Tartalomjegyzék

Előszó

A digitális kommunikáció, adattárolás, -védelem és -feldolgozás tudományos alaposságú vizsgálata Claude Shannon 1948-as, A kommunikáció matematikai elmélete című cikkével vette kezdetét, így az információ- és kódelmélet tudományága alig fél évszázados múltra tekint vissza. Napjainkban azonban szinte észrevétlenül, ám mégis kikerülhetetlenül jelen van az élet és az infokommunikációs ipar tömegszolgáltatásainak számos területén. Lehetővé teszi a filmek és zeneművek korábban elképzelhetetlenül jó minőségű és gazdaságos tárolását DVD-n illetve CD-n, és az alkotások tetszőleges számú lejátszását minőségromlás nélkül. Ki álmodott volna a múlt század derekán arról, hogy egy egész estés mozifilm elférhet egy korongon, akár többféle szinkronnal együtt? Gondolhatunk az utóbbi években rohamosan elterjedt második és harmadik generációs mobiltelefonokra, vagy a közeljövőben hasonló karriert befutó intelligens chipkártyákra. Említhetnénk ezeken kívül számos más számítástechnikai és távközlési alkalmazást.

Ahhoz, hogy ezekben az információtovábbító illetve -tároló berendezésekben az információkezelés egyszerre legyen gazdaságos és biztonságos, szükség van olyan informatikusokra, akik a különböző kódolási technikákat és algoritmusokat készségszinten tudják használni.

Az informatikusok és villamosmérnökök oktatásában éppen ezért régóta lényeges szerepet kap ez a terület. A Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karán, a mérnök-informatikus egyetemi szakon a kezdetektől a tanterv részei az Információelmélet és a Kódelmélet tárgyak, öt éve pedig az Adatbiztonság tárgy is. Ezek a tárgyak a kétciklusú képzésben az eredeti célkitűzéssel az MSc szintre kerülnek.

A kétszintű mérnökképzés bevezetésével fel kell készülnünk ezen ismeretanyag alapjainak az eddigiektől célkitűzésében és módszertanában eltérő átadására. A BSc, vagyis az alapképzés jól használható gyakorlati ismeretek elsajátítására épít, míg az MSc, vagyis a mesterképzés feladata az elméleti háttér részletes bemutatása, amely alapján a mérnök képes az információtechnológia területén fejlesztési, tervezési illetve kutatási problémák megoldására. A BME mérnök-informatikus BSc képzésében önálló alaptárgyként jelenik meg a Kódolástechnika, melyet a hallgatók egy félévben, heti négy órában tanulnak. Jegyzetünk ehhez a tárgyhoz készült, de törekedtünk arra, hogy az elkészült mű más felsőoktatási intézmények informatikus alapképzéseibe is könnyűszerrel beilleszthető legyen. A jegyzetnek a mérnök-informatikus alapképzésen kívül jóval szélesebb a potenciá-

lis olvasóközönsége. Az egyetemi és főiskolai oktatásban haszonnal forgathatják más mérnökszakos vagy matematikus, alkalmazott matematikus szakos hallgatók is, de nem szabad megfeledkeznünk a végzett mérnökök szakmai önképzéséről vagy szervezett tanfolyamok keretében folyatott továbbképzéséről sem.

A korlátozott terjedelem és az oktatásra fordítható korlátos idő miatt nem lehet célunk a bevezető matematikai ismeretek ismételt elmondása, hiszen ezzel a hallgatók más alapozó tárgyak keretében ismerkednek meg (a BME mérnök-informatikus szakán ilyen tárgy például a Bevezetés a számításelméletbe vagy a Valószínűségszámítás). Ugyanakkor csak az elemi lineáris algebrai és a bevezető valószínűségszámítási tudásra építünk.

Célkitűzésünk, hogy ismertessük az információ átvitelének illetve tárolásának alapvető kódolási algoritmusait és ezek tulajdonságait. A szóba jövő információtechnológiai feladatokat négy csoportba szokás sorolni:

- hibajavító kódolás,
- adatbiztonság,
- adattömörítés és
- veszteséges forráskódolás.

Ehhez igazodnak a következő fejezetek, amelyekben a legfontosabbb technikák után alkalmazási példákat adunk (pl. CD, GSM, Internet, videotömörítés), majd gyakorló feladatok és azok megoldásai következnek. Végül a legfontosabb ismeretek összefoglalása és irodalomjegyzék zárja a fejezeteket.

Ez a jegyzet a Korszerű Mérnökért Alapítvány „Tankönyv, szakkönyv, jegyzet” projektjének keretében készült. Ezúton is szeretnénk köszönetet mondani az Alapítvány támogatásáért.

Budapest, 2006. december 18.

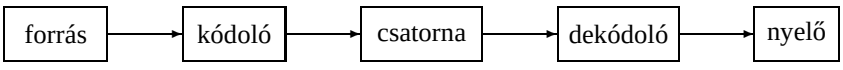
Buttyán Levente
Györfi László
Győri Sándor
Vajda István

1. fejezet

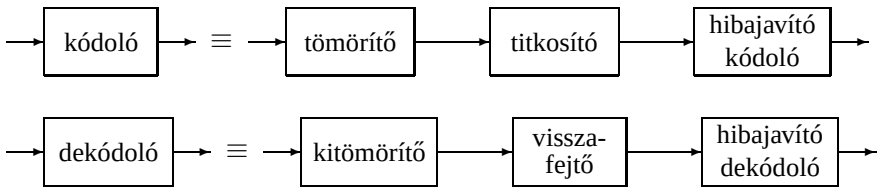
Bevezetés

A következő fejezetekben a hírközlési rendszerek működését egzakt matematikai eszközökkel fogjuk vizsgálni. Persze sokféle matematikai modellt állíthatunk hírközlési feladatokra, de kezdetben az egyik legegyszerűbbet választva jól kifejezhetjük a probléma lényegét. A hírközlés alapfeladata az, hogy valamely jelsorozatot („információt”) el kell juttatni egyik helyről a másikra (vagy tárolni kell). A távolságot (vagy időt) áthidaló hírközlési eszköz — a csatorna — azonban csak meghatározott típusú jeleket képes átvinni. Az információforrás által előállított jelfolyamot kódolással meg kell feleltetni egy, a csatorna által használt jelekből álló jelfolyamnak. A felhasználó (vevő, nyelő) a csatorna kimenetén pontosan, vagy megközelítőleg visszaállítja, dekódolja az üzenetet. Az 1.1. ábrán egy ilyen rendszer blokkdiagrammja látható.

A kódoló általános esetben három részből áll (1.2. ábra). Az első a forráskódoló vagy tömörítő, amelynek célja a forrás kimenetén jelenlévő felesleges ismétlődések, függőségek, vagyis az ún. redundancia eltávolítása. Ez a tömörítés akkor lehetséges, ha vagy az egyes betűk nem egyformán valószínűek, vagy az egymás után következő betűk nem függetlenek. Megkövetelhetjük, hogy a forrásdekódoló (kitömörítő) pontosan helyreállíthassa az adatokat, de megelégedhetünk részleges helyreállíthatósággal is. A forráskódoló tehát a forrás üzeneteit gazdaságosan, tömören reprezentálja. Ezzel foglalkozik a 4. és 5. fejezet. A kódoló második része a titkosító (3. fejezet), amely egyrészt az adatvédelmet garantálja, vagyis azt, hogy illetéktelenek ne férhessenek hozzá az üzenet tartalmához, másrészt pedig a hitelesítést oldja meg, vagyis annak bizonyítását, hogy az üzenet valóban a feladótól származik. A harmadik rész a csatornakódoló vagy hibajavító kódoló (2. fejezet), amelynek feladata a tömörítőével éppen ellentétes. Irányított módon visz be re-



1.1. ábra. Hírközlési rendszer blokkdiagramja



1.2. ábra. A kódoló és a dekódoló felépítése

dundanciát az adatokba úgy, hogy a csatorna átviteli hibái javíthatók legyenek. Ezeknek megfelelően az 1.2. ábrán látható dekódoló is három részből áll: csatornadekódoló (hibajavító dekódoló), visszafejtő és forrásdekódoló (kitömörítő).

A második és a harmadik rész feladatának érzékeltetésére tegyük fel, hogy egy ügyfél az interneten akar egy banki tranzakciót lebonyolítani. Ekkor nyilván elvárja, hogy a megadott adatok pontosan legyenek továbbítva (hibajavító kódolás), más személy ne tudja meg ezeket az adatokat még akkor sem, ha az információtovábbítás nyilvános hálózaton, például mobil eszközön történik (titkosítás), a bank számára pedig bizonyított legyen, hogy valóban ő kezdeményezte a tranzakciót (hitelesítés, digitális aláírás).

A forráskódoló és -dekódoló együttese foglalja magába a forráskódot, a csatornakódoló és -dekódoló együttese pedig a csatornakódot. A csatorna a kommunikációs rendszer tervezője számára adott, meg nem változtatható tulajdonságokkal rendelkező modell, amely leírja, hogyan zajlik az adatok átvitele vagy tárolása. A tervező ennek figyelembe vételével azonban szabadon megválaszthatja a forráskódot és a csatornakódot úgy, hogy ez minél jobb adattovábbítást eredményezzen.

2. fejezet

Hibajavító kódolás

A hibakorlátozó kódolás célja információ hibázó kommunikációs csatornákon történő megbízható átvitele illetve hibázó adattárolókon történő megbízható tárolása. E célból az információt kisebb egységekre bontjuk, majd redundáns információval bővítve kódszavakba képezzük, kódoljuk. A redundancia hozzáadása teszi lehetővé az információ védelmét hibázás esetén. A hibakorlátozó kódolás két fő technikája a hibajelzés illetve a hibajavítás. Hibajelzés esetén a cél annak eldöntése, hogy történt-e meghibásodás az információ továbbítása illetve tárolása során. Hibajavítás esetén ezen túlmenően a hibák kijavítása is feladat. Több kapcsolatos kérdés merül fel: hogy történjen a kódszavakba képezés, azaz a kódolás, ha célunk a minél erősebb hibakorlátozó képesség, azon feltétel mellett, hogy a rekonstrukció (dekódolás) a számításigényét tekintve hatékony maradjon.

Az alábbiakban a hibakorlátozó kódok konstrukciós és dekódolási alapelveit tekintjük át, a hangsúlyt a kapcsolatos fogalmakra és alapvető algoritmusokra helyezve.

2.1. Kódolási alapfogalmak

A hibajavító kódolás alapvető módszereit a 2.1. ábrán látható egyszerű hírközlési struktúra kapcsán vizsgáljuk.

Az $\mathbf{u}$ és $\mathbf{u}'$ vektorok koordinátái egy F halmazból veszik értékeiket, mely halmazt **forrásábécé**nek nevezzük. A kódoló a k hosszú $\mathbf{u}$ vektort (az **üzenet**et) egy n hosszú $\mathbf{c}$ vektorba (a **kódszó**ba) képezi le. A $\mathbf{c}$ koordinátái egy Q halmazból veszik értékeiket. A Q-t **kódábécé**nek vagy csatorna bemeneti ábécének fogjuk hívni.

2.1. ábra. Hírközlési rendszer blokkdiagramja

A csatorna kimenete **v**, szintén egy n hosszú vektor, melynek koordinátái szintén Q-beliek.

Egy

$$\mathbf{c} = (c_1, \ldots, c_n)$$

bemeneti és

$$\mathbf{v} = (v_1, \ldots, v_n)$$

kimeneti sorozat esetén azt mondjuk, hogy az m-edik időpontban a csatorna hibázott, ha $c_m \neq v_m$. Jelölje $d(\mathbf{c},\mathbf{v})$ azon i pozíciók számát, ahol $c_i \neq v_i$.

$d(\mathbf{c},\mathbf{v})$ neve a $\mathbf{c},\mathbf{v}$ sorozatok **Hamming-távolság**a, és azt mondjuk, hogy a **c** sorozat küldésekor és a **v** sorozat vételekor a hibák száma $t = d(\mathbf{c},\mathbf{v})$. Ezt az esetet nevezzük **egyszerű hibázás**nak, amikor a hiba helye és értéke egyaránt ismeretlen. $d(\mathbf{c},\mathbf{v})$ valóban távolság, hiszen

$$d(\mathbf{c},\mathbf{v}) \geq 0,$$

$$d(\mathbf{c},\mathbf{v}) = d(\mathbf{v},\mathbf{c}),$$

és igaz a háromszög-egyenlőtlenség:

$$d(\mathbf{c},\mathbf{v}) \leq d(\mathbf{c},\mathbf{w}) + d(\mathbf{w},\mathbf{v}).$$

Kód (blokk-kód) alatt a Q^n halmaz egy C részhalmazát értjük, azaz C minden eleme egy n hosszú vektor, melynek koordinátái Q-beliek. C elemeit kódszavaknak nevezzük. A továbbiakban a $C(n,k,d)$, illetve rövidítettebb formában $C(n,k)$ jelölést alkalmazzuk, kiemelve a kód paramétereit. A **kódolás** egy invertálható függvény, mely k hosszú F-beli sorozatot — üzenetet — képez le egy kódszóba, formalizálva:

$$f : F^k \to C,$$

és minden különböző **u**, **u**′-re $f(\mathbf{u})$, $f(\mathbf{u}')$ is különböző.

Dekódolás alatt két függvény egymásutánját értjük. Az egyik a csatorna kimenetének n hosszú szegmensét képezi le C-be, azaz igyekszik eltalálni a küldött kódszót, a másik pedig az f függvény inverze, tehát

$$g : Q^n \to C, \qquad f^{-1} : C \to F^k.$$

Mivel f egyértelműen meghatározza f^{-1}-et, ezért dekódolás alatt a későbbiekben csak a g függvényt értjük. A g dekódoló függvényként az algebrai hibajavító kódok elméletében speciális függvényt választunk, nevezetesen a **v** vektorhoz megkeressük azt a $\mathbf{c}' \in C$ kódszót, mely Hamming-távolság szerint hozzá a legközelebb van, vagy ha több ilyen van, akkor az egyiket, tehát teljesül, hogy ha $\mathbf{c}' = g(\mathbf{v})$, akkor

$$d(\mathbf{c}',\mathbf{v}) = \min_{\mathbf{c}\in C} d(\mathbf{c},\mathbf{v}).$$

A dekódolás feladata ezek után arra a messze nem triviális feladatra szűkül, hogy egy **v** vett szóhoz hogyan keressük meg a hozzá legközelebbi $\mathbf{c}'$ kódszót anélkül, hogy minden $d(\mathbf{c},\mathbf{v})$-t kiszámítanánk. Ha mégis kiszámítjuk ezeket a távolságokat, és minden **v**-hez megkeressük a hozzá legközelebbi **c** kódszót, majd a neki megfelelő üzenetet, akkor elvben azt eltárolhatjuk, és így egy táblázathoz jutunk, melynek címét **v** adja, tartalma pedig a **v**-nek megfelelő dekódolt üzenet. Ez a **táblázatos dekódolás**nak a legegyszerűbb, de legpazarlóbb esete, hiszen a táblázat q^n darab üzenetből áll, ahol q a Q kódábécé elemszáma.

2.1. példa (ismétléses kód). Tekintsük azt a nagyon egyszerű kódolást, amikor bináris forrásunk egyes bitjeit tekintjük üzenetnek, s háromszor megismételve küldjük a kommunikációs csatornába a következő leképezés szerint:

$$\begin{array}{lll} u & c_1\,c_2\,c_3 & \\ 0 \rightarrow & 0\;0\;0 & \mathbf{c}_1 \\ 1 \rightarrow & 1\;1\;1 & \mathbf{c}_2 \end{array}$$

A kód egy hibát képes javítani, mivel 1 hiba esetén a vett szó az átküldött kódszótól egy, míg a másik kódszótól kettő Hamming-távolságra van. A kód egy illetve kettő hibát képes jelezni, mivel ezen esetekben a vett szó nem lehet kódszó. A konstrukció általánosítható: legyen az ismétlések száma n, $n \geq 3$ páratlan szám. Könnyen látható, hogy a kód $(n-1)/2$ hibát képes javítani, valamint $n-1$ hibát jelezni.

2.2. példa (egyszerű paritáskód). Tekintsük azt a feladatot, amikor a forrás a következő négy lehetséges üzenetet bocsátja ki, a 00, 01, 10, 11 üzeneteket, amelyekhez a kódoló a következő, 3 hosszú kódszavakat rendeli hozzá:

$$\begin{array}{lll} u_1u_2 & c_1\,c_2\,c_3 & \\ 0\;0 \rightarrow & 0\;0\;0 & \mathbf{c}_1 \\ 0\;1 \rightarrow & 0\;1\;1 & \mathbf{c}_2 \\ 1\;0 \rightarrow & 1\;0\;1 & \mathbf{c}_3 \\ 1\;1 \rightarrow & 1\;1\;0 & \mathbf{c}_4 \end{array}$$

azaz az u_1u_2 bitet kiegészítjük egy paritásbittel. A kód egy hibát képes jelezni. A kód nem képes hibát javítani: pl. ha $\mathbf{c}_1$ kód továbbításakor a második bit meghibásodik, akkor a $\mathbf{v} = (0,1,0)$ vett szó azonos, egy távolságra lesz a $\mathbf{c}_1$ és $\mathbf{c}_2$ kódszavakhoz.

2.3. példa. A 2.2 példában szereplő üzenetekhez rendeljünk 5 hosszú kódszavakat:

$$\begin{array}{lll} u_1u_2 & c_1\,c_2\,c_3\,c_4\,c_5 & \\ 0\;0 \rightarrow & 0\;0\;0\;0\;0 & \mathbf{c}_1 \\ 0\;1 \rightarrow & 0\;1\;1\;0\;1 & \mathbf{c}_2 \\ 1\;0 \rightarrow & 1\;0\;1\;1\;0 & \mathbf{c}_3 \\ 1\;1 \rightarrow & 1\;1\;0\;1\;1 & \mathbf{c}_4 \end{array}$$

A g és az f^{-1} függvényt a következő táblázat foglalja össze:

$v_1\,v_2\,v_3\,v_4\,v_5$		$c'_1\,c'_2\,c'_3\,c'_4\,c'_5$		$u'_1\,u'_2$
0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1	→	0 0 0 0 0	→	0 0
0 1 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0	→	0 1 1 0 1	→	0 1
1 0 1 1 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 1	→	1 0 1 1 0	→	1 0
1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0	→	1 1 0 1 1	→	1 1
0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0	→	0 0 0 0 0	→	0 0
0 0 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 1 0 0	→	0 1 1 0 1	→	0 1

Az első 24 kimeneti szó dekódolásakor nincs probléma, hiszen minden 6 szóból álló csoport minden szava olyan, hogy vagy kódszó (az első helyen álló), vagy a kódszó egy bitjének megváltoztatásával (egy hibával) képződött. A 25–28. szavak mindegyike $\mathbf{c}_1$-től és $\mathbf{c}_4$-től 2, míg $\mathbf{c}_2$-től és $\mathbf{c}_3$-tól 3 távolságra van. Itt önkényesen

a dekódolás eredménye $\mathbf{c}_1$ (jobb érvünk nincs, mint az, hogy jobban szeretjük az almát, mint a cseresznyét). A 29–32. szavak mindegyike $\mathbf{c}_1$-től és $\mathbf{c}_4$-től 3, míg $\mathbf{c}_2$-től és $\mathbf{c}_3$-tól 2 távolságra van. Itt (szintén önkényesen) a dekódolás eredménye $\mathbf{c}_2$.

A későbbiekben kiderül, hogy a kódoló f függvény leglényegesebb tulajdonsága a C kód egy paramétere, amit **kódtávolság**nak nevezünk, és $d_{\min}$-nel jelölünk:

$$d_{\min} = \min_{\substack{\mathbf{c}\neq\mathbf{c}' \\ \mathbf{c},\mathbf{c}'\in C}} d(\mathbf{c},\mathbf{c}').$$

A 2.1. és a 2.3. példákban $d_{\min} = 3$, míg a 2.2. példában $d_{\min} = 2$.

A **hibajelzés** a hibakorlátozó kódolás azon feladata, amikor a vevőben csupán detektálni akarjuk a hibázás tényét, azaz azt kérdezzük, hogy van-e hiba. Nyilván egy $\mathbf{v}$ vett szó esetén akkor tudjuk a hibázást észrevenni, ha $\mathbf{v}$ nem kódszó, amire garancia, hogy ha $\mathbf{c}$ küldött kódszó esetén

$$d_{\min} > d(\mathbf{v},\mathbf{c}),$$

azaz a hibák számára

$$d_{\min} > t,$$

tehát egy $d_{\min}$ kódtávolságú kód minden, legfeljebb $d_{\min} - 1$ számú hibát jelezni tud.

Mivel a 2.1. és a 2.3. példákban $d_{\min} = 3$, ezért ez a kód 2 hibát tud jelezni, míg a 2.2. példa kódja $d_{\min} = 2$ miatt 1-et.

Hibajavítás esetén azt kérdezzük, hogy ha t a hibák száma, akkor mi biztosítja, hogy a $\mathbf{v}$ vett szóból a $\mathbf{c}$ küldött kódszó egyértelműen visszaállítható legyen, azaz minden más $\mathbf{c}'$ kódszóra

$$d(\mathbf{v},\mathbf{c}') > d(\mathbf{v},\mathbf{c}) \tag{2.1}$$

legyen. Mivel a Hamming-távolság valóban távolság, ezért teljesíti a háromszög-egyenlőtlenséget, azaz

$$d(\mathbf{v},\mathbf{c}') \geq d(\mathbf{c},\mathbf{c}') - d(\mathbf{v},\mathbf{c}), \tag{2.2}$$

tehát (2.1) úgy biztosítható, hogy

$$d(\mathbf{c},\mathbf{c}') - d(\mathbf{v},\mathbf{c}) > d(\mathbf{v},\mathbf{c}),$$

ugyanis, ha ez utóbbi teljesül, akkor (2.1) is teljesül, azaz minden $\mathbf{c}' \neq \mathbf{c}$-re

$$d(\mathbf{c},\mathbf{c}') > 2d(\mathbf{v},\mathbf{c}),$$

vagyis

$$\frac{d_{\min}}{2} > d(\mathbf{v},\mathbf{c}).$$

Összefoglalva: **egyszerű hibázás** esetén $\left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$ hiba javítható.

A 2.1. és a 2.3. példa kódja 1 hibát tud javítani, míg a 2.2. példa kódjának hibajavító képessége 0.

Gyakran fordul elő olyan hibázás, amikor tudjuk, hogy egy pozícióban hiba lehet, vagyis tudjuk, hogy más pozíciókban nincs hiba, tehát a hiba helyét ismerjük, csak a hiba értékét nem. Az ilyen hibát **törléses hibá**nak nevezzük. Egyszerűen belátható, hogy minden $d_{\min}-1$ törléses hiba javítható, ugyanis a legrosszabb esetben sem fordulhat elő, hogy két $\mathbf{c},\mathbf{c}'$ kódszó ugyanazon, de legfeljebb $d_{\min}-1$ pozíciójának törlésével ugyanazt a szót kapnánk.

A 2.1. és a 2.3. példa kódja 2 törléses hibát tud javítani, míg a 2.2. példa kódja 1-et.

Nyilván adott n kódszóhosszúság és $d_{\min}$ kódtávolság esetén nem lehet akármilyen nagy méretű kódot konstruálni:

2.1. tétel (Singleton-korlát). *Egy M kódszóból álló, n hosszú és d_{min} kódtávolságú kódra*

$$M \leq q^{n-d_{min}+1}.$$

BIZONYÍTÁS: Legyen k egy természetes szám, melyre

$$q^{k-1} < M \leq q^k.$$

Mivel a $k-1$ hosszú különböző sorozatok száma q^{k-1}, ezért $q^{k-1} < M$ miatt létezik két kódszó $\mathbf{c}$ és $\mathbf{c}'$, melyek az első $k-1$ koordinátában megegyeznek. Ezekre

$$d(\mathbf{c},\mathbf{c}') \leq n-k+1,$$

következésképpen

$$d_{\min} \leq n-k+1,$$

azaz

$$M \leq q^k \leq q^{n-d_{\min}+1}.$$ ■

Jellegzetes esetben $M = q^k$, vagyis a kódoló k hosszú forrásszegmensekhez rendel n hosszú vektorokat. Azt mondjuk ilyenkor, hogy a kódunk (n,k) paraméterű. Ebben az esetben a Singleton-korlát alakja

$$d_{\min} \leq n-k+1.$$

2.1. definíció. *Azon kódot, melyre a Singleton-korlátban egyenlőség áll,* ***maximális távolságú*** *vagy* ***MDS*** *(maximum distance separable) kódnak nevezzük.*

A 2.1. példában $k=1$, $n=3$, $d_{\min}=3$, tehát ez a kód MDS kód. Ugyanakkor a 2.3. példában $k=2$, $n=5$, $d_{\min}=3$, így ez a kód nem MDS.

A 2.1. és a 2.3. példák kódjai egy hibát képesek javítani. Ehhez a következő szemléletes geometriai képet rendelhetjük. A kódszavak mint középpontok körül képzeljünk el 1 Hamming-sugarú gömböket, azaz a gömb felületén olyan —

kódszóhossz hosszúságú — bináris szavak találhatók, amelyeknek a középpontban levő szótól való Hamming-távolsága 1. Ha egy hiba keletkezik, akkor a vett szó a leadott kódszó körüli gömbön helyezkedik el. Más szavakkal ezen gömbök a dekódolási tartományok. Tekintsük először a 2.1. példa kódját. A két gömb a következő szavakat tartalmazza:

$$(0,0,0) : (1,0,0),(0,1,0),(0,0,1)$$

$$(1,1,1) : (0,1,1),(1,0,1),(1,1,0)$$

A két gömb tartalmazza az összes 3 bit hosszúságú bináris szót, teljesen kitölti 3 bit hosszúságú bináris szavak terét. A 2.3. példák kódja esetén a négy 1 Hamming-sugarú gömb nem tölti ki az 5 bit hosszúságú bináris szavak terét. Ugyanakkor nem is tudjuk növelni 2 Hamming-sugárra a gömbök méretét anélkül, hogy azok átlapolódnának. Abban az esetben, ha gömbi dekódolási tartományokkal hézagmentesen képesek vagyunk lefedni a teret perfekt kódokról beszélünk. Ennek megfelelően a 2.1. példa kódja perfekt, míg a 2.3. példa kódja nem az. Nagyon egyszerű összefüggés adódik az 1 hibát javító bináris perfekt kódok paraméterei közötti összefüggésre. A tér elemeinek száma 2^n, a kódszavak száma 2^k, így egy gömb elemeinek száma ezek hányadosa, azaz 2^{n-k}. Másfelől, egy 1 Hamming sugarú gömbben $1+n$ elem van, így adódik, hogy 1 hibát javító bináris perfekt kódok esetén

$$1+n = 2^{n-k}. \tag{2.3}$$

A következő szakaszban mutatunk egy az ismétléses kódnál hatékonyabb perfekt kódot, a bináris Hamming-kódot. A (2.3) összefüggés könnyen általánosítható tetszőleges $t \geq 1$ hiba javítás esetére: a képlet bal oldalán a gömb elemeinek száma, a középpontól $0,1,2,\ldots,t$ Hamming-távolságra levő szavak számának összege, azaz

$$V(n,t) = 1+n+\binom{n}{2}+\cdots+\binom{n}{t}$$

áll. A fenti gondolatmenet alapján adódó Hamming-korlát bináris esetben az alábbi

$$V(n,t) \leq 2^{n-k}.$$

A Hamming-korlát nembináris esetben a következő alakot ölti:

$$\sum_{i=0}^{t}\binom{n}{i}(q-1)^i \leq q^{n-k}$$

2.2. Lineáris kódok

Bináris lineáris kódok

Ebben a szakaszban először bináris kódok egy fontos csoportjával ismerkedünk meg. A továbbiakban a kódjainkban szereplő kódszavakat alkotó szimbólumok legyenek 0 vagy 1 értékűek, az összeadás és a szorzás pedig a bináris összeadás és a bináris szorzás, azaz a modulo 2 összeadás és a modulo 2 szorzás.

Vezessük be a lineáris kód fogalmát:

2.2. definíció. *Egy bináris C kód **lineáris**, ha a C halmaza lineáris tér, azaz ha minden $\mathbf{c}, \mathbf{c}' \in C$-re $\mathbf{c} + \mathbf{c}' \in C$.*

A lineáris kód definíciójából következik, hogy a $\mathbf{0}$ vektor eleme minden lineáris kódnak, vagyis minden lineáris kód esetén a $\mathbf{0}$ kódszó. Egyszerűen belátható, hogy a 2.3. és 2.2. példa kódja lineáris.

A lineáris kódok jelentőségét az adja, hogy az egyes üzenetekhez tartozó kódszavak viszonylag egyszerűen generálhatók, és ugyancsak egyszerű módszer található a vett kódszavak hibamentességének vizsgálatára, vagyis a hibadetektálásra, és a hibák javítása sem bonyolult. A következőkben e módszereket fogjuk bemutatni.

Jelentsen C továbbra is egy lineáris kódot, a kódszóhossz legyen n. Ekkor C az n hosszúságú bináris koordinátájú vektorok terének egy altere; „kódszó” helyett gyakran „vektor”-t fogunk mondani.

A valós vektortérben megszokott lineáris függetlenség és bázis fogalmak itt is teljesen hasonlóan értelmezhetők, vagyis

2.3. definíció. *A $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in C$ vektorok lineárisan függetlenek, ha $\alpha_i \in \{0,1\}$ mellett*

$$\sum_{i=1}^{k} \alpha_i \mathbf{g}_i = 0$$

csak úgy állhat elő, ha $\alpha_i = 0$ minden $i = 1, 2, \ldots, k$-ra.

2.4. definíció. *A $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in C$ vektorok a C lineáris tér egy bázisát alkotják, ha lineárisan függetlenek, továbbá igaz az, hogy minden $\mathbf{c} \in C$ vektor előállítható*

$$\mathbf{c} = \sum_{i=1}^{k} u_i \mathbf{g}_i \tag{2.4}$$

alakban, ahol $u_i \in \{0,1\}$ minden $i = 1, 2, \ldots, k$-ra.

Az utóbbi definícióban a bázist alkotó vektorok lineáris függetlenségéből következik, hogy a kódszavak fenti típusú előállítása egyértelmű is, ha ugyanis létezne két különböző előállítás valamely $\mathbf{c} \in C$-re, tehát

$$\mathbf{c} = \sum_{i=1}^{k} u_i \mathbf{g}_i$$

és

$$\mathbf{c} = \sum_{i=1}^{k} y_i \mathbf{g}_i,$$

ahol nem áll fenn $u_i = y_i$ minden i-re, akkor a két egyenletet kivonva egymásból a nullvektornak egy nem triviális előállítását kapnánk a bázisvektorokkal, ami ellentmondana azok lineáris függetlenségének.

A (2.4) egyenlőség felírható mátrixalakban:

$$\mathbf{c} = \mathbf{uG}, \tag{2.5}$$

ahol $\mathbf{u} = (u_1, u_2, \ldots, u_k)$, $\mathbf{G}$ pedig a bázisvektorokból mint sorvektorokból álló mátrix. A (2.5) egyenlettel tehát egy k-dimenziós és egy n-dimenziós vektort rendelünk össze lineáris transzformációval, mégpedig kölcsönösen egyértelmű módon. Azt fogjuk mondani, hogy az $\mathbf{u}$ üzenethez a $\mathbf{c}$ kódszó tartozik.

A k-dimenziós $\mathbf{u}$ vektorokkal 2^k-féle üzenetet fejezhetünk ki, s ezeket kódoljuk a C kóddal. C elemei azonban n-dimenziós vektorok, és n nem kisebb k-nál, hiszen k az n-dimenziós vektorok C alterének dimenziószáma. A $k = n$ esetnek nincs most jelentősége, ha k kisebb, mint n, akkor viszont világos, hogy nem minden vektort kell felhasználni kódszónak, vagyis kódunk redundáns lesz, s ezt a redundanciát tudjuk hibajavításra felhasználni.

Az üzenetekhez a kódszavakat a $\mathbf{G}$ mátrix segítségével rendeljük hozzá, vagyis a $\mathbf{G}$ mátrix jelöli ki az n-dimenziós vektortérnek a kódot jelentő C alterét, a kódot $\mathbf{G}$ „generálja”.

2.5. definíció. *A fenti tulajdonságú* $\mathbf{G}$ *mátrixot a* C *kód* ***generátormátrix****ának nevezzük.*

Vegyük észre, hogy ha nem törődünk azzal, hogy melyik kódszó melyik üzenethez tartozik, csak a kódszavak halmazát tekintjük, akkor $\mathbf{G}$ nem egyértelmű, vagyis több mátrix is generálhatja ugyanazt a C kódszóhalmazt. A következő definíció egy megfeleltetést definiál az üzenetek és a kódszavak között.

2.6. definíció. *Egy* (n,k) *paraméterű lineáris kód* ***szisztematikus****, ha minden kódszavára igaz, hogy annak utolsó* $n-k$ *szimbólumát elhagyva éppen a neki megfelelő* k *hosszúságú üzenetet kapjuk, más szavakkal a* k *hosszú üzenetet egészítjük ki* $n-k$ *karakterrel.*

A 2.1. szakaszban már leszögeztük, hogy dekódolás alatt csak az esetleges hibák kijavítását értjük, aminek eredményeképp egy kódszót kapunk. Az üzenetvektor visszanyeréséhez még el kell ugyan végezni a kódolás inverz műveletét, ez azonban rendszerint triviális lépés, szisztematikus kód esetén például csak el kell hagyni a kódszó egy részét (a végét).

Szisztematikus kód esetén a generátormátrix is egyértelmű, mégpedig

$$\mathbf{G} = (\mathbf{I}_k, \mathbf{B}) \tag{2.6}$$

alakú, ahol $\mathbf{I}_k$ a $k \times k$ méretű egységmátrix, $\mathbf{B}$ pedig $k \times (n-k)$ méretű mátrix. Az $\mathbf{u}$ üzenethez tartozó $\mathbf{c}$ kódszó szerkezete tehát:

$$\mathbf{c} = (u_1, u_2, \ldots, u_k, c_{k+1}, c_{k+2}, \ldots, c_n).$$

A $\mathbf{c}$ első k koordinátájából álló szegmensét **üzenetszegmens**nek, az utolsó $n-k$ koordinátájából állót **paritásszegmens**nek nevezzük.

A lineáris kódok további tulajdonságai elvezetnek az ígért egyszerű hibadetektáláshoz illetve hibajavításhoz.

2.7. definíció. *Ha egy $n-k$ sorból és n oszlopból álló $\mathbf{H}$ mátrixra*

$$\mathbf{H}\mathbf{c}^T = \mathbf{0}$$

*akkor és csak akkor, ha $\mathbf{c} \in C$ ($\mathbf{c}^T$ a $\mathbf{c}$ transzponáltja), akkor $\mathbf{H}$-t a C kód **paritásellenőrző mátrix**ának nevezzük. (Röviden paritásmátrixot fogunk mondani.)*

$\mathbf{H}$ segítségével tehát meg tudjuk állapítani, hogy egy vett szó valóban kódszó-e.

2.2. tétel. *Ha $\mathbf{G}$ és $\mathbf{H}$ ugyanazon C lineáris kód generátormátrixa illetve paritásmátrixa, akkor*

$$\mathbf{H}\mathbf{G}^T = \mathbf{0}.$$

BIZONYÍTÁS: Jelölje Q^k a k hosszú bináris sorozatok halmazát. Ekkor minden $\mathbf{u} \in Q^k$-hoz létezik $\mathbf{c} \in C$, amire $\mathbf{c} = \mathbf{u}\mathbf{G}$. Ugyanakkor $\mathbf{c} \in C$ miatt $\mathbf{H}\mathbf{c}^T = \mathbf{0}$, azaz

$$\mathbf{H}\mathbf{c}^T = \mathbf{H}(\mathbf{u}\mathbf{G})^T = \mathbf{H}\mathbf{G}^T\mathbf{u}^T = \mathbf{0}.$$

Az utolsó egyenlőség pedig csak úgy állhat fönn minden $\mathbf{u} \in Q^k$-ra, ha $\mathbf{H}\mathbf{G}^T = \mathbf{0}$, amint állítottuk. ■

A 2.2. tétel alapján szisztematikus generátormátrix felhasználásával könnyen előállíthatjuk a kód egy paritásellenőrző mátrixát. Keressük $\mathbf{H}$-t

$$\mathbf{H} = (\mathbf{A}, \mathbf{I}_{n-k})$$

alakban. A 2.2. tétel alapján

$$\mathbf{H}\mathbf{G}^T = (\mathbf{A}, \mathbf{I}_{n-k})(\mathbf{I}_k, \mathbf{B})^T = \mathbf{A} + \mathbf{B}^T = \mathbf{0}.$$

Azaz

$$\mathbf{A} = -\mathbf{B}^T$$

kell teljesüljön. (Bináris esetben $-\mathbf{B}^T = \mathbf{B}^T$.)

2.4. példa. Adjuk meg a 2.3. példa kódjának szisztematikus generátormátrixát, ha van, és a paritásmátrixot! Ezt úgy kapjuk meg, ha **G** első sora $\mathbf{c}_3$, míg a második $\mathbf{c}_2$, mivel ekkor az első 2×2-es részmátrix egységmátrix:

$$\mathbf{G} = \begin{pmatrix} 1\,0\,1\,1\,0 \\ 0\,1\,1\,0\,1 \end{pmatrix}.$$

A fentiek alapján a paritásmátrix:

$$\mathbf{H} = \begin{pmatrix} 1\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1 \end{pmatrix}.$$

Ugyanígy kapjuk a 2.2. példa szisztematikus generátormátrixát és paritásmátrixát:

$$\mathbf{G} = \begin{pmatrix} 1\,0\,1 \\ 0\,1\,1 \end{pmatrix},$$

$$\mathbf{H} = \begin{pmatrix} 1\,1\,1 \end{pmatrix}.$$

A következőkben a súly fogalmát definiáljuk, majd megmutatjuk, hogy lineáris kódoknál a minimális súly a kódtávolsággal egyenlő. (Emlékeztetünk, hogy két kódszó távolsága azon koordinátáik száma, ahol a két kódszó különbözik.)

2.8. definíció. *Egy* **c** *vektor* ***súly****a a koordinátái között levő nem nulla elemek száma, jelölése* $w(\mathbf{c})$.

2.9. definíció. *Egy* C *kód* ***minimális súly****án a*

$$w_{min} = \min_{\substack{\mathbf{c} \in C \\ \mathbf{c} \neq \mathbf{0}}} w(\mathbf{c})$$

számot értjük.

2.3. tétel. *Ha* C *lineáris kód, akkor a kódtávolsága megegyezik a minimális súlyával, azaz*

$$d_{min} = w_{min}.$$

BIZONYÍTÁS:

$$d_{\min} = \min_{\mathbf{c} \neq \mathbf{c}'} d(\mathbf{c}, \mathbf{c}') = \min_{\mathbf{c} \neq \mathbf{c}'} w(\mathbf{c} - \mathbf{c}') = \min_{\mathbf{c}'' \neq \mathbf{0}} w(\mathbf{c}'') = w_{\min},$$

ahol az utolsó előtti egyenlőség felírásakor a C kód linearitását használtuk ki, ebből következik ugyanis, hogy $\mathbf{c}'' = \mathbf{c} - \mathbf{c}'$ is kódszó, továbbá, az is, hogy minden kódszó előáll ilyen különbség alakjában. (Utóbbi ahhoz szükséges, hogy a minimum képzésekor valóban minden $\mathbf{c}'' \in C$-t figyelembe vehessünk.) ■

A 2.3. tétel jelentősége abban áll, hogy segítségével a $d_{\min}$ definíció alapján történő kiszámításához szükséges $\frac{|C|(|C|-1)}{2}$ műveletet a $w_{\min}$ kiszámításához szükséges $|C|-1$ műveletre redukálhatjuk. ($|C|$-vel a C elemszámát jelöltük.)

A 2.1. és a 2.3. példák nemzérus kódszavaira tekintve látható, hogy a minimális súly 3, míg a 2.3. példa esetén a minimális súly 2.

Szindróma dekódolás

A **H** mátrix hasznosnak bizonyul dekódolás során.

2.10. definíció. *Az* $\mathbf{s} = \mathbf{e}\mathbf{H}^T$ *mennyiséget* ***szindróma****nak nevezzük.*

Legyen az adott kódszó **c**, a vett szó **v**. Az $\mathbf{e} = \mathbf{v} - \mathbf{c}$ vektort **hibavektor**nak nevezzük. Vegyük észre, hogy

$$\mathbf{H}\mathbf{v}^T = \mathbf{H}(\mathbf{c}+\mathbf{e})^T = \mathbf{H}\mathbf{c}^T + \mathbf{H}\mathbf{e}^T = \mathbf{H}\mathbf{e}^T,$$

vagyis $\mathbf{H}\mathbf{v}^T$ értéke csak a hibavektortól függ, az adott kódszótól nem. A szindróma tehát a hibavektor egy lineáris leképezése.

A dekódolás leggyakoribb módja a **szindróma dekódolás**. A fentiek alapján a dekódolás a következőképpen mehet végbe: a vett **v** szóból kiszámítjuk az $\mathbf{s}^T = \mathbf{H}\mathbf{v}^T = \mathbf{H}\mathbf{e}^T$ szindrómát, ennek alapján megbecsüljük a hibavektort, s ezt **v**-ből levonva megkapjuk a kódszóra vonatkozó becslésünket.

A szindrómának hibamintára történő leképezési módját táblázatba szokás foglalni, az ún. **standard elrendezési táblázat**ba.

Valamely **e** hibaminta által generált halmaz (szokásos nevén mellékosztály) az $\mathbf{e}+\mathbf{c}$, $\mathbf{c} \in C(n,k)$ vektorok halmaza. Adott mellékosztály elemeihez azonos szindróma tartozik. Az $\mathbf{e} = \mathbf{0}$ zérus hibavektorhoz tartozó mellékosztály a $C(n,k)$ kóddal azonos. Ha egy **e** hibaminta $\mathbf{e} = \mathbf{e}' + \mathbf{c}$ alakban írható fel, akkor a két hibaminta (**e** és $\mathbf{e}'$) azonos mellékosztályt generál. Azonos mellékosztályba tartozó hibaminták közül válasszuk ki a legkisebb súlyút, s azt mellékosztály-vezetőnek nevezzük. Ennek megfelelően a standard elrendezési táblázat az alábbi struktúrájú:

szindróma	mellékosztály-vezető			
$\mathbf{s}^{(0)}$	$\mathbf{e}^{(0)} = \mathbf{0}$	$\mathbf{c}^{(1)}$		$\mathbf{c}^{(2^k-1)}$
$\mathbf{s}^{(1)}$	$\mathbf{e}^{(1)}$	$\mathbf{c}^{(1)} + \mathbf{e}^{(1)}$		$\mathbf{c}^{(2^k-1)} + \mathbf{e}^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{s}^{(2^{n-k}-1)}$	$\mathbf{e}^{(2^{n-k}-1)}$	$\mathbf{c}^{(1)} + \mathbf{e}^{(2^{n-k}-1)}$		$\mathbf{c}^{(2^k-1)} + \mathbf{e}^{(2^{n-k}-1)}$

mellékosztály elemek

A $w(\mathbf{e}^{(i+1)}) \geq w(\mathbf{e}^{(i)})$, $\mathbf{e}^{(0)} = \mathbf{0}$, $i = 0,1,\ldots,2^{n-k}-2$ a szokásos sorrend. Könnyen látható, hogy a táblázat elemei különbözőek. Egy soron belül ez nyilvánvaló. Különböző sorokat tekintve tegyük fel, hogy $\mathbf{e}^{(i)} + \mathbf{c}^{(j)} = \mathbf{e}^{(k)} + \mathbf{c}^{(m)}$, ahol $i > k$. Mivel ebből $\mathbf{e}^{(i)} = \mathbf{e}^{(k)} + \mathbf{c}^{(m)} - \mathbf{c}^{(j)} = \mathbf{e}^{(k)} + \mathbf{c}^{(n)}$ következik, ahol $\mathbf{c}^{(j)}, \mathbf{c}^{(m)}, \mathbf{c}^{(n)} \in C(n,k)$, ezért $\mathbf{e}^{(i)}$-nek is az $\mathbf{e}^{(k)}$ mellékosztály-vezetőjű sorban kell lennie, ami ellentétes kiindulási feltételünkkel.

Az $\mathbf{e}^{(i)}$, $i = 1,2,\ldots,2^{n-k}-1$ mellékosztály-vezetőket **javítható hibamintá**knak nevezzük, ugyanis ha a $\mathbf{v}$ vett szó szindrómája $\mathbf{s}^{(i)}$, akkor a $\widehat{\mathbf{c}} = \mathbf{v} - \mathbf{e}^{(i)}$ kódszóra döntünk. A szindróma dekódolásnak ezt az — elsősorban elvi — módját **táblázatos dekódolás**nak nevezzük. (Megjegyezzük, hogy a fenti dekódolási módszer nembináris ábécé esetére történő kiterjesztésekor 2^{n-k} helyett q^{n-k} áll.)

A szindrómát használó táblázatos dekódoló tárja a vizsgált bináris esetben 2^{n-k} darab hibavektort tartalmaz, és a táblázat elemeit a szindróma segítségével címezzük. Következésképp a táblázatos módszer gyakorlatban addig használható, amíg gyorselérésű tárunk mérete lehetővé teszi a javítható hibaminták tárolását.

2.5. példa. Adjuk meg a javítható hibamintákat a

$$\mathbf{H} = \begin{pmatrix} 0\,0\,1\,1\,1 \\ 0\,1\,0\,1\,0 \\ 1\,0\,0\,1\,1 \end{pmatrix}$$

mátrixszal adott kód esetére.

A dekódolási táblázat a következő:

szindróma	javítható hibaminták
000	00000
001	10000
010	01000
011	00110
100	00100
101	00001
110	01100
111	00010

Tehát a standard elrendezés fenti táblázata alapján történő szindróma dekódolással az egyszeres hibák és a 00110, 01100 két hibát tartalmazó hibaminták javíthatók.

Illusztrációként egy klasszikusnak számító kódot mutatunk be, mely **bináris Hamming-kód** néven ismeretes. Konstrukciónkat az alábbi tételre alapozzuk:

2.4. tétel. *C lineáris kód kódtávolsága legalább d^* akkor és csak akkor, ha a paritásellenőrző mátrixa tetszőlegesen választott d^*-1 oszlopa lineárisan független.*

A 2.4. tétel alapján 1 hibát javító bináris kódot kapunk, ha **H** tetszőleges két oszlopa lineárisan független, azaz oszlopai különbözők. Mivel a különböző, nemzérus, $n-k$ hosszú bináris vektorok száma $2^{n-k}-1$, ezért ezen vektorokat használva a **H** mátrix különböző oszlopaiként, az

$$n = 2^{n-k} - 1$$

összefüggésre jutunk, ami azt is jelenti, hogy a kapott kód perfekt tulajdonságú. Ennek alapján bináris Hamming-kód paraméterei az alábbi számpárok ($d_{min} = 3$):

$n =$ 3	$k =$ 1
7	4
15	11
31	26
63	57
127	120

2.6. példa. A $(7,4)$ paraméterű Hamming-kód paritásmátrixa

$$\mathbf{H} = \begin{pmatrix} 1\,1\,0\,1\,1\,0\,0 \\ 1\,0\,1\,1\,0\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1 \end{pmatrix}.$$

A generátormátrixa ebből könnyen kiszámítható a már szerepelt $\mathbf{A} = -\mathbf{B}^T$ összefüggés alapján:

$$\mathbf{G} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,1\,0\,1 \\ 0\,0\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1 \end{pmatrix}.$$

2.3. Véges test

Hatékony hibajavító kódok konstrukciójához szükséges, hogy a nembináris Q kódábécé struktúrált legyen, mely például úgy lehetséges, hogy műveleteket vezetünk be Q-n.

2.11. definíció. *Egy Q halmazt **test**nek nevezünk, ha értelmezve van tetszőleges két eleme között két művelet, amelyeket összeadásnak illetve szorzásnak nevezünk, $+$ illetve $*$ szimbólumokkal jelöljük, és Q rendelkezik a következő tulajdonságokkal:*

1. *Q az összeadásra nézve kommutatív csoport, azaz*
   a) *Minden $\alpha, \beta \in Q$ esetén $\alpha + \beta \in Q$, tehát Q az összeadásra nézve zárt.*
   b) *Minden $\alpha, \beta, \gamma \in Q$ esetén $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$ (asszociativitás).*

c) *Létezik egy 0-val jelölt eleme Q-nek úgy, hogy minden $\alpha \in Q$-re $0+\alpha=\alpha+0=\alpha$. 0-t nullelemnek nevezzük.*

d) *Minden $\alpha \in Q$-hez létezik $\beta \in Q$ úgy, hogy $\alpha+\beta=0$. β-t az α additív inverzének nevezzük és $-\alpha$-val jelöljük.*

e) *Minden $\alpha,\beta \in Q$-re $\alpha+\beta=\beta+\alpha$ (kommutativitás).*

2. *$Q\setminus\{0\}$ a szorzásra nézve kommutatív csoport, azaz*

a) *Minden $\alpha,\beta \in Q\setminus\{0\}$ esetén $\alpha\cdot\beta \in Q\setminus\{0\}$ (zártság).*

b) *Minden $\alpha,\beta,\gamma \in Q\setminus\{0\}$ esetén $(\alpha\cdot\beta)\cdot\gamma=\alpha\cdot(\beta\cdot\gamma)$ (asszociativitás).*

c) *Létezik egy 1-gyel jelölt eleme $Q\setminus\{0\}$-nak úgy, hogy $1\cdot\alpha=\alpha\cdot 1=\alpha$. 1-et egységelemnek nevezzük.*

d) *Minden $\alpha \in Q\setminus\{0\}$ esetén létezik $\beta \in Q\setminus\{0\}$ úgy, hogy $\alpha\cdot\beta=\beta\cdot\alpha=1$. β-t az α multiplikatív inverzének nevezzük, és α^{-1}-gyel jelöljük.*

e) *Minden $\alpha,\beta \in Q\setminus\{0\}$-ra $\alpha\cdot\beta=\beta\cdot\alpha$ (kommutativitás).*

3. *Minden $\alpha,\beta,\gamma \in Q$-re $\alpha\cdot 0=0\cdot\alpha=0$ és $\alpha\cdot(\beta+\gamma)=(\alpha\cdot\beta)+(\alpha\cdot\gamma)$ (disztributivitás).*

Egyszerű konvenciókkal egy Q testben definiálható a kivonás és az osztás a következő módon: $\alpha-\beta$ alatt az α-nak és a β additív inverzének összegét értjük, azaz $\alpha+(-\beta)$-t. α/β alatt az α-nak és a β multiplikatív inverzének a szorzatát értjük, azaz $\alpha\cdot\beta^{-1}$-et, amennyiben β nem 0.

Példák testre:

- Valós számok halmaza a valós összeadással és szorzással.
- Racionális számok halmaza a valós összeadással és szorzással.
- Komplex számok halmaza a komplex összeadással és szorzással.
- $\{0,1\}$ a bináris összeadással és szorzással.

Egy q elemszámú Q testet **véges test**nek nevezünk és GF(q)-val jelöljük.

Mielőtt a véges testek aritmetikáját tárgyalnánk, néhány a továbbiakban felhasznált fontos tulajdonságukat ismertetjük.

Egy GF(q) esetén q nem lehet bármilyen:

2.5. tétel. *Egy GF(q) esetén $q=p^m$ alakú, ahol p prímszám, tehát q vagy prímszám, vagy prímhatvány.*

2.1. lemma. *Minden $0 \neq a \in$ GF(q)-ra*

$$a^{q-1}=1.$$

2.2. lemma. *Minden $0 \neq a \in GF(q)$-ra létezik egy legkisebb m természetes szám, amit az a **elem rendjé**nek nevezünk, melyre*

$$a^m = 1,$$

és az $a, a^2, \ldots, a^m$ elemek mind különbözők. m osztója $q-1$-nek.

2.12. definíció. *Egy $\alpha \in GF(q)$-t a $GF(q)$ **primitív elem**ének nevezünk, ha α rendje $q-1$.*

2.6. tétel. *Minden GF(q)-ban létezik primitív elem.*

Aritmetika GF(p)-ben

2.7. tétel. *A $G = \{0, 1, \ldots, p-1\}$ halmaz a **modulo** p **aritmetiká**val egy p prímszám esetén véges test, azaz a testműveletek*

$$a + b = a + b \mod p,$$

$$a \cdot b = a \cdot b \mod p,$$

ahol $+$ illetve $\cdot$ jelöli a valós összeadást illetve szorzást.

2.7. példa. GF(3)
A GF(3) testben a műveletek modulo 3 összeadás és szorzás. A kapcsolatos műveleti táblákat láthatjuk alább:

$+$	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

$*$	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

2.8. példa. GF(7)

elem ($\neq 0$)	hatványai	rendje
1	1	1
2	2, 4, 1	3
3	3, 2, 6, 4, 5, 1	6 (primitív elem)
4	4, 2, 1	3
5	5, 4, 6, 2, 3, 1	6 (primitív elem)
6	6, 1	2

A prímitív elem egyrészt igen fontos hatékony kódok konstrukciójakor, másrészt GF(q)-beli szorzások és osztások elvégzésekor. Ha α a GF(q) egy primitív eleme, akkor bevezethetjük egy $a \in \mathrm{GF}(q)$ testelem α alapú logaritmusát az

$$\alpha^{\log a} = a$$

egyenlet (egyértelmű) megoldásával, ahol $a \neq 0$. Ha a, b a GF(q) nem 0 elemei, akkor

$$a \cdot b = \alpha^{\log a} \cdot \alpha^{\log b} = \alpha^{\log a + \log b},$$

tehát egy α alapú logaritmustábla és egy inverzlogaritmus-tábla segítségével a szorzás (illetve az osztás) visszavezethető valós összeadásra (illetve kivonásra).

A következőkben nagyon hasznosnak bizonyulnak a GF(q) feletti polinomok, így többek között egy fontos kódcsalád (a ciklikus kódok) leírásában, illetve a prímhatvány méretű véges testek aritmetikája generálásakor fogjuk használni őket.

Véges test feletti polinomok

GF(q) feletti vektorok reprezentálására, és vektorok közötti szorzás kényelmes bevezetésére egy célszerű eszköz a polinomreprezentáció:

2.13. definíció. $a(x) = a_0 + a_1 x + \ldots + a_m x^m$ ***GF(q) feletti*** *m-edfokú* ***polinom****, ha*

$$a_i \in \mathrm{GF}(q), \qquad i = 0, \ldots, m, \quad a_m \neq 0,$$

$$x \in \mathrm{GF}(q).$$

A polinom m fokszámát $\deg a(x)$ *jelöli. (Az* $a(x) \equiv 0$ *polinom fokszáma definíció szerint legyen* $-\infty$*.)*

2.14. definíció. $a(x) = b(x)$*, ha* $a_i = b_i$ *minden i-re.*

Műveletek polinomok között:

1. Polinomok összeadása: $c(x) = a(x) + b(x)$ tagonként történik GF(q) feletti műveletekkel: $c_i = a_i + b_i$. Nyilvánvalóan

$$\deg c(x) \leq \max\{\deg a(x), \deg b(x)\}.$$

2. Polinomok szorzása: $c(x) = a(x)b(x)$ minden tagot minden taggal szorzunk, majd az azonos fokú tagokat csoportosítjuk (az összeadások és szorzások GF(q) felettiek):

$$c_i = \sum_{j=0}^{\min\{i, \deg a(x)\}} a_j \cdot b_{i-j}.$$

Nyilván

$$\deg c(x) = \deg a(x) + \deg b(x)$$

2.9. példa. Ha GF(2) felett $a(x) = 1+x$ és $b(x) = 1+x+x^3$, akkor $a(x)+b(x) = x^3$ és $a(x)b(x) = 1+x^2+x^3+x^4$.

2.8. tétel (Euklidészi osztás polinomokra). *Adott $a(x)$ és $d(x) \neq 0$ esetén egyértelműen létezik olyan $q(x)$, $r(x)$, hogy*

$$a(x) = q(x)d(x) + r(x),$$

és $\deg r(x) < \deg d(x)$.

2.15. definíció. *$r(x)$-et az $a(x)$-nek $d(x)$-re vonatkozó maradékának nevezzük. Jelölés: $r(x) = a(x) \mod d(x)$.*

2.16. definíció. *$d(x)$ osztja $a(x)$-et, ha $a(x) \mod d(x) = 0$. Ezt a továbbiakban $d(x) \mid a(x)$ formában fogjuk jelölni.*

2.17. definíció. *$b \in GF(q)$ gyöke az $a(x)$ polinomnak, ha $a(b) = 0$.*

2.9. tétel. *Ha c az $a(x)$ polinom gyöke, akkor az előáll*

$$a(x) = b(x)(x-c)$$

alakban.

2.10. tétel. *Egy k-adfokú polinomnak legfeljebb k gyöke lehet.*

Aritmetika GF(p^m)-ben

Lényeges különbség van a prím illetve prímhatvány méretű testek aritmetikája között. Prím méretű testben a modulo aritmetika megfelelt. Prímhatvány méret esetén sajnos a modulo aritmetika nem teljesíti a testaxiómákat, például egy 4 elemű halmazban $2 \cdot 2 \mod 4 = 0$, tehát két nem 0 elem szorzata 0 lenne, ami sérti a 2. a) axiómát. A GF(p^m) feletti aritmetika konstrukciója azért alapvető fontosságú, mert manapság a hibajavító kódokat tömegesen alkalmazzuk számítástechnikai környezetben, ahol a természetes ábécé a GF(2^8), vagyis a bájt.

A GF(p^m)-beli elemek legyenek a $0, 1, \ldots, p^m - 1$ számok, melyeknek m hosszú vektorokat feleltetünk meg, ahol a koordináták GF(p)-beliek. Ezt megfogalmazhatjuk például úgy is, hogy a $0, 1, \ldots, p^m - 1$ számokat p-s számrendszerben írjuk fel. Ezek után a GF(p^m)-beli aritmetikát m hosszú vektorok közötti műveletekkel definiáljuk. A két művelet közül az összeadás az egyszerűbb: két vektor összegén a koordinátánkénti GF(p)-beli összeget értjük, vagyis a koordinátánkénti mod p összeget. A szorzás egy kicsit bonyolultabb. A két m hosszú vektort legfeljebb $(m-1)$-edfokú polinom formájában reprezentáljuk, és összeszorozzuk. Az eredmény fokszáma meghaladhatja $(m-1)$-et, ezért itt egy speciális polinom szerinti maradékot képezünk. Ezt a speciális polinomot irreducíbilis polinomnak nevezzük, és ez a polinom ugyanolyan szerepet játszik, mint a prímszám a GF(p)-beli aritmetikában.

2.18. definíció. *A GF(p) feletti, nem nulladfokú $P(x)$ polinomot **irreducíbilis polinom**nak nevezzük, ha nem bontható fel két, nála alacsonyabb fokú GF(p) feletti polinom szorzatára, azaz nincs GF(p) feletti $a_1(x), a_2(x)$ polinom, melyekre*

$$P(x) = a_1(x) \cdot a_2(x)$$

és

$$0 < \deg(a_i(x)) < \deg(P(x)), \qquad i = 1,2.$$

Bizonyítás nélkül megjegyezzük, minden véges testben található tetszőleges fokszámú irreducíbilis polinom. Példát mutatunk viszont arra, hogy hogyan lehet GF(2) feletti irreducíbilis polinomokat generálni. A definícióból következik, hogy minden elsőfokú polinom (x és $x+1$) irreducíbilis. Ha találunk olyan másodfokú polinomot, mely különbözik az x^2, az $x(x+1)$ és az $(x+1)^2$ mindegyikétől, akkor találtunk irreducíbilis másodfokú polinomot. Egy ilyen van: x^2+x+1. Más testben és nagyobb fokszám esetén ennél hatékonyabb konstrukciókat érdemes használni, de bináris esetben így is találhatók irreducíbilis polinomok, amelyeket táblázatban foglalunk össze:

fokszám	irreducíbilis polinom
2	x^2+x+1
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^3+x^2+1$
9	x^9+x^4+1

2.11. tétel. *Legyen p egy prím, m egy természetes szám, $P(x)$ egy GF(p) feletti m-edfokú irreducíbilis polinom és $Q = \{0,1,\ldots,p^m-1\}$. Egy $a \in Q$-nak és $b \in Q$-nak kölcsönösen egyértelműen feleltessünk meg GF(p) feletti, legfeljebb $(m-1)$-edfokú $a(x)$ és $b(x)$ polinomot. $a+b$ definíció szerint az a $c \in Q$, melynek megfelelő $c(x)$ polinomra*

$$c(x) = a(x) + b(x).$$

$a \cdot b$ az a $d \in Q$, melynek megfelelő $d(x)$ polinomra

$$d(x) = \{a(x) \cdot b(x)\} \mod P(x).$$

Ezzel az aritmetikával Q egy GF(p^m).

2.10. példa. Készítsük el a GF(2^2)-beli aritmetikát! Tudjuk, hogy a $P(x) = x^2 + x + 1$ egy másodfokú irreducíbilis polinom. A kölcsönösen egyértelmű megfelelte-

téseket egy táblázatban foglaljuk össze:

testelemek	$m=2$ hosszú vektorok	polinomok
0	00	0
1	01	1
2	10	x
3	11	$x+1$

Az összeadást egyszerűen a 2 hosszú vektorok koordinátánkénti bináris összegével kapjuk. Nézzünk a szorzásra példát! $2 \cdot 3$-at úgy számoljuk ki, hogy a 2-nek és a 3-nak megfelelő polinomot összeszorozzuk, és vesszük a $P(x)$ szerinti maradékot:

$$x(x+1) = 1 \pmod{x^2+x+1},$$

amely megfelel az 1 testelemnek. Az összeadó és a szorzó tábla ennek megfelelően a bináris vektorokra:

$+$	**00**	**01**	**10**	**11**
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

$\cdot$	**00**	**01**	**10**	**11**
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

majd testelemekre

$+$	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\cdot$	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

2.4. Nembináris lineáris kód

Ebben a szakaszban kódok egy fontos csoportjával ismerkedünk meg, melyek a 2.2. szakaszban megismert bináris lineáris kódok kiterjesztései nembináris esetre.

A továbbiakban a kódjainkban szereplő kódszavakat alkotó szimbólumokat vegyük GF(q)-ból, a lehetséges szimbólumok tehát a $0,1,2,\ldots,q-1$ számoknak feleltethetők meg.

2.19. definíció. *Egy C kód **lineáris**, ha a C halmaz lineáris tér GF(q) fölött, azaz ha minden $\mathbf{c},\mathbf{c}' \in C$-re*

$$\mathbf{c}+\mathbf{c}' \in C$$

illetve $\beta \in$ GF(q) esetén

$$\beta\mathbf{c} \in C.$$

A 2.2. szakaszhoz hasonló módon belátható, hogy tetszőleges C lineáris kódhoz létezik egy k lineárisan független sorból és n oszlopból álló $\mathbf{G}$ mátrix, melyre

$$\mathbf{c} = \mathbf{uG}, \tag{2.7}$$

ahol a k hosszú $\mathbf{u}$ üzenethez a $\mathbf{c}$ kódszó tartozik, és a $\mathbf{G}$ mátrixot a $\mathbf{C}$ kód **generátormátrix**ának nevezzük.

A bináris esethez hasonlóan a C lineáris kódhoz egy $n-k$ sorból és n oszlopból álló $\mathbf{H}$ mátrixot **paritásmátrix**nak nevezünk, amennyiben

$$\mathbf{Hc}^T = 0$$

akkor és csak akkor teljesül, ha $\mathbf{c} \in C$.

A bináris eset másolataként kaphatjuk, hogy

2.12. tétel. *Minden C lineáris kódnak van paritásellenőrző mátrixa.*

Példaként bemutatjuk a **nembináris Hamming-kód**ot. Ismét 1 hibát javító kódot akarunk konstruálni. A bináris esetben a hiba javításához elég volt ismerni a hiba helyét, amihez elégséges volt, ha a $\mathbf{H}$ paritásmátrix minden oszlopa különböző. Nembináris esetben nemcsak a hiba helyét, hanem a hiba értékét is meg kell állapítani, ezért a $\mathbf{H}$ mátrix oszlopait úgy választjuk, hogy azok nem 0-k, mind különbözők legyenek, és az első nem 0 elem minden oszlopban 1 értékű legyen. Ekkor, ha egy hiba esetén az az i-edik helyen fordul elő és értéke e_i, akkor a szindróma $\mathbf{s} = e_i\mathbf{a}_i$, ahol $\mathbf{a}_i^T$ a $\mathbf{H}$ i-edik oszlopa. Tehát a hiba értéke, e_i éppen a szindróma első nem 0 értéke, míg $\mathbf{a}_i = \frac{\mathbf{s}}{e_i}$, amiből az i visszakereshető.

Ha $\mathbf{H}$ tartalmazza az összes lehetséges, a fenti módon megengedett oszlopvektort, akkor

$$n = \frac{q^{n-k}-1}{q-1},$$

azaz

$$1 + n(q-1) = q^{n-k},$$

másrészt a Hamming-korlát miatt

$$1 + n(q-1) \leq q^{n-k},$$

tehát

2.13. tétel. *A maximális hosszúságú nembináris Hamming-kód perfekt kód.*

A nembináris Hamming-kódok közül különösen érdekes az az eset, amikor a kód szisztematikus és a paritásszegmens hossza 2, azaz $n-k=2$. Legyen α a GF(q) egy nem 0 eleme, melynek rendje $m \geq 2$. Válasszunk $n \leq (m+2)$-t és $k = (n-2)$-t. Ekkor a paritásmátrix:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-3} & 0 & 1 \end{pmatrix}.$$

Ez egy $(n, n-2)$ paraméterű nembináris Hamming-kód paritásmátrixa.

A 2.2. tétel alkalmazásával nyerjük a kód generátormátrixát:

$$\mathbf{G} = \begin{pmatrix} 1\,0\,0\,0\cdots 0\,-1 & -1 \\ 0\,1\,0\,0\cdots 0\,-1 & -\alpha \\ 0\,0\,1\,0\cdots 0\,-1 & -\alpha^2 \\ \vdots \qquad\quad \ddots \qquad\quad & \vdots \\ 0\,0\,0\,0\cdots 1\,-1 & -\alpha^{n-3} \end{pmatrix}.$$

Mivel ez a kód 1 hibát tud javítani, ezért $d_{\min} \geq 3$, de a Singleton-korlát miatt $d_{\min} \leq n-k+1 = 3$, ezért

2.14. tétel. *Az $(n, n-2)$ paraméterű nembináris Hamming-kód MDS kód.*

2.11. példa. Írjuk fel a GF(7) feletti, $(8,6)$ paraméterű Hamming-kód generátormátrixát és paritásmátrixát! GF(7)-ben a 3 primitív elem (lásd a 2.8. példát), tehát

$$\mathbf{H} = \begin{pmatrix} 1\,1\,1\,1\,1\,1\,1\,0 \\ 1\,3\,2\,6\,4\,5\,0\,1 \end{pmatrix}$$

$$\mathbf{G} = \begin{pmatrix} 1\,0\,0\,0\,0\,0\,-1\,-1 \\ 0\,1\,0\,0\,0\,0\,-1\,-3 \\ 0\,0\,1\,0\,0\,0\,-1\,-2 \\ 0\,0\,0\,1\,0\,0\,-1\,-6 \\ 0\,0\,0\,0\,1\,0\,-1\,-4 \\ 0\,0\,0\,0\,0\,1\,-1\,-5 \end{pmatrix} = \begin{pmatrix} 1\,0\,0\,0\,0\,0\,6\,6 \\ 0\,1\,0\,0\,0\,0\,6\,4 \\ 0\,0\,1\,0\,0\,0\,6\,5 \\ 0\,0\,0\,1\,0\,0\,6\,1 \\ 0\,0\,0\,0\,1\,0\,6\,3 \\ 0\,0\,0\,0\,0\,1\,6\,2 \end{pmatrix}.$$

Reed–Solomon-kód

Ebben a szakaszban a lineáris kódok egyik leggyakrabban használt osztályával, a **Reed–Solomon-kód**okkal, azok különböző konstrukcióival ismerkedünk meg.

2.1. konstrukció. *Legyenek $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ a GF(q) különböző elemei ($n \leq q$), és $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ ($u_i \in$ GF(q)) a k hosszúságú üzenetszegmens, amelyhez az*

$$u(x) = u_0 + u_1 x + \ldots + u_{k-1} x^{k-1}$$

üzenetpolinomot rendeljük. Ekkor a Reed–Solomon-kódnak az **u** *üzenethez tartozó n hosszú* **c** *kódszavát a következő módon állítjuk elő:*

$$\begin{aligned} c_0 &= u(\alpha_0) \\ c_1 &= u(\alpha_1) \\ c_2 &= u(\alpha_2) \\ &\vdots \\ c_{n-1} &= u(\alpha_{n-1}). \end{aligned}$$

Egyszerűen belátható, hogy a Reed–Solomon-kód lineáris, és a generátormátrixa

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} \\ \vdots & & & \ddots & \vdots \\ \alpha_0^{k-1} & \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_{n-1}^{k-1} \end{pmatrix}.$$

2.15. tétel. *Az (n,k) paraméterű Reed–Solomon-kód kódtávolsága*

$$d_{min} = n-k+1,$$

vagyis a Reed–Solomon-kód maximális távolságú.

BIZONYÍTÁS:

$$\begin{aligned} w(\mathbf{c}) &= |\{\mathbf{c} \text{ nem 0 koordinátái}\}| = \\ &= n - |\{\mathbf{c} \text{ 0 koordinátái}\}| \geq \\ &\geq n - |\{u(x) \text{ gyökei}\}| \geq \\ &\geq n-(k-1), \end{aligned}$$

tehát

$$w_{\min} \geq n-k+1.$$

Ugyanakkor a 2.1. tétel és a 2.3. tétel miatt

$$n-k+1 \geq d_{\min} = w_{\min},$$

következésképp az állítást bebizonyítottuk. ■

Az (n,k) paraméterű Reed–Solomon-kód tehát $n-k$ hibát tud jelezni, $\left\lfloor \frac{n-k}{2} \right\rfloor$ egyszerű hibát javítani és $n-k$ törléses hibát javítani. Ez utóbbi azt is jelenti, hogy az **u** ismeretlenre vonatkozó

$$\mathbf{uG} = \mathbf{c}$$

n darab egyenletből bármelyik $n-k$ egyenlet elhagyásával egy egyértelműen megoldható egyenletrendszer marad, tehát a **G** mátrix minden $k \times k$-s négyzetes részmátrixa invertálható.

2.2. konstrukció. *Legyen α a GF(q) egy nem 0 eleme, melynek rendje m, $m \geq n$ és a 2.1. konstrukcióban legyen $\alpha_0 = 1, \alpha_1 = \alpha, \ldots, \alpha_{n-1} = \alpha^{n-1}$. Ekkor a generátormátrix:*

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(k-1)(n-1)} \end{pmatrix}$$

2.5. Ciklikus kódok

2.20. definíció. *Egy*

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$$

vektor ciklikus eltoltja az

$$S\mathbf{c} = (c_{n-1}, c_0, \ldots, c_{n-2}).$$

*S-et a **ciklikus eltolás** operátorának nevezzük.*

2.21. definíció. *A C kódot **ciklikus**nak nevezzük, ha bármely kódszó ciklikus eltoltja is kódszó.*

2.12. példa. Legyen C a

000
101
110
011
111

vektorok halmaza. Egyszerűen belátható, hogy C ciklikus. Megjegyezzük, hogy a ciklikusságból nem következik a linearitás, például a 2. és az 5. kódszó összege 010, amely nem eleme a kódnak.

A ciklikus eltolás operátorát kényelmesebben tudjuk kezelni, ha a kódszavakat mint vektorokat a már megszokott polinomos formában reprezentáljuk.

2.22. definíció. *Rendeljünk polinomot az egyes kódszavakhoz a következő módon:*

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \;\mapsto\; c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1},$$

ekkor a $\mathbf{c}$ *kódszónak megfeleltetett* $c(x)$ *polinomot **kódszópolinom**nak vagy röviden kódpolinomnak nevezzük. A kódszópolinomok halmazát* $C(x)$*-szel jelöljük.*

2.3. lemma. *Legyen* $c'(x)$ *a* $\mathbf{c}$ *kódszó* $S\mathbf{c}$ *eltoltjához rendelt kódszópolinom, ekkor*

$$c'(x) = [xc(x)] \mod (x^n - 1).$$

2.16. tétel. *Minden* (n,k) *paraméterű, ciklikus, lineáris* C *kódban a nem azonosan nulla kódszópolinomok között egyértelműen létezik egy minimális fokszámú* $g(x)$ *polinom, amelynek legmagasabb fokú tagja együtthatója* 1. $g(x)$ *fokszáma* $n-k$*, és egy* $\mathbf{c} \in C$ *akkor és csak akkor, ha* $g(x) \mid c(x)$*, azaz létezik egy* $u(x)$ *polinom úgy, hogy* $c(x) = g(x)u(x)$*.*

2.23. definíció. $g(x)$*-et a kód **generátorpolinom**jának nevezzük.*

2.17. tétel. *Minden ciklikus, lineáris kód $g(x)$ generátorpolinomjára*

$$g(x) \mid x^n - 1.$$

Másrészről, ha egy $g(x)$ főpolinomra $g(x) \mid x^n - 1$, akkor létezik egy lineáris ciklikus kód, melynek $g(x)$ a generátorpolinomja.

A paritásmátrixnak is van polinomos megfelelője:

2.24. definíció. *Egy $g(x)$ generátorpolinomú lineáris, ciklikus kód esetén a*

$$h(x) = \frac{x^n - 1}{g(x)}$$

*polinomot **paritásellenőrző polinom**nak nevezzük.*

2.18. tétel. *Egy lineáris, ciklikus kódra $c(x)$ akkor és csak akkor kódszópolinom, ha*

$$c(x)h(x) = 0 \mod (x^n - 1)$$

és

$$\deg(c(x)) \leq n - 1.$$

Ciklikus kódok szisztematikus generálása

A ciklikus kódok előnyös tulajdonságai egyrészt a generálási lehetőségek sokféleségében, másrészt egyszerű dekódolási eljárásokban jelentkeznek. Egy lineáris ciklikus kódot lehet például a generátorpolinom és az üzenetpolinom szorzásával generálni. Ez a módszer megfogalmazható egy olyan **G** generátormátrix segítségével is, amelyhez legegyszerűbben úgy juthatunk el, ha **G** sorai a $g(x)$-nek megfelelő vektor eltoltjai:

$$\mathbf{G} = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 & 0 & \cdots & & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-2} & g_{n-k-1} & 1 & \cdots & & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & g_0 & g_1 & g_2 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k-1} & 1 \end{pmatrix},$$

kihasználva, hogy $g(x)$ főpolinom, így $g_{n-k} = 1$.

Egy másik generálási módszer kapcsán azt is megmutatjuk, hogy

2.19. tétel. *Minden lineáris ciklikus kód generálható szisztematikusan.*

BIZONYÍTÁS: Vegyük észre, hogy $g_0 \neq 0$, mert ha 0 lenne, akkor a **g**-t balra eltolva **g**-nél 1-gyel kisebb fokszámú kódszópolinomot kapnánk, ami lehetetlen. $g_0 \neq 0$ miatt viszont a Gauss-eliminációt balról jobbra végrehajtva szisztematikus generátormátrixot kapunk. ■

A **szisztematikus generálás** egy praktikus módszere a következő:
Legyen $u(x)$ egy legfeljebb $(k-1)$-edfokú üzenetpolinom és

$$c(x) = u(x)x^{n-k} - [u(x)x^{n-k}] \mod g(x),$$

akkor $c(x)$ kódszópolinom, mivel $c(x) = 0 \mod g(x)$. Ezen generálás szisztematikus: a $c(x)$-et definiáló egyenlőség jobb oldalának első tagja adja az üzenetszegmenst, míg a második tagja a paritásszegmenst.

2.13. példa. Tekintsük a GF(2) feletti $g(x) = 1+x+x^3$ polinomot! Mivel

$$x^7 - 1 = (1+x)(1+x+x^3)(1+x^2+x^3),$$

ezért $g(x)$ osztja (x^7-1)-et, tehát $g(x)$ egy $(7,4)$ paraméterű bináris, lineáris, ciklikus kód generátorpolinomja. Egy generátormátrixához jutunk a $g(x)$ eltoltjaival:

$$\mathbf{G} = \begin{pmatrix} 0\,0\,0\,1\,0\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,1\,0\,1\,1\,0\,0 \\ 1\,0\,1\,1\,0\,0\,0 \end{pmatrix}.$$

A második módszer szerinti szisztematikus generátormátrixhoz úgy jutunk el, ha kiszámítjuk az $[x^{3+i}] \mod (1+x+x^3)$ maradékokat $i = 0,1,2,3$-ra, melyek

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

Ezek alapján

$$\mathbf{G} = \begin{pmatrix} 1\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1\,1 \end{pmatrix},$$

amely a már jól ismert $(7,4)$ paraméterű Hamming-kód szisztematikus generátormátrixa.

Ciklikus Reed–Solomon-kód

A Reed–Solomon-kódok legfontosabb gyakorlati előállítási módja az alábbi tételen alapszik:

2.20. tétel. *A Reed–Solomon-kódok esetén legyen az n kódszóhossz egyenlő az ott szereplő α elem m rendjével. Ekkor a kód ciklikus, és generátorpolinomja*

$$g(x) = \prod_{i=1}^{n-k} (x - \alpha^i),$$

továbbá paritásellenőrző polinomja

$$h(x) = \prod_{i=n-k+1}^{n} (x - \alpha^i),$$

tehát a nem rövidített Reed–Solomon-kód ciklikus.

2.21. tétel. *A 2.20. tétel Reed–Solomon-kódjának paritásellenőrző mátrixa*

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha^{n-k} & \alpha^{2(n-k)} & \cdots & \alpha^{(n-1)(n-k)} \end{pmatrix}$$

2.6. Dekódolási algoritmus

A 2.2. szakaszban lineáris kódok táblázatos dekódolását mutattuk be. A táblázat sorainak száma q^{n-k}, azaz a módszer gyakorlati alkalmazhatóságát a paritásszegmens hossza és a kódábécé mérete határozza meg. Ha a táblázat mérete meghaladja a tárkapacitásunkat, nem tudjuk előre tárolni az egyes szindróma értékekhez tartozó javítható hibamintákat, ehelyett a hibamintát a vett szó szindrómájának meghatározása után mindig újra kiszámítjuk. Az általános hibajavító algoritmusok bemutatása meghaladja ezen jegyzet kereteit, ugyanakkor módunk van arra, hogy egyszerűbb esetekben bemutassuk az általános algoritmusok alapvető lépéseit. Az alábbiakban Reed–Solomon-kód esetén egy hiba javításának, illetve a $t \geq 1$ törlés javításának algoritmusát mutatjuk be.

Egy hibát javító ciklikus Reed–Solomon-kód generátorpolinomja

$$g(x) = (x - \alpha)(x - \alpha^2),$$

következésképp tetszőleges $c(x)$ kódszó esetén $c(\alpha) = c(\alpha^2) = 0$. A vett szó $\mathbf{v} = \mathbf{c} + \mathbf{e}$ felbontása polinomos alakban $v(x) = c(x) + e(x)$. Az egy hiba javítás esetét tekintve $e(x) = ex^i$, $e \in GF(q)$, $i \in \{0,1,\ldots,n-1\}$, ahol e a hiba értéke, és i a hiba helye. A 2.21. tételbeli paritásellenőrző-mátrix alapján láthatjuk, hogy $s = (s_1, s_2)$, $s_i \in GF(q)$, $i = 1,2$ szindróma vektor komponenseit ekvivalensen kiszámíthatjuk a következő módon: $s_1 = v(\alpha)$, $s_2 = v(\alpha^2)$. Innen

$$s_1 = v(\alpha) = c(\alpha) + e(\alpha) = 0 + e(\alpha) = e(\alpha) = e \cdot \alpha^i$$
$$s_2 = v(\alpha^2) = c(\alpha^2) + e(\alpha^2) = 0 + e(\alpha^2) = e(\alpha^2) = e \cdot \alpha^{2i},$$

ahonnan

$$e \cdot \alpha^i = s_1$$
$$e \cdot \alpha^{2i} = s_2,$$

egyenletrendszer adódik e, i ismeretlenekben. Így $s_2/s_1 = \alpha^i$, amiből az i hibahelyet kapjuk, majd ezután meghatározzuk az $e = s_1\alpha^{-i}$ hibaértéket.

Törléses hiba esetén ismerjük a hibahelyeket, de továbbra sem ismerjük a hiba értékeit. A dekódolási algoritmus alapja ismét a szindrómákra vonatkozó lineáris egyenletrendszer

$$\begin{aligned} s_1 &= v(\alpha) = e_1 \cdot \alpha^{i_1} + \cdots + e_t \cdot \alpha^{i_t} \\ s_2 &= v(\alpha^2) = e_1 \cdot \alpha^{2i_1} + \cdots + e_t \cdot \alpha^{2i_t} \\ &\vdots \\ s_t &= v(\alpha^t) = e_1 \cdot \alpha^{ti_1} + \cdots + e_t \cdot \alpha^{ti_t} \end{aligned}$$

ahol $e(x) = e_1 \cdot x^{i_1} + \cdots + e_t \cdot x^{i_t}$, továbbá $t \leq n-k$.

2.14. példa. Egy GF(11) feletti $g(x) = (x-2)(x-4)$ generátorpolinomú Reed–Solomon-kód dekóderéhez érkezett vett szóból 2 karakter törlődött:

0.	1.	2.	3.	4.	5.	6.	7.	8.	9.
(8	2	0	0	2	0	0	?	0	?)

Határozzuk meg a törlődött karaktereket!

$u, v \in GF(11)$ ismeretleneket bevezetve az ismeretlen értékekre, a kódszó polinom alakban a következő:

$$c(x) = ux^9 + vx^7 + 2x^4 + 2x + 8.$$

A $c(2) = 0$, $c(4) = 0$ egyenletek alapján:

$$u \cdot 2^9 + v \cdot 2^7 + 2 \cdot 2^4 + 2 \cdot 2 + 8 = 0,$$
$$u \cdot 4^9 + v \cdot 4^7 + 2 \cdot 4^4 + 2 \cdot 4 + 8 = 0,$$

$GF(11)$ feletti egyenletrendszer adódik, amelynek megoldása $u = 0$, $v = 0$.

2.7. Kódkombinációk

A standard kódkonstrukciók során kapott kódok paraméterei nem mindig illeszkednek közvetlenül az adott alkalmazásban megkövetelt értékekhez. Hatékony, ugyanakkor egyszerű módszerek léteznek arra, hogy változtassuk a kódszóhossz, üzenethossz, kódtávolság paraméterek értékét az eredeti konstrukcióhoz képest. Az alábbiakban ezen módszereket tekintjük át röviden.

Kódátfűzés és a csomós hibák javítása

Adott $C(n,k)$ kód m-szeres **átfűzés**ével egy $C^m = C(mn,mk)$ kódot kapunk, olyan módon, hogy a C kód $\mathbf{c}^{(i)}$, $i = 1,\ldots,m$ m darab kódszavát egy $m \times n$ dimenziós mátrixba rendezzük soronként, s a C^m átfűzéses kód $\mathbf{c}$ kódszavát ezen mátrix oszlapainak sorrendben való kiolvasásával képezzük. Azaz a kódszavakat (komponens szavakat) fésű módon egymásba toljuk:

$$\mathbf{c} = \left(c_0^{(1)}, c_0^{(2)}, \ldots, c_0^{(m)}, c_1^{(1)}, c_1^{(2)}, \ldots, c_1^{(m)}, \ldots, c_{n-1}^{(1)}, c_{n-1}^{(2)}, \ldots, c_{n-1}^{(m)}\right) \tag{2.8}$$

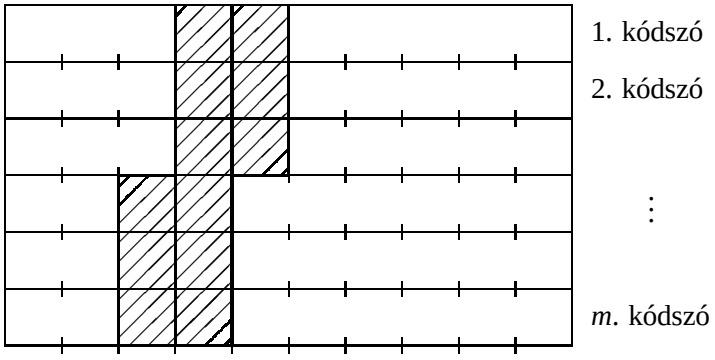
Lineáris kódot átfűzve nyilván lineáris kódot kapunk. Az is könnyen látható, hogy ha d a C kód kódtávolsága, akkor a C^m átfűzéses kód távolsága is d marad. Lineáris C kódot tekintve legyen $\mathbf{c}^{(i)}$, $i = 1,\ldots,m$ sorozat egyik kódszavának súlya d, míg a többi kódszó legyen a zérus kódszó. Általános esetben tekintsük a C kódbeli kódszavak $\mathbf{c}^{(1,i)}$, $i = 1,\ldots,m$, $\mathbf{c}^{(2,i)}$, $i = 1,\ldots,m$ két sorozatát, ahol $\mathbf{c}^{(1,1)}$ és $\mathbf{c}^{(2,1)}$ távolsága d, míg $\mathbf{c}^{(1,i)} = \mathbf{c}^{(2,i)}$, $i = 2,\ldots,m$. (A később bemutatott CD példájában $m = 2, n = 28, k = 24$.)

Ciklikus kódot átfűzve ciklikus kódot kapunk. Legyen S az egyszeri ciklikus jobbra léptetés operátora. Könnyen ellenőrizhető, hogy a (2.8) szerinti $\mathbf{c}$ kódszó $S\mathbf{c}$ ciklikus eltoltja az $S\mathbf{c}^{(m)}, \mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \ldots, \mathbf{c}^{(m-1)}$ sorozat átfűzésének felel meg, s mivel $S\mathbf{c}^{(m)} \in C$, ezért $S\mathbf{c} \in C^m$ is fennáll.

Mint láttuk, a kódtávolság nem változik átfűzés során, ami azt jelenti, hogy a C^m átfűzéses kód szokásos képességei (véletlen hibák javítása, törlésjavítás, detekciós képesség) romlanak az átfűzéssel, hiszen ezen képességek m-szeres kódszóhosszon érvényesek. Ha valaki itt arra gondolna, hogy például az egyes komponensszavak javítóképessége nem változott, s így a teljes javító képesség a komponensek m-szeresének tűnik, az ott hibázik, hogy t javítóképesség azt jelenti, hogy tetszőleges t pozícióban eshet hiba, nem pedig azt, hogy az a komponens szavaknak megfelelően kerül „szétosztásra”. Ezen a ponton felmerül a természetes kérdés: egyáltalán mire jó akkor az átfűzés? A válasz: hibacsomók javítására.

A hibavektor egy l hosszúságú szegmense **hibacsomó** l hosszal, ha a szegmens első és utolsó karaktere nem zérus. Egy kód l hosszúságú hibacsomót javító, ha minden legfeljebb l hosszúságú hibacsomó javítható.

2.22. tétel. *A C^m átfűzéses kód $m \cdot t$ hosszúságú hibacsomót javító, ahol t a C kód javítóképessége.*



2.2. ábra. Kódátfűzés $t = 2$ esetén

BIZONYÍTÁS: A (2.8) szerinti **c** kódszóban egy legfeljebb $m \cdot t$ hosszúságú hibacsomónak megfelelő hibázás a komponens szavakban legfeljebb t számú hibát okozhat, amit azok javítani képesek. (A $t = 2$ esetet szemlélteti a 2.2. ábra.) ■

Egy tetszőleges lineáris $C(n,k)$ kód n,k paramétere alapján a kód l **hibacsomójavító képesség**ére az alábbi egyszerű korlát adható:

2.23. tétel. *Egy $C(n,k)$ lineáris kód l hibacsomójavító képességére fennáll, hogy $l \leq \lfloor \frac{n-k}{2} \rfloor$.*

A tételbeli korlát Reiger-korlát néven ismert. Azokat a hibacsomó javító kódokat, amelyre $l = \lfloor \frac{n-k}{2} \rfloor$ fennáll, **Reiger-optimális**nak hívjuk.

MEGJEGYZÉS: Egy MDS tulajdonságú lineáris kód Reiger-optimális.

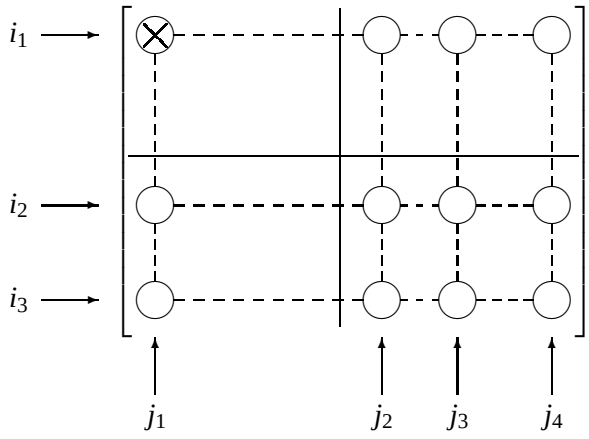
Szorzatkód

Egy $C_1(n_1,k_1,d_1)$ és egy $C_2(n_2,k_2,d_2)$ lineáris kód (komponenskódok) felhasználásával $C_1 \times C_2(n_1 \cdot n_2, k_1 \cdot k_2, d_1 \cdot d_2)$ **szorzatkód**ot készíthetünk, amelynek kódszavai $n_1 \times n_2$ dimenziós mátrixok, ahol a mátrix sorai C_1 kódbeli, oszlopai C_2 kódbeli kódszavak. Szisztematikus komponenskódok esetén a szorzatkódbeli mátrix-kódszó bal felső $k_1 \times k_2$ dimenziós minorja tartalmazza az üzenetet. A mátrix-kódszavakat soronként kiolvasva kapjuk a szorzatkód — soros — kódszavát. A kapott $C_1 \times C_2(n_1 \cdot n_2, k_1 \cdot k_2)$ kód lineáris.

A mátrix-kódszó képzése a következőképp történik. Az első k_1 oszlopot a $C_2(n_2,k_2)$ kód alapján szisztematikus kódolással kapjuk, kiegészítve a k_2 hosszú üzenetszegmenst $n_2 - k_2$ hosszú paritásszegmenssel (2.3. ábra). Az első k_2 sort a $C_1(n_1,k_1)$ kód alapján szisztematikus kódolással kapjuk, kiegészítve a k_1 hosszú üzenetszegmenst $n_1 - k_1$ hosszú paritásszegmenssel. A mátrix jobb alsó sarkába kerül a paritások paritása, amit — mint azt hamarosan belátjuk — képezhetjük akár az első k_1 oszlop, akár az első k_2 sor paritásai alapján szisztematikus kódolással a C_2 illetve C_1 kódbeli szavakkal. A fenti módon képezett szorzatkódot —

$$\left[\begin{array}{cccc|ccc}
c_0^{(0)} & c_1^{(0)} & \cdots & c_{k_1-1}^{(0)} & c_{k_1}^{(0)} & \cdots & c_{n_1-1}^{(0)} \\
c_0^{(1)} & c_1^{(1)} & \cdots & c_{k_1-1}^{(1)} & c_{k_1}^{(1)} & \cdots & c_{n_1-1}^{(1)} \\
\vdots & & & & & & \vdots \\
c_0^{(k_2-1)} & c_1^{(k_2-1)} & \cdots & c_{k_1-1}^{(k_2-1)} & c_{k_1}^{(k_2-1)} & \cdots & c_{n_1-1}^{(k_2-1)} \\
\hline
\vdots & & & & & & \vdots \\
c_0^{(n_2-1)} & c_1^{(n_2-1)} & \cdots & c_{k_1-1}^{(n_2-1)} & c_{k_1}^{(n_2-1)} & \cdots & c_{n_1-1}^{(n_2-1)}
\end{array}\right]$$

2.3. ábra. A szorzat-kódszó képzése

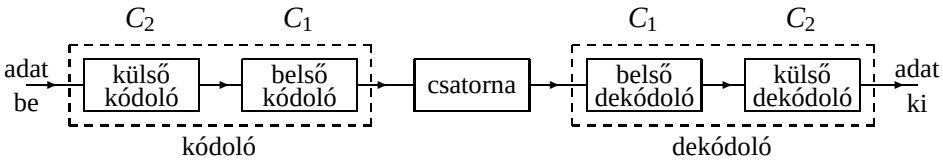


2.4. ábra. A paritások paritásainak képzése

amelynek sorai illetve oszlopai az alapkódok kódszavai — kanonikus elrendezésűnek nevezzük.

A paritások paritásai képzésével kapcsolatos alábbi gondolatmenetünket illusztrálja a 2.4. ábra.

Képezzük azt a $C_1 \times C_2$ kódbeli kódszót, amelynek üzenetmátrixa csak az (i_1, j_1) koordinátájú helyen tartalmaz nullától különböző elemet. Ehhez az üzenethez képezzük a C_2 illetve C_1 kódolás szerint a $C_1 \times C_2$ kódbeli kódszó i_1-edik sorát és j_1-edik oszlopát. A kétdimenziós paritásszegmens jobb felső illetve bal alsó részmátrixa az i_1-edik sor illetve a j_1-edik oszlop kivételével csak 0 elemeket tartalmaz. Innen már egyszerűen látszik, hogy a jobb alsó részmátrixot megkaphatjuk, akár az (i_2, j_1) illetve (i_3, j_1) elemekből C_2 kód szerinti, akár $(i_1, j_2), (i_1, j_3)$ illetve (i_1, j_4) elemekből C_1 kód szerinti kódolással. S miután a $C_1 \times C_2$ kód lineáris, ezért tetszőleges üzenetmátrixú szorzatkód kódszót a 2.4. ábrán is illusztrált elem-kódszavakból koordinátánként vett összeadással képezhetjük.



2.5. ábra. Kaszkád kódoló

Annak igazolása, hogy a szorzatkód kódtávolsága a komponenskódok távolságainak szorzata, a minimális nemzérus súlyú, azaz $d_1 \cdot d_2$ súlyú kódszó előállításával történhet. Válasszunk ehhez egy-egy minimális súlyú kódszót a C_1 illetve C_2 kódból, amelyeket jelöljön $\mathbf{c}'$ illetve $\mathbf{c}''$, ekkor egy minimális súlyú mátrix-kódszó az i-edik sorában a $\mathbf{c}''$ kódszót tartalmazza, ha $\mathbf{c}'$ i-edik komponense 1, egyébként a csupa zérus kódszó kerül a sorba. Az, hogy a kapott kódszó minimális súlyú, onnan látható, hogy ha nem minimális súlyú $\mathbf{c}''$ kódszót helyeznénk el valamelyik sorba, akkor több nemzérus oszlopot kellene elhelyezni a mátrixban a C_1 kódszavai közül és viszont.

2.15. példa. Az egyik legismertebb és egyben legegyszerűbb konstrukciójú hibajavító kód a **kétdimenziós paritáskód**. Ez egy $C \times C$ szorzatkód, ahol a C komponenskód $(n, n-1, 2)$ paraméterű egy paritásbittel rendelkező, egy hibát jelző bináris kód. A kapott szorzatkód kódtávolsága 4, azaz egyszerű paritásbites konstrukcióval 1 hiba javítására vagy 3 hiba jelzésére alkalmas kódot kaptunk.

Kaszkád kódok

Vegyünk egy $C_1(n_1, k_1, d_1)$ GF(q) feletti és egy $C_2(N_2, K_2, D_2)$ GF(q^{k_1}) feletti lineáris kódot, amelyből az alábbi módon generálhatjuk a szisztematikus, $C(n_1N_2, k_1K_2, d)$ paraméterű GF(q) feletti **kaszkád kód** kódszavait. A k_1K_2 hosszú üzenetet osszuk fel K_2, egyenként k_1 hosszú szegmensre. A C_2 kód egy k_1 hosszú üzenetszegmenst egy üzenetkarakternek vesz, és K_2 ilyen karakter alkot számára egy üzenetszegmenst, amelyből N_2 karakter hosszúságú kódszót képez $N_2 - K_2$ paritáskarakternek az üzenethez való illesztésével. A C_2-beli kódszó elkészülte után a kódszó mindegyik koordinátáját a C_1 kód kódolója újra k_1 hosszúságú üzenetként értelmezi, és $n_1 - k_1$ paritáskarakterrel kiegészíti. Így kapjuk az n_1N_2 hosszú kódszót, ami a kaszkád kód adott k_1K_2 hosszú üzenethez tartozó kódszava. A kaszkád kód kódtávolsága $d \geq d_1D_2$.

A C_1 kódot **belső**, a C_2 kódot **külső kód**nak is nevezik. A kód az elnevezését onnan kapta, hogy a külső kód kódolójának és a belső kód kódolójának a kaszkádba kötése képezi a generált kód kódolóját (2.5. ábra).

A dekódolás során először a C_1 kódszavakat dekódoljuk, majd értelemszerűen, a C_1 kódszavai paritásszegmensének törlése után a C_2 kódszó dekódolását végezzük el.

A kaszkád kódok igen alkalmasak az együttes csomós és véletlen hibák javítására, ahol a csomós hibákat a C_2 kód, a véletlen hibákat a C_1 kód javítja elsősorban. A C_2 kód egy karakterének tetszőleges meghibásodása legfeljebb k_1 méretű q-áris hibaszámnak felel meg. Ugyanakkor ritka egyedi hibák javítása C_1-beli kódszavakban könnyen elvégezhető, míg ezen egyedi hibák C_2-beli karakterszintű javítása „pazarlás" lenne.

Rövidített kód

Egy $C(n,k)$ kód **rövidítés**ével egy $C(n-i,k-i)$, $1 \leq i < k$ kódot kapunk olyan módon, hogy a $C(n,k)$ szisztematikus kód kódszavai közül csak azokat hagyjuk meg, amelyek az első i karakterén zérust tartalmazó üzenetekhez rendeltek. Ekkor a $C(n,k)$ kód i karakterrel történő rövidítéséről beszélünk. Mivel a rövidített kód kódszavai a $C(n,k)$ kód kódszavai is egyben, ezért a rövidített kód minimális távolsága legalább akkora, mint az eredeti kódé volt. Praktikusan természetesen a kódoló és a dekódoló úgy van kiképezve, hogy az első i zérus karaktert nem is továbbítjuk, s a dekóder zérusnak tekinti azokat. A kódrövidítés elsődleges célja a kódhossznak az alkalmazásbeli paraméterekhez való igazítása.

2.16. példa. Konstruáljunk kódszórövidítés módszerével 5 bit hosszon 1 bit hiba javítására bináris kódot. Ehhez rövidítsük a $g(x) = x^3 + x + 1$ generátorpolinomú Hamming-kódot. A rövidített kód az alapkód altere, amelynek szavai az eredeti kód rövidítésnek megfelelő számú 0 bittel kezdődő kódszavai. Az alapkód szisztematikus generátormátrixa

$$\mathbf{G} = \begin{pmatrix} 1000101 \\ 0100111 \\ 0010110 \\ 0001011 \end{pmatrix}$$

amelynek alapján a keresett alteret a

$$\mathbf{G} = \begin{pmatrix} 10110 \\ 01011 \end{pmatrix}$$

mátrix generálja, ahonnan a keresett kód szavai:

$$(00000), (10110), (01011), (11101).$$

Paritásbittel bővítés

A paritáskarakterrel történő kiegészítés után a $C(n,k)$ bináris lineáris alapkódból egy $\widehat{C}(n+1,k)$ lineáris kódot kapunk, amelynek a minimális távolsága az alapkód d minimális távolságával azonos, ha d páros, illetve $d+1$ lesz, ha d páratlan.

A $\mathbf{H}_{\widehat{C}}$ paritásmátrix $\mathbf{H}_C$ ismeretében az alábbi alakú:

$$\mathbf{H}_{\widehat{C}} = \begin{pmatrix} 1\,1 & 1 & \cdots & 1 \\ & & & 0 \\ & \mathbf{H}_C & & \vdots \\ & & & 0 \\ & & & 0 \end{pmatrix} \tag{2.9}$$

2.17. példa. Adjuk meg a $C(7,4)$ Hamming-kód $\widehat{C}(8,4)$ paritásbittel bővített kódjának paritásmátrixát. Az x^3+x+1 generátorpolinomú Hamming-kód bővítésével a (2.9) képlet alapján

$$\mathbf{H}_{\widehat{C}} = \begin{pmatrix} 1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,1\,0 \\ 0\,1\,0\,1\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,1\,1\,0 \end{pmatrix}.$$

A kapott $\widehat{C}(8,4)$ kód nyilván nem ciklikus, például a (10001101) a $\widehat{C}$ kódszava, de (11000110) már nem az. A bővített kód minimális távolsága 4, ezért 3 véletlen hiba detektálására alkalmas.

2.8. Hibajavítás és hibajelzés hibavalószínűsége

A bináris kommunikációs csatorna egyik modellje az emlékezetnélküli bináris szimmetrikus csatorna p hibázási valószínűséggel (röviden emlékezetnélküli BSC(p)). Ekkor az egyes bitek átvitele során függetlenül adódnak a meghibásodások, továbbá annak az eseménynek valószínűsége, hogy 0 (1) bit átvitele esetén hibásan 1 (0) bit jelenik meg a csatorna kimenetén, p valószínűségű.

A kódszavak körüli döntési tartományok t sugarú gömbök, ezért a következő felső becslést kapjuk a hibajavító dekódolás kódszó-hibavalószínűségére:

$$P_e \leq 1 - \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i}. \tag{2.10}$$

Ha a kód perfekt, akkor a t sugarú gömbök hézagmentesen kitöltik az n bit hosszú szavak terét, ezért egy t-nél nagyobb Hamming-súlyú hiba mindig téves dekódolásra vezet. Így (2.10) jobb oldali formulája a hibavalószínűség pontos értékét adja. Például Hamming-kód esetén a

$$P_{e,\mathrm{corr}} = 1 - ((1-p)^n + np(1-p)^{n-1}) \tag{2.11}$$

formulát kapjuk. Táblázatos szindróma dekódolás esetén tetszőleges kódra ugyancsak pontosan meg tudjuk adni a hibajavítás hibavalószínűségét a tekintett emlékezetnélküli BSC(p) csatorna esetén. A javítható hibavektorok halmazát jelölje

B, amely perfekt kód esetén a zéró vektor körüli t sugarú gömb, egyéb esetekben ennél bővebb (lásd az alábbi példát). A hibavalószínűség formulája az alábbi:

$$P_{e,\text{det}} = \sum_{c \in B \setminus \{0\}} p^{w(c)} (1-p)^{n-w(c)}. \tag{2.12}$$

Tekintsük most a hibajelzés esetét. Ezesetben pontosan akkor keletkezik hiba, ha a vett szó egy nemzérus kódszóval egyezik meg, amely különbözik a továbbított kódszótól. Lineáris kód esetén ez az esemény ekvivalens azzal az eseménnyel, hogy a hibavektor valamely nemzérus kódszóval egyenlő. Így kapjuk egy C lineáris hibajelző blokk kód és emlékezetnélküli BSC(p) csatorna esetén az alábbi általános formulát a hibajelzés hibavalószínűségére:

$$P_{e,\text{det}} = \sum_{c \in C \setminus \{0\}} p^{w(c)} (1-p)^{n-w(c)}. \tag{2.13}$$

2.18. példa. Tekintsük a következő generátormátrixszal definiált kódot:

$$\mathbf{G} = \begin{pmatrix} 11001 \\ 01110 \end{pmatrix}.$$

A kódot emlékezetnélküli BSC(p) csatornán hibajelzésre illetve hibajavításra használjuk. Adjuk meg mindkét alkalmazás esetén a hibázás valószínűségét!

1. Hibajelzés esete:
   Hibajelzés hibája akkor következik be, ha a hibavektor pontosan egy nemzérus kódszónak felel meg. A (2.13) képletet alkalmazva, tekintettel hogy a kódnak 3 nemzérus kódszava lözül kettő kódszó 3 súlyú, egy 4 súlyú, a következő eredmény adódik: $P_{e,\text{det}} = 2p^3(1-p)^2 + p^4(1-p)$.

2. Hibajavítás esete:
   A standard elrendezési táblázat az alábbi:

00000	11001	01110	10111
00001	11000	01111	10110
00010	11011	01100	10101
00100	11101	01010	10011
01000	10001	00110	11111
10000	01001	11110	00111
00011	11010	01101	10100
10001	01000	11111	00110

   Ezen táblázatban minden lehetséges vett szó fel van sorolva, s oszloponként látható az egyes kódszavak döntési tartománya. Az utolsó két sorban vannak olyan szavak, amelyek azonos távolságban vannak több kódszótól is, s ezek döntési tartományba helyezése önkényes (szerencsére az adott statisztikai csatornamodell erre érzéketlen). Ennek megfelelően dekódolási hiba akkor

keletkezik, ha egy kódszó úgy hibásodik meg, hogy a vett szó már kívül esik a kódszó körüli döntési tartományon. Az egyes kódszavak átküldését azonos valószínűségűnek véve

$$P_e = 1 - ((1-p)^5 + 5p(1-p)^4 + 2p^2(1-p)^3).$$

2.9. Alkalmazások

Teletext kód. A $(7,4)$-es Hamming-kódot egy páratlan paritásúra kiegészítő paritásbittel kapunk egy $(8,4)$ paraméterű, továbbra is egy hibát javító kódot, amelyet a Teletextben használnak.

CRC. A ciklikus kódok gyakorlata a leghosszabb múlttal a hibajelzés területén rendelkezik, amikor a kód bináris, a kódokat a szabványok generátorpolinomjuk segítségével adják meg és generálásuk a 2.19. tétel bizonyításában leírt módon, a generátorpolinom szerinti maradékos osztással, szisztematikusan történik. Ezeket a kódokat **CRC** kódoknak hívják (Cyclic Redundancy Check). A hibajelzést legtöbbször zajos, visszacsatolásos csatornáknál használják, amikor a vevő hiba detektálása esetén értesíti az adót, amely ezután az adást ugyanazzal a kóddal vagy egy jobbal megismétli. Ezt az eljárást **ARQ**-nak nevezzük (Automatic Repeat reQuest).

A CCITT 16 paritásbitet tartalmazó szabványában a CRC generátorpolinomja

$$g_1(x) = x^{16} + x^{12} + x^5 + 1.$$

Ezt a generátorpolinomot alkalmazzák pl. az SNC 2653 (Polinomial Generator Checker), Intel 82586 (Local Communication Controller), Intel 8274 (Multi-Protocol Serial Controller), Signetics 2652 (Multi-Protocol Communications Circuit) integrált áramkörökben. A két utóbbiban még választhatjuk a

$$g_2(x) = x^{16} + x^{15} + x^2 + 1$$

polinomot is. Az Intel 82586-os Ethernet chip tartalmaz egy 32 bites generátorpolinomot is:

$$g_3(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

A 2.17. tétel értelmében ezek a polinomok akkor generátorpolinomjai ciklikus kódoknak, ha osztják az $x^n - 1$ polinomot, tehát csak bizonyos kódszóhosszakra ciklikus kódok. Ugyanakkor erre nem figyelmeztetik a felhasználót. Ez azért nem okoz problémát, mert tetszőleges üzenethossz esetén azért jó kódot kapunk, ugyanis az vagy eleve ciklikus, vagy egy ciklikus kód rövidítése. Legyen C egy (n,k) paraméterű szisztematikus lineáris kód és $k' < k$. Egy

$$\mathbf{u}' = (u'_0, u'_1, \ldots, u'_{k'})$$

üzenethez rendeljük a k hosszú

$$\mathbf{u} = (0,0,\ldots,0,u'_0,u'_1,\ldots,u'_{k'})$$

üzenetet, ahhoz a

$$\mathbf{c} = (0,0,\ldots,0,u'_0,u'_1,\ldots,u'_{k'},c_{k+1},c_{k+2},\ldots,c_n)$$

kódszót és ahhoz a rövidített kódszót:

$$\mathbf{c}' = (u'_0,u'_1,\ldots,u'_{k'},c_{k+1},c_{k+2},\ldots,c_n).$$

Az ilyen $\mathbf{c}'$ kódszavak C' halmazát nevezzük a C kód rövidített kódjának. Nyilván

$$n - n' = k - k'$$

és C' kódtávolsága legalább akkora, mint a C kódé.

Ez utóbbi miatt elég összefoglalni a CRC kódok alapvető tulajdonságait akkor, amikor az ciklikus kód, azaz az n kódszóhosszra a generátorpolinom osztja $x^n - 1$-et.

Ezek után legyen n az a legkisebb természetes szám, melyre $g_1(x) \mid x^n - 1$, és jelölje C az $(n, n-16)$ paraméterű, ciklikus, lineáris kódot, melynek a generátorpolinomja $g_1(x)$, ekkor

1. tulajdonság: $n = 2^{15} - 1 = 32767$.
2. tulajdonság: C jelez minden legfeljebb 3 súlyú hibát.
3. tulajdonság: C jelez minden páratlan súlyú hibát.
4. tulajdonság: C jelez minden olyan hibát, ahol a hibahelyek maximumának és minimumának a távolsága kisebb, mint 16. (Ez utóbbit úgy szokás mondani, hogy a kód jelez minden legfeljebb 16 hosszú hibacsomót.)

Közvetlen műholdas műsorszórás. A közvetlen műholdas műsorszórás (Direct Broadcasting Satellite, DBS) digitalizált hangját is hibajavító kóddal védik. Így a D2-MAC/PACKET szabványa szerint az egyik változatban a 14 bites hangminta felső 11 bitjét egy $(16,11)$ paraméterű kóddal kódolják, ami a $(15,11)$ paraméterű Hamming-kód kiegészítése egy páratlan paritásbittel. A másik változatban a 10 bites hangminta felső 6 bitjét kódolják egy $(11,6)$-os kóddal. Megjegyezzük még, hogy a csomagolt, kódolt beszédmintákat egy olyan csomagfejjel látják el, melyet 2 hibát javító $(71,57)$ ill. $(94,80)$ paraméterű úgynevezett BCH-kóddal védenek, míg a legfontosabb adatokat, az úgynevezett szolgáltatásazonosítást egy három hibát javító $(23,12)$ paraméterű Golay-kóddal kódolják. Ennek a kódnak a kódtávolsága 7, ezért 3 hibát tud javítani. Könnyen ellenőrizhető, hogy a kód paramétereire a Hamming-korlátban az egyenlőség teljesül:

$$\sum_{i=0}^{3} \binom{23}{i} = 2^{11},$$

tehát ez a kód perfekt. Eredetileg Golay a kódot szisztematikus generátormátrixával adta meg:

$$\mathbf{G} = \begin{pmatrix}
1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,1\,0\\
0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,1\\
0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\\
0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,0\\
0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,0\\
0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\\
0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\\
0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,1\,1\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,1\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,0\,0\,0\,1\,0\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1
\end{pmatrix}$$

Kiderült, hogy ez a kód ciklikus is, és a generátorpolinomja

$$g(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1.$$

Ugyanilyen paraméterű kódot kapunk a

$$g'(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$$

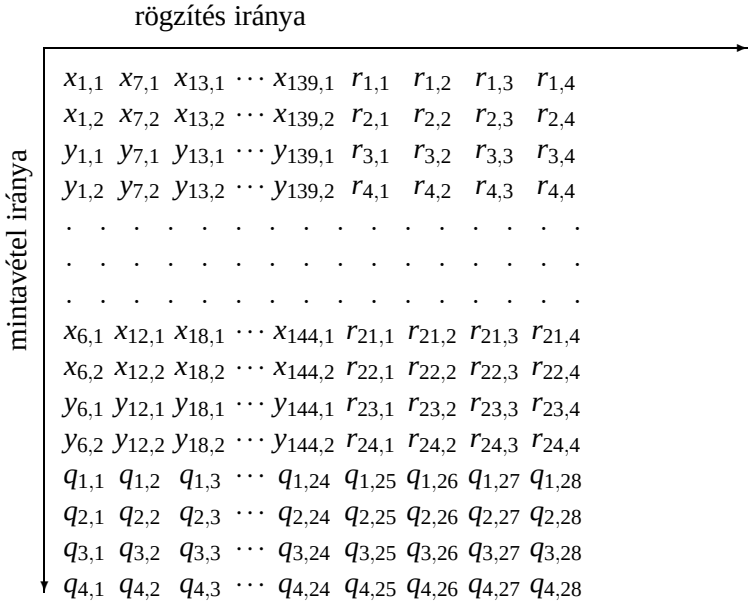
generátorpolinommal. Ezek valóban 23 hosszú ciklikus kódok generátorpolinomjai, ugyanis

$$(x-1)g(x)g'(x) = x^{23} - 1.$$

Mivel a $g(x)$ együtthatói közül 7 darab 1, ezért a minimális súly nem lehet 7-nél nagyobb. Megmutatható, hogy pontosan 7:

2.24. tétel. *A* $(23,12)$ *paraméterű Golay-kód egy 3 hibát javító perfekt, lineáris, ciklikus kód.*

Compact Disc hibavédelme. A digitális hangrögzítésben (CD és DAT) alkalmazott hibavédelem Reed–Solomon-kódra épül. A kódolási eljárás lényegét közelítőleg a következő módon lehet összefoglalni: a 44.1 kHz-cel mintavételezett és 16 bitre kvantált mintákat két bájtban ábrázoljuk, és egy mátrixba írjuk be oszlopfolytonosan.



$$
\begin{array}{cccccccccc}
x_{1,1} & x_{7,1} & x_{13,1} & \cdots & x_{139,1} & r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} \\
x_{1,2} & x_{7,2} & x_{13,2} & \cdots & x_{139,2} & r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} \\
y_{1,1} & y_{7,1} & y_{13,1} & \cdots & y_{139,1} & r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} \\
y_{1,2} & y_{7,2} & y_{13,2} & \cdots & y_{139,2} & r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} \\
\vdots & & & & & & & & \\
x_{6,1} & x_{12,1} & x_{18,1} & \cdots & x_{144,1} & r_{21,1} & r_{21,2} & r_{21,3} & r_{21,4} \\
x_{6,2} & x_{12,2} & x_{18,2} & \cdots & x_{144,2} & r_{22,1} & r_{22,2} & r_{22,3} & r_{22,4} \\
y_{6,1} & y_{12,1} & y_{18,1} & \cdots & y_{144,1} & r_{23,1} & r_{23,2} & r_{23,3} & r_{23,4} \\
y_{6,2} & y_{12,2} & y_{18,2} & \cdots & y_{144,2} & r_{24,1} & r_{24,2} & r_{24,3} & r_{24,4} \\
q_{1,1} & q_{1,2} & q_{1,3} & \cdots & q_{1,24} & q_{1,25} & q_{1,26} & q_{1,27} & q_{1,28} \\
q_{2,1} & q_{2,2} & q_{2,3} & \cdots & q_{2,24} & q_{2,25} & q_{2,26} & q_{2,27} & q_{2,28} \\
q_{3,1} & q_{3,2} & q_{3,3} & \cdots & q_{3,24} & q_{3,25} & q_{3,26} & q_{3,27} & q_{3,28} \\
q_{4,1} & q_{4,2} & q_{4,3} & \cdots & q_{4,24} & q_{4,25} & q_{4,26} & q_{4,27} & q_{4,28}
\end{array}
$$

Nevezetesen egy 24×24-es mátrix oszlopai egymás után következő 6 mintavételi időpontban vett két minta (bal és jobb hangcsatorna) $2 \times 2 = 4$ bájtját tartalmazzák. Ha $x_{i,1}, x_{i,2}$ jelöli a jobb csatorna mintáját az i-edik időpillanatban, és $y_{i,1}, y_{i,2}$ a bal csatornáét, akkor a fenti ábra mutatja a minták beírását a táblázatba. A kapott 24×24-es mátrix minden oszlopát kódoljuk egy $(28, 24)$ paraméterű, GF(2^8) feletti szisztematikus Reed–Solomon-kóddal. A j-edik oszlop paritásbájtjait jelöltük $q_{1,j}, q_{2,j}, q_{3,j}, q_{4,j}$-vel. Ennek a kódnak a kódtávolsága 5, tehát 4 hibát tud jelezni, 2 egyszerű hibát tud javítani és 4 törléses hibát tud javítani. A digitális lemezen előforduló hibák jól modellezhetők egy kétállapotú csatornával. Az egyik állapotot nevezzük JÓ állapotnak, melyben átlagosan 10000–20000 bitideig tartózkodik, és ekkor a hibák előfordulása független egymástól és valószínűsége kb. 10^{-4}. A másik állapotot nevezzük ROSSZ állapotnak, amiben 30–40 bitideig tartózkodik, és ekkor gyakorlatilag használhatatlan a vétel. Ekkor azt mondjuk, hogy a hibázás csomós (burst-ös). Az ilyen csatornák kódolására találták ki a kódátfűzés (interleaving) technikát, amikor az előbbi mátrixot sorfolytonosan olvassák ki, de előtte minden sort kódolnak ugyanazzal a $(28, 24)$ paraméterű Reed–Solomon-kóddal. A j-edik sor paritásbájtjait jelöli $r_{j,1}, r_{j,2}, r_{j,3}, r_{j,4}$. Ennek előnye az, hogy a fizikailag összefüggő, csomós hiba hatását több kódszóra osztja szét.

A Sony és a Philips megegyezett a fentihez hasonló (kicsit bonyolultabb) kódolásban azért, hogy a tömeges digitális hanglemezgyártás elindulhasson. A verseny nyitott viszont a lejátszó készülékben, vagyis a dekódolás terén. A különböző dekódolások igaziból a következő egyszerű eljárás finomításai: számítsuk ki soronként a szindrómát! Ha a szindróma 0, akkor azzal a sorral készen vagyunk. Ha 1 hiba volt, akkor azt kijavítjuk. Ha 2 hiba volt, akkor azt kijavítjuk, és az oszloponkénti javításhoz ezeket a hibahelyeket megjegyezzük, azaz mesterségesen törléses hibákat generálunk. Minden egyéb esetben az egész sort törléses hiba-

ként regisztráljuk. Ezek után oszloponként javítunk, ha ott legfeljebb két törléses hiba volt (emlékeztetünk, hogy 4 törléses hibát képes a rendszer javítani). Ha a hibák száma nagyobb, mint 2, akkor a környező hibátlan mintákból interpolálunk. Látható, hogy a hibajavítás nem használja ki a Reed–Solomon-kód hibajavítási lehetőségeit, aminek elsősorban technológiai okai vannak, mivel a dekódolás bonyolultsága a javítandó hibák számának négyzetével arányos, és itt igen gyorsan kell dekódolni (a forrás sebessége $2 \cdot 44100 \cdot 16 = 1.4112$ Mbit/sec)

2.10. Feladatok

Lineáris blokk-kódok

2.1. feladat. Egy $\{0,1,2\}$ kódábécéjű GF(3) feletti lineáris kód generátormátrixa:

$$\mathbf{G} = \begin{pmatrix} 1021 \\ 0122 \end{pmatrix}$$

Adja meg a kódszavakat, valamint a d minimális távolságot!

2.2. feladat. Egy lineáris bináris kód paritásellenőrző mátrixa
$\mathbf{H} = \begin{pmatrix} 1\,1\,1\,1\,1\,1 \end{pmatrix}$. Adja meg a kód következő paramétereit: n,k,d, kódszavak száma!

2.3. feladat. Egy lineáris bináris blokk-kód generátormátrixa:

$$\mathbf{G} = \begin{pmatrix} 10110 \\ 01101 \end{pmatrix}$$

Adja meg

a) a kód paramétereit: n,k,d,

b) standard elrendezési táblázatát,

c) szindróma dekódolási táblázatát,

d) a kódszó dekódolási hibavalószínűséget emlékezetnélküli BSC(p) esetére!

2.4. feladat. Egy lineáris bináris kód paritásellenőrző mátrixa:

$$\mathbf{H} = \begin{pmatrix} 11100 \\ 10010 \\ 11001 \end{pmatrix}$$

Adja meg a szindróma dekódolási táblázatot!

4. fejezet

Adattömörítés

A 4. és 5. fejezetben a forráskódolásra fogunk koncentrálni. Feltételezzük, hogy a csatorna ideális, vagy másképp, a csatornakód olyan jó, hogy a forráskódolóból kilépő adatsorozat pontosan megegyezik a forrásdekódolóba belépő adatsorozattal.

A forráskódolás lehet veszteségmentes és veszteséges. Veszteségmentes forráskódolás, vagyis az ún. adattömörítés esetében megköveteljük, hogy az eredeti forrásadatok pontosan helyreállíthatók legyenek. Veszteséges forráskódolás esetén csak annyit várunk el, hogy a dekódoló által visszaállított adatok valamilyen értelemben eléggé közel legyenek az eredetihez. Az adott alkalmazástól függ, hogy a kettő közül melyik módszert használjuk. Például egy szöveg tömörítésekor azt szeretnénk, hogy a visszaállított szöveg ne tartalmazzon hibát. Egy kép tömörítésekor viszont megelégszünk azzal, ha a visszaállított kép látványa az eredetihez hasonló minőségű.

Ebben a fejezetben bevezetjük a veszteségmentes adattömörítés fogalmát. Ennek célja az, hogy egy üzenetet, amely egy véges halmaz (forrásábécé) elemeiből álló véges sorozat, egy másik véges halmaz (kódábécé) elemeiből álló sorozattal reprezentáljunk úgy, hogy ez a sorozat a lehető legrövidebb (és belőle az üzenet visszaállítható) legyen. A legismertebb példa erre az, amikor a kódábécé a $\{0,1\}$ halmaz, és egy üzenetet (pl. írott szöveg, fájl) bináris sorozatok formájában kódolunk tárolás, illetve átvitel céljából.

Az első tömörítő eljárás a 19. század közepén Samuel F. B. Morse (1791–1872) által kidolgozott Morse-kód volt. A távírón átküldött betűket ti–tá (rövid–hosszú, mai szóhasználattal bináris) sorozatokkal kódolta. Morse észrevette, hogy mivel az ábécé egyes betűi gyakrabban fordulnak elő, mint mások, ha ezekhez rövidebb sorozatokat rendel, akkor ezzel lerövidítheti az üzenetek átküldéséhez szükséges időt.

A tömörítés minőségét a tömörítési aránnyal jellemezhetjük, ami a tömörített hossznak és az eredeti adatsorozat hosszának az aránya. Mindenki számára világos, hogy a tömörítési aránynak, a tömöríthetőségnek van alsó határa. Az adattömörítés természettörvényét Claude E. Shannon (1916–2001) fedezte fel, amikor kiszá-

mította a tömörítési arány elvi alsó határát, a forrásentrópiát, és megadott olyan kódolási eljárásokat, amelyek ezt az elvi alsó határt elérik.

A következő szakaszokban megismerkedünk az üzenetek egy természetesen adódó kritérium szerinti kódolásával — az egyértelműen dekódolható kódolással —, mely később vizsgálataink középpontjában áll majd. Bevezetjük a diszkrét valószínűségi változó entrópiáját, és megmutatjuk, hogy az entrópia milyen alapvető szerepet játszik a kódolással kapcsolatban. Az elkövetkezőkben mindig azt tartjuk szem előtt, hogy az üzenetek kódja minél rövidebb legyen, ezért a kódok általunk vizsgált fő tulajdonsága az átlagos kódszóhossz lesz. A fejezetet az univerzális forráskódolással és annak gyakorlati alkalmazásaival zárjuk.

4.1. Prefix kódok

Jelöljön X egy diszkrét valószínűségi változót, amely az $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ véges halmazból veszi értékeit. Az X-et a továbbiakban véletlen betűnek, az $\mathcal{X}$ halmazt **forrásábécé**nek, elemeit pedig betűknek nevezzük.

$\mathcal{Y}$ jelöljön egy s elemű $\{y_1, y_2, \ldots, y_s\}$ halmazt. Ezt **kódábécé**nek nevezzük. $\mathcal{Y}^*$ jelölje az $\mathcal{Y}$ elemeiből álló véges sorozatok halmazát. $\mathcal{Y}^*$ elemeit **kódszavak**nak nevezzük. Egy

$$f : \mathcal{X} \to \mathcal{Y}^*$$

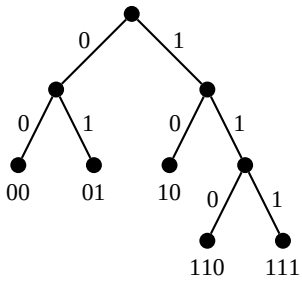
függvényt, amely megfeleltetést létesít a forrásábécé és a kódszavak között, **kód**nak nevezünk. Az $\mathcal{X}$ elemeiből alkotott véges sorozatok az **üzenet**ek vagy közlemények (értékeiket az $\mathcal{X}^*$ halmazból veszik). Amennyiben az f kód értékkészlete különböző hosszú kódszavakból áll, úgy változó szóhosszúságú kódolásról beszélünk.

4.1. definíció. *Az $f : \mathcal{X} \to \mathcal{Y}^*$ kód **egyértelműen dekódolható**, ha minden véges kódbetűsorozat legfeljebb egy közlemény kódolásával állhat elő, azaz ha $\mathbf{u} \in \mathcal{X}^*$, $\mathbf{v} \in \mathcal{X}^*$, $\mathbf{u} = u_1 u_2 \ldots u_k$, $\mathbf{v} = v_1 v_2 \ldots v_m$, $\mathbf{u} \neq \mathbf{v}$, akkor $f(u_1) f(u_2) \ldots f(u_k) \neq f(v_1) f(v_2) \ldots f(v_m)$. (Itt az $f(u) f(u')$ a két kódszó egymás után írását [konkatenáció] jelenti.)*

MEGJEGYZÉS:

a) Az egyértelmű dekódolhatóság több, mint az invertálhatóság. Ugyanis legyen $\mathcal{X} = \{a, b, c\}, \mathcal{Y} = \{0, 1\}$ és $f(a) = 0, f(b) = 1, f(c) = 01$. Ekkor az $f : \mathcal{X} \to \mathcal{Y}^*$ leképezés invertálható, viszont a 01 kódszót dekódolhatjuk $f(a) f(b) = 01$ szerint *ab*-nek, vagy $f(c) = 01$ szerint *c*-nek is.

b) Az előbbi definícióban szereplő kódolási eljárást, amikor egy közlemény kódját az egyes forrásbetűkhöz rendelt kódszavak sorrendben egymás után írásával kapjuk, betűnkénti kódolásnak nevezzük. Természetesen a kódolást teljesen általánosan egy $g : \mathcal{X}^* \to \mathcal{Y}^*$ függvénnyel is definiálhatnánk, de belátható, hogy



4.1. ábra. Egy prefix kód kódfája

a betűnkénti kódolás és annak természetes kiterjesztése, a blokk-kódolás elegendő számunkra, azaz a tömörítési arány minimuma elérhető blokkonkénti kódolással.

c) Az egyértelmű dekódolhatóság végtelen sok eset ellenőrzését igényli, ezért a gyakorlatban használhatatlan követelmény.

4.2. definíció. *Az f kód **prefix**, ha a lehetséges kódszavak közül egyik sem folytatása a másiknak, vagyis bármely kódszó végéből bármekkora szegmenst levágva nem kapunk egy másik kódszót.*

Egy prefix kód ábrázolható az ún. kódfával (4.1. ábra), amelynek levelei jelentik az üzeneteket. A gyökértől a levelekig vezető éleken szereplő kódbetűket összeolvasva kaphatók meg a kódszavak. A prefix kódok egyben egyértelműen dekódolhatóak is, mivel egy adott kódszó karakterei a kódfa gyökerétől indulva egyértelmű utat jelölnek ki egy levélig, s ez lesz a dekódolt üzenet.

Az eddig bevezetett fogalmak illusztrálására nézzük a következő példákat:

4.1. példa. $\mathcal{X} = \{a,b,c\}$; $\mathcal{Y} = \{0,1\}$ A kód legyen a következő: $f(a) = 0,\ f(b) = 10,\ f(c) = 11$. Könnyen ellenőrizhető, hogy a kód prefix.

Ha az *abccab* üzenetet kódoljuk, akkor a 0101111010 kódbetűsorozatot kapjuk. A kódból az üzenet visszafejtése nagyon egyszerű a prefix tulajdonság miatt; többek között ez a gyors dekódolási lehetőség teszi vonzóvá a prefix kódokat.

4.2. példa. $\mathcal{X} = \{a,b,c,d\}$; $\mathcal{Y} = \{0,1\}$; $f(a) = 0,\ f(b) = 01,\ f(c) = 011,\ f(d) = 0111$. Jól látható, hogy a kód nem prefix, de egyértelműen dekódolható, hiszen a 0 karakter egy új kódszó kezdetét jelzi.

Bebizonyítható, hogy minden egyértelműen dekódolható kódhoz létezik vele ekvivalens (azonos kódszóhosszú) prefix kód, tehát nem vesztünk semmit, ha az egyértelmű dekódolhatóság helyett a speciálisabb, és ezért könnyebben kezelhető prefix tulajdonságot követeljük meg.

4.2. Átlagos kódszóhossz, entrópia

Legyen X egy $\mathcal{X}$ értékű valószínűségi változó, amit kódolni akarunk. Vezessük be a következő $p:\ \mathcal{X}\to[0,1]$ függvényt:

$$p(x)=\mathbf{P}\{X=x\},\qquad x\in\mathcal{X}.$$

A $p(x)$ függvény tehát az x forrásbetűhöz annak valószínűségét rendeli.

4.3. definíció. *Az X valószínűségi változó **entrópiá**ját, $H(X)$-et, a*

$$H(X)=\mathbf{E}(-\log p(X))=-\sum_{i=1}^{n}p(x_i)\log p(x_i).$$

összeggel definiáljuk.

MEGJEGYZÉS: A $\log z$ jelölés a z pozitív szám kettes alapú logaritmusát jelenti. Mivel a 4.3. definíció megkívánja, a logaritmusfüggvénnyel kapcsolatban a következő „számolási szabályokat” vezetjük be ($a\geq 0, b>0$):

$$0\log\frac{0}{a}=0\log\frac{a}{0}=0;\qquad b\log\frac{b}{0}=+\infty;\qquad b\log\frac{0}{b}=-\infty.$$

(E szabályok az adott pontban nem értelmezett függvények folytonos kiterjesztései.) A definícióból közvetlenül látszik, hogy az entrópia *nemnegatív*. Vegyük észre, hogy az entrópia értéke valójában nem függ az X valószínűségi változó értékeitől, csak az eloszlásától.

4.4. definíció. *Egy f kód **átlagos kódszóhossz**án az*

$$\mathbf{E}|f(X)|=\sum_{i=1}^{n}p(x_i)|f(x_i)|$$

várható értéket értjük.

4.3. példa. A 4.1. példa kódja esetén legyen $p(a)=0.5,\ p(b)=0.3, p(c)=0.2$, ekkor az entrópia

$$H(X)=-0.5\cdot\log 0.5-0.3\cdot\log 0.3-0.2\cdot\log 0.2\approx 1.485,$$

az átlagos kódszóhossz pedig

$$\mathbf{E}|f(X)|=0.5\cdot 1+0.3\cdot 2+0.2\cdot 2=1.5.$$

Bebizonyítható, hogy az entrópiával alsó korlát adható az átlagos kódszóhosszra:

4.1. tétel. *Tetszőleges egyértelműen dekódolható $f:\ \mathcal{X}\to\mathcal{Y}^*$ kódra*

$$\mathbf{E}|f(X)|\geq\frac{H(X)}{\log s}. \tag{4.1}$$

Az entrópia néhány tulajdonsága: Legyenek X és Y valószínűségi változók, amelyek a véges $\mathcal{X}$ illetve $\mathcal{Y}$ halmazból veszik értékeiket.

4.2. tétel.

a) Ha az X valószínűségi változó n különböző értéket vehet fel pozitív valószínűséggel, akkor

$$0 \leq H(X) \leq \log n,$$

és a bal oldalon egyenlőség akkor és csak akkor áll fenn, ha X 1-valószínűséggel konstans, a jobb oldalon pedig akkor és csak akkor, ha X egyenletes eloszlású, azaz $p(x_i) = \frac{1}{n},\ i = 1,\ldots,n$.

b) X és Y diszkrét valószínűségi változókra

$$H(X,Y) \leq H(X) + H(Y),$$

és az egyenlőség szükséges és elégséges feltétele X és Y függetlensége. ($H(X,Y)$ az (X,Y) valószínűségi változópár együttes eloszlásához rendelt entrópiát jelöli.)

c) Az X tetszőleges $g(X)$ függvényére

$$H(g(X)) \leq H(X),$$

és itt az egyenlőség szükséges és elégséges feltétele az, hogy g invertálható legyen.

4.3. Shannon–Fano-kód

Miután láttuk, hogy egy X valószínűségi változó egyértelműen dekódolható kódjának átlagos kódszóhosszára a $\frac{H(X)}{\log s}$ mennyiség alsó korlátot ad, most megmutatjuk, hogy prefix kóddal ezt a korlátot jól meg lehet közelíteni.

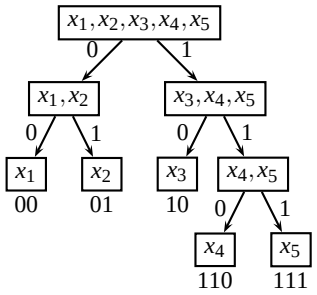
A **Shannon–Fano-kód** kostrukciójához feltehetjük, hogy

$$p(x_1) \geq p(x_2) \geq \cdots \geq p(x_{n-1}) \geq p(x_n) > 0,$$

mert különben az $\mathcal{X}$ elemeinek átindexelésével elérhetjük ezt. A konstrukció technikája miatt felhasználjuk az $y_i \mapsto i-1,\ i = 1,\ldots,s$ megfeleltetést. Legyenek a w_i számok a következők:

$$w_1 = 0, \qquad w_i = \sum_{l=1}^{i-1} p(x_l), \qquad i = 2,\ldots,n.$$

Kezdjük a w_j számokat felírni s-alapú számrendszerbeli tört alakban. A w_j-hez tartozó törtet olyan pontosságig írjuk le, ahol az először különbözik az összes többi w_i, $i \neq j$ szám hasonló alakbeli felírásától. Miután ily módon n darab véges hosszú



4.2. ábra. A 4.4. példa Shannon–Fano-kódjának előállítása

történt kapunk, az $f(x_j)$ legyen a w_j-hez tartozó tört az egészeket reprezentáló nulla nélkül.

A Shannon–Fano-kódot előállíthatjuk egy másik algoritmussal is. Osszuk fel a szimbólumok valószínűségeinek csökkenő sorozatát két részre úgy, hogy a sorozat első felében lévő valószínűségek összege és a sorozat második felében lévő valószínűségek összege a lehető legkevésbé térjen el egymástól. Legyen az első részben lévő szimbólumok kódszavának első bitje 0, a második részben lévőké pedig 1. Ezután alkalmazzuk ezt az eljárást rekurzívan az első és a második részben lévő szimbólumokra, s így megkapjuk a kódszavak további bitjeit. Az algoritmus akkor ér véget, ha már minden részben csak egyetlen szimbólum szerepel.

4.4. példa. Legyen $n = 5$ és $p(x_1) = 0.35,\ p(x_2) = 0.2,\ p(x_3) = 0.2,\ p(x_4) = 0.15,\ p(x_5) = 0.1$ és $\mathcal{Y} = \{0,1\}$. Keressük meg az ehhez tartozó Shannon–Fano-kódot. Használjuk a második algoritmust. A valószínűségek két részre osztása után az első részben az x_1 és x_2 szimbólumok lesznek ($p(x_1)+p(x_2) = 0.55$), a második részben pedig az x_3, x_4 és x_5 szimbólumok ($p(x_3)+p(x_4)+p(x_5) = 0.45$). Így az x_1 és x_2 szimbólumok kódszavának első bitje 0, míg az x_3, x_4 és x_5 szimbólumoké 1 lesz. Az x_1 és x_2 részt kettéosztva elkészültünk ezek kódszavaival: az x_1-é 00, az x_2-é pedig 01 lesz. A második részt ismét kettéosztva az x_3 egyedül marad, így ennek kódszava 10, az x_4 és x_5 kódszavát pedig egy újabb kettéosztás után kapjuk: x_4 kódszava 110, míg x_5-é 111 lesz.

Könnyedén belátható, hogy az így kapott kód prefix, mivel kódfával ábrázolható.

4.3. tétel. *Létezik olyan $f : \mathcal{X} \to \mathcal{Y}^*$ prefix kód, mégpedig a Shannon–Fano-kód, amelyre*

$$\mathbf{E}|f(X)| < \frac{H(X)}{\log s} + 1.$$

Blokk-kódolás

A betűnkénti kódolásnak van egy természetes általánosítása, a **blokk-kódolás**. Ezt formálisan egy $f : \mathcal{X}^m \to \mathcal{Y}^*$ leképezéssel definiálhatjuk, ahol tehát a forrásábécé betűiből alkotott rendezett m-eseket tekintjük forrásszimbólumoknak és ezeknek feleltetünk meg kódszavakat. A helyzet tulajdonképpen nem változik a betűnkénti kódolás esetéhez képest, hiszen egy új $\widehat{\mathcal{X}}$ forrásábécét definiálhatunk az $\widehat{\mathcal{X}} = \mathcal{X}^m$ jelöléssel, és az eddig elmondottak mind érvényben maradnak. Az egyértelmű dekódolhatóság definíciója ugyanaz marad, mint a betűnkénti kódolás esetében, az előbbieket szem előtt tartva. Legyen $\mathbf{X} = (X_1, \ldots, X_m)$ egy valószínűségi vektorváltozó, melynek koordinátái az $\mathcal{X}$-ből veszik értékeiket. Az entrópia 4.3. definíciójából jól látszik, hogy az csakis az eloszlástól függ. Mivel $\mathbf{X}$ is csak véges sok különböző értéket vehet fel, a 4.3. definíciót közvetlenül alkalmazhatjuk. Az $\mathbf{X}$ entrópiája tehát a

$$p(\mathbf{x}) = p(x_1, \ldots, x_m) = \mathbf{P}\{X_1 = x_1, \ldots, X_m = x_m\}, \qquad x_1, \ldots, x_m \in \mathcal{X},$$

jelölést bevezetve a következő:

$$\begin{aligned} H(\mathbf{X}) &= -\sum_{\mathbf{x} \in \mathcal{X}^m} p(\mathbf{x}) \log p(\mathbf{x}) = \\ &= -\sum_{x_1 \in \mathcal{X}} \cdots \sum_{x_m \in \mathcal{X}} p(x_1, \ldots, x_m) \log p(x_1, \ldots, x_m). \end{aligned}$$

Az $\mathbf{X} = (X_1, \ldots, X_m)$ entrópiájára a $H(\mathbf{X})$ jelölés mellett, ahol ez a célszerűbb, gyakran a $H(X_1, \ldots, X_m)$ jelölést fogjuk használni.

A betűnkénti átlagos kódszóhossz definíciója értelemszerűen módosul a blokk-kódolás esetében: a kódszóhossz várható értékét el kell osztani az egy blokkot alkotó forrásbetűk számával, vagyis a betűnkénti átlagos kódszóhosszon a következő mennyiséget értjük:

$$\frac{1}{m}\mathbf{E}|f(\mathbf{X})| = \frac{1}{m}\sum_{\mathbf{x} \in \mathcal{X}^m} p(\mathbf{x})|f(\mathbf{x})|$$

Értelemszerűen a 4.1. tétel állítása blokk-kódolás esetén is igaz:

$$\frac{1}{m}\mathbf{E}|f(\mathbf{X})| \geq \frac{\frac{1}{m}H(\mathbf{X})}{\log s}.$$

Másrészről a 4.3. tételből következik, hogy a Shannon–Fano-kódra

$$\frac{1}{m}\mathbf{E}|f(\mathbf{X})| \leq \frac{\frac{1}{m}H(\mathbf{X})}{\log s} + \frac{1}{m}.$$

Az m blokkhossz növelésével általában javul az adattömörítés hatékonysága, azaz csökken az egy betűre jutó átlagos kódszóhossz. Példaként tekintsük az irodalmi angol szöveg tömörítésének problémáját. Csak a kis betűket és a legfontosabb írásjeleket megtartva kaphatunk egy 32 elemű ábécét, vagyis tömörítés nélkül

egy betűt 5 bittel tudunk reprezentálni. Ha becsüljük az X_1 eloszlását, akkor a $H(X_1)$-re körülbelül 4.03 adódik, tehát $m = 1$ esetén a Shannon–Fano-kódra a

$$4.03 \leq \mathbf{E}|f(X_1)| \leq 4.03 + 1 = 5.03$$

adódik, ami még nem előrelépés a tömörítetlen 5 bithez képest. $m = 2$ esetén $H(X_1, X_2)/2 \approx 3.32$, tehát

$$3.32 \leq \frac{1}{2}\mathbf{E}|f(X_1, X_2)| \leq 3.32 + \frac{1}{2} = 3.82,$$

$m = 3$ esetén

$$3.1 \leq \frac{1}{3}\mathbf{E}|f(X_1, X_2, X_3)| \leq 3.43,$$

és $m = 4$ esetén

$$2.8 \leq \frac{1}{4}\mathbf{E}|f(X_1, X_2, X_3, X_4)| \leq 3.05.$$

4.5. definíció. *Az $\mathbb{X}$ információforrást az $X_1, X_2, \ldots$ valószínűségi változók végtelen sorozatával modellezzük, azaz a forrás az i-edik időpillanatban az X_i jelet bocsátja ki. Az X_i valószínűségi változók mindegyike ugyanabból a véges $\mathcal{X} = \{x_1, \ldots, x_n\}$ halmazból, a forrásábécéből veszi az értékét. Az $\mathbb{X}$ forrást statisztikai tulajdonságaival jellemezzük, vagyis adottnak vesszük, ha minden véges dimenziós eloszlását ismerjük.*

Információforrások egy tág osztályára megmutatható, hogy az $\frac{1}{m}H(X_1, \ldots, X_m)$ betűnkénti entrópia monoton csökken, tehát az m növelésével az alsó határ csökken.

Markov-láncok

A következőkben egy olyan struktúrát, az ún. Markov-láncot ismerjük meg, amellyel hatékonyan leírhatók egyes redundáns karaktersorozatok, és szemléltethető a blokkonkénti kódolás hatékonysága.

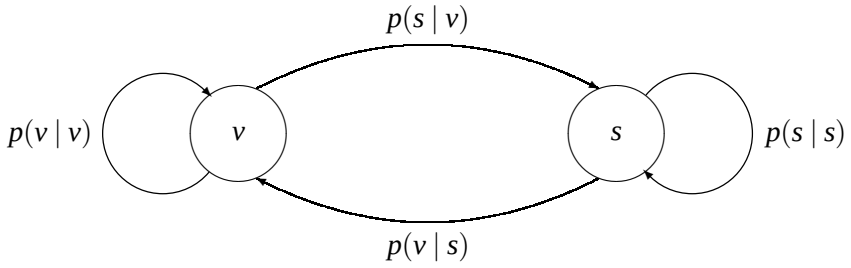
Az $X_1, X_2, \ldots$ valószínűségi változók Markov-láncot alkotnak, ha

$$\mathbf{P}\{X_k = x_k \mid X_1 = x_1, X_2 = x_2, \ldots, X_{k-1} = x_{k-1}\} = \mathbf{P}\{X_k = x_k \mid X_{k-1} = x_{k-1}\} \quad (4.2)$$

minden $k \geq 2$-re és minden lehetséges $x_1, x_2, \ldots, x_k$ sorozatra. A X_i értékeit a Markov-lánc állapotainak, az állapotok $\mathcal{X}$ halmazát pedig a Markov-lánc állapotterének nevezzük. Feltesszük, hogy $|\mathcal{X}| < \infty$, vagyis az állapottér véges. Az $X_1, X_2, \ldots$ Markov-lánc homogén, ha az ún. egylépéses átmenetvalószínűségek nem függenek a változók indexétől, vagyis

$$\mathbf{P}\{X_k = x_2 \mid X_{k-1} = x_1\} = \mathbf{P}\{X_2 = x_2 \mid X_1 = x_1\}$$

minden $x_1, x_2 \in \mathcal{X}$-re és minden $k \geq 2$-re.



4.3. ábra. Egy fekete-fehér kép Markov-lánc modellje

Egy Markov-lánc eloszlása az alábbi láncszabály alapján megadható a kezdeti eloszlás és az egylépéses átmenetvalószínűségek segítségével:

$$\begin{aligned}
&\mathbf{P}\{X_m = x_m, \ldots, X_1 = x_1\} = \\
&= \mathbf{P}\{X_m = x_m \mid X_{m-1} = x_{m-1}, \ldots, X_1 = x_1\} \cdots \mathbf{P}\{X_2 = x_2 \mid X_1 = x_1\}\mathbf{P}\{X_1 = x_1\} = \\
&= \mathbf{P}\{X_m = x_m \mid X_{m-1} = x_{m-1}\} \cdots \mathbf{P}\{X_2 = x_2 \mid X_1 = x_1\}\mathbf{P}\{X_1 = x_1\} = \\
&= \prod_{i=2}^{m} p(x_i \mid x_{i-1}) p(x_1).
\end{aligned}$$

A $\mathbf{P}\{X_k = x_2 \mid X_{k-1} = x_1\}$ átmenetvalószínűségek ismeretében a Markov-modell jól használható szövegek és képek tömörítésére. Angol nyelvű szövegek entrópiájának meghatározására elsőként Shannon végzett kísérleteket. A nyelvi redundanciát modellezte Markov-láncokkal. A 26 betűs angol ábécét használó szövegek esetén másodrendű Markov-lánc modell használatával 3.1 bit/betű tömörítési arányt ért el. Ez 2.4 bit/betű-re szorítható le, ha betűk helyett szavak szerepelnek a modellben. Az angol nyelv entrópiájának becslésére Shannon kísérleti személyeket is alkalmazott a statisztikai modellek helyett. A résztvevőknek a szöveg aktuális betűjét kellett kitalálniuk a megelőző 100 betűs környezet alapján. Így alsó határnak 0.6 bit/betű, míg felsőnek 1.3 bit/betű adódott. Minél hosszabb környezetet vizsgálunk, annál jobb a jóslás eredménye, azonban a környezet hosszával exponenciálisan növekszik a lehetséges megelőző állapotok száma.

4.5. példa. Egy fekete-fehér kép sorait modellezzük egy $\{X_i\}$ homogén Markov-lánccal a 4.3. ábrán látható módon. A v állapot jelöli azt, hogy az aktuális képpont világos, míg az s, hogy sötét. $p(v)$ a világos, $p(s)$ a sötét képpont kezdeti valószínűsége. Az átmenetvalószínűségek közül például $p(v \mid s)$ jelenti annak az eseménynek a valószínűségét, amikor egy sötét képpont után egy világos következik.

Számítsuk ki egy tetszőleges Markov-lánc entrópiáját:

$$H(X_1, \ldots, X_m) = -\sum_{x_1, \ldots, x_m} p(x_1, \ldots, x_m) \log p(x_1, \ldots, x_m) =$$

$$
\begin{aligned}
&= -\sum_{x_1,\ldots,x_m} p(x_1,\ldots,x_m)\log\prod_{i=2}^{m} p(x_i \mid x_{i-1})p(x_1) = \\
&= -\sum_{i=2}^{m}\sum_{x_1,\ldots,x_m} p(x_1,\ldots,x_m)\log p(x_i \mid x_{i-1}) \\
&\quad -\sum_{x_1,\ldots,x_m} p(x_1,\ldots,x_m)\log p(x_1) = \\
&= -\sum_{i=2}^{m}\sum_{x_{i-1},x_i} p(x_{i-1},x_i)\log p(x_i \mid x_{i-1}) - \sum_{x_1} p(x_1)\log p(x_1) = \\
&= -(m-1)\sum_{x_1,x_2} p(x_1,x_2)\log p(x_2 \mid x_1) - \sum_{x_1} p(x_1)\log p(x_1) = \\
&= (m-1)H(X_2 \mid X_1) + H(X_1).
\end{aligned}
$$

A $-\sum_{x_1,x_2} p(x_1,x_2)\log p(x_2 \mid x_1)$ mennyiséget feltételes entrópiának nevezzük, és $H(X_2 \mid X_1)$-gyel jelöljük. A betűnkénti entrópia ekkor így alakul:

$$\frac{1}{m}H(X_1,\ldots,X_m) = \frac{1}{m}H(X_1) + \frac{m-1}{m}H(X_2 \mid X_1).$$

Nagy m blokkhossz esetén az első tag elhanyagolható, ezért a betűnkénti entrópia közelítőleg megegyezik a $H(X_2 \mid X_1)$ feltételes entrópiával.

4.6. példa. Legyenek a 4.5. példa valószínűségei a következőek:

$$p(s \mid s) = 0.77, \quad p(v \mid s) = 0.23, \quad p(s \mid v) = 0.02, \quad p(v \mid v) = 0.98$$

Megmutatható, hogy ha

$$p(s) = 0.08, \quad p(v) = 0.92,$$

akkor a Markov-lánc azonos eloszlású. Ennek az eloszlásnak az entrópiája:

$$H(X_1) = -0.92\log 0.92 - 0.08\log 0.08 = 0.402$$

Ugyanez a modellünk szerint:

$$
\begin{aligned}
H(X_2 \mid X_1) &= -\sum_{x_1}\sum_{x_2} p(x_2 \mid x_1)p(x_1)\log p(x_2 \mid x_1) = \\
&= -p(s \mid s)p(s)\log p(s \mid s) - p(v \mid s)p(s)\log p(v \mid s) \\
&\quad -p(s \mid v)p(v)\log p(s \mid v) - p(v \mid v)p(v)\log p(v \mid v) = \\
&= -0.77\cdot 0.08\cdot\log 0.77 - 0.23\cdot 0.08\cdot\log 0.23 \\
&\quad -0.02\cdot 0.92\cdot\log 0.02 - 0.98\cdot 0.92\cdot\log 0.98 = \\
&= 0.192
\end{aligned}
$$

Látható, hogy kevesebb, mint a felére csökkent az entrópia az átmenetvalószínűségek ismerete miatt.

4.4. Optimális kódok, bináris Huffman-kód

A 4.1. tétel alsó korlátot ad az egyértelműen dekódolható kódok átlagos kódszóhosszára, a 4.3. tétel pedig mutat egy olyan kódkonstrukciót, ahol ezt a korlátot jól megközelíthetjük. A jó kód konstrukciójának problémáját persze ennél általánosabban is felvethetjük: konstruáljuk meg az optimális, azaz legkisebb átlagos kódszóhosszú kódot, ha adott az X valószínűségi változó eloszlása. Először is gondoljuk át, hogy optimális kód valóban létezik. Ugyan véges kódábécé esetén is az egyértelműen dekódolható vagy prefix kódok halmaza végtelen, de a bizonyos átlagos kódszóhossznál (pl. $\frac{H(X)}{\log s}+1$) jobb kódok halmaza véges. Másodszor, vegyük észre, hogy az optimális kód nem feltétlenül egyértelmű; egyenlő valószínűségekhez tartozó kódszavakat felcserélhetünk, csakúgy, mint az egyenlő hosszú kódszavakat, anélkül, hogy az átlagos kódszóhosszat ezzel megváltoztatnánk.

Egy optimális egyértelműen dekódolható kód konstruálását egy optimális prefix kód konstruálására lehet visszavezetni. Ezért a következőkben kimondjuk az optimális prefix kódok néhány tulajdonságát. A továbbiakban az egyszerűség kedvéért a bináris, $s=2$ esettel foglalkozunk; feltesszük, hogy $\mathcal{Y}=\{0,1\}$. Az általános, $s>2$ eset bonyolultabb, és a dolog lényege így is jól látható.

4.4. tétel. *Ha az $f:\mathcal{X}\to\{0,1\}^*$ prefix kód optimális, és $\mathcal{X}$ elemei úgy vannak indexelve, hogy $p(x_1)\geq p(x_2)\geq\cdots\geq p(x_{n-1})\geq p(x_n)>0$, akkor feltehető, hogy f-re a következő három tulajdonság teljesül:*

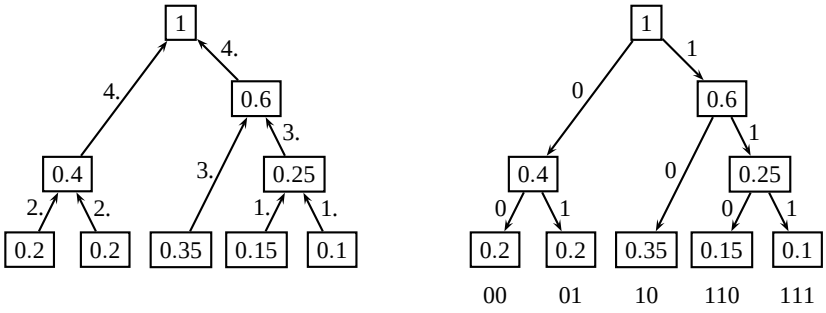
a) $|f(x_1)|\leq|f(x_2)|\leq\cdots\leq|f(x_{n-1})|\leq|f(x_n)|$, *vagyis nagyobb valószínűségekhez kisebb kódszóhosszak tartoznak.*

b) $|f(x_{n-1})|=|f(x_n)|$, *vagyis a két legkisebb valószínűségű forrásbetűhöz tartozó kódszó egyenlő hosszú.*

c) Az $f(x_{n-1})$ és az $f(x_n)$ kódszavak csak az utolsó bitben különböznek.

4.5. tétel. *Tegyük most fel, hogy a 4.4. tétel feltételei teljesülnek, és hogy a $\{p(x_1),p(x_2),\ldots,p(x_{n-1})+p(x_n)\}$ valószínűségeloszláshoz ismerünk egy g optimális bináris prefix kódot. (Az x_{n-1} és x_n forrásbetűket összevonjuk egy $\bar{x}_{n-1}$ szimbólumba; $p(\bar{x}_{n-1})=p(x_{n-1})+p(x_n)$). Ekkor az eredeti $\{p(x_1),p(x_2),\ldots,p(x_{n-1}),p(x_n)\}$ eloszlás egy optimális f prefix kódját kapjuk, ha a $g(\bar{x}_{n-1})$ kódszót egy nullával, illetve egy egyessel kiegészítjük (a többi kódszót pedig változatlanul hagyjuk).*

Az előző tétel alapján már megadhatjuk az optimális prefix kód Huffman-féle konstrukcióját: A két legkisebb valószínűség összevonásával addig redukáljuk a problémát, amíg az triviális nem lesz, vagyis amíg összesen két valószínűségünk marad. Ezt $n-2$ lépésben érhetjük el. Ezután az összevonások megfordításával, mindig a megfelelő kódszó kétféle kiegészítésével újabb $n-2$ lépésben felépítjük az optimális kódot, a Huffman-kódot. Vegyük észre, hogy ez egy ún. mohó algoritmus, azaz az egyes lépésekben mindig lokálisan optimális döntést hoz.



4.4. ábra. A 4.7. példa Huffman-kódjának előállítása

A Huffman-kód prefix, ezért azt bináris faként is ábrázolhatjuk. A leveleket a forrásszimbólumokkal címkézzük, az éleken pedig a kódábécé $(\{0,1\})$ elemei szerepelnek. A csúcsokban a szimbólumok valószínűségei állnak. Egy forrásszimbólumhoz tartozó kódszót úgy kapunk meg, hogy a fa gyökerétől a megfelelő levélig húzódó út élein szereplő kódbetűket sorban összeolvassuk (konkatenáljuk). A Huffman-kódolás szemléletesen úgy történik, hogy kiindulásként felvesszük a forrásszimbólumokhoz tartozó leveleket, a csúcsokba a forrásszimbólumok valószínűségeit írjuk, majd lépésenként mindig a két legkisebb értéket tartalmazó csúcs fölé teszünk egy új csúcsot (szülőt), s ebbe a két régi érték összegét írjuk. Az eljárás végén kialakul az összefüggő fa, melynek a legutolsó lépésben megkapott csúcsa lesz a gyökér.

```
INPUT: lista={az n féle szimbólumnak megfelelő
              izolált csúcsok}

FOR i=1 to n-1
   csúcs=Csúcsot_Felvesz()
   csúcs.bal=Legkisebb_Értékű_Csúcsot_Kivesz(lista)
   csúcs.jobb=Legkisebb_Értékű_Csúcsot_Kivesz(lista)
   csúcs.valószínűség=
      (csúcs.bal).valószínűség+(csúcs.jobb).valószínűség
   Beilleszt(lista,csúcs)
ENDFOR
RETURN Legkisebb_Értékű_Csúcsot_Kivesz(lista)
```

A következő példában nyomon követhetjük az algoritmus egyes lépéseit.

4.7. példa. Legyen $n = 5$ és $p(x_1) = 0.35,\ p(x_2) = 0.2,\ p(x_3) = 0.2,\ p(x_4) = 0.15,\ p(x_5) = 0.1$, és keressük meg az ehhez tartozó Huffman-kódot. A 4.4. ábra bal oldalán az egyes lépéseket az összevonásokat jelölő nyilak melletti sorszámozással szemléltettük. A 0.4 és 0.6 valószínűségek optimális prefix kódja nyilván a 0 és az 1 (vagy fordítva). Az összevonások megfordításával elvégezhetjük a Huffman-kód felépítését. A 4.4. ábra jobb oldalán a lefelé mutató nyilakon feltüntettük, hogy az adott lépésben hogyan különböztettük meg a kódszavakat a 0 vagy

az 1 bit hozzárendelésével. Végül a bináris fa gyökerétől indulva és a levelekig szintenként lefelé haladva kiolvashatjuk az így kapott kódszavakat, amelyeket a levelek alatt is feltüntettünk. Tehát $f(x_1) = 10,\ f(x_2) = 00,\ f(x_3) = 01,\ f(x_4) = 110,\ f(x_5) = 111$.

A bemenet néhány speciális eloszlása esetén azonnal meg tudjuk adni a Huffman-kódot, az algoritmus tényleges lefuttatása nélkül is.

4.8. példa. A bemenet ún. diadikus eloszlása esetén, vagyis amikor az egyes szimbólumok valószínűségei a 2 negatív egész kitevős hatványaiként írhatók fel

$$p(x_i) = 2^{-\alpha_i}, \quad \alpha \in \mathbb{Z},$$

az egyes kódszavak α_i hosszúak, így az átlagos kódszóhossz

$$\mathbf{E}|f(X)| = \sum_{i=1}^{n} p(x_i)|f(x_i)| = \sum_{i=1}^{n} p(x_i)\,(-\log p(x_i)) = \sum_{i=1}^{n} 2^{-\alpha_i}\alpha_i.$$

4.9. példa. A bemenet egyenletes eloszlása, azaz

$$p(x_i) = \frac{1}{n}$$

esetén a kódszavak hossza k vagy $k-1$, ahol $k = \lceil \log n \rceil$. A Huffman-algoritmus a k hosszú kódszavak számát a lehető legkisebbre választja meg. Ezt legegyszerűbben úgy kaphatjuk meg, ha kiindulunk egy $k-1$ mélységű teljes bináris fából, és ennek néhány leveléhez további 2-2 csúcsot kapcsolunk a k-adik szinten. Egy levél ily módon való kettéágaztatása eggyel növeli a levelek, s ezzel együtt a kódszavak számát. Ennek eredményeként $2n - 2^k$ darab k hosszú és $2^k - n$ darab $k-1$ hosszú kódszót kapunk.

Mivel a Huffman-kód optimális, azaz a legkisebb átlagos kódszóhosszú prefix kód, ezért a 4.3. tétel miatt az átlagos kódszóhossza az entrópiánál legfeljebb 1 bittel nagyobb. Valójában ennél erősebb állítás is igaz: a Huffman-kód átlagos kódszóhossza az entrópiánál legfeljebb $p_{\max} + 0.086$ értékkel nagyobb, ahol $p_{\max}$ a leggyakoribb szimbólum valószínűsége. A gyakorlatban, nagy bemeneti ábécé esetén $p_{\max}$ értéke kicsi, így a Huffman-kód átlagos kódszóhosszának eltérése az entrópiától szintén kicsi, különösen ha az eltérést az entrópia arányában nézzük. Ugyanakkor kis ábécéméret és nagyon eltérő valószínűségek esetén $p_{\max}$, s így az átlagos kódszóhossz entrópiától való eltérése is meglehetősen nagy lehet. Ezen részben segíthetünk a blokk-kódolással, de sajnos ez újabb problémák forrása lehet.

4.10. példa. Tegyük fel, hogy a bemeneten kapott tömörítendő adatsorozat karakterei egy háromelemű ábécéből veszik fel az értékeiket, és az egyes szimbólumok egymástól függetlenül a következő eloszlás szerintiek:

$$p(x_1) = 0.95, \quad p(x_2) = 0.03, \quad p(x_3) = 0.02.$$

Ekkor a bemenet entrópiája 0.335 bit/szimbólum, míg a betűnkénti Huffman-kód átlagos kódszóhossza 1.05 bit/szimbólum, amely az entrópia 213%-a. Ha két-két szimbólum alkotta blokkra végezzük el a Huffman-kódolást, akkor 0.611 bit/szimbólum átlagos kódszóhosszat kapunk. Ezt folytatva egészen 8 szimbólum méretű blokkokig kell elmennünk ahhoz, hogy az átlagos kódszóhossz és az entrópia különbsége elfogadható értékre csökkenjen. Vegyük észre, hogy ehhez $3^8 = 6561$ elemű ábécé tartozik. Ilyen nagy méretű kódok alkalmazása több szempontból is előnytelen. Egyrészt már a kód tárolása (illetve a vevőhöz való átküldése) is sok helyet foglal. Másrészt sok időt és erőforrást igényel a dekódolás. Harmadrészt, ha egy kicsit is megváltozik a szimbólumok valószínűsége, az jelentősen ronthatja a kód hatékonyságát.

Láthatjuk, hogy sokkal hatékonyabb, ha blokkokat kódolunk egyes szimbólumok helyett. Minél nagyobb blokkméretet választunk, annál kisebb lesz a veszteség. Ugyanakkor a Huffman-kód nagy blokkméret esetén a gyakorlatban kevésbé alkalmazható. Egyrészt a kódszó elküldése csak a teljes m hosszú blokk beolvasása után lehetséges, és emiatt nagy m esetén jelentős késleltetés keletkezhet. Másrészt m hosszú blokkok Huffman-kódolásához az összes lehetséges m-hosszú sorozathoz tartozó kódszót elő kell állítanunk, ami a kód méretének exponenciális növekedését jelenti, és ez praktikus szempontból is korlátot állít m növelése elé. Olyan eljárásra van szükségünk, amely az egyes blokkokhoz úgy rendel kódszavakat, hogy közben nem kell meghatároznunk az összes többi ugyanolyan hosszú blokk kódszavát. Ezt a követelményt teljesíti a 4.5. szakaszban ismertetett aritmetikai kódolás, amelyet kifejezetten blokkonkénti kódoláshoz alkalmaznak. Jellegzetessége, hogy valós időben történik a kódszó előállítása és visszafejtése, tehát nem lép fel jelentős késleltetés, akármilyen hosszú blokkokat is kódolunk.

4.5. Aritmetikai kódolás

Az aritmetikai kódolás a Shannon–Fano-kód természetes kiterjesztése. Alapötlete, hogy a valószínűségeket intervallumokkal ábrázolja, amihez a valószínűségek pontos kiszámítása szükséges. Egyetlen kódszót rendel a bemeneti adathalmaz egy blokkjához, vagyis blokkonkénti kód. Egy kódszó a $[0, 1)$ intervallum egy jobbról nyílt részintervallumának felel meg, és megadása annyi bittel történik, amennyi már egyértelműen megkülönböztethetővé teszi bármely más részintervallumtól. A rövidebb kódszavak hosszabb intervallumokat, vagyis nagyobb valószínűségű bemeneti blokkokat kódolnak. A gyakorlatban a részintervallumokat fokozatosan finomítjuk a blokk szimbólumainak valószínűségei szerint, és mihelyt eldől a kimenet egy bitje (elegendően kicsi lesz az intervallum aktuális hossza), továbbítjuk azt.

Az aritmetikai kódolást ideális, azaz végtelen pontosságú lebegőpontos aritmetikával mutatjuk be. A gyakorlatban természetesen csak véges pontosságú aritmetikák léteznek, de a módszer ezeken is implementálható.