Kapak Konusu: Geometrik Kombinatorik

Kodlar

Sibel Özkan* / *ozkansi@auburn.edu*

Diyelim ki çok uzaklarda bir yerdesiniz ve Türkiye'den sorulan bir soruya "evet" ya da "hayır" yanıtlarından birini vermek istiyorsunuz. Örneğin uzaktan ÖSS sınavına giriyor olabilirsiniz, ya da daha dramatik bir örnek: Biri size "benimle evlenir misin?" diye bir mesaj atmış olabilir.

Diyelim, evet mesajı • olarak, hayır mesajı da •• olarak kodlanıp yollanıyor ve diyelim evlenme teklifini reddetmek istediniz. "Çok onur duydum, ancak..." anlamına "hayır" düğmesine bastınız. •• şifreli mesajı yola çıktı... Mesaj, dağları, ovaları, engin okyanusları aşacak, tayfunları, boraları delip geçecek, doluya tipiye tutulacak, güneş ışığına, meteor yağmurlarına, radyo dalgalarına maruz kalacak ve ta Türkiye'ye gidecek... Yolda mesajın başına bir kaza gelmesi muhtemel. Kazayla noktalardan biri silinse örneğin, taliplinize "hayır ••" yerine "evet •" mesajı gider. Ya da teklifi kabul edip sevinç gözyaşları içinde "evet" mesajı yolladınız ama yolda şeytan doldurdu ve mesajınıza yeni bir nokta eklendi, yolladığınız • mesajı yerine •• olarak vardı... Trajikomik bir durum!

Kodlar kuramı oldukça yeni bir dal sayılır. Claude Shannon'ın 1948'de yazdığı bir makale [Sha], elektronik mühendisliğinde bilişim (enformasyon) kuramı adı verilen dalın ve bu dalın altında hata düzelten kodlar alanının doğmasını sağlamıştır.

Oysa evet mesajı • olarak, hayır mesajı •••• olarak gönderilseydi, bir noktanın eklenmesi ya da silinmesi halinde, mesaj onarılıp gene doğru yanıt algılanabilirdi.

Hayır mesajının ••• olarak yollanması olası yanlışın giderilmesine yaramaz. Çünkü bu hayır (•••) mesajından bir noktanın silinmesiyle evet (•) mesajına bir nokta eklenmesi aynı sonucu verir (••). Kazayla •• mesajını alan aygıt bu anlamsız mesajın evet mi yoksa hayır mı olarak algılanması gerektiğini anlayamaz, ikilemde kalır.

Hayır mesajını •••••• olarak yollamak daha da güvencelidir. Böylece iki noktanın eklenmesi ya da iki noktanın silinmesine karşı da önlemimizi almış oluruz. Ancak mesaj uzun olduğunda yerine ulaşması zaman alır ve düğün dernek olana kadar müstakbel eşler karta kaçabilir.

Yollanan sinyalin uzunluğunu artırdığından, kazalara karşı alınan önlemler yollanan sinyalin yerine varış süresini uzatır. Doğrulukla hız arasında makul bir karar almalıyız.

Yukarda yollanmak istenen mesajı kodladık: evet yerine •, hayır yerine •••• mesajlarını yolladık. Alfabemizde bir tek harf vardı: •. Bu harfle • ve •••• sözcüklerini yazdık. Uygulamada yukardaki gibi bir değil, birden çok simge kullanılır, daha çok iki simgeli (harfli) bir alfabe kabul edilir: {0, 1}.

Anlamlı sözcükler bu alfabeyle yazılan sözcüklere dönüştürülür ve 0100110 gibi kodlanmış mesajlar yollanır.

Yukardaki örnekte olduğu gibi yolda mesaja yeni simgeler eklenmez, ancak mesajın simgelerinden biri değişebilir. Örneğin, yollanan 0100110 mesajı, yerine 0101110 olarak ulaşabilir.

Evet/hayır örneğimize geri dönelim: Eğer "evet"i 01 olarak, "hayır"ı da 10 olarak yollarsak ve Türkiye'ye yanlışlıkla 11 mesajı ulaşırsa, mesajı alan kişi ya da aygıt bu 11'in 01 mi yoksa 10 mı olarak algılanması gerektiğini algılayamaz.

"Evet"i 11 olarak, "hayır"ı da 00 olarak yollasanız da olası bir yanlışlık düzeltilemez.

Ancak, "evet"i 000 olarak, "hayır"ı da 111 olarak yollarsak, o zaman 000, 001, 010 ve 100 mesajları 000 olarak, 011, 101, 110 ve 111 mesajları da 111 olarak algılanır. Çünkü, örneğin, 001 mesajı 000'a 111'e olduğundan "daha yakındır".

Elbette, bu durumda bile yollanan 000 mesajı, yerine 011 olarak ulaşabilir, ama bunun olasılığı çok daha düşüktür. Daha güvenilir bir sistem istiyorsak, o zaman mesajları daha uzun sözcüklerle kodlamalıyız, örneğin evet ve hayır yerine 00000 ve 11111 şifrelerini seçmeliyiz.

* Auburn Üniversitesi Matematik Bölümü doktora öğrencisi. Bu yazı için [AK], [And], [CV] ve [Han]'dan yararlanılmıştır.

Kodlar Kuramı. Dediğimiz gibi, kod oluşturmak için kod sözcüklerine, kod sözcüklerini yazabilmek için de bir ***alfabe***ye ihtiyacımız var. Alfabe olarak q farklı simgeyi içeren sonlu bir küme kullanabiliriz. Uygulamada genellikle $q = 2$ alınır ve bu durumda alfabe {0, 1} olarak kabul edilir.

Seçtiğimiz alfabeyle yazacağımız her sözcüğe ***kod sözcüğü***, bu sözcüklerden oluşan kümeye de ***kod*** denir. Örneğin, tüm anlamlı Türkçe sözcükler kümesi, 29 harfli bir alfabesi olan bir kod olarak, ya da Türkiye'deki tüm telefon numaraları {0, 1, 2, ..., 9} alfabesini kullanan 10 uzunluğunda (alan kodları dahil) bir blok kod örneği olarak düşünülebilir. Biz tüm sözcüklerin uzunluklarının aynı alındığı ve kullanımı en yaygın olan ***blok kodları***nı inceleyeceğiz.

{0, 1} alfabesiyle n uzunluğunda 2^n tane sözcük yazabiliriz. $n = 0, 1, 2, 3$ ve 4 için bunları aşağıda göreceksiniz. Sözcüklerin nasıl "alfabetik sırayla" yazıldıklarına dikkatinizi çekeriz. 0 uzunluğunda tek sözcük vardır: Hiç simgesi olmayan ***boş sözcük***: $\langle\rangle$.

$n = 0$	$n = 1$	$n = 2$	$n = 3$	$n = 4$	
$\langle\rangle$	0	00	000	0000	1000
	1	01	001	0001	1001
		10	010	0010	1010
		11	011	0011	1011
			100	0100	1100
			101	0101	1101
			110	0110	1110
			111	0111	1111

Bir kod seçmek için önce sözcüklerimizin uzunluğunu seçmeliyiz, kodlanan bütün sözcüklerin ***uzunluğu*** aynı olacak.

Kodlamak istediğimiz sözcük sayısı arttıkça, uzunluk da artar elbet, çünkü sözgelimi, {0, 1} alfabesiyle 4 uzunluğunda sadece 16 sözcük yazabiliriz. Örneğin Türkçe alfabeyi {0, 1}'le kodlayabilmek için uzunluğun en az 5 olması gerekir.

Ama olası yanlışları düzeltebilen bir kod yazmak için uzunluğu daha da artırmamız gerekir. Sözgelimi, sadece evet ve hayır'ı kodlamak için uzunluğun 1 olması yeter; ama bir hata düzeltebilen bir kod yazmak istiyorsak, uzunluk en az 3 olmalı; iki hata düzelten bir kod yazmak içinse uzunluk en az 5 olmalı.

Bir ***ikili kod***, belli bir n için, 0 ve 1'lerden oluşmuş n uzunluğunda bir diziler kümesidir. Örneğin,

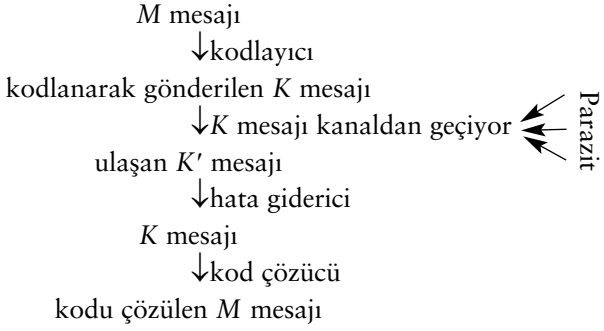
$$C = \{000, 111\}$$

bir ikili koddur. Burada $n = 3$. C'nin elemanlarına ***kod sözcükleri*** denir.

Kodlar kuramı, bilginin bir yerden bir yere doğru ve etkin biçimde iletilmesiyle ilgilenir. Günümüzde kompakt disk kayıtlarında hatayı en aza indirmek, telefon hatları üzerinden ya da bir uydudan düzgün bilgi akışını sağlamak gibi çeşitli uygulamaları vardır. Bilgi akışı esnasında mesajlarda oluşabilecek hataları düzelten kodlarla tanışmamız 1940'lara dayanır.

Bilgi akışının gerçekleştiği ortama ***kanal*** denir; telefon hatları ya da atmosfer kanal örnekleridir. ***Parazit*** ya da ***gürültü*** adını verdiğimiz istenmeyen dış etkenler varan mesajın gönderilenden farklı olmasına neden olabilir.

Aşağıdaki şemada bilgi akışı gösterilmektedir. Yollamak istediğimiz mesaja M diyelim. Örneğin M, "evet" mesajı olabilir. Bu mesaj kodlanarak K kod sözcüğü haline getirilir. Sözgelimi, K, 000 olabilir. K, gideceği yere doğru kanalda yol alırken parazite maruz kalır ve ulaşacağı yere K yerine K' olarak ulaşır. K', 010 olabilir. Bu aşamada bir hata giderici devreye girer ve yanlış olan K' mesajını K kod sözcüğü olarak düzeltir. Hata giderici 010 mesajının 000'a 111'den daha yakın olduğunu anlar.



Kanalın en az derecede parazite maruz kalmasını sağlamak hataları önlemenin bir yoludur elbette, ama bu bir dereceye kadar mümkündür, ayrıca matematiğe değil mühendisliğe girer, bizim konumuz değil yani. Hataları düzeltmek ise matematiğin konusudur.

Hata düzeltici kodlamanın ardındaki düşünce, kod sözcüklerini birbirlerinden yeterince farklı seçip, gönderilen sözcüğün birkaç terimi parazit nedeniyle yanlış iletilse bile, iletilen sözcüğün hâlâ tanınabilir, yani asıl sözcüğe diğer bütün kod sözcüklerine olduğundan daha "yakın" olmasını sağlamaktır.

Bu yakınlığı ölçmek için bir uzaklık fonksiyonu kullanılır. x ve $y \in C$ ise, x ve y'nin aralarındaki ***Hamming uzaklığı*** $d(x, y)$, x'in y'den farklı olan koordinatlarının sayısıdır. Örneğin,

$d(000, 111) = 3$,
$d(0100101, 1101101) = 2$,
$d(01101010, 01101010) = 0$.

C kodunun ***minimum uzaklığı*** $d(C)$, C'nin içindeki bütün $x \neq y$ kod sözcükleri için $d(x, y)$'nin aldığı en küçük değeridir:

$$d(C) = \min\{d(x, y) : x, y \in C, x \neq y\}.$$

Örnek: C = {1101000, 0110100, 0011010, 0001101, 1000110, 0100011, 1010001} olsun. Her $x \neq y$ için kod sözcüğü için, $d(x, y) = 4$ olduğuna göre (denemeden inanmayın bence!) $d(C) = 4$'tür.

C kodunun tek bir simge hatasını düzeltmeye olanak verebilmesi için, herhangi iki kod sözcüğünün arasındaki uzaklık en az 3 olmalıdır, yani $d(C) \geq 3$ olmalıdır, bu çok bariz. C kodunun iki simge hatasını düzeltmeye olanak verebilmesi için de, $d(C) \geq 5$ olmalıdır. Genel olarak, C kodunun e simge hatasını düzeltmeye olanak verebilmesi için, $d(C) \geq 2e + 1$ olmalıdır. Bu durumda, C'ye ***e-hata düzelten kod*** denir, yani C en fazla e tane hata düzeltme özelliğine sahiptir.

Soru: C = {0101010, 1001100, 0011001, 1110000, 0100101, 1000011, 0010110} ise C'nin hata düzeltme kapasitesi nedir?

Kodları tanımlarken tasarımlarda yaptığımız gibi bazı parametreler kullanmak işimizi kolaylaştırır. Alfabesinde q harf olan, n uzunluğunda M tane kod sözcüğü içeren ve minimum uzaklığı d olan bir koda q-(n, M, d) kodu diyelim. Örneğin yukardaki sorudaki C bir 2-(7, 7, 4) kodudur.

Kodlar ve Geometri. Yukardaki C kodunun sözcüklerini bir matrisin satırları olarak yazalım:

$$\begin{matrix} 0\,1\,0\,1\,0\,1\,0 \\ 1\,0\,0\,1\,1\,0\,0 \\ 0\,0\,1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,1\,0\,1 \\ 1\,0\,0\,0\,0\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \end{matrix}$$

Sütunlara $l_1, l_2, ..., l_7$, sıralara $P_1, P_2, ..., P_7$ diyelim ve P ve l'leri birazdan tanımlayacağımız bir "geometri"nin noktaları ve doğruları (ya da tasarımcıların dilinde, blokları) olarak görelim. Eğer i-inci sırada ve j-inci sütunda 1 varsa $P_i \in l_j$ olsun, 0 varsa $P_i \notin l_j$ olsun. Böylece 7 nokta ve 7 doğrudan oluşan bir "geometri" elde etmiş oluruz. Bu geometrinin aynen sayfa 28'de resmini çizdiğimiz geometri olduğuna dikkatinizi çekerim.

Bunun tersini de yapabiliriz. Nokta ve doğrulardan oluşan sonlu bir "geometri" verilmiş olsun, örneğin nokta ve bloklardan oluşmuş bir (v, k, λ) tasarımı verilmiş olabilir. Bu geometrinin doğrularına $l_1, l_2, ..., l_b$, noktalarına $P_1, P_2, ..., P_v$ diyelim. Geometrinin ***oluşum matrisi***, (i, j) girdisi, $P_i \in l_j$ ya da $P_i \notin l_j$ durumuna göre 1 ya da 0 olan $v \times b$ boyutlu matristir. Yani oluşum matrisinin i-inci sıradaki ve j-inci sütundaki girdisi, eğer $P_i \in l_j$ ise 1, $P_i \notin l_j$ ise 0'dır. Oluşum matrisinin satırlarını bir kodun kod sözcükleri olarak görebiliriz.

Bu fikirden birazdan yararlanacağız.

Kodlar ve Simetrik Tasarımlar. Nokta ve doğrulardan oluşan bir geometriden bir kodun nasıl elde edilebileceğini gördük. Şimdi herhangi bir geometri alacağımıza bir simetrik tasarım alalım (bkz. sayfa 31) ve bu tasarımın yarattığı koda göz atalım. Bakalım böyle bir kodun bir özelliği var mı?

C'nin kod sözcükleri herhangi bir 2-(v, k, λ) simetrik tasarımının oluşum matrisinin satırları olsun. Blok sayısı b ise, tasarım simetrik olduğundan, $b = v$'dir. Dolayısıyla C, v uzunluğunda bir koddur.

Her blokta k tane nokta olduğuna göre, bu matrisin her satırında, yani her kod sözcüğümüzde, k tane 1 ve $v - k$ tane 0 vardır. Ayrıca, her blok çifti λ tane ortak eleman içerdiğinden, herhangi iki kod sözcüğünün aynı λ koordinatında 1 vardır. Dolayısıyla ilk sözcükte geri kalan $k - \lambda$ tane 1 diğer sözcüktekinden farklı yerlerdedir, ikinci sözcük

Mars'tan Mesaj Var!

Mars'ın yörüngesine oturtulan ilk uzay aracı olan Mariner 9'dan fotoğraf alınmasında yararlanılan kod, yazının sonunda göreceğimiz Reed-Muller kodlarından biridir. Bu kod çok hızlı bir algoritmayla çözüldüğünden Mariner 9'dan saniyede 16.200 bit veri aktarımı gerçekleştirilmiş, Mars'tan fotoğraf almayı ilk başaran Mariner 4'ün saniyede 25/3 bit veri aktarım hızının kat kat üstüne çıkılmıştır [Hil].

için de aynısı geçerlidir. Yani, herhangi iki kod sözcüğünün $2(k-\lambda)$ koordinatı birbirinden farklıdır. Bu da bize $d(C)$'nin $2(k-\lambda)$ olduğunu gösterir.

$2(k-\lambda) > 2(k-\lambda-1)+1$ olduğundan, simetrik tasarımımızın oluşum matisinden elde ettiğimiz C kodu, $(k-\lambda-1)$-hata düzelten bir kod olur.

Kodlar ve Latin Kareler. Birbirine dik latin karelerden de kod üretebiliriz. Hatta birbirine dik ne kadar fazla latin kare bulursak, elde ettiğimiz kod o kadar fazla hata düzeltir. Bir örnekle başlayalım. Sayfa 9'da bulduğumuz birbirine dik ve derecesi 5 olan dört latin kareden herhangi ikisini seçelim.

$$A = \begin{matrix}12345\\23451\\34512\\45123\\51234\end{matrix} \qquad B = \begin{matrix}12345\\34512\\51234\\23451\\45123\end{matrix}$$

olsun. A ve B'nin girdileri sırasıyla a_{ij} ve b_{ij} olsun.

$$C = \{(i, j, a_{ij}, b_{ij}) \mid i, j = 1, 2, 3, 4, 5\}$$

olsun. Böylece alfabesinde 5 harf olan 25 sözcüklü ve 4 uzunluğunda bir kod buluruz. 1111, 3241, 4425 bu kodun kod sözcüklerinden bazılarıdır.

Teorem 1. *q-$(4, q^2, 3)$ kodunun olması için gerek ve yeter koşul, q dereceden birbirine dik iki latin karenin olmasıdır.*

Kanıt: Önce, q dereceden birbirine dik $A = (a_{ij})_{ij}$ ve $B = (b_{ij})_{ij}$ latin karelerinin olduğunu varsayalım. Üstte yaptığımız gibi

$$C = \{(i, j, a_{ij}, b_{ij}) \mid i, j = 1, 2, ..., q\}$$

kodunu tanımlayalım. Bu kodun uzunluğunun 4 olduğu ve q^2 tane sözcük içerdiği bariz. İlk iki koordinat son iki koordinatı belirlediğinden minimum uzaklık en az 1'dir. Öte yandan, iki latin karenin birbirine dik olmasından dolayı, son iki koordinat da ilk iki koordinatı belirler. Dolayısıyla minimum uzaklık en az 2'dir. Şimdi minimum uzaklığın en az 3 olduğunu gösterelim. İki değişik (i, j, a_{ij}, b_{ij}) ve (k, l, a_{kl}, b_{kl}) kod sözcükleri alalım. Eğer $i \neq k$ ve $j \neq l$ ise uzaklık en az 3'tür. Eğer $i = k$ ya da $j = l$ ise, yani ya sıra ya da sütun aynıysa o zaman hem $a_{ij} \neq a_{kl}$ olur hem de $b_{ij} \neq b_{kl}$, yani minimum uzaklık 3'tür.

Şimdi de q-$(4, q^2, 3)$ kodunun olduğunu varsayalım. Alfabemizin 1, 2, ..., q sayılarından oluştuğunu varsayabiliriz. Şimdi yukardaki inşayı ters çevirerek üçüncü ve dördüncü koordinatlardan birbirine dik iki latin kare elde etmek işten bile değildir. □

Böylece örneğimizdeki kodun minimum uzaklığının 3 olduğunu yani bir hata düzeltebildiğini görmüş olduk. Dik latin karelerle daha fazla hata düzeltebilen kodlar üretebiliriz:

Teorem 2. *q-$(n, q^2, n-1)$ parametreli bir kodun olması için yeter ve gerek koşul, q dereceli $n-2$ tane birbirine dik latin kare olmasıdır.*

Bunu $n = 4$ iken kanıtladık. Bu daha genel teoremin kanıtında da aynı yol izlenir. Ayrıntıları okura bırakıyoruz.

Ne yazık ki q dereceli $n-2$ tane birbirine dik latin karenin her zaman bulunup bulunmayacağını bilmiyoruz, dolayısıyla yukardaki teoremin uygulama alanı bulunan dik kare sayısıyla sınırlı.

Lineer Kodlar. Kod sözcüklerinin genellikle iki harfli {0, 1} alfabesiyle yazıldığını söyledik. Bundan böyle $\mathsf{F}_2 = \{0, 1\}$ olsun. F_2'nin elemanlarını oldukça doğal biçimde toplayabiliriz:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 0$$

Bunun bildiğimiz toplamadan tek farkı $1 + 1 = 0$ eşitliği. Buna modülo 2 toplama dendiğini birçok okur biliyordur herhalde. Dikkat ederseniz F_2'nin iki elemanının toplamı ancak ve ancak iki eleman birbirine eşitse 0'dır, yoksa 1'dir. Bu basit olgu, Teorem 3'ün kanıtında önemli olacak.

Eğer kodun uzunluğu n ise, her kod sözcüğü 011001100 gibi n uzunluğunda bir 0-1 dizisidir. n uzunluğundaki 0-1 diziler kümesini F_2^n olarak gösterelim. Demek ki uzunluğu n olan bir C kodu, F_2^n kümesinin bir altkümesidir.

n uzunluğundaki dizileri, yani F_2^n'nin elemanlarını F_2'nin toplamasıyla uyumlu biçimde toplayabiliriz: İki diziyi toplamak için, dizilerin terimlerini teker teker toplarız. Sözgelimi,

$$\begin{array}{r} 1001110110 \\ +\ 1100010101 \\ \hline 0101100011 \end{array}$$

Eğer $C \subseteq \mathsf{F}_2^n$ bir kodsa, iki kod sözcüğünün toplamının gene bir kod sözcüğü olması için hiçbir neden yoktur, bazı $x, y \in C$ için, $x + y$ dizisi C'de olmayabilir.

Eğer her $x, y \in C$ için, $x + y \in C$ ise, yani C toplama altında kapalıysa C'ye ***lineer kod*** denir. Line-

er kodlar diğer kodlara göre daha sevimli kodlardır, çünkü çözülme algoritmaları daha çabuktur.

Her terimi 0 olan 00 ... 0 sıfır dizisi her lineer kodda bulunmak zorundadır, çünkü eğer $x \in C$ ise, $x + x$ dizisi C'dedir ve her terimi 0'dır. (Burada bir kodun boşküme olamayacağını varsaydık, bundan sonra da hep bu varsayımda bulunacağız.)

00 ... 0 dizisini $\underline{0}$ olarak göstereceğiz.

Örnek. {000, 111} ve {000, 110, 101, 011} kodları lineerdir ama yedi sözcüklü {1101000, 0110100, 0011010, 0001101, 1000110, 0100011, 1010001} kodu lineer değildir, çünkü $\underline{0}$ = 0000000 dizisini içermez, ayrıca 1101000 ve 0110100 kodlarının toplamı olan 1011100 sözcüğü kodda değildir.

Alıştırma. C, içinde çift sayıda 1'in olduğu n uzunluğundaki 0-1 dizileri olsun. C'nin lineer bir kod olduğunu kanıtlayın. C'nin kaç elemanı vardır?

Lineer kodların diğer kodlara olan avantajını göstermek için uzaklıkla yakından bağlantılı olan ağırlık kavramını tanımlayalım: Bir sözcüğün ***ağırlığı***, sözcüğün içindeki 0'dan farklı terimlerin sayısıdır. Biz ikili kodlardan sözettiğimize göre, bunu, sözcükteki 1'lerin sayısı olarak alabiliriz. Bir x sözcüğünün ağırlığı $w(x)$ olarak gösterilir. Örneğin $w(1101101) = 5$. Aşağıdaki kolay teorem ağırlıkla uzaklığın ilişkisini gösterir.

Teorem 3. *Her x, y kod sözcükleri için,*

$$w(x) = d(x, \underline{0})$$

ve

$$d(x, y) = w(x + y).$$

Kanıt: Toplamanın tanımından doğrudan çıkar. Ayrıntılar okura bırakılmıştır. □

Sonuç 4. *C lineer bir kod, $w(C)$ de C'nin sözcüklerinin ağırlığının en küçüğü olsun. O zaman,*

$$d(C) = w(C).$$

Kanıt: C'nin lineer kod oluşundan, $d(C)$'nin tanımından ve Teorem 3'ten çıkar. □

M sözcük içeren bir kodda minimum uzaklığı bulabilmek için, tüm sözcükleri karşılaştırmamız gerektiğinden, sözcükler arasında

$$\binom{M}{2} = \frac{M(M-1)}{2}$$

tane karşılaştırma yapmamız gerekir. Oysa M sözcüklü bir **lineer** kodda M tane sözcüğün ağırlıklarına hesaplamak kodun uzaklığını bulmada yeterli olacaktır. Örneğin, C = {000000, 101000, 100010, 001010, 000011, 100001, 001001, 101011} lineer kodunun sekiz sözcüğünün birbirlerine olan uzaklığını ölçmek için 8×7/2 = 28 hesap yapmalıyız, oysa ağırlıklarını 8 basit işlemle hesaplayabiliriz; sonuç, 101011 kod sözcüğünden dolayı, 4 çıkar.

Doğuraylar ve Taban. Lineer kodların daha birçok avantajı vardır, ama biz şimdilik bu kadarıyla yetinip lineer kod elde etme yöntemlerini görelim.

Örnek 1. $w_1, w_2 \in \mathsf{F}_2^n$ olsun. O zaman,

$$\{0, w_1, w_2, w_1 + w_2\}$$

lineer bir koddur. Bu kod $\langle w_1, w_2 \rangle$ olarak gösterilir. Ama dikkat! Bu kodda illa dört sözcük olmayabilir. Örneğin w_1, w_2'ye eşitse kodda en fazla iki sözcük vardır. Eğer $w_1 = w_2 = 0$ ise o zaman kodda sadece bir sözcük vardır.

Örnek 2. $w_1, w_2, w_3 \in \mathsf{F}_2^n$ ise,

$$\{0, w_1, w_2, w_3, w_1 + w_2, w_1 + w_3, w_2 + w_3, w_1 + w_2 + w_3\}$$

lineer bir koddur ve $\langle w_1, w_2, w_3 \rangle$ olarak gösterilir. Ama gene dikkat! Bu kodda illa 8 sözcük olmayabilir. Örneğin $w_1 + w_2 + w_3$ sözcüğü 0'a eşit olabilir, bu taktirde $w_3 = w_1 + w_2$ olur ve kodumuz bir önceki örnekteki $\langle w_1, w_2 \rangle$ koduna eşit olur.

Örnek 3. $w_1, w_2, ..., w_k \in \mathsf{F}_2^n$ olsun. Bu k sözcüğü olabilecek her biçimde birbirleriyle toplayalım. $0, w_1, w_2, ..., w_k, w_1 + w_2, w_1 + w_3, w_2 + w_3, ..., w_1 + w_2 + w_3, ..., w_1 + w_2 + ... + w_k$, gibi sözcükler elde ederiz. Çok bariz bir biçimde bu küme toplama altında kapalıdır, yani böylece bir lineer kod elde ederiz. Bu lineer kod $\langle w_1, w_2, ..., w_k \rangle$ olarak gösterilir ve $w_1, w_2, ..., w_k$ sözcüklerine bu kodun ***doğuray***ları adı verilir. Aynı kodun birçok değişik doğurayı olabileceğini de belirtelim.

Bu kodda kaç kod sözcüğü vardır? Tümevarımla bunlardan en fazla 2^k tane olduğu kolaylıkla gösterilebilir.

Eğer $w_1, w_2, ..., w_k$ doğuraylarının bir grubunun toplamı bir başka grubun toplamına eşitse, o zaman 2^k'dan daha az sözcük elde ederiz elbette. Bu durumda doğuraylardan biri diğerlerinin toplamı olarak elde edilir. Örneğin,

$$w_1 + w_2 + w_3 + w_6 = w_1 + w_3 + w_4 + w_5 + w_7$$

ise, sadeleştirerek, önce,

$$w_2 + w_6 = w_4 + w_5 + w_7$$

elde ederiz, sonra bu beş sözcükten herhangi birini diğerlerinin toplamı olarak yazabiliriz. Örneğin,

$$w_7 = w_2 + w_4 + w_5 + w_6;$$

dolayısıyla w_7'yi doğurayların arasından atarsak sözcük kaybına uğramayız.

Yukardaki örnekteki gibi hareket ederek, gereksiz doğurayları teker teker atabiliriz. Geriye, her biri gerekli olan doğuraylar kalır. Örneğin doğuraylar olarak şu sözcükleri alalım:

w_1 = 11010 w_5 = 10001
w_2 = 01101 w_6 = 01000
w_3 = 00110 w_7 = 10100
w_4 = 00011

Burada

$$w_5 = w_1 + w_2 + w_3$$

olduğundan, w_5'i doğuraylardan atabiliriz. Ama

$$w_6 = w_2 + w_3 + w_4$$

olduğundan w_6'yı da atabiliriz. Ayrıca

$$w_7 = w_1 + w_2 + w_4$$

olduğundan w_7'yi de atabiliriz. Geriye sadece

w_1 = 11010
w_2 = 01101
w_3 = 00110
w_4 = 00011

kalır ve artık bunlardan birini atamayız, yoksa sözcük kaybederiz. Bu dört sözcüğe kodun ***tabanı*** adı verilir. Bundan da bu kodda $2^4 = 16$ tane sözcük olduğu anlaşılır. Kodumuzdaki diğer bütün sözcükler bu tabanın "lineer kombinasyonu"dur ve tabanın hiçbir kombinasyonu gereksiz değildir.

Bir kodun birden çok tabanı olabilir.

Alıştırmalar

1. Doğurayları 4 ağırlıklı sözcükler olan n uzunluğundaki lineer kodda kaç eleman vardır?

2. Doğurayları 3 ağırlıklı sözcükler olan n uzunluğundaki lineer kodda kaç eleman vardır?

Herhangi bir kodu tanımlamak için içindeki tüm sözcükleri yazmamız gerekebilirken, lineer bir kodu tanımlarken k sözcükten oluşan tabanı vermek yeterlidir. Örneğin, yukardaki kodun 16 elemanını sıralamak yerine, tabanının 4 elemanını 4×5'lik bir matris olarak sıraya dizebiliriz:

11010
01101
00110
00011

Bu da lineer kodların bir başka avantajıdır.

Golay Kodu. Marcel J.E. Golay, 1949'da kodlar kuramının en ilgi çekici makalelerinden birini yayımlamıştır [Gol]. Tek sayfalık bu makalede Golay şimdi açıklayacağımız kodları tanıtmıştır.

(11, 6, 3)-simetrik tasarımının oluşum matrisinin devriğine M diyelim. M'yi sayfa 31'deki gri alanda görmüştük, 11×11 boyutunda bir matristi. Şimdi aşağıdaki matrise bakalım.

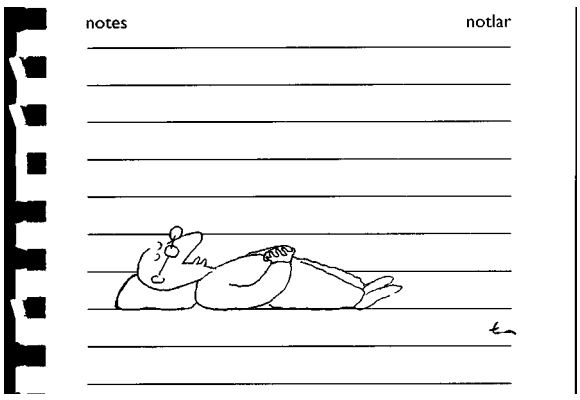
$$G = \begin{array}{|c|c|} \hline \mathrm{Id}_{12} & M \\ \hline & 1\ 1\ \ldots\ 1 \\ \hline \end{array}$$

Yani, G, 12×23 boyutlu şu matris olsun.

$$\begin{array}{cc}
1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1 \\
0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0 \\
0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,1\,0\,1\,0\,0\,0\,1\,1\,1 \\
0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,1 \\
0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 & 1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,1 \\
0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0 & 1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,1\,1\,1\,0\,1\,1\,0\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0 & 1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,0\,1\,1\,1\,0\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1 & 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1
\end{array}$$

Bu matrisin 12 sırasını 23 uzunluğunda bir kodun doğurayları olarak düşünelim. Böylece en fazla 2^{12} elemanlı bir kod elde ederiz. Bu kodun tam 2^{12} tane elemanı olduğunu göstermek zor değildir, yani sıralar kodun bir tabanını oluştururlar. Bu koda ***Golay kodu*** denir.

Golay kodunun minimum uzaklığı 7'dir, dolayısıyla 3 hata düzeltir.



Reed-Muller Kodları

Bilgi taşıma ve hata düzeltme kapasitesi bakımından çok yararlı bu ikili lineer kodları 1954'te D.E. Muller bulmuştur [Mul]. Aynı yıl I.S. Reed bu kodları çözme algoritmalarını geliştirmiştir [Ree].

Bir m doğal sayısı verilmiş olsun. $\mathsf{F}_2[x_1, ..., x_m]$ polinom halkasını ele alalım. Yalnız bu halkada her $i = 1, ..., m$ için, $x_i^2 = x_i$ kuralının geçerli olduğunu varsayalım. O zaman örneğin, $x_i^3 = x_i^2 = x_i$ olur, hatta her $m > 0$ doğal sayısı için $x_i^m = x_i$ olur. Birkaç örnek daha:

$$(x_1 + x_2)^2 = x_1^2 + 2x_1x_2 + x_2^2 = x_1 + x_2,$$
$$(x_1 + x_2)^3 = x_1^3 + 3x_1^2x_2 + 3x_1x_2^2 + x_2^2$$
$$= x_1 + 6x_1x_2 + x_2 = x_1 + x_2,$$
$$(x_1 + x_2)(x_1 + x_3) = x_1 + x_1x_2 + x_1x_3 + x_2x_3.$$

Böylece yeni bir halka elde ederiz. Bu halka polinom halkasından değişiktir, çünkü polinom halkasında $x_i^2 = x_i$ gibi bir kural geçerli değildir. Bu yeni halkaya R_m adını verelim.

R_m'nin her elemanını F_2^m kartezyen çarpımından F_2'ye giden bir fonksiyon olarak görebiliriz; nitekim her $f(x_1, ..., x_m)$ polinomu, F_2^m'nin $(a_1, ..., a_m)$ elemanını F_2'nin $f(a_1, ..., a_m)$ elemanına götüren fonksiyonu doğurur. Her a_i ya 0 ya da 1 olduğundan, $a_i^2 = a_i$'dir ve x_i^2'nin doğurduğu fonksiyon x_i'nin doğurduğu fonksiyona eşittir. Bunun tersi de doğrudur: F_2^m'den F_2'ye giden her fonksiyon R_m'nin bir elemanı tarafından yukardaki kuralla verilir. Bu son dediğimizi kanıtlamayacağız, ama kanıtı çok zor değildir. Sadece bir örnek verelim: F_2^m'nin $(1, 0, ..., 0)$ elemanını 1'e, diğer elemanlarını 0'a götüren fonksiyon, R_m'nin

$$x_1(x_2 + 1) ... (x_m + 1)$$

elemanı tarafından verilir.

F_2^m'den F_2'ye giden bir fonksiyon F_2^m'nin 2^m tane elemanında aldığı değer tarafından belirlendiğinden, F_2^m'den F_2'ye giden fonksiyonları 2^m uzunluğunda diziler olarak görebiliriz.

Sonuç olarak R_m kümesiyle uzunluğu 2^m olan 0-1 dizileri arasında küme olarak bir fark yoktur. Ayrıca R_m kümesindeki fonksiyon toplaması işlemi fonksiyonlarla belirlenen dizilerin toplamasına tekabül eder. Yani R_m kümesiyle uzunluğu 2^m olan 0-1 dizileri arasında toplama açısından da bir fark yoktur.

Şimdi $0 \le r < m$ olsun ve R_m'de derecesi en fazla r olan elemanlara bakalım. Bu elemanlar, $i_1 + ... + i_m \le r$ için, R_m'nin $x_1^{i_1}x_2^{i_2} ... x_m^{i_m}$ türünden yazılan "monom"larının toplamıdır. İşte $RM(r, m)$ ***Reed-Muller kodu***, R_m'nin bu tür elemanlarının kümesidir ve yukarda da açıklamaya çalıştığımız gibi uzunluğu 2^m olan bazı 0-1 dizilerinden oluşur ve lineerdir.

Alıştırma. $RM(r, m)$ kodunun,

$$\binom{m}{0} + \binom{m}{1} + ... + \binom{m}{r}$$

elemandan oluşan bir tabanı olduğunu ve uzaklığının 2^{m-r} olduğunu kanıtlayın. Bu kod kaç hata düzeltir?

Örnek 1. RM(2, 3) kodunun uzunluğu $2^3 = 8$'dir. Yukardaki alıştırmaya göre $1 + 3 + 3 = 7$ elemanlı bir tabanı vardır, dolayısıyla toplam kod sözcüğü sayısı $2^7 = 128$'dir. Uzaklığı ise $2^{3-2} = 2$'dir. Hiç hata düzeltemez.

Örnek 2. Şimdi RM(1, 5) koduna bakalım. Uzunluğu $2^5 = 32$. Tabanında $1 + 5 = 6$ eleman var. Demek ki $2^6 = 64$ tane kod sözcüğü var. Uzaklığı $2^{5-1} = 16$. Demek ki 7 hata düzeltebiliyor, çok iyi!

Bu, Mariner 9'dan fotoğraf alımında kullanılan 2-(32, 64, 16)-kodudur.

RM(1, 5) kodunun tabanının elemanlarını teker teker yazabiliriz: 1, x_1, x_2, x_3, x_4, x_5. Eğer F_2^5'in 16 elemanını 00000'dan 11111'e kadar küçükten büyüğe doğru sıralarsak, bu 1, x_1, x_2, x_3, x_4, x_5 fonksiyonları, F_2^5'in 16 elemanında aldıkları değerlere göre sırasıyla,

f_0 = 1111111111111111111111111111111
f_1 = 0000000000000000111111111111111
f_2 = 0000000011111111000000011111111
f_3 = 0000111100001111000011110000111
f_4 = 0011001100110011001100110011001
f_5 = 0101010101010101010101010101010

dizilerine tekabül ederler. Kodumuzun sözcükleri yukardaki 5 sözcüğün toplamlarından oluşur. Böylesine önemli bir görevde kullanılmış bir kodu gördüğünüz gibi biz bile kolaylıkla inşa edebildik. ♣