: if we apply the procedure for partitioning X to a point $x \in X$, we say that we are applying the X-procedure to x. (Here, "applying the procedure" simply means that we apply g^{-1} and f^{-1} alternately until the procedure terminates, or does not terminate at all.) Similarly, if we apply the procedure for partitioning Y to a point $y \in Y$, we say that we are applying the Y-proceudre to y.

For the first bijection, $f : X_X \to Y_X$, the first thing that we have to check is that f does indeed map elements of X_X to Y_X as advertised. If $x \in X_X$, then that means applying the X-procedure to x fails at some step because we cannot apply g^{-1}. Now, x is mapped by f to the element $f(x)$, and consider what happens when we apply the Y-procedure to $f(x)$: first we apply $f^{-1}(f(x))$, which is clearly valid since $f(x)$ lies in the range of f. Next, we apply g^{-1} and f^{-1} alternately, and we see that the procedure for doing so *exactly mirrors* the steps we take when we apply the X-procedure to x. In other words, the X-procedure applied to x terminates because we cannot apply g^{-1}, *if and only if* the Y-procedure applied to $f(y)$ terminates because we cannot apply g^{-1}. This proves that $f(x) \in Y_X$.

Next, we must show that f is a bijection between these sets. Consider any point $y \in Y_X$: when we apply the Y-procedure to y, the first thing we do is attempt to apply f^{-1}. However, since $y \in Y_X$, the Y-procedure cannot stop here (we must terminate when we cannot apply g^{-1} anymore). This says that if $y \in Y_X$, then we can successfully apply f^{-1} to y, which is what we wanted to show. (Think about it: we already know that f is injective. Saying that we can apply f^{-1} to any $y \in Y_X$ is saying that every $y \in Y_X$ lies in the range of f, so f is also surjective, so f is a bijection.)

We have shown the bijection between X_X and Y_X, and in principle, we must now prove the bijections between X_Y and Y_Y, and between X_∞ and Y_∞. The proofs end up looking almost identical to the one described above, so let's be brief: g^{-1} maps X_Y to Y_Y, because both x and $g^{-1}(x)$ must terminate when we are unable to apply f^{-1}; and this is a bijection because we know that if $x \in X_Y$, then we succeed at applying g^{-1}. f maps X_∞ to Y_∞ because if the procedure never terminates for x, then it clearly never terminates for $f(x)$ either; and f is a bijection because if $y \in Y_\infty$, then because the Y-procedure applied to y never terminates, we must succeed at applying f^{-1}. □

Now, let's see an example of what sort of bijection is constructed by the theorem.

Example 7.2. We will construct a bijection $\varphi : \mathbb{N} \to \mathbb{N}$ (here, $X = Y = \mathbb{N}$) from the two injections $f(x) = 2x$ and $g(x) = 2x$. From the Cantor-Schröder-Bernstein Theorem 7.1, we know that the bijection will consist of $f(x) = 2x$ and $g^{-1}(x) = x/2$; it only remains to determine what the sets X_X, X_Y, and X_∞ are.

1. Suppose that we take $x \in X$, and try to apply g^{-1} to x. (If this fails, then we know that $x \in X_X$.) This will fail exactly when x is odd, or equivalently, when $x \equiv 1 \pmod 2$.

2. Suppose that we successfully apply g^{-1} to x, which means that x is even. What happens when we fail to apply f^{-1}? That means that x is even, but $x/2$ is odd, which is to say that $x \equiv 2 \pmod 4$. In this case, $x \in X_Y$.

3. Assume that the previous two steps succeeded, so that x is divisible by 4. If we cannot apply g^{-1} here, then that means x is even, $x/2$ is even, but $x/4$ is odd,

which is equivalent to $x \equiv 4 \pmod 8$. Here, $x \in X_X$.

4. The pattern continues...

We can write the full bijection φ as follows: if $x \equiv n/2 \pmod n$, where n is some *even* power of 2 (such as 2^2, 2^4, 2^6, etc.), then we map x to $x/2$ (this is g^{-1}); otherwise, we map x to $2x$ (this is f).

Example 7.3. We turn our attention to the question posed at the beginning: can we find a bijection from $(0, 1]$ to $[0, 1]$? We supply two injections. The first injection, $f : (0, 1] \to [0, 1]$, is obvious. For the second injection, we can define $g(x) = x/2 + 1/4$. (Basically, we squash the interval $[0, 1]$ to $[0, 1/2]$, and then add a constant to prevent any input from mapping to 0.) The Cantor-Schröder-Bernstein Theorem 7.1 supplies a bijection φ, so we know that the cardinalities of $(0, 1]$ and $[0, 1]$ are the same (in fact, the same as the real numbers, as you can show with the theorem). However, we choose to skip the explicit construction of this bijection as we did in the example above.

7.2 References

The proof of the Cantor-Schröder-Bernstein Theorem was taken from *Real Analysis: Modern Techniques and Their Applications* by Folland.

Extra Note 8

Generating Functions

8.1 Introduction

This week, we will introduce an especially interesting method of counting: **generating functions**! Let us jump directly into the definition of a generating function:

Definition 8.1. If a_i is a sequence of numbers, then the **generating function** associated with the sequence a_i is the function

$$G(x) = \sum_{i=1}^{\infty} a_i x^i. \tag{8.1}$$

In other words, $G(x)$ is the Taylor series with the sequence a_i as the coefficients. How can we use this mathematical object to help us count?

Example 8.2. 21 TAs are nervously sitting in a circle. The professors have a total of 5 gold stickers, which they want to award to the most dedicated TAs. Assume that no TA is awarded more than one gold sticker. In how many different ways can the professors award the gold stickers?

We can easily solve this question using the techniques of counting that we have already learned, but let's see how we can use generating functions can be used. For each TA, we can give him or her either 0 gold stickers, or 1 gold sticker. We represent this with the polynomial $1 + x$, where $1 = x^0$ represents 0 gold stickers, and x represents 1 gold stickers. We claim that the total number of ways to distribute the gold stickers is the coefficient of x^5 in the polynomial $(1 + x)^{21}$.

Why is this true? We can view the polynomial multiplication of $(1+x)^{21}$ in the following way: we have 21 factors of $(1 + x)$, and every term in the product $(1 + x)^{21}$ is formed by choosing either 1 or x from each factor. For example, to form the term x^{21}, we need to choose x from each of the 21 factors. The coefficient of x^5 in $(1 + x)^{21}$ is the number

of ways to choose 1 or x from each of the terms, such that the sum of the degrees of the chosen terms is 5; this exactly corresponds to the number of ways to choose 0 or 1 gold stickers for each TA, such that the total number of stickers given away is 5.

8.2 Counting Change

Example 8.3. We want to count how many ways we make change for 73 cents from pennies, nickels, dimes, and quarters.

Consider the generating function

$$G(x) = (1 + x + x^2 + \cdots)(1 + x^5 + x^{10} + \cdots)(1 + x^{10} + x^{20} + \cdots)(1 + x^{25} + x^{50} + \cdots).$$

The answer to our counting problem is the coefficient of the x^{73} term in the above generating function. Here is why: the coefficient of x^{73} counts the number of ways to choose one term from each of the four factors above, such that the sum of the degrees is 73, where the four factors above represent the number of pennies, nickels, dimes, and quarters that we use respectively. For example, $x^3 \cdot 1 \cdot x^{20} \cdot x^{50}$ represents choosing 3 pennies, no nickels, 2 dimes, and 2 quarters. We see that there is an exact correspondence between the number of ways to produce the term x^{73}, and the number of ways to make change for 73 cents.

In the example above, we obtain a concise representation for the generating function if we use the identity for the sum of a geometric series, $\sum_{i=0}^{\infty} x^i = 1/(1-x)$ (provided that $|x| < 1$). Hence, we can write

$$G(x) = \frac{1}{1-x} \cdot \frac{1}{1-x^5} \cdot \frac{1}{1-x^{10}} \cdot \frac{1}{1-x^{25}}.$$

Example 8.4. What if, in the example above, the order of the coins we choose matters? For instance, we would consider one penny and one quarter to be distinct from one quarter and one penny. Can we still count the number of ways to make change for 73 cents?

Yes, we can! First we consider: what if we are only allowed to use one coin? Then we can choose a penny, a nickel, a dime, or a quarter, but not more than one; we can represent this choice by $x + x^5 + x^{10} + x^{25}$. The number of ways to make ordered change for 73 cents with one coin is the coefficient of x^{73} in this expression, which is 0 of course (it is not possible with just a single coin).

What about if we were allowed two coins? Then we have two choices, so we can represent this by $(x + x^5 + x^{10} + x^{25})^2$. Similarly, if we are allowed three coins, then we would write this choice as $(x + x^5 + x^{10} + x^{25})^3$, and so forth.

In our original problem, we made no mention of how many coins we are allowed. In fact, we are allowed to use any number of coins that we wish, so in particular, we must sum

over all of the numbers of coins we can use. The generating function is thus:

$$G(x) = \sum_{i=0}^{\infty}(x + x^5 + x^{10} + x^{25})^i = \frac{1}{1 - (x + x^5 + x^{10} + x^{25})}$$

In this generating function, we look for the coefficient of x^{73}, as we did before.

In the examples above, we see that we can often find closed-form representations for generating functions using formulae for infinite series. Closed-form expressions are very useful for carrying out the analysis of a combinatorics problem, so generating functions find much use in practice. We leave you with one final teaser: generating functions can also be used to solve linear recurrence relations, such as $F_n = F_{n-1} + F_{n-2}$. With this technique, we obtain the closed-form of the Fibonacci numbers:

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n \tag{8.2}$$

Extra Note 9

Generating Functions II

This edition of the extra notes is brought to you by **Jonathan Xia**!

9.1 Introduction

Essentially, a generating function is a way of storing a sequence inside a (possibly infinite) polynomial. Suppose that $a_0, a_1, a_2, \ldots$ is a sequence of numbers. The corresponding generating function for this sequence is

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots .$$

Technically, since this is an "infinite polynomial", the proper term is "power series". In real analysis, we would be concerned about whether or not this infinite sum would converge. For example, the power series

$$1 + x + x^2 + x^3 + \cdots$$

would converge if and only if $|x| < 1$. However, when using generating functions, the polynomial is merely a convenient way of storing the information of a sequence, so usually we won't care about what values of x make the sequence actually converge.

Generating functions can provide a different kind of structure to problems.

Example 9.1. A "fair die" is a die that rolls 1, 2, 3, 4, 5, 6, each with probability 1/6. A "fair pair of dice" is a pair of dice that rolls a sum of 2 with probability 1/36, a sum of 3 with probability 2/36, and so on. In other words, a fair pair of dice will roll sums with the same probabilities as a pair of fair dice. Assume that a die must have a positive integer on each face. Is a fair pair of dice necessarily a pair of fair dice?

In fact, we can actually do this! Let's see how we can arrive there. The generating function for just one die is

$$f(x) = \frac{1}{6}x + \frac{1}{6}x^2 + \frac{1}{6}x^3 + \frac{1}{6}x^4 + \frac{1}{6}x^5 + \frac{1}{6}x^6.$$

We are basically taking the probabilities of each roll of the die, and encoding it in a polynomial. Now, since we are looking at two dice, we look at $f(x) \cdot f(x)$. Watch what happens when we multiply it out:

$$\begin{aligned} f(x) \cdot f(x) = &\frac{1}{36}x^2 + \frac{2}{36}x^3 + \frac{3}{36}x^4 + \frac{4}{36}x^5 + \frac{5}{36}x^6 + \frac{6}{36}x^7 \\ &+ \frac{5}{36}x^8 + \frac{4}{36}x^9 + \frac{3}{36}x^{10} + \frac{2}{36}x^{11} + \frac{1}{36}x^{12} \end{aligned}$$

This is just the probabilities of each possible outcome! For example, if we look at the coefficient of x^9, we have $4/36$, which is the probability of rolling a 9. Make sure you understand how this happens (when looking at the coefficient of x^9, we're adding up all the possibilities of getting a sum of 9).

Suppose our two dice had generating functions $g(x)$ and $h(x)$. Then,

$$g(x)h(x) = f(x)^2.$$

is what we want. Well,

$$f(x)^2 = \frac{1}{36}(x + x^2 + x^3 + x^4 + x^5 + x^6)^2.$$

Now, let's try to factor this expression. We have

$$\begin{aligned} \frac{1}{36}(x + x^2 + x^3 + x^4 + x^5 + x^6)^2 &= \frac{1}{36}x^2(1 + x + x^2 + x^3 + x^4 + x^5)^2 \\ &= \frac{1}{36}x^2(1 + x^3)^2(1 + x + x^2)^2 \\ &= \frac{1}{36}x^2(1 + x)^2(1 - x + x^2)^2(1 + x + x^2)^2. \end{aligned}$$

So, we see that we have a bunch of irreducible polynomials. What we can try to do is mix and match them. Let's say we gave $g(x)$ these factors:

$$\begin{aligned} g(x) &= \frac{1}{6}x(1 + x)(1 + x + x^2)(1 - x + x^2)^2 \\ &= \frac{1}{6}(x + x^3 + x^4 + x^5 + x^6 + x^8) \end{aligned}$$

and gave $h(x)$ the others:

$$h(x) = \frac{1}{6}x(1 + x)(1 + x + x^2) = \frac{1}{6}(x + 2x^2 + 2x^3 + x^4)$$

So, if we had a die of $1, 3, 4, 5, 6, 8$ as faces and a die of $1, 2, 2, 3, 3, 4$ as faces, then rolling these two dice will give the exact same probability distribution as two fair dice!

We can prove combinatorial identities with generating functions as well. Recall the **Binomial**

Theorem which states that

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n.$$

Exercise: Why is the Binomial Theorem true in the context of generating functions?

Then, consider $(1+x)^{m+n} = (1+x)^m(1+x)^n$. Consider the coefficient of x^k in both of these generating functions. The coefficient of x^k in $(1+x)^{m+n}$ is $\binom{m+n}{k}$.

Example 9.2. Prove *Vandermonde's Identity*, which states

$$\binom{m+n}{k} = \binom{m}{0}\binom{n}{k} + \binom{m}{1}\binom{n}{k-1} + \binom{m}{2}\binom{n}{k-2} + \cdots + \binom{m}{k}\binom{n}{0}.$$

Let's look at the coefficient of x^k in $(1+x)^m \cdot (1+x)^n$. Since

$$(1+x)^m = \binom{m}{0} + \binom{m}{1}x + \binom{m}{2}x^2 + \cdots + \binom{m}{m}x^m$$

and

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n,$$

we can see that the coefficient of x^k is the right hand side of Vandermonde's identity.

9.2 Distribution Generating Functions

The stars and bars problem that was given in lecture can be related to generating functions too! Let's first go over the problem we know:

Example 9.3. Let's count how many ordered triples of non-negative integers (a, b, c) satisfy the equation

$$a + b + c = 30.$$

The number a can be $0, 1, 2, \ldots$ and so on, so let's say a gets a generating function of

$$1 + x + x^2 + x^3 + \cdots .$$

Also, b would have the same generating function, and so will c. Now, if we look at the coefficient of x^{30} in

$$(1 + x + x^2 + x^3 + \cdots)^3.$$

we will get our answer. Make sure you see how! We can brute force multiply this out:

$$(1 + x + x^2 + \cdots)(1 + x + x^2 + \cdots) = 1 + 2x + 3x^2 + 4x^3 + \cdots$$

and then

$$(1+x+x^2+x^3+\cdots)(1+2x+3x^2+4x^3+\cdots) = (1+3x+6x^2+10x^3+\cdots).$$

In general,

$$(1+x+x^2+x^3+\cdots)^n = \binom{n-1}{n-1} + \binom{n}{n-1}x + \binom{n+1}{n-1}x^2 + \binom{n+2}{n-1}x^3 + \cdots .$$

Exercise: verify that the above formula holds for small values of n. Try to prove this statement with induction!

Now, something cool about generating functions is that we can actually write them in closed forms to make them compact. So, we could write the above statement as

$$\frac{1}{(1-x)^n} = \binom{n-1}{n-1} + \binom{n}{n-1}x + \binom{n+1}{n-1}x^2 + \binom{n+2}{n-1}x^3 + \cdots$$

where we used the fact that $(1-x)^{-1} = 1+x+x^2+x^3+\cdots$.

Therefore, the entire stars and bars problem is essentially encapsulated in the expression $(1-x)^{-n}$. Pretty neat! Let's see how we could approach a problem that would be very tricky normally but can be blown apart with generating functions.

Example 9.4. I have 100 candies and I want to distribute them to 5 kids. However, the two youngest kids must have at most one candy, and the two oldest kids both insist on having an odd number of candies. How many ways can I give out the candies?

Let's first write out the generating functions for these kids. The youngest kids both have generating functions of $1+x$. The two oldest kids have generating functions $x+x^3+x^5+\cdots$. Therefore, we want the coefficient of x^{100} of

$$(1+x)^2(1+x+x^2+\cdots)(x+x^3+x^5+\cdots)^2.$$

Now, this is a jumble of stuff, but what if we wrote the closed form for this expression?

$$(1+x)^2\frac{1}{1-x}\left(\frac{x}{1-x^2}\right)^2 = \frac{x^2}{(1-x)^3}.$$

So, we want to find the coefficient of x^{98} of $(1-x)^{-3}$, but that's just

$$\binom{100}{2} = 4950.$$

9.3 Generating Functions for Partitions

A *partition* of a positive integer n is a set of positive integers that sum to n (and we do not care about the order, so this is different from stars and bars). The partitions of 5 would be:

$$\begin{aligned} 5 &= 5 \\ &= 4+1 \\ &= 3+2 \\ &= 3+1+1 \\ &= 2+2+1 \\ &= 2+1+1+1 \\ &= 1+1+1+1+1 \end{aligned}$$

There isn't really a nice closed-form formula for how many partitions a positive integer n has. However, the generating function for partitions captures all of that information.

We can think about a partition as how many 1s it has, how many 2s, etc. So, let's get a sequence about the number of ways to have a certain number of 1s. We have

$$f_1(x) = 1 + x + x^2 + x^3 + \cdots .$$

where the coefficient of x^k is the number of ways to have k ones. Obviously, the number of ways to have k ones is just 1, since we don't care about the order. Similarly, the generating function for the number of ways to have 2s is

$$f_2(x) = 1 + x^2 + x^4 + x^6 + \cdots .$$

We can then see that the number of partitions of an integer n is the coefficient of x^n in

$$(1 + x + x^2 + \cdots)(1 + x^2 + x^4 + x^6 + \cdots)(1 + x^3 + x^6 + \cdots)\cdots .$$

Let's write this in a more compact form:

$$\frac{1}{(1-x)(1-x^2)(1-x^3)\cdots}$$

All the information about partitions is wrapped into that expression right there!

Example 9.5. Show that the number of partitions of n in which no part appears exactly once is equal to the number of partitions in which no part is one greater or one less than a multiple of 6. For example, if $n = 7$, then the partitions with no part appearing exactly once are $1+1+1+2+2$ and $1+1+1+1+1+1+1$, and the partitions with no part one greater or one less than a multiple of 6 are $2+2+3$ and $3+4$.

Not all the details will be given here, but we can use the ideas about generating functions to prove that both quantities are equal. Note that this problem can be solved with bijections, but the bijection isn't obvious at all! However, generating functions provide us a way to justify this with pure algebra.

The generating function for the partitions which no part appears exactly once is

$$(1+x^2+x^3+x^4+\cdots)(1+x^4+x^6+x^8+\cdots)(1+x^6+x^9+\cdots)\cdots,$$

which can be simplified to

$$\left(1+\frac{x^2}{1-x}\right)\left(1+\frac{x^4}{1-x^2}\right)\left(1+\frac{x^6}{1-x^3}\right)\cdots.$$

The second type of partition, which is no part is one greater or one less than a multiple of 6 is given by

$$\frac{1}{(1-x^2)(1-x^3)(1-x^4)(1-x^6)(1-x^8)(1-x^9)(1-x^{10})\cdots}.$$

Don't take our word for it! Make sure you convince yourself why that is! Then, can you show that these two are indeed the same thing?

Exercise: Finish off the above problem and show that the two are equal.

9.4 Solving Linear Recurrences

Suppose we have the famous recursion $a_n = a_{n-1} + a_{n-2}$, $a_0 = 0$ and $a_1 = 1$. Let's try to write the generating function for the Fibonacci sequence (this was mentioned in the first note). We can write it as

$$f(x) = x + x^2 + 2x^3 + 3x^4 + 5x^5 + 8x^6 + 13x^7 + \cdots.$$

How can we simplify this? Let's multiply this by x and x^2 and see what we get

$$xf(x) = x^2 + x^3 + 2x^4 + 3x^5 + 5x^6 + \cdots$$

and

$$x^2f(x) = x^3 + x^4 + 2x^5 + 3x^6 + 5x^7 + \cdots.$$

Now, we can take advantage of the Fibonacci recursion and add $xf(x)$ and $x^2f(x)$ to get

$$x + xf(x) + x^2f(x) = f(x).$$

Make sure you see how we got this! Now, we can solve for $f(x)$ to get

$$f(x) = \frac{x}{1-x-x^2}.$$

We have wrapped the Fibonacci sequence into yet another function. Awesome! The denominator is a polynomial, and it is in fact a very special polynomial. It is related to the *characteristic polynomial* of the linear recursion. Basically, the characteristic polynomial will show up in the denominator of a generating function of a linearly recursive sequence, but with the coefficients backwards.

So, if we have a recursion that was $a_n - 3a_{n-1} + 2a_{n-2} = 0$, then the characteristic polynomial would be $x^2 - 3x + 2$, and so our denominator would be $1 - 3x + 2x^2$. Why? Because if we think about the polynomial, we have

$$\cdots + a_{n-2}x^{n-2} + a_{n-1}x^{n-1} + a_n x^n + \cdots$$

and so when we multiply this by $1 - 3x + 2x^2$ we would get 0, since they cancel out by the recursion!

This is helpful because we are now able to factor the denominator and do partial fraction decomposition. So,

$$\begin{aligned} f(x) &= \frac{x}{\sqrt{5}}\left(\frac{1}{\phi_1 - x} - \frac{1}{\phi_2 - x}\right), \\ \phi_1 &= \frac{-1+\sqrt{5}}{2}, \\ \phi_2 &= \frac{-1-\sqrt{5}}{2}. \end{aligned}$$

Then, using the formula for a geometric series, we can recover the closed-form formula for the Fibonacci numbers.

In general, linear recurrences can be solved this way, where we factor the denominator, perform partial fraction decomposition, and get a bunch of geometric series.

9.5 Roots of Unity Filter

We should be pretty familiar with the following combinatorial identity:

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n} = 2^n,$$

which can be proved by considering the number of ways to form a subset of $\{1, 2, 3, \ldots, n\}$. You may have also seen

$$\binom{n}{0} - \binom{n}{1} + \binom{n}{2} - \binom{n}{3} + \cdots = 0.$$

In fact, both of these follow quite easily from the Binomial Theorem, since we know

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n$$

and plugging in $x = 1$ and $x = -1$ respectively will give the desired identities.

Adding these two identities will give us

$$\binom{n}{0} + \binom{n}{2} + \binom{n}{4} + \cdots = 2^{n-1}.$$

Now what is

$$\binom{n}{0} + \binom{n}{3} + \binom{n}{6} + \cdots ?$$

or

$$\binom{n}{0} + \binom{n}{4} + \binom{n}{8} + \cdots ?$$

Now, a root of unity is a root of the polynomial $x^m = 1$. The third roots of unity would be the solutions to $x^3 = 1$. What are these solutions? They are:

$$e^{2\pi i/3}, e^{4\pi i/3}, e^{6\pi i/3}.$$

You can look up roots of unity to learn about where these magical numbers came from. Now, let's let $\omega = e^{2\pi i/3}$. Then, we have $\omega, \omega^2, \omega^3$ as the roots of $x^3 = 1$. Now, let's plug these values into $(1+x)^n$. Also, there is an important property of roots of unity: their sum is 0. So, we can add up the results and get

$$(1+\omega)^n + (1+\omega^2)^n + (1+\omega^3)^n = 3\binom{n}{0} + \binom{n}{1}(\omega + \omega^2 + \omega^3) + \binom{n}{2}(\omega + \omega^2 + \omega^3) + \cdots .$$

Only every third term survives. Now, we know that

$$\begin{aligned} \omega &= -\frac{1}{2} + \frac{\sqrt{3}}{2}i, \\ \omega^2 &= -\frac{1}{2} - \frac{\sqrt{3}}{2}i, \\ \omega^3 &= 1. \end{aligned}$$

We have

$$\frac{1}{3}\left(\left(\frac{1}{2} + \frac{\sqrt{3}}{2}i\right)^n + \left(\frac{1}{2} - \frac{\sqrt{3}}{2}i\right)^n + 2^n\right) = \binom{n}{0} + \binom{n}{3} + \cdots .$$

These values are actually very easy to compute if we convert the complex numbers to exponential form. We have

$$\frac{1}{3}(e^{in\pi/3} + e^{-in\pi/3} + 2^n) = \frac{2}{3}\cos\left(\frac{n\pi}{3}\right) + \frac{2^n}{3}.$$

Isn't that crazy? We have trigonometric functions and complex numbers popping up in a sum of binomial coefficients! And there you have it:

$$\binom{n}{0} + \binom{n}{3} + \binom{n}{6} + \cdots = \frac{2^n}{3} + \frac{2}{3}\cos\left(\frac{n\pi}{3}\right).$$

Try the formula for some small values of n and see if it works (it will, of course, but it is still an amazing fact).

We could do the same trick for 4, and obtain

$$\binom{n}{0} + \binom{n}{4} + \binom{n}{8} + \cdots = \frac{2^n + (1+i)^n + 0^n + (1-i)^n}{4}.$$

Converting to exponential form and simplifying we get

$$\frac{1}{4}\left(2^n + 2^{n/2} \cdot 2\cos\left(\frac{n\pi}{4}\right)\right).$$

Test it! In fact, we can apply this to generating functions in general. We plug in the kth root of unity so we can "filter out" all the terms that we don't want.

Extra Note 10

Pairwise Independent Hash Functions

A family of hash functions, with the form $h : \{0, \ldots, n-1\} \to \{0, \ldots, p-1\}$ for simplicity, often needs to satisfy the property of **pairwise independence** for use in algorithms. Intuitively, pairwise independence is the property that for two distinct inputs i and j, knowing the value $h(i)$ tells you nothing about $h(j)$. We will see how to construct a family of hash functions with this property when p is a prime number.

10.1 Model

When we speak of pairwise independence, where is the source of randomness? To clarify the situation, we have a fixed (non-random) family of hash functions $\mathcal{H}$, and the randomness lies in the fact that we are choosing a hash function h from $\mathcal{H}$ *uniformly at random*. Hence, we can regard h as a *random function*, and so we can discuss the probability of the event $\{h(i) = h(j)\}$.

First, we should note that the purpose of a hash function is to "evenly spread out its inputs among its outputs". Typically we visualize the output set $\{0, \ldots, p-1\}$ as a set of p buckets. One way of looking at a hash function is to imagine that the input is a ball, which you are throwing into one of p bins. However, this view misses one crucial fact: *hash functions are deterministic*. In other words, if you look at where a single input is hashed, then it appears to be randomly chosen out of the p buckets; but if we receive the *same* input again, then it is hashed to the *same* bucket as before.

If you give the idea some careful thought, then you should be impressed. It is an astonishing property which makes hash functions incredibly useful for algorithms.

Pairwise independent hash functions guarantee a little more: for any distinct i and j, and any two values y_i and y_j (not necessarily distinct), we must have the following hold:

$$\mathbb{P}(h(i) = y_i \cap h(j) = y_j) = \frac{1}{p^2}. \tag{10.1}$$

You should compare (10.1) to the definition of independence and try to understand in what way the above definition captures the notion of independence for hash functions. (Remember

that the probability above is taken over the randomness of the hash function, because we are choosing a *random* hash function from our family $\mathcal{H}$.)

10.2 The Family of Hash Functions

The scheme is simple. Let $\mathcal{H}$ be the family of hash functions of the form $h(x) = ax+b \bmod p$, where $a, b \in \{0, \ldots, p-1\}$. Observe that it is very easy to pick a hash function randomly from $\mathcal{H}$: instead of generating all hash functions in $\mathcal{H}$ and picking one randomly, we can simply pick a and b uniformly at randomly in the set $\{0, \ldots, p-1\}$ and we will achieve the same effect. The fact that we can generate random hash functions from $\mathcal{H}$ makes it a very attractive family to study!

So, we consider a hash function h and we regard a and b as chosen uniformly at random from $\{0, \ldots, p-1\}$. Also, a and b are chosen independently of each other. Next, we seek to calculate the probability in (10.1). If $h(i) = y_i$ and $h(j) = y_j$, then

$$\begin{aligned} ai + b &\equiv y_i \pmod{p}, \\ aj + b &\equiv y_j \pmod{p}. \end{aligned}$$

In the system of equations above, a and b are the variables. We are given i, j, y_i, and y_j, and we are asking the question: what values of a and b solve the above system of equations?

Subtracting the second equation from the first, we have $a(i-j) \equiv y_i - y_j \pmod{p}$. However, we are assuming that $i \neq j$, which implies that $i - j$ has a multiplicative inverse modulo p. Hence, we can solve for $a = (i-j)^{-1}(y_i - y_j) \bmod p$. Once we have solved for a, we can plug in this value for a into either of the equations and solve: $b = y_i - ai \bmod p = y_j - aj \bmod p$.

What have we found? We have showed that there is a *unique* pair (a, b) that solves the system of equations. Since there are p^2 total choices for (a, b), the probability of choosing the one (a, b) pair which solves the system of equations is $1/p^2$, which verifies (10.1).

10.3 k-Wise Independence

One can extend the results to generate k-wise independent hash functions satisfying

$$\mathbb{P}(h(i_1) = y_1 \cap \cdots \cap h(i_k) = y_k) = \frac{1}{p^k}$$

for distinct $i_1, \ldots, i_k$ and any $y_1, \ldots, y_k$. The technique is to use a family of hash functions which have the form of degree-k polynomials taken modulo p.

Extra Note 11

Moment-Generating Functions

11.1 Introduction

A **moment-generating function** is a *transform* which allows us to conveniently calculate quantities of interest for random variables. Formally, it is defined as:

Definition 11.1. Let X be a random variable. The **moment-generating function** of X is the function:

$$\boxed{M_X(\theta) = E[e^{\theta X}]} \tag{11.1}$$

In other words, the MGF is the expectation of the random variable $e^{\theta X}$.

Example 11.2. Let's see an example of how to compute the MGF of a random variable. Let $X \sim \text{Poisson}(\lambda)$. Then

$$M_X(\theta) = E[e^{\theta X}] = \sum_{x=0}^{\infty} e^{\theta x} \frac{e^{-\lambda}\lambda^x}{x!} = e^{-\lambda} \sum_{x=0}^{\infty} \frac{(\lambda e^{\theta})^x}{x!} = e^{-\lambda} e^{\lambda e^{\theta}} = e^{\lambda(e^{\theta}-1)}.$$

We have used the identity

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}. \tag{11.2}$$

We will return to this example shortly. Our immediate goal is to explain the meaning behind the name "moment-generating function".

Theorem 11.3. *Let X be a random variable with MGF $M_X(\theta)$. Then:*

$$\boxed{E[X^k] = M_X^{(k)}(0)} \tag{11.3}$$

Proof. We can carry out the computation directly:

$$M_X^{(k)}(\theta) = \frac{\mathrm{d}^k}{\mathrm{d}\theta^k} \sum_{x=0}^{\infty} e^{\theta x} P(X = x) = \sum_{x=0}^{\infty} x^k e^{\theta x} P(X = x)$$

Setting $\theta = 0$, we obtain

$$M_X^{(k)}(0) = \sum_{x=0}^{\infty} x^k P(X = x) = E[X^k]. \qquad \square$$

Technical Remark: In the proof above, we interchanged the order of differentiation and summation. This is *not* always valid; there are conditions that must be met in order to justify this step. However, for the purpose of this exposition, we will assume that all random variables under consideration are "well-behaved" so that the above proof holds.

Recall that the Taylor series of a function is given by

$$f(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \cdots$$

The above theorem implies that the Taylor expansion of the MGF is (note that $M_X(0) = 1$)

$$M_X(\theta) = 1 + \frac{E[X]}{1!}\theta + \frac{E[X^2]}{2!}\theta^2 + \cdots$$

Therefore, we can see that the MGF is indeed a generating function, where the kth coefficient is given by $E[X^k]/k!$. The values $E[X^k]$ are often known as the **moments** of a distribution, which explains the name.

11.2 Sums of Random Variables

Here is one of the reasons why the MGF is so useful for computation.

Theorem 11.4. *Let X and Y be independent random variables. Then:*

$$\boxed{M_{X+Y}(\theta) = M_X(\theta) M_Y(\theta)} \tag{11.4}$$

Proof.

$$M_{X+Y}(\theta) = E[e^{\theta(X+Y)}] = E[e^{\theta X} e^{\theta Y}] = E[e^{\theta X}]E[e^{\theta Y}] = M_X(\theta) M_Y(\theta) \qquad \square$$

Previously, we have proven that if $X \sim \text{Poisson}(\lambda)$ and $Y \sim \text{Poisson}(\mu)$, then $X + Y \sim \text{Poisson}(\lambda+\mu)$. We will prove this fact again in a much simpler way using the MGF function.

Theorem 11.5. *Suppose that X and Y are independent random variables with distributions $X \sim \text{Poisson}(\lambda)$ and $Y \sim \text{Poisson}(\mu)$. Then $X + Y \sim \text{Poisson}(\lambda + \mu)$.*

Proof. Observe:

$$M_{X+Y}(\theta) = M_X(\theta)M_Y(\theta) = e^{\lambda(e^\theta - 1)}e^{\mu(e^\theta - 1)} = e^{(\lambda+\mu)(e^\theta - 1)}$$

This is precisely the MGF function of a $\text{Poisson}(\lambda + \mu)$ distribution. □

Actually, if you were especially observant, you will have noticed that we cheated in the last line of the above proof. After all, we did not answer the question: does the moment-generating function completely determine the distribution? Is there potentially another probability distribution besides $\text{Poisson}(\lambda + \mu)$ which happens to have the same MGF? We will overlook these details; if you are interested, you should take further courses in probability.

The Poisson distribution is especially nice because the sum of independent Poisson distributions is Poisson. This amazing property does *not* hold for most distributions. In fact, the distribution of the sum of random variables is often very complicated and messy. Indeed, this is the reason why we prefer to work with the MGF when we have a sum of independent random variables.

11.3 Cumulants

Definition 11.6. The **cumulant-generating function** of a random variable X is defined as:

$$\boxed{C_X(\theta) = \log M_X(\theta)} \tag{11.5}$$

It seems like all we did was take the logarithm of the MGF. Why is this useful?

Theorem 11.7. *Let X be a random variable with cumulant-generating function $C_X(\theta)$. We have the following facts:*

1. $C_X(0) = 0$.
2. $C'_X(0) = E[X]$.
3. $C''_X(0) = \text{var}(X)$.

Proof.

1. Since $M_X(0) = 1$, then $C_X(0) = \log M_X(0) = \log 1 = 0$.
2. We compute $C'_X(\theta)$:

$$C'_X(\theta) = \frac{\mathrm{d}}{\mathrm{d}\theta} \log M_X(\theta) = \frac{M'_X(\theta)}{M_X(\theta)}$$

Plugging in $\theta = 0$, and using $M'_X(0) = E[X]$, we have $C'_X(0) = E[X]$.

3. We compute $C''_X(\theta)$:

$$C''_X(\theta) = \frac{\mathrm{d}}{\mathrm{d}\theta}\frac{M'_X(\theta)}{M_X(\theta)} = \frac{M''_X(\theta)M_X(\theta) - (M'_X(\theta))^2}{(M_X(\theta))^2}$$

Now, we use $M''_X(0) = E[X^2]$ to obtain $C''_X(0) = E[X^2] - E[X]^2 = \mathrm{var}(X)$.

□

The cumulant-generating function is often useful because taking the logarithm before taking the derivative often makes calculations simpler. Intuitively, logarithms convert products into sums, and we would much rather differentiate sums rather than products.

Example 11.8. Let us compute the mean and variance of the Poisson distribution. If $X \sim \mathrm{Poisson}(\lambda)$, then $M_X(\theta) = e^{\lambda(e^\theta - 1)}$. Therefore, $C_X(\theta) = \lambda(e^\theta - 1)$. Differentiating, we obtain

$$C'_X(0) = \lambda e^\theta\Big|_{\theta=0} = \lambda.$$

and

$$C''_X(0) = \lambda e^\theta\Big|_{\theta=0} = \lambda.$$

Moment-generating functions find many applications, both because they aid computations (as we have seen above) and because of their useful properties (such as convexity) which are useful for proving theorems. We hope you found this exposition interesting, and we certainly encourage you to read more about these objects in your spare time.

Extra Note 12

Jensen's Inequality

12.1 Convex Functions

This extra note focuses on another highly useful inequality in probability: **Jensen's inequality**. Jensen's inequality can be applied to any convex function, so first, we will need to introduce the notion of convexity.

Definition 12.1. A function ϕ is **convex** if for all $x < y$, for all $0 \leq \lambda \leq 1$,

$$\phi((1-\lambda)x + \lambda y) \leq (1-\lambda)\phi(x) + \lambda\phi(y).$$

Fix two points x, y, with $x < y$. Observe that as λ varies from 0 to 1, then the function $(1-\lambda)\phi(x) + \lambda\phi(y)$ traces out the straight line from $\phi(x)$ to $\phi(y)$. Specifically, for a fixed value of λ, $(1-\lambda)\phi(x) + \lambda\phi(y)$ is a point on the line. On the other hand, $\phi((1-\lambda)x + \lambda y)$ is the point on the curve ϕ at the same x-value. Therefore, the definition of convexity states that the curve ϕ always lies *below* any straight line connecting two points on the curve.

Another description of convex functions is that they lie *above* their corresponding tangent lines. In other words, given a point x_0, there exists a line $l(x)$ such that $l(x) \leq \phi(x)$ for all x, and the line is "tangent" to the curve in the sense that $l(x_0) = \phi(x_0)$.

One common description of convex functions is that they are *bowl-shaped*. You may have previously learned that convexity is related to the second derivative of the function. To see how we can relate the two definitions, recall from Taylor's theorem that for some $x_0 \in (x, y)$,

$$\phi(y) = \phi(x) + \phi'(x)(y-x) + \frac{\phi''(x_0)}{2}(y-x)^2.$$

If $\phi''(x) \geq 0$ always, then $\phi(y) - \phi(x) - \phi'(x)(y-x) \geq 0$ always. However, $\phi(x) + \phi'(x)(y-x)$ is precisely the tangent line to ϕ at x, so we see that $\phi''(x) \geq 0$ is a sufficient condition for convexity. The definition of convexity given above, however, can be applied with more generality, since it does not assume that ϕ is differentiable.

12.2 Jensen's Inequality

Now, we present the inequality.

Theorem 12.2 (Jensen's Inequality)**.** *Suppose that ϕ is any convex function. Then $\phi(E[X]) \leq E[\phi(X)]$, provided that the expectations exist.*

Proof. We make use of the fact that ϕ lies above its tangent line. Let $\mu = E[X]$ and consider the line l which is tangent to ϕ at μ, that is, we have $l(\mu) = \phi(\mu)$ and for all x, $\phi(x) \geq l(x)$. Then,

$$E[\phi(X)] \geq E[l(X)] = l(E[X]) = l(\mu) = \phi(\mu) = \phi(E[X]).$$

We wrote $E[l(X)] = l(E[X])$ because of linearity of expectation. □

Here are a few applications.

Example 12.3. Apply Jensen's inequality with the convex function x^2.

$$E[X]^2 \leq E[X^2]$$

This provides a proof that the variance is non-negative.

Example 12.4. The function $\phi(x) = |x|^p$ is convex for $p \geq 1$. Let $Y = |X|^a$ and apply Jensen's inequality to Y with the convex function $\phi(x) = x^{b/a}$, where $b > a$. Then,

$$E[Y]^{b/a} \leq E[Y^{b/a}].$$

Substituting $Y = |X|^a$ yields

$$E[|X|^a]^{b/a} \leq E[|X|^b].$$

Rearranging the inequality gives **Lyapunov's inequality**:

$$E[|X|^a]^{1/a} \leq E[|X|^b]^{1/b}$$

Consider positive numbers $x_1, \ldots, x_n > 0$. Let X be uniformly distributed over the x_i, i.e. $P(X = x_i) = 1/n$ for all i. Observe that

$$E[X] = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

$$E[\phi(X)] = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i).$$

Example 12.5. Apply Jensen's inequality to the convex function $\phi(x) = -\log x$. We have

$$-\log(E[X]) \leq E[-\log X]$$

which gives

$$\log(E[X]) \geq E[\log X].$$

Taking the exponential of both sides yields

$$E[X] \geq e^{E[\log X]}.$$

Now, return to our setting where $P(X = x_i) = 1/n$. The above inequality tells us

$$\frac{1}{n}\sum_{i=1}^{n} x_i \geq e^{n^{-1}\sum_{i=1}^{n}\log x_i} = \prod_{i=1}^{n} e^{n^{-1}\log x_i} = \prod_{i=1}^{n} x_i^{1/n} = \left(\prod_{i=1}^{n} x_i\right)^{1/n}.$$

This provides another proof of Cauchy's AM-GM inequality, which was the subject of the first extra note.

Example 12.6. Apply Jensen's inequality to the convex function $\phi(x) = 1/x$ (the function is convex as long as we restrict ourself to positive inputs). We have

$$\frac{1}{E[X]} \leq E[X^{-1}]$$

or

$$E[X] \geq \frac{1}{E[X^{-1}]}.$$

In the setting above, we have

$$\frac{1}{n}\sum_{i=1}^{n} x_i \geq \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i}\right)^{-1}.$$

The latter is sometimes called the **harmonic mean**, so we have proven that the arithmetic mean is at least as big as the harmonic mean.

These are just a few of the inequalities you can derive using Jensen's inequality. There is also a version of Jensen's inequality that works for conditional expectation.

Extra Note 13

Martingales

13.1 Introduction

We will look at a very interesting application of conditional expectation: **martingales**. The subject of martingales requires more probability theory than we have covered, so we present only a simplified treatment here.

Definition 13.1. A sequence of random variables $\{X_n : n \in \mathbb{N}\}$ is a **martingale** if it satisfies

$$E[X_{n+1} \mid X_1, \ldots, X_n] = X_n \tag{13.1}$$

There is a natural *gambling* interpretation of this definition: X_n is your fortune at time n, and you are making a series of bets. The martingale property says that, conditioned on your fortune being $X_1, \ldots, X_n$ in the first n time steps, your expected fortune at time $n+1$ is X_n, that is, you don't expect to win or lose any money on the $(n+1)$th bet. A concise way of describing a martingale is: a martingale describes a *fair game*. Note that the martingale property implies

$$E[X_{n+1}] = E[E[X_{n+1} \mid X_1, \ldots, X_n]] = E[X_n]$$

so that $E[X_n] = E[X_{n-1}] = \cdots = E[X_0]$, i.e. your expected fortune is always the same.

Let us see a few examples of martingales before we prove results about them.

Example 13.2. Suppose that X_i are independent random variables with $E[X_i] = 0$. Then the sequence $S_n = \sum_{i=1}^{n} X_i$ is a martingale. We verify the martingale property:

$$\begin{aligned} E[\underbrace{S_{n+1}}_{S_n + X_{n+1}} \mid X_1, \ldots, X_n] &= E[S_n \mid X_1, \ldots, X_n] + E[X_{n+1} \mid X_1, \ldots, X_n] \\ &= E[S_n \mid X_1, \ldots, X_n] + \underbrace{E[X_{n+1}]}_{=0} = S_n \end{aligned}$$

First, we split S_{n+1} into $S_n + X_{n+1}$ and applied linearity of expectation. Since S_n is a

function of $X_1, \ldots, X_n$, then $E[S_n \mid X_1, \ldots, X_n] = S_n$. (Think about the MMSE property: the best function of $X_1, \ldots, X_n$ to estimate S_n is simply S_n itself.) Since X_{n+1} is independent of $X_1, \ldots, X_n$, then $E[X_{n+1} \mid X_1, \ldots, X_n] = E[X_{n+1}] = 0$.

Intuitively, this example reinforces our interpretation of martingales as fair games: X_n is the bet you make at time n, and if we assume that each bet is fair ($E[X_n] = 0$), then your fortune (the sum of your winnings so far) is a martingale.

Example 13.3. Suppose that X_i are independent random variables with $E[X_i] = 1$. Then the sequence $P_n = \prod_{i=1}^{n} X_i$ is a martingale. We verify the martingale property:

$$E[\underbrace{P_{n+1}}_{P_n \cdot X_{n+1}} \mid X_1, \ldots, X_n] = E[P_n X_{n+1} \mid X_1, \ldots, X_n] = P_n E[X_{n+1} \mid X_1, \ldots, X_n]$$
$$= P_n \underbrace{E[X_{n+1}]}_{=1} = P_n$$

Since P_n is a function of $X_1, \ldots, X_n$, conditioning on $X_1, \ldots, X_n$ effectively turns P_n into a constant that can be moved outside of the conditional expectation. Again, X_{n+1} is independent of $X_1, \ldots, X_n$, so $E[X_{n+1} \mid X_1, \ldots, X_n] = E[X_{n+1}] = 1$.

Here, the interpretation is that your fortune is *multiplied* by X_n at time n, and $E[X_n] = 1$ ensures that the bets are still fair.

13.2 Martingale Transforms

Suppose we have a martingale X_n. The martingale represents your fortune at time n, so $X_n - X_{n-1}$ represents how much money you gained on the nth bet. Although we cannot control the outcome of the wagers, perhaps we can profit by varying the *amount* of money that we wager. Formally, we say that a sequence of random variables H_n is **predictable** if it is a function of $X_1, \ldots, X_{n-1}$ (which is to say that H_n can only depend on the information you have *before* making the nth bet). We interpret H_n to be the amount of money you wager on the nth bet. Originally, we gained $X_n - X_{n-1}$ on the nth bet, and now we stand to gain $Y_n - Y_{n-1} = H_n(X_n - X_{n-1})$ on the nth bet, where Y_n is a sequence of random variables representing the fortune at time n under this new betting system.

Notation: We write $Y = H \cdot X$ and call Y a **martingale transform** or a **discrete-time stochastic integral**.

Theorem 13.4 (Martingale Transforms)**.** *Let X_n be a martingale and H_n be predictable. Then $Y = H \cdot X$ is also a martingale.*

Proof. First, we observe that the martingale property is equivalent to

$$E[X_{n+1} - X_n \mid X_1, \ldots, X_n] = 0 \tag{13.2}$$

We verify this version of the martingale property.

$$\begin{aligned} E[Y_{n+1} - Y_n \mid Y_1, \ldots, Y_n] &= E[H_{n+1}(X_{n+1} - X_n) \mid Y_1, \ldots, Y_n] \\ &= H_{n+1} E[X_{n+1} - X_n \mid Y_1, \ldots, Y_n] = 0 \end{aligned}$$

since H_{n+1} is a function of $Y_1, \ldots, Y_n$ and X_n is a martingale. □

Saying that Y_n is a martingale means that Y_n is also a fair game. Under the new betting system, no matter how inventive, you stand to do no better on average than your original X_n! Therefore, we can interpret the theorem as saying: you cannot cheat a fair game.

13.3 Stopping Times

Definition 13.5. A **stopping time** is a random variable T which takes on values in $\mathbb{N}$, such that $1_{(T=n)}$ is a function of $X_1, \ldots, X_n$ for every n.

Think of T as the time at which you stop betting. The condition that $1_{(T=n)}$ is a function of $X_1, \ldots, X_n$ is interpreted as: your decision to stop betting after the nth bet can only depend on information up until the nth bet, and not on the future.

Proposition 13.6. *In order to show that T is a stopping time, it is equivalent to show that $1_{(T \leq n)}$ is a function of $X_1, \ldots, X_n$ for every n.*

Proof. Sufficient:

$$1_{(T=n)} = 1_{(T \leq n)} - 1_{(T \leq n-1)}$$

and both $1_{(T \leq n)}$ and $1_{(T \leq n-1)}$ are functions of $X_1, \ldots, X_n$ by assumption, so $1_{(T=n)}$ is a function of $X_1, \ldots, X_n$.

Necessary: Suppose that $1_{(T=n)}$ is a function of $X_1, \ldots, X_n$ for all n. Then

$$1_{(T \leq n)} = \sum_{i=0}^{n} 1_{(T=i)}$$

and each $1_{(T=i)}$ is a function of $X_1, \ldots, X_n$, so $1_{(T \leq n)}$ is a function of $X_1, \ldots, X_n$. □

Can we start with a martingale and profit by choosing a judicious stopping time? "Quit while you're ahead", in a nutshell. Our final result says "no":

Theorem 13.7. *Suppose that X_n is a martingale and let T be a stopping time. Define Y_n to be the "stopped" process, that is $Y_n = X_{\min(T,n)}$ (so $Y_n = X_n$ until T, and then $Y_n = X_T$ afterwards). Then Y_n is a martingale.*

Proof. Define $H_n = 1_{(n \leq T)}$. This reflects the following betting strategy: bet one dollar at the start, and continue until time T, and then bet no more afterwards. H_n is predictable: note that the event $\{n \leq T\}$ is the complement of the event $\{n > T\} = \{T \leq n-1\}$, so $H_n = 1_{(n \leq T)} = 1 - 1_{(T \leq n-1)}$, which is a function of $X_1, \ldots, X_{n-1}$. Next, we observe that $Y_n = \sum_{i=1}^{n} (Y_i - Y_{i-1}) = \sum_{i=1}^{n} (X_i - X_{i-1}) 1_{(i \leq T)} = \sum_{i=1}^{n} H_i (X_i - X_{i-1})$, so Y_n is indeed a martingale transform of X_n. Finally, we apply Theorem 13.4 to $Y_n = H \cdot X$ and we obtain that Y is a martingale as well. □

13.4 More Martingales

Of course, there are many more interesting results on martingales. We can define a **submartingale** by the property

$$E[X_{n+1} \mid X_1, \ldots, X_n] \geq X_n \tag{13.3}$$

and a **supermartingale** by the property

$$E[X_{n+1} \mid X_1, \ldots, X_n] \leq X_n \tag{13.4}$$

Martingales have useful convergence properties and can be used to prove a number of inequalities such as Doob's inequality. Martingales are fascinating because we can attach gambling interpretations in order to make sense of what's going on, yet end up with a completely mathematical result. The martingale transform result is particularly fun: by choosing various gambling strategies, we can obtain new martingales!

Extra Note 14

PageRank Algorithm

14.1 Introduction

This week's extra note discusses **PageRank**, the algorithm behind Google's search engine. This edition of the extra notes was written by **Allen Tang**!

14.2 History

Perhaps you have never thought about how Google search works, even though you use it every single day. It turns out the the whole algorithm under the search function is a very complicated mechanism. In this extra note, we will look at one part of the Google search system which uses a Markov chain to model the importance of each webpage. When you put a keyword ("query") in the search box, Google first searches for all webpages that have your keywords, ranks all of them, and returns to you a list of results sorted by importance or relevance. It took scientists a long time to develop such an elegant and precise algorithm to measure the importance of each page. In 1998, Larry Page (guess why the algorithm is called PageRank) and Sergey Brin, two graduate students at Stanford University, came up with the PageRank algorithm to quantize the importance of each page. Later on, they founded Google to serve billions of people who use the internet. Of course, Google, as a fast-growing technology company, continues to improve the algorithm and incorporate other algorithms from artificial intelligence and machine learning. The original PageRank algorithm is still considered one of the most influential algorithms in the fields of stochastic processes and information retrieval.

14.3 Assumptions

The basic assumption of PageRank is that the importance of a web page is determined by the number of pages that point to it. Obviously, a page with more incoming links is more important than a page with less incoming links. Also, if a very important page P has a link to another page P', then P' is also important. For example, the Berkeley EECS homepage should have a really high importance score because a lot of other websites have links on their

sites pointing to Berkeley EECS homepage (since we are the best). The more links point to the Berkeley EECS homepage, the more important the webpage is. Think about academic paper citations. If a lot of papers cite the same paper P (in our case, a lot of webpages have links to the same webpage P), then the paper (webpage) P must be very influential.

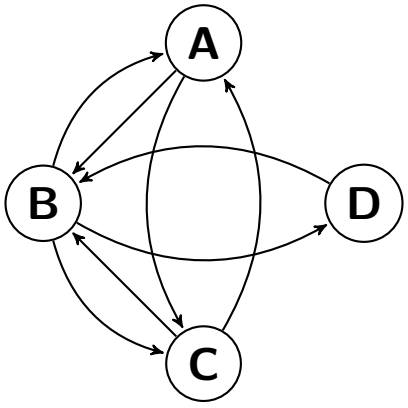
14.4 A Simple Example

Let's first consider a very simple network with only four websites: Professor Walrand's homepage (A), the Berkeley EECS homepage (B), the CS 70 homepage (C), and the wikipage about Soda Hall (D).

Suppose that we have the following situation:

- Professor Walrand has links on his website pointing to the Berkeley EECS homepage and the CS 70 homepage.
- The Berkeley EECS homepage has links on its website pointing to Professor Walrand's website, the CS 70 homepage, and the wikipage about Soda Hall.
- The CS 70 homepage has links to Professor Walrand's website and the Berkeley EECS homepage.
- The wikipage about Soda Hall has links to the Berkeley EECS homepage.

Now, we can visualize these relationships by representing the connections between the four websites in the graphical model below. Each node represents a webpage and an arrow pointing from page P to page P' means that there is a link on page P that points to page P'.



You may have noticed that this graphical model captures a Markov chain process. Suppose you are at one page (state) P, and given all of the links on that page, you will pick one link with some probability and go to that page (next state). You then pick another link on the new page and travel to another page. This process continues forever. Since we have a Markov chain process, we can start by finding the transition probability matrix.

Let's start by defining some notation. Let N be the total number of webpages (states), which in this case is equal to 4. Let $L(p)$ be the number of out-going links in a page p. In graph theory, $L(p)$ is the out-degree of node p. In the example above, we have $L(D) = 1, L(A) = L(C) = 2, L(B) = 3$. Now, suppose you are at page A with 2 links on it. You will uniformly choose one link as your next destination. In general, if there is a link from page P to page P', then the transition probability distribution is a simple uniform distribution. If there is no link from page P to page P', the transition probability should be simply 0. Mathematically, we can create a $N \times N$ matrix P by defining the (i, j) entry as:

$$P(i,j) = \begin{cases} \dfrac{1}{L(i)}, & \text{if there is a link from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

Below, we write the transition matrix P corresponding to the example. We can do a sanity check: the entries in each row must always sum up to 1.

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Now it's time to reveal the PageRank algorithm. We need to first initialize a vector π_0 representing the importance of each webpage. At the beginning, we initialize all entries to 1. Because each incoming link increases the importance of each webpage, we start to update the importance of each page by adding the current importance values of the incoming links. This is the same as multiplying π_0 with P. The algorithm will repeatedly replace π_n by $\pi_n P$ until it converges.

Here is the numeric computation of our example:

$$\begin{aligned}
\pi_0 &= \begin{bmatrix}1.000 & 1.000 & 1.000 & 1.000\end{bmatrix} & \pi_0 P &= \begin{bmatrix}0.833 & 2.000 & 0.833 & 0.333\end{bmatrix} \\
\pi_0 P^2 &= \begin{bmatrix}1.083 & 1.167 & 1.083 & 0.667\end{bmatrix} & \pi_0 P^3 &= \begin{bmatrix}0.931 & 1.750 & 0.931 & 0.389\end{bmatrix} \\
\pi_0 P^4 &= \begin{bmatrix}1.049 & 1.320 & 1.049 & 0.583\end{bmatrix} & \pi_0 P^5 &= \begin{bmatrix}1.049 & 1.320 & 1.049 & 0.583\end{bmatrix} \\
\pi_0 P^6 &= \begin{bmatrix}1.026 & 1.404 & 1.026 & 0.544\end{bmatrix} & \pi_0 P^7 &= \begin{bmatrix}0.981 & 1.570 & 0.981 & 0.468\end{bmatrix} \\
\pi_0 P^8 &= \begin{bmatrix}1.013 & 1.449 & 1.013 & 0.523\end{bmatrix} & \pi_0 P^9 &= \begin{bmatrix}1.007 & 1.473 & 1.007 & 0.512\end{bmatrix} \\
\pi_0 P^{10} &= \begin{bmatrix}0.995 & 1.512 & 0.995 & 0.491\end{bmatrix} & \pi_0 P^{11} &= \begin{bmatrix}1.004 & 1.486 & 1.004 & 0.507\end{bmatrix} \\
\pi_0 P^{12} &= \begin{bmatrix}0.997 & 1.510 & 0.997 & 0.495\end{bmatrix} & \pi_0 P^{13} &= \begin{bmatrix}1.002 & 1.492 & 1.002 & 0.503\end{bmatrix} \\
\pi_0 P^{14} &= \begin{bmatrix}0.998 & 1.506 & 0.998 & 0.497\end{bmatrix} & \pi_0 P^{15} &= \begin{bmatrix}1.001 & 1.496 & 1.001 & 0.502\end{bmatrix} \\
\pi_0 P^{16} &= \begin{bmatrix}0.999 & 1.503 & 0.999 & 0.499\end{bmatrix} & \pi_0 P^{17} &= \begin{bmatrix}1.001 & 1.498 & 1.001 & 0.501\end{bmatrix} \\
\pi_0 P^{18} &= \begin{bmatrix}1.000 & 1.502 & 1.000 & 0.500\end{bmatrix} & \pi_0 P^{19} &= \begin{bmatrix}1.000 & 1.499 & 1.000 & 0.501\end{bmatrix} \\
\pi_0 P^{20} &= \begin{bmatrix}1.000 & 1.501 & 1.000 & 0.500\end{bmatrix} & \pi_0 P^{21} &= \begin{bmatrix}1.000 & 1.500 & 1.000 & 0.500\end{bmatrix} \\
\pi_0 P^{22} &= \begin{bmatrix}1.000 & 1.500 & 1.000 & 0.500\end{bmatrix} & & \cdots
\end{aligned}$$

We can see that after 20 iterations, our importance vector x converges. We obtain our final importance vector $\pi = \begin{bmatrix} 1 & 1.5 & 1 & 0.5 \end{bmatrix}$, which makes sense. The Berkeley EECS homepage (B) has the most incoming links, so it should achieve the highest score, while the wikipage on Soda Hall (D) has the fewest incoming links, whichs yield the lowest score. PageRank algorithm actually quantifies the importance of each webpage in the network.

14.5 Stationary Distribution

Recall that in the setting of Markov chains, we define π as a stationary distribution if it is invariant under the transition probability matrix: $\pi = \pi P$. In our case, we are wondering what the distribution of the importance vector will be when it stabilizes in the long run. In fact, the stationary distribution π is exactly what we are looking for. If a chain reaches a stationary distribution, then it maintains that distribution for all future times. A stationary distribution represents a steady state (or an equilibrium) in the chain's behavior. Essentially, the PageRank algorithm finds the stationary distribution of the importance vector under the transition probability matrix. In our case, we want to solve

$$\pi = \pi P,$$

$$\begin{bmatrix} \pi(0) & \pi(1) & \pi(2) & \pi(3) \end{bmatrix} = \begin{bmatrix} \pi(0) & \pi(1) & \pi(2) & \pi(3) \end{bmatrix} \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

If you have taken any linear algebra course, you should be able to solve this linear system. We can get exactly the same result as before: $\pi = \begin{bmatrix} 1 & 1.5 & 1 & 0.5 \end{bmatrix}$. (This is also very easy to verify!) It is also worth pointing out that π is the left eigenvector corresponding to the eigenvalue 1. Usually, people also normalize the vector π so that it becomes a probability distribution that sums up to 1, which is known as the probabilistic eigenvector. If you are interested in solving PageRank using linear algebra, I highly recommend you to read the paper *The $25,000,000,000 Eigenvector: The Linear Algebra Behind Google* by Kurt Bryan and Tanya Leise.

14.6 Summary

This extra note only discusses a simplified version of PageRank. There are a lot of follow-up questions that you can ask yourself:

1. What will happen to the stationary probability distribution if there are dangling nodes (nodes that do not contain any outgoing links)?
2. What will happen to the stationary probability distribution if there are disconnected nodes (nodes that do not contain any incoming or outgoing links)?
3. What will happen if there are billions of webpages?

The last question is actually a very realistic question: it is a challenge that is faced by Google. If you plan to simply calculate the stationary probability distribution by solving the linear system, the cost can be very expensive! Therefore, there are a lot of methods that build upon the PageRank algorithm to further optimize the speed (and also to yield more accurate importance scores based on other information). Anyway, PageRank is a very popular application of Markov chains. If you are interested in learning more about it, I highly recommend that you take EE 126 (Probability and Random Processes). You may also get a chance to implement the PageRank algorithm in an iPython notebook.

Extra Note 15

Beta Distribution

In this note, we introduce a specific family of continuous probability distributions known as the **Beta distribution** and study some of its fascinating properties.

15.1 Order Statistics

Suppose that $X_1, \ldots, X_n$ are i.i.d. random variables with common density $f_X(x)$ and CDF $F_X(x)$. Let $X^{(k)}$ be the kth smallest of $X_1, \ldots, X_n$, so that $X^{(1)}$ is the minimum of the points and $X^{(n)}$ is the maximum of the points. We can derive the density of $X^{(k)}$: recall that $f_{X^{(k)}}(x)$ has the interpretation $P(X^{(k)} \in (x, x + \mathrm{d}x)) \approx f_{X^{(k)}}(x) \cdot \mathrm{d}x$. In order for the kth smallest point to lie in the interval $(x, x + \mathrm{d}x)$, we must have the following:

1. $k - 1$ points must lie in the interval $(-\infty, x)$, which has probability $(F_X(x))^{k-1}$.
2. One point must lie in the interval $(x, x + \mathrm{d}x)$, which has probability $f_X(x)\,\mathrm{d}x$.
3. $n - k$ points must lie in the interval $(x + \mathrm{d}x, \infty)$, which has probability $(1 - F_X(x))^{n-k}$.

In addition, there are n ways to choose which of the n points lies in the interval $(x, x + \mathrm{d}x)$, and out of the remaining $n - 1$ points, there are $\binom{n-1}{k-1}$ ways to choose which points lie in the interval $(-\infty, x)$. Hence, we have

$$f_{X^{(k)}}\,\mathrm{d}x = P(X^{(k)} \in (x, x + \mathrm{d}x)) = n\binom{n-1}{k-1} f_X(x)(F_X(x))^{k-1}(1 - F_X(x))^{n-k} \cdot \mathrm{d}x$$

so our result is

$$f_{X^{(k)}} = n\binom{n-1}{k-1} f_X(x)(F_X(x))^{k-1}(1 - F_X(x))^{n-k} \tag{15.1}$$

Although (15.1) is a complicated result, we will only need the special case when the X_i are i.i.d. $U[0, 1]$ random variables. In this case, we have $f_X(x) = 1$ and $F_X(x) = x$ for $0 < x < 1$, so the density of $X^{(k)}$ is

$$f_{X^{(k)}}(x) = \frac{n!}{(k-1)!(n-k)!} x^{k-1}(1 - x)^{n-k}, \qquad 0 < x < 1 \tag{15.2}$$

15.2 Beta Distribution

We define the beta distribution by specifying the density function. Let $\beta(r, s)$ denote the beta distribution with parameters r and s (we will see the significance of the parameters soon). If $X \sim \beta(r, s)$, then the density of X is

$$f_X(x) = \frac{1}{B(r, s)} x^{r-1}(1-x)^{s-1}, \qquad 0 < x < 1 \tag{15.3}$$

where $B(r, s)$ is a normalizing constant. First, we observe that if $X^{(k)}$ is the kth smallest point out of n i.i.d. $U[0, 1]$ random variables (see the previous section), then $X^{(k)} \sim \beta(k, n-k+1)$. This provides an interpretation for the parameters of the $\beta(r, s)$ distribution: there are k points less than or equal to $X^{(k)}$, and $n - k + 1$ points greater than or equal to $X^{(k)}$. By setting $n = r + s - 1$ and $k = r$ in (15.2), we can obtain an expression for $B(r, s)$ for integer values of r and s:

$$\frac{1}{B(r, s)} = \frac{(r+s-1)!}{(r-1)!(s-1)!} \tag{15.4}$$

Remember that $B(r, s)$ is a normalizing constant for the density (15.3), which implies

$$B(r, s) = \int_0^1 x^{r-1}(1-x)^{s-1}\,\mathrm{d}x = \frac{(r-1)!(s-1)!}{(r+s-1)!} \tag{15.5}$$

Remembering (15.5) can actually be useful in solving integrals quickly.

We can solve for the moments of the beta distribution:

$$E[X] = \int_0^1 \frac{1}{B(r, s)} x^r(1-x)^{s-1}\,\mathrm{d}x = \frac{B(r+1, s)}{B(r, s)} = \frac{(r+s-1)!}{(r-1)!\cancel{(s-1)!}} \frac{r!\cancel{(s-1)!}}{(r+s)!} = \frac{r}{r+s}$$

$$E[X^2] = \int_0^1 \frac{1}{B(r, s)} x^{r+1}(1-x)^{s-1}\,\mathrm{d}x = \frac{B(r+2, s)}{B(r, s)} = \frac{(r+s-1)!}{(r-1)!\cancel{(s-1)!}} \frac{(r+1)!\cancel{(s-1)!}}{(r+s+1)!}$$

$$= \frac{r(r+1)}{(r+s)(r+s+1)}$$

so we have

$$\mathrm{var}(X) = \frac{r(r+1)}{(r+s)(r+s+1)} - \frac{r^2}{(r+s)^2} = \frac{rs}{(r+s)^2(r+s+1)}$$

15.3 Flipping Coins

Note that the beta distribution is supported on $(0, 1)$ so there is a natural interpretation of the beta distribution as a distribution over *probability values*. Suppose that we have a biased coin, but we do not know what the bias of the coin is. Assume that we have a *prior* distribution over the bias of the coin: $X \sim \beta(r, s)$. Since the expectation of the beta distribution is $r/(r+s)$, that means that we believe the bias of the coin is close to $r/(r+s)$, but we do

not have enough information to say the true bias with any certainty. We decide to flip the coin more times in order to get a better idea of the bias, and we obtain h heads and t tails. The question is: what is our posterior distribution of X, that is, how should we update our belief about the bias of the coin?

Of course, the answer is Bayes Rule, but with continuous random variables. To be more clear, let $X \sim \beta(r, s)$ be a random variable which represents our prior belief about the bias of the coin, and let A be the event that we flip h heads and t tails. What is the conditional distribution of X after observing A, $f_{X \mid A}(x)$?

Bayes Rule tells us that the answer is

$$P(X \in (x, x + \mathrm{d}x) \mid A) = \frac{P(X \in (x, x + \mathrm{d}x) \cap A)}{P(A)} = \frac{P(X \in (x, x + \mathrm{d}x))P(A \mid X = x)}{P(A)}$$

When we use the density interpretation, we have

$$f_{X \mid A}(x)\,\mathrm{d}x = \frac{f_X(x)P(A \mid X = x)\,\mathrm{d}x}{P(A)}$$

which gives the equation

$$f_{X \mid A}(x) = \frac{f_X(x)P(A \mid X = x)}{P(A)}$$

We know that $f_X(x) = (B(r, s))^{-1}x^{r-1}(1 - x)^{s-1}$ for $0 < x < 1$. $P(A \mid X = x)$ is the probability of flipping h heads and t tails if the bias of our coin is x, which is the binomial distribution with $h + t$ trials and probability of success x. Therefore,

$$P(A \mid X = x) = \frac{(h + t)!}{h!t!}x^h(1 - x)^t$$

To obtain $P(A)$, we should integrate $P(A \mid X = x)$ against the density of X:

$$\begin{aligned} P(A) &= \int_{-\infty}^{\infty} P(A \mid X = x)f_X(x)\,\mathrm{d}x = \int_0^1 \frac{(h + t)!}{h!t!}x^h(1 - x)^t \cdot \frac{1}{B(r, s)}x^{r-1}(1 - x)^{s-1}\,\mathrm{d}x \\ &= \frac{(h + t)!}{h!t!}\frac{1}{B(r, s)}\int_0^1 x^{r+h-1}(1 - x)^{s+t-1}\,\mathrm{d}x = \frac{(h + t)!}{h!t!}\frac{B(r + h, s + t)}{B(r, s)} \end{aligned}$$

Putting these pieces together, we obtain

$$\begin{aligned} f_{X \mid A}(x) &= \frac{1}{\cancel{B(r, s)}}x^{r-1}(1 - x)^{s-1} \cdot \frac{\cancel{(h + t)!}}{\cancel{h!t!}}x^h(1 - x)^t \cdot \frac{\cancel{h!t!}}{\cancel{(h + t)!}}\frac{\cancel{B(r, s)}}{B(r + h, s + t)} \\ &= \frac{1}{B(r + h, s + t)}x^{r+h-1}(1 - x)^{s+t-1}, \qquad 0 < x < 1 \end{aligned}$$

We have found that the posterior distribution has the $\beta(r + h, s + t)$ distribution! We now turn to the interpretation of this key result.

We stated that the expectation of the $\beta(r, s)$ distribution is $r/(r + s)$, which would be our best guess for the bias of the coin if we had observed r heads and s tails in $r + s$ coin flips. The result above states that if we observe h more heads and t more tails, then we now have the $\beta(r+h, s+t)$ distribution, which is consistent with the following interpretation: the first parameter represents how many heads we have seen, and the second parameter represents how many tails we have seen. The amazing part is that if we start with a belief according to the beta distribution, then we never have to use another distribution: repeated applications of Bayes Rule will always yield another beta distribution.

Furthermore, our result almost provides an algorithm for estimating the bias of a coin: start with a beta distribution, and keep flipping coins. If we observe heads, we increment the first parameter of the beta distribution. If we observe tails, we increment the second parameter of the beta distribution. Of course, this is an extremely cheap computation, which makes the beta distribution a convenient family of distributions for the estimation problem.

This leads to a minor problem, which is: how should we initialize the parameters of the beta distribution? Regardless of the choice of the initial parameters, after enough flips, the beta distribution will converge to the correct probability. It is true that our choice of initial parameters certainly influences how long it takes for our distribution to converge. Instead of worrying about this problem, however, we may view it as another advantage of the beta distribution: by initializing our initial parameters, we can incorporate our prior belief into the algorithm. Here are some examples to illustrate this point:

If we take an ordinary coin, we may be reasonably confident that the coin is fair, so we can start with a $\beta(500, 500)$ distribution. The fact that $r = s$ reflects the fact that we think the coin is fair, and our choice of the number 500 represents the strength of our belief: we belief strongly in the fairness of the coin, so we initialize the beta distribution with 500 observations of heads and 500 observations of tails. On the other hand, if the coin was given to you by a friend, perhaps you believe that your friend is not malicious (so you still believe that the coin is fair), but you do not trust the coin as much as you would an ordinary coin. In this case, you may decide to initialize the algorithm with a $\beta(20, 20)$ distribution. Finally, suppose that the coin was given to you by a gambler, and you suspect that the coin may be loaded (so that it is more likely to come up heads). In this case, you may encode your belief with a $\beta(40, 20)$ distribution (your suspicions amount to the same information as having observed 40 heads and 20 tails prior to flipping the coin).

In summary, the beta distribution is a useful class of continuous probability distributions that enjoys interesting properties. Perhaps you can think of other contexts in which the coin flipping interpretation of the beta distribution is of interest.