Információelmélet

Nagy Szilvia

2006. augusztus 14.

Tartalomjegyzék

1. Bevezetés

Már az állatvilág egyedei is (sőt bizonyos fokon a növények is) cserélnek információt, így növelik saját, fajtársaik és életközösségeik egyéb tagjainak fennmaradási esélyeit. Az információközlésnek a riasztástól kezdve a táplálékszerzésen és szaporodáson keresztül a csapat (csorda, falka, stb.) összetartozásának növeléséig, az összetettebb közösségek szervezéséig igen sokféle célja van. A közlés módozata is többféle lehet, gondoljunk csak a méhek bonyolult mozgására, vagy a feromonok és egyéb szagjelek kibocsátására, a vizuális és hangjelzésekre. Az emberek, ahogy telnek az évek, évszázadok, egyre összetettebb, magasabb szervezettségű közösségekben élnek, amelyek jó működéséhez egyre több információközlésre van szükség. Eleinte, amikor még csak a kisebb közösség éppen együtt levő tagjai kommunikáltak egymással, elegendő volt a mutogatás és a kezdetleges beszéd. Amikor a messzebb tartózkodó törzstagokkal és a szomszédos törzsekkel is meg kellett értetniük magukat, dob-, füst-, vagy egyéb fény- és hangjeleket használtak. Az írás kifejődésével már nem csak térbeli, de időbeli távolságokat is át tudtak hidalni. Idővel elkezdték a hírközlésre szolgáló rendszereket gépiesíteni, így azok egyre nagyobb távolságra egyre több információt, tudtak továbbítani és egyre több felhasználó igényeit tudták kielégíteni. Jó ideje már a továbbított információ igen nagy részét is gépek szintetizálják, gondoljunk csak az automata gyártósorok vezérlésében felhasznált adatokra, vagy akár a minden hónapban legenerált számlakivonatunkra. Napjainkban az információcsere egyre nagyobb hányada zajlik gépiesítve, és ahogy a dolgok állnak, ez az arány egyre nőni fog. És abban, hogy eddig fejlődhettek a hírközlő rendszerek, igen nagy szerepe volt (van) az információelméletnek.

Az információelméletet C. E. Shannon 1948-ban írt munkájától szokták eredeztetni [1]. Ő maga azt írja, hogy az elmélet alapjait Nyquist [2], [3] és Hartley [4] tette le, több mint húsz évvel korábban, a gyökerek pedig egészen Boltzmannig nyúlnak vissza, aki a XIX. század végén bevezette az entrópia fogalmát a statisztikus fizikában. (Hogy mi köze van a rendezetlenség mértékének, az entrópiának az információhoz, az rögtön a jegyzet első fejezetéből ki fog derülni.) Az információelmélet mára matematikailag alaposan kidolgozott, sokrétűen felhasznált tudományággá fejlődött, hogy a tudományterület egyik első magyarországi művelőjét idézzem: „A híradástechnikában ma az információelmélet hasonló szerepet tölt be, mint a kémiában a Mengyelejev-féle periódusos rendszer. Mint ahogy a periódusos rendszer felfedezése előtt is ismerték az elemeket és léteztek kémiai módszerek, az első híradástechnikai berendezések is jóval

megelőzték az információelméletet." Az információelmélet fejlődése nem csak lehetővé tette a már meglévő hírközlő eszközök rendszerezését, hanem egészen új módszerek, távközlési ágak kifejlődéséhez vezetett.

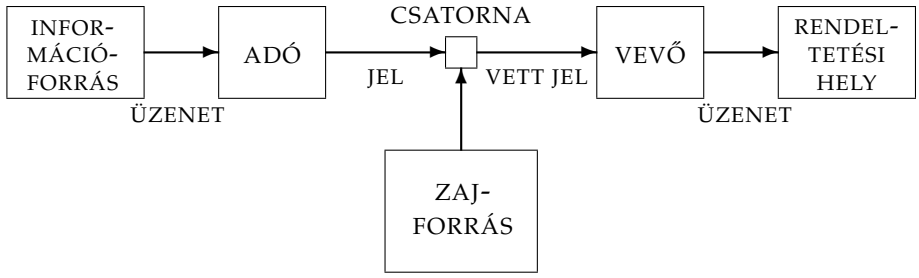
A jegyzet a Széchenyi István Egyetem elsőéves villamosmérnök BSC hallgatóinak készült, s amellett, hogy az információ- és kódoláselmélet alapfogalmait leírja, igyekszik a gyakorlati felhasználásokra is utalni és a szükséges valószínűségszámítási és lineáris algebrai alapokat is megadni.

A munka az elején a szükséges absztrakt információelméleti fogalmakat tárgyalja. Ezután könyv két fő részből tevődik össze, az első az adattömörítésről vagy forráskódolásról szól, a második a hibajavító kódokról. A kettő között rövid fejezet szól az információ átjuttatásáról a hírközlési csatornán. Végül néhány ajánlott, történelmi jelentőségű és/vagy a szerző által felhasznált könyv, cikk, illetve honlap címe sorakozik.

2. Az információelmélet alapfogalmai

2.1. A Shannon-féle hírközlési modell

A hírközlés során egy üzenetet el kell juttatni egy helyről (időpontból) a másikra valamilyen csatornán keresztül. Shannon [6] az általános hírközlési rendszer blokkvázlatát a következőképpen írta le:



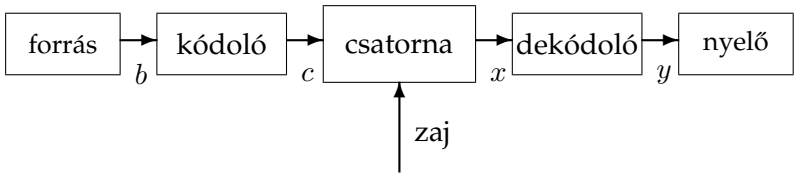
A modell egyes részei a következők:

- Az **információforrás** egy olyan apparátus, amely valamilyen jeleket: *üzenet*eket bocsát ki. Ezeket az jeleket kell eljuttatni a rendeltetési helyére. Az üzenet többféle típusú lehet: betűk, vagy szimbólumok sorozata (pl. az SMS vagy a távirat), egyszerű folytonos időfüggvény (telefon), vagy ezek csoportja (sztereó rádió), esetleg olyan függvények csoportja, amelyek időtől és térkoordinátáktól is függenek (TV).
- Az **adó** valamilyen műveletet hajt végre az üzeneten, hogy az alkalmas legyen a csatornán való átjutásra. Ez a művelet lehet viszonylag egyszerű, mint például a régi telefonoknál, ahol a hangot egyszerűen a hangnyomással arányos árammá alakították, vagy lehet összetett, mint a mostani telefonoknál, ahol a hangot többféle módon transzformálják, mintavételezik, és csak annyi információt hagynak meg belőle, amennyiből a vevő oldali berendezés szintetizálni tud egy az eredeti beszédet megfelelően közelítő jelet. Ezután esetleg hibajavító kódolást is alkalmaznak.
- A **csatorna** pusztán az a közeg, amelyen átjut a jel az adótól a vevőig; egy réz érpár, koax kábel, üvegszál és lézerfény, egy rádiófrekvenciasáv, vagy akár egy mágneses vagy optikai lemez. Az üzenet átvitele során a jelhez zaj adódhat, amit az ábrán a zajforrás jelképez.
- A **vevő** eredetileg csak az adó által végrehajtott művelet inverzét hajtotta végre, hogy visszaállítsa a vett jelből az üzenetet. Ha

az adó több részből összetevődő operációt hajt végre az üzeneten, akkor az egyes részműveletek inverzét fordított sorrendben szokták a vevőben elvégezni. A hibajavító kódok fejlődésével a vevőre az adó műveleteinek invertálásán túl egyre több és bonyolultabb feladat hárult; először csak hibajelzés és az üzenet megismételtetése, majd hibajavítás, interpoláció.

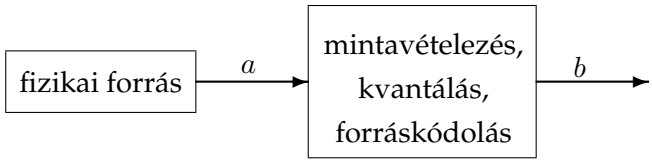
- A **rendeltetési hely** az a személy vagy gép, akinek az üzenet szól.

A jegyzet a Shannon-féle hírközlési modell kissé módosított változatát fogja alkalmazni:



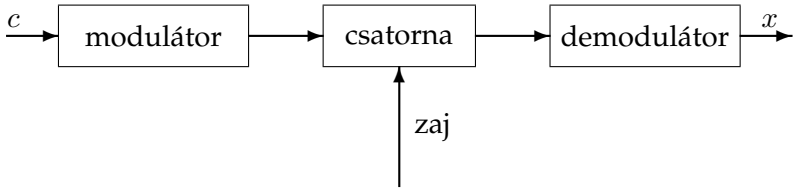
2.1. ábra. A hírközlési modell

- A félév során alapvetően diszkrét jelekből, szimbólumokból álló üzenetekkel fogunk foglalkozni. Feltesszük tehát, hogy a **forrás** által kibocsátott b üzenet diszkrét szimbólumok tömörített sorozata. Ha a továbbítandó jel folytonos lenne, azt előbb átalakítjuk diszkrét jelekké: mintavételezzük és kvantáljuk. A forrásba értjük a tömörítő vagy *forráskódoló* eljárást is. A forrás blokkvázlata:



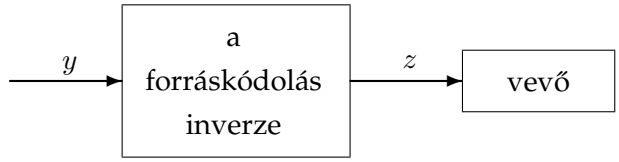
- A **kódoló**n alapvetően a hibajavító kódolást végrehajtó csatornakódolót értjük, melynek a bemenete a b tömörített üzenet, a kimenete pedig a c kódolt szimbólumsorozat. Erről szól a félév második fele. A csatornakódolás során ügyelni kell arra, hogy a csatorna tulajdonságait – zaj, veszteség – figyelembe véve az eredmény lehetőleg rövid, de nagy valószínűséggel érthetően visszafejthető legyen a vevőnél.
- A **csatorná**n olyan csatornákat értünk, amelyeknek a bemeneti és kimeneti jelkészlete is véges sok elemből áll, azaz *diszkrét* csatornákat.

Igen sok, jól használható, de folytonos jelekkel működő csatorna van. Ezeken lehet diszkrét jeleket is továbbítani, ha a diszkrét jeleket megfelelően átalakítjuk. Ez az átalakítás, vagy általánosabban a kódolt üzenet szimbólumainak a csatornával kompatibilissé tétele a **modulátor** feladata. A fizikai csatorna kimenetén természetesen egy demodulátorra is szükség van, amely a csatorna jeleit visszatranszformálja a vevő dekódolója számára értelmezhető diszkrét jelekké. A csatorna blokkvázlata:



A zaj hozzáadása az átvitt jel torzulásához, így az átvitt szimbólumok egymásba alakulásához vezet.

- A **dekódoló** hibajavítást végez, azaz eldönti, hogy a kapott szimbólumsorozat hibás-e, és ha igen, akkor milyen szimbólumsorozatból keletkezhetett. Ezután – illetve bizonyos esetekben, mint például a Viterbi-algoritmus, ezzel párhuzamosan – következik a csatornakódolás inverz-művelete.
- A **vevő** végrehajtja a forráskódolás inverz műveletét, illetve ha az eredeti üzenet folytonos volt, akkor a diszkrét jelből visszaállítja azt. A blokkdiagramja:



A forrás- és a csatornakódoló közé sok esetben egy tikosító eljárás is bekerül. Ezesetben a dekódoló és a forráskódoló inverze között megvan a titkosító eljárás visszafejtője is.

A 2.1 ábrán látható modellnek a részeit fogjuk a következő fejezetekben tanulmányozni.

2.2. Az információ és az entrópia

Matematikai kitérő - Valószínűségszámítási emlékeztető. A valószínűségszámítás során sokszor végrehajtható *kísérlet*ek eredményeit szokták **esemény**nek nevezni, a kísérlet fogalmát elég tágan értelmezve. A kísérlet lefolyását annyi sok tényező befolyásolja, hogy azokat nem tudjuk, vagy nem akarjuk figyelembe venni, hanem azt mondjuk róla, hogy véletlenszerű.

Kísérlet lehet például az, hogy elküldünk egy jelet, és megfigyeljük a fogadott jelet, de akár vizsgálhatjuk a kettes kifutópályán adott idő alatt felszálló repülőgépek számát is, vagy azt, hogy leesik-e a labda, amit elejtünk. Ezen kísérletek kimenetelei lehetnek a következő események: például, hogy 800 kHz-es szinuszos jelet adva a fogadott jel $1{,}6$ MHz-es szinusz lesz, vagy egy $1{,}6$ MHz-es szinusz és egy 800 kHz-es koszinusz jel összege; hogy reggel 8 és 10 óra között két gép szállt fel, vagy egy sem, mert köd volt; vagy hogy a labda leesik a földre, illetve nem esik le. Szokás az eseményeket az ABC betűivel jelölni. Egy A esemény **ellentett esemény**én minden olyan esetet értünk, amikor A nem következik be, és $\overline{A}$-val jelöljük.

Nagyon sokszor elvégezve a kísérletet és megfigyelve, hogy az adott esemény megtörtént-e, valószínűséget lehet rendelni az eseményhez: egy A esemény bekövetkezésének **valószínűsége**

$$p(A) = \frac{\text{azon kísérletek száma, mikor bekövetkezett az } A \text{ esemény}}{\text{az összes kísérlet száma}}.$$

Ha például száz esetből nyolcvanszor kétszeres frekvenciájú jelet fogadtunk, 15-ször az $1{,}6$ MHz frekvenciájú szinusz és egy 800 kHz-es koszinusz jel összegét, a többi esetben valami mást, akkor az első esemény valószínűsége $0{,}8$, a másodiké $0{,}15$, a harmadiké (hogy a fogadott jel sem $1{,}6$ MHz-es szinusz, sem pedig az $1{,}6$ MHz-es szinusz és egy 800 kHz-es koszinuszos jel összege) $0{,}05$. Ha a menetrend szerint két gép száll fel a vizsgált időpontban a kettes kifutópályáról, elég kicsi a valószínűsége annak, hogy egy sem, vagy négy induljon, de még mindig sokkal nagyobb, mint annak, hogy ne essen le a labda, ha leejtjük a Földön. Ha sohasem fordulhat elő egy esemény, a valószínűsége nulla, ha mindig bekövetkezik, akkor pedig 1.

Vizsgálhatunk egyszerre több eseményt is: akár ugyanannak a kísérletnek az eredményét több szempontból, akár többféle párhuzamos kísérlet eredményeit. Az első esetre példa, hogy a fogadott jel szinuszos,

háromszög vagy egyéb, illetve 800 kHz-es frekvenciájú, $1{,}6$ MHz-es, 400 kHz-es, vagy másféle; az utóbbira pedig, hogy vizsgáljuk a vett jelet is valamilyen szempontból, és mondjuk a leadott jelet is, vagy akár az előző vett jelet. Definiálhatjuk két – vagy több – esemény összegét és szorzatát is, minkét esetben egy másik (összetett) eseményt kapunk:

- Két esemény szorzatán azt értjük, ha mindkét esemény bekövetkezik. Az A és B események együttes bekövetkezésének valószínűségét $p(A \cdot B)$-vel vagy $p(AB)$-vel jelöljük.
- Két esemény összegén azt értjük, ha vagy az egyik vagy a másik, vagy mindkét esemény bekövetkezik. Az alkalmazott jelölés: $p(A + B)$.

E két mennyiség között összefüggés áll fenn:

$$p(A + B) = p(A) + p(B) - p(AB). \tag{2.1}$$

Egy esemény akkor elemi esemény, ha nem bontható fel részeseményekre, azaz nem áll elő más események összegeként.

A valószínűségszámítás szigorú matematikai rendszerét, az axiómáit Kolmogorov fektette le egy 1933-as írásában. Kolmogorov egy kísérlet lehetséges kimeneteit, az **elemi események** összességét **eseménytér**nek nevezi. Jelöljük az eseményterünket Ω-val. Az Ω lehetséges részhalmazai az események. Az $\mathcal{S}$ halmaz legyen az összes esemény halmaza. Az események között lehet műveleteket definiálni: összeadást és szorzást, amelyek speciálisan viselkednek. Ha O-val jelöljük a lehetetlen eseményt, akkor $\mathcal{S}$ az összeadással és szorzással akkor alkot σ-algebrát, ha

1. mind $\Omega \in \mathcal{S}$, mind pedig $O \in \mathcal{S}$, azaz a biztos esemény és a lehetetlen esemény is az $\mathcal{S}$ halmazban van,
2. ha $A_i \in \mathcal{S}$ minden i-re, akkor $\sum_i A_i \in \mathcal{S}$, azaz $\mathcal{S}$-beli események összegzése nem visz ki $\mathcal{S}$-ből,
3. ha $A_1 \in \mathcal{S}$ és $A_2 \in \mathcal{S}$, akkor $A_1 \cdot A_2 \in \mathcal{S}$, azaz két $\mathcal{S}$-beli esemény szorzata is $\mathcal{S}$-en belüli esemény lesz.

Minden $A \in \mathcal{S}$ eseményhez hozzárendelünk egy $0 \leq p(A) \leq 1$ számot, az esemény **valószínűség**ét, melyre teljesülnek a következők:

1. $p(\Omega) = 1$,
2. ha $A_i \cdot A_j = 0$ minden $j \neq i$ számpárosra, akkor

$$p\left(\sum_i A_i\right) = \sum_i p(A_i).$$

Ez azt jelenti, hogy ha az események kölcsönösen kizárják egymást (de csak akkor), akkor a belőlük képezett összeg-esemény valószínűsége megegyezik az események valószínűségeinek az összegével.

Ha információt akarunk továbbítani valamilyen csatornán, akkor igen fontos tudni, hogy adott bemeneti jel esetén milyen valószínűséggel kapjuk meg az egyes kimeneti jeleket, így szükség lesz a feltételes valószínűség fogalmára is. Ha A-val és B-vel két, a kísérlettel kapcsolatos eseményt jelölünk, és a B esemény valószínűsége nem nulla, akkor az A eseménynek a B feltétel melletti **feltételes valószínűség**én azt értjük, hogy milyen valószínűséggel következik be A, ha már B bekövetkezett, és $p(A|B)$-vel jelöljük:

$$p(A|B) = \frac{p(A \cdot B)}{p(B)}, \tag{2.2}$$

ahol $p(A \cdot B)$ az A és B események szorzatának, az $A \cdot B$ eseménynek a bekövetkezési valószínűsége. A feltételes és együttes bekövetkezési valószínűségekre igaz a következő néhány állítás:

- Ha vizsgáljuk az egy adott B eseményt feltéve létrejöhető A_1, $A_2, \ldots A_n$ eseményeket, azok feltételes valószínűségeinek összege 1 lesz, azaz

$$\sum_{i=1}^{n} p(A_i|B) = 1. \tag{2.3}$$

- Ha vizsgáljuk az egy adott B eseményt feltéve létrejöhető A_1, $A_2, \ldots A_n$ eseményeket, azok B-vel együttes előfordulási valószínűségeinek összege megadja a B esemény valószínűségét, azaz

$$\sum_{i=1}^{n} p(A_i \cdot B) = p(B). \tag{2.4}$$

- Ha az összes lehetséges $A_1,A_2,\dots A_n$ és $B_1,B_2,\dots B_m$ események együttes előfordulási valószínűségeinek összegét vesszük, 1-et kapunk, vagyis

$$\sum_{j=1}^{m}\sum_{i=1}^{n} p(A_i \cdot B_j) = \sum_{j=1}^{m} p(B_j) = 1. \tag{2.5}$$

Ha tudjuk, hogy a kísérletünk az $A = \{A_1,A_2,\dots,A_n\}$ *szám*halmaz elemeinek valamelyikét eredményezi, méghozzá sorrendben $p_1, p_2, \dots, p_n$ valószínűséggel, akkor meg tudjuk mondani, mi a kísérlet eredményének várható értéke. Visszatérve a repülős példára: ha $0{,}9$ valószínűséggel két gép száll fel az adott pályáról reggel 8 és 9 óra között, $0{,}06$ valószínűséggel csak egy, és $0{,}02$ valószínűséggel három, illetve nulla, akkor a felszállt repülőgépek számának várható értéke $0{,}9\cdot 2+0{,}06\cdot 1+0{,}02\cdot 3+0{,}02\cdot 0 = 1{,}92$ lesz. A kísérlet A eredményének **várható értéke** tehát

$$\langle A\rangle = \sum_{i=1}^{n} p_i A_i. \tag{2.6}$$

Az ettől az értéktől vett átlagos eltérést **szórás**nak nevezzük, és a következőképpen definiáljuk:

$$D(A) = \sqrt{\left\langle (A - \langle A\rangle)^2 \right\rangle}. \tag{2.7}$$

Ha azt vizsgáljuk, hogy két kísérlet A és B kimenetei mennyire hasonlítanak egymásra, erről az A és B korrelációja adhat tájékoztatást:

$$R(A,B) = \frac{\langle (A - \langle A\rangle)\cdot(B - \langle B\rangle)\rangle}{D(A)\cdot D(B)}. \tag{2.8}$$

2.2.1. Az információ

Shannon szerint [6] hírközlés alapvető feladata az, hogy egy valahol – az adónál – kiválasztott üzenetet vagy pontosan, vagy pedig valamilyen közelítéssel reprodukáljunk egy másik helyen – a vevőnél. Az üzenetnek többnyire van valamilyen jelentése, amely függ a körülményektől. Egy mérnök számára azonban nincs igazi jelentőssége ezeknek a körülményeknek. Ami igazán fontos az, hogy a szóban forgó üzenet olyan, amelyet a *lehetséges üzenetek halmazá*ból választottak ki. A hírközlő rendszereket olyanra kell tervezni, hogy *bármely* lehetséges üzenetet továbbítani tudják.

Az információelmélet felépítéséhez szükség van az információ számszerű – tartalomtól, eredettől, helyzettől független – mérésére.

Az **információ**, definíciója szerint valamely véges számú, előre ismert lehetőség közül az egyik megnevezése.

Ahelyett a kérdés helyett azonban, hogy *mekkora információt jelent* az a kijelentés, hogy a lehetséges m darab esemény – a forrás lehetséges m kimenete – közül éppen az i-edik, A_i, következett be, feltehetjük úgy is a kérdést, *mi volt a bizonytalanság* arra nézve, hogy a lehetséges $\{A_1, A_2, \ldots, A_m\}$ állapotok közül az i-edik jön létre. Az információ mértéke tehát egyenlő azzal a bizonytalansággal, amelyet megszüntet.

Ha véges sok lehetséges eseményünk van, akkor annak a mérésére, hogy mekkora információt jelent az, hogy egy lehetséges esemény bekövetkezet, tulajdonképpen bármely olyan függvény jó lenne, amely a lehetőségek számától monoton módon függ. Hartley azonban rámutatott, hogy a legideálisabb ilyen függvény a logaritmus. Ennek több oka is van:

- Mérnöki szempontból az a legfontosabb, hogy igen sok, gyakorlati jelentőségű paraméter a lehetséges kimenetek számának logaritmusával skálázódik. Például, ha fix számú szimbólummal dolgozunk, akkor az adásidő megduplázása a lehetséges üzenetek számát négyzetre emeli. Ez azt jelenti, hogy az üzenetek számának logaritmusa kétszereződik. Ha kétállapotú tárolókkal dolgozunk, akkor egy plusz tároló a lehetséges állapotokat megkétszerezi, vagy az üzenetek számának (kettes alapú) logaritmusát növeli eggyel.
- Matematikailag jól kezelhető a kimenetek számának logaritmusa.
- Intuitív megfontolások is a logaritmus függvényt részesítik előnyben. Például természetesen úgy gondoljuk, hogy kétszerannyi adásidő alatt kétszerannyi információt tudunk közölni, vagy hogy két egyforma lemezen kétszerannyi információt tudunk tárolni, mint egyen. Ezek a megfontolások mind afelé mutatnak, hogy az állapotok számának logaritmusa igen jó információmérő függvény.

Nézzünk egy klasszikus példát. Ha a lehetséges kimeneteket tartalmazó m elemű halmaz elemeit azonosítani szeretnénk – például barkochbázunk úgy, hogy csak előre megadott m dologra gondolhatunk (mondjuk a számokra 1-től m-ig) –, és $2^k \leq m \leq 2^{k+1}$, akkor legfeljebb $k+1$ kérdés után megtudjuk, mire gondolt a másik, azaz azonosítani tudjuk az elemet.

Hartley az információt az

$$I = \log_2 m \tag{2.9}$$

mennyiséggel azonosította.

A logaritmus alapja azért kettő, mert két lehetséges válasz van a kérdésekre: igen és nem. A számítástechnikában használt bináris számrendszer és a leggyakrabban használt kétállapotú tárolók miatt is előnyös a logaritmus alapjának a kettőt választani. Ez a definíció azonban csak igen speciális esetekre alkalmazható, csak akkor, ha minden lehetséges kimenet valószínűsége azonos, $1/m$. Ha a lehetséges események előfordulása nem egyenlő valószínűségű, azaz az események valamilyen statisztikát követnek, akkor kicsit módosítani kell a definíciónkon.

Minél váratlanabb egy eredmény annál nagyobb információtartalmat rendelnek bekövetkezéséhez.

Legyen a lehetséges események halmaza $A = \{A_1, A_2, \ldots, A_m\}$, az A_1 esemény valószínűsége p_1, az A_2-é p_2,..., az A_m-é pedig p_m. (A valószínűségekre igaz, hogy $\sum_{i=1}^{m} p_i = 1$.) Ha az i-edik esemény következett be, annak az információtartalma

$$I(A_i) = I(p_i) = \log_2 \frac{1}{p_i} = -\log_2 p_i. \tag{2.10}$$

Ezt a definíciót C. E. Shannon fogalmazta meg először 1948-as cikkében [1].

Az információ mértékegysége ez esetben a **bit** (a binary digit szavak Tukey által javasolt összevonásából). Természetesen más alapú logaritmussal is lehet definiálni az információt, csak akkor az egysége más lesz, természetes alapú logaritmus esetén például az egység a *nat*, tízes alapú logaritmussal *hartley*.

2.1. példa: Hány bit lehet egy nat? Hány hartley egy bit?

Megoldás: 1 nat információt akkor nyerünk, ha a bekövetkezett esemény p valószínűségére igaz, hogy

$$-\ln p = \ln \frac{1}{p} = 1,$$

azaz $p = 1/e$. Ennek az eseménynek az információtartalma bit egységekben:

$$-\log_2 \frac{1}{e} = \log_2 e \approx 1{,}4427.$$

A 1 bit információtartalmú esemény előfordulási valószínűsége $1/2$. Ennek a hartleyben kifejezett információtartalma

$$-\lg \frac{1}{2} = \lg 2 \approx 0{,}301.$$

Ha az események mindegyike $p_i = \frac{1}{m}$ valószínűségű, visszakapjuk az információ Hartley-féle definícióját:

$$I\left(p = \frac{1}{m}\right) = -\log_2 \frac{1}{m} = \log_2 m.$$

2.2. példa: Statisztikai elemzések alapján az angol nyelvű szövegekben az E betű fordul elő leggyakrabban, $p_{\mathrm{E}} = 0{,}1073$ valószínűséggel, a következő leggyakoribb betű a T, melyre $p_{\mathrm{T}} = 0{,}0856$, a legritkább pedig a Q, $p_{\mathrm{Q}} = 0{,}00063$ előfordulási valószínűséggel. Adjuk meg e három betű vétele során nyert információt különkülön az egyes betűkre és a „TEQ" sorozatra, ha feltesszük azt, hogy az egyes helyeken a betűk bekövetkezése egymástól független esemény.

Megoldás: A feladat első részéhez egyszerűen behelyettesítjük a valószínűségeket a (2.10) képletbe:

$$\begin{aligned} I(\mathrm{E}) &= -\log_2 p_{\mathrm{E}} = -\log_2 0{,}1073 \approx 3{,}22 \\ I(\mathrm{T}) &= -\log_2 p_{\mathrm{T}} = -\log_2 0{,}0856 \approx 3{,}55 \\ I(\mathrm{Q}) &= -\log_2 p_{\mathrm{Q}} = -\log_2 0{,}00063 \approx 10{,}63 \end{aligned}$$

Az, hogy a szövegben az egyes pozíciókban betűk előfordulása független esemény, elég erős közelítés, hiszen például a „Q" betű után szinte mindig „U" következik az angol szavakban, a „T" után meg gyakran jön „H". Ha azonban független eseményeknek tekintjük a betűk egymásutánját, akkor a „TEQ" sorozat előfordulási valószínűsége az egyes karakterek előfordulási valószínűségeinek szorzata, azaz $p_{\mathrm{TEQ}} = p_{\mathrm{T}} \cdot p_{\mathrm{E}} \cdot p_{\mathrm{Q}}$. Így a sztring információtartalma:

$$I(\mathrm{TEQ}) = -\log_2 p_{\mathrm{TEQ}} = -\log_2 p_{\mathrm{T}} - \log_2 p_{\mathrm{E}} - \log_2 p_{\mathrm{Q}} \approx 3{,}22 + 3{,}55 + 10{,}63 = 17{,}4.$$

Az információ tulajdonságai:

1. Csak az esemény valószínűségének függvénye.
2. Nem negatív: azaz $I \geq 0$.
3. Additív: ha az események száma $m = m_1 \cdot m_2$ alakban két természetes szám szorzataként áll elő, azaz az események csoportosíthatók m_1 darab m_2 elemű csoportra, akkor az információk összeadódnak:
$$I(m_1 \cdot m_2) = I(m_1) + I(m_2). \tag{2.11}$$
Ez azt jelenti, hogy ugyanakkora információra van szükség egy elem azonosítására, hogy ha az egész halmazban keressük, mint ha előbb

azonosítanánk az m_2 elemű részhalmazt, amelyben van, majd a részhalmazon belül keresnénk meg az elemet.

4. Monoton: a logaritmus függvény monoton növekvő, így a $\log_2(1/p) = -\log_2 p$ monoton csökkenő. Ez azt jelenti, hogy ha $p_i < p_j$, akkor $I(A_i) > I(A_j)$. Minél nagyobb valószínűségű tehát egy (A_j) esemény bekövetkezése, annál kisebb információval szolgál, és minél valószínűtlenebb az (A_i) esemény annál nagyobb az információtartalma, ha mégis előfordul.
5. Normálás: ez lényegében az információ egységének kiválasztása. Legyen $I(A_i) = 1$, ha $p_i = 1/2$, így a **bit** egységet kell használni.

2.2.2. Az entrópia

Ha m darab, egymást kizáró eseményt veszünk, feltehetjük azt a kérdést, milyen átlagos információtartalommal bír az, hogy tudjuk, az *egyik* bekövetkezett. A válasz az **információ várható értéke**: $\sum_{i=1}^{m} p_i I(p_i)$. Ezt az átlagos információt nevezték el entrópiának.

Shannon információdefinícióját felhasználva az **entrópia** a

$$H(p_1,p_2,\ldots,p_m) = -\sum_{i=1}^{m} p_i \log_2 p_i \tag{2.12}$$

alakot ölti.

(A $p_i \log_2 p_i$ kifejezés $p_i = 0$ esetben a L'Hospital-szabály szerint nullát ad.)

2.3. példa: Tegyük fel, hogy a forrásunk öt szimbólumot bocsáthat ki, a_1-et, a_2-t, a_3-at, a_4-et és a_5-öt, mindegyiket egyforma valószínűséggel. Mekkora az egy szimbólum kibocsátásával átadott átlagos információ, azaz mekkora a forrás, mint halmaz entrópiája?

Megoldás: Egy-egy szimbólum kibocsátásának valószínűsége $0{,}2$, így a keresett entrópia:

$$\begin{aligned} H(a_1,a_2,a_3,a_4,a_5) =& -p_1\log_2 p_1 - p_2\log_2 p_2 - p_3\log_2 p_3 - p_4\log_2 p_4 - p_5\log_2 p_5 = \\ =& -5\cdot 0{,}2\log_2 0{,}2 \approx 2{,}32 \end{aligned}$$

2.4. példa: Tegyük fel, hogy a forrásunk továbbra is öt szimbólumot bocsáthat ki, a_1-et, a_2-t, a_3-at, a_4-et és a_5-öt, azonban más és más valószínűséggel. Legyen $p_{a_1} = 0{,}35$, $p_{a_2} = 0{,}20$, $p_{a_3} = 0{,}10$, $p_{a_4} = 0{,}20$ és $p_{a_5} = 0{,}15$. Mekkora az egy szimbólum kibocsátásával átadott átlagos információ?

Megoldás: Az entrópia (2.12) definíciójába behelyettesítve a valószínűségeket:

$$H(a_1,a_2,a_3,a_4,a_5) = -p_1\log_2 p_1 - p_2\log_2 p_2 - p_3\log_2 p_3 - p_4\log_2 p_4 - p_5\log_2 p_5 =$$

$$= -0{,}35 \log_2 0{,}35 - 0{,}2 \log_2 0{,}2 - 0{,}1 \log_2 0{,}1 - 0{,}2 \log_2 0{,}2 - \\ -0{,}15 \log_2 0{,}15 \approx 2{,}20$$

2.5. példa: A forrásunk továbbra is öt szimbólumot bocsásson ki, $p_{a_1} = 0{,}60$, $p_{a_2} = 0{,}12$, $p_{a_3} = 0{,}08$, $p_{a_4} = 0{,}05$ és $p_{a_5} = 0{,}15$ valószínűségekkel. Mekkora az egy szimbólum kibocsátásával átadott átlagos információ?

Megoldás: Az entrópia (2.12) definíciójába behelyettesítve a valószínűségeket:

$$H(a_1,a_2,a_3,a_4,a_5) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - p_3 \log_2 p_3 - p_4 \log_2 p_4 - p_5 \log_2 p_5 = \\ = -0{,}6 \log_2 0{,}6 - 0{,}12 \log_2 0{,}12 - 0{,}08 \log_2 0{,}08 - \\ -0{,}05 \log_2 0{,}05 - 0{,}15 \log_2 0{,}15 \approx 1{,}73$$

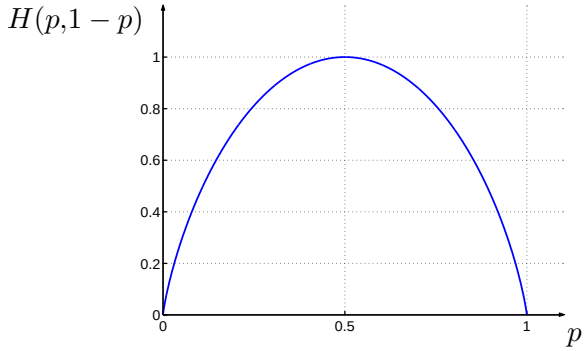
Figyeljük meg, hogy minél közelebb van a rendszer az egyenletes valószínűség-eloszláshoz, annál nagyobb az entrópiája. Hasonlóképpen statisztikus fizikában minél közelebb volt a rendszer a teljesen rendezetlen állapothoz annál nagyobb volt az entrópiája, avagy a rendezetlenségének a mértéke. A teljesen rendezetlen állapot az, amelyben semmiféle strukturáltság nem figyelhető meg, ahol a részecskék (állapot)térbeli eloszlása egyenletes.

Az entrópia a következő tulajdonságokkal rendelkezik:

1. Nem negatív, azaz $H(p_1,p_2,\ldots,p_m) \geq 0$.
2. Folytonos függvénye a valószínűségeknek, azaz ha a $(p_1,p_2,\ldots,p_m)$-ek közül az egyik p_i valószínűséget kicsit megváltoztatjuk (a többit is úgy, hogy az összeg 1 maradjon), akkor az entrópia is csak kicsit fog megváltozni.
3. Ha $p_k = 1$ és $p_i = 0$, az $i = 1,\ldots,k-1,k+1,\ldots,m$ indexű eseményekre, akkor $H(p_1,p_2,\ldots,p_m) = 0$, azaz egy biztosan bekövetkező és bármennyi lehetetlen eseményből álló halmaz nem hordoz információt.
4. $H(p_1,p_2,\ldots,p_m,0) = H(p_1,p_2,\ldots,p_m)$, azaz egy lehetetlen esemény hozzáadása az eseményhalmazhoz nem befolyásolja az entrópiát.
5. $H(p_1,p_2,\ldots,p_m) \leq H(1/m,1/m,\ldots,1/m) = \log_2 m$, azaz a legnagyobb várható információtartalma egy eseménynek akkor van, ha minden esemény azonos valószínűséggel következik be. Speciálisan, ha egy eseményből és az ellentett eseményéből álló halmazunk van, akkor az entrópia a

$$H(A,\overline{A}) = H(p,1-p) = -p \log 2p - (1-p) \log_2(1-p)$$

függvény szerint változik, amelyet az alábbi ábrán láthatunk.



Leolvasható, hogy a függvény maximuma a $p = (1-p) = 1/2$-nél van.

6. $H(p_1,p_2,\ldots,p_m)$ szimmetrikus a változói felcserélésére:

$$H(p_1,\ldots,p_{k-1},p_\ell,p_{k+1},\ldots,p_{\ell-1},p_k,p_{\ell+1},\ldots,p_m) = \\ = H(p_1,p_2,\ldots,p_m),$$

bármely $k \neq \ell$ párosra.

2.2.3. A kölcsönös és a feltételes entrópia és tulajdonságaik

Tegyük fel, hogy két eseményhalmazunk van, az $A = \{A_1,A_2,\ldots,A_{m_1}\}$, és a $B = \{B_1,B_2,\ldots,B_{m_2}\}$, és ez a két halmaz valamilyen módon kapcsolódik egymáshoz. Például az A halmaz a forrás szimbólumkészlete vagy a leadott jeleké, a B pedig a lehetséges vett jelek halmaza. Ezenkívül minden A-beli esemény bekövetkezése maga után von egy B-beli eseményt (vagy fordítva). Az A halmaz i-edik és a B halmaz j-edik elemének *együttes bekövetkezési valószínűsége*

$$p_{i,j} = p(A_i \cdot B_j). \tag{2.13}$$

Ha a két esemény független – tehát ha az, hogy milyen B_j-t kapok nem függ attól, hogy milyen volt az A_i –, akkor a $p(A_i \cdot B_j) = p(A_i) \cdot p(B_j)$, egyébként az együttes valószínűség mindig kisebb, mint az egyedi valószínűségek szorzata, azaz $p(A_i \cdot B_j) \leq p(A_i) \cdot p(B_j)$.

Az A_i és B_j események *együttes bekövetkezésekor nyert információ*

$$I(A_i \cdot B_j) = -\log_2 p(A_i \cdot B_j), \tag{2.14}$$

amely – mivel az együttes valószínűség sosem nagyobb, mint bármely esemény előfordulásának valószínűsége – sosem kisebb, mint az egyes eseményekhez rendelhető információ, azaz

$$I(A_i \cdot B_j) \geq I(A_i) \qquad \text{és} \qquad I(A_i \cdot B_j) \geq I(B_j). \tag{2.15}$$

Egyenlőség csak akkor áll fenn, ha a másik esemény (az első egyenlőtlenség esetében B_j, a másodikában A_i) biztosan bekövetkezik.

Az A és B eseményhalmazok együttes, vagy **kölcsönös entrópiája** a

$$H(A \cdot B) = -\sum_{i=1}^{m_1}\sum_{j=1}^{m_2} p_{i,j} \log_2 p_{i,j} \tag{2.16}$$

mennyiség. Ha A és B elemei egymástól függetlenek, akkor (de csak akkor) igaz, hogy a kölcsönös entrópiájuk egyenlő az entrópiáik összegével.

Ha az A eseményhalmaz egy-egy elemének bekövetkezése maga után vonja valamely B-beli elem bekövetkezését, például az A elemei a forrás szimbólumkészlete, a B pedig a lehetséges vett jelek halmaza, akkor érdemes feltenni a kérdést, hogy ha csak B elemeit tudjuk megfigyelni, és B_j-t észleltük, akkor mekkora átlagos bizonytalanság maradt azzal kapcsolatban, hogy melyik A-beli elem váltotta ki azt. Ez az átlagos bizonytalanság, illetve az ezt megszüntető átlagos információmennyiség az A halmaznak a B-re vonatkoztatott **feltételes entrópiája**:

$$\begin{aligned} H(A|B) &= -\sum_{j=1}^{m_2} p(B_j) \sum_{i=1}^{m_1} p(A_i|B_j) \log_2 p(A_i|B_j) = \\ &= -\sum_{i=1}^{m_1}\sum_{j=1}^{m_2} p(A_i \cdot B_j) \log_2 p(A_i|B_j). \end{aligned} \tag{2.17}$$

Ha felhasználjuk azt, hogy $p(A_i \cdot B_j) = p(B_j)p(A_i|B_j)$ akkor levezethetjük a kölcsönös és a feltételes entrópia közötti összefüggést:

$$H(A \cdot B) = H(B) + H(A|B) = H(A) + H(B|A). \tag{2.18}$$

Szemléletesen ez azt jelenti, hogy a két esemény együttes megfigyelésével nyert átlagos információ egyenlő az egyik esemény megfigyelése során kapott információnak és az egyik esemény megfigyelése után a másikra maradt átlagos bizonytalanságot megszüntetni képes információ mennyiségének összegével.

Logikus, hogy egy esemény közvetlen megfigyelésével nem nyerhetünk átlagosan kevesebb információt, mint akkor, ha egy másik eseményt előtte már megfigyeltünk, és csak aztán figyeljük meg az előzőt, azaz

$$H(A) \geq H(A|B) \geq 0. \tag{2.19}$$

3. Forráskódolás

Forráskódoláson a hírközléselméletben azt értjük, hogy a már diszkrét jelekké átalakított üzenetet egy másik, szintén véges elemszámú szimbólumkészlettel lehetőleg igen tömören reprezentáljuk. Az eredeti üzenet kvantálásával, azaz diszkrét jelekké való átalakításával egyelőre nem foglalkozunk. Adott tehát egy véges elemszámú halmazunk, A, az úgynevezett *forrásábécé*, és azt képezzük le egy másik véges elemszámú B halmaz, a *kódábécé* elemeiből álló sorozatokra (kódszavakra) úgy, hogy bármilyen üzenetet dekódolni lehessen, és minél rövidebb kódszavakat kapjunk.

3.1. Diszkrét információforrások

Az üzenetet létrehozó apparátus a forrás, a mi esetünkben az üzenet véges sok elemből felépülő sorozat.

Ha a forrás által kibocsátott üzenet egy olyan diszkrét jelekből álló sorozat, amelynek minden szimbóluma egy véges elemszámú halmaz – a forrásábécé – eleme, akkor **diszkrét információforrás**ról beszélünk.

A forrás felfogható egy olyan gépezetnek, amely betűről betűre generálja az üzenetet, és a soron következő szimbólumot úgy választja ki, hogy figyelembe veszi a forrásábécé egyes elemeihez rendelt előfordulási valószínűségeket és esetleg a korábbi választásait. Azokat a rendszereket, amelyek egy valószínűséghalmaz felhasználásával hoznak létre diszkrét sorozatokat, *sztochasztikus rendszerek*nek nevezzük. A sztochasztikus folyamatok matematikájából csak igen kis szeletet fogunk itt megismerni.

Ha a forrásunk az $A = \{A_1, A_2, \ldots, A_n\}$ szimbólumhalmazzal, mint forrásábécével működik és *csak* az egyes szimbólumok előfordulási valószínűségét, azaz csak a $P = \{p_1, p_2, \ldots, p_n\}$ halmazt használja fel az üzenet létrehozásakor, akkor **emlékezet nélküli forrás**ról beszélünk. Tekintsük a forrás egymást követő N szimbólum-kibocsátását, mint eseményeket. Emlékezet nélküli forrás esetén ezek az események függetlenek egymástól. A forrás ezen kívül **stacionárius** is, ha minden alkalommal az $A = \{A_1, A_2, \ldots, A_n\}$ halmaz elemei közül bocsát ki jelet, méghozzá mindig $p_1, p_2, \ldots, p_n$ valószínűséggel, azaz sem a kódábécé, sem pedig annak eloszlása nem változik az időben.

Előfordulhat azonban, hogy az, hogy milyen szimbólum volt a forrás kimenetén az előző néhány lépésben, befolyásolja az aktuális választást. Ha egy emberi – például magyar – szöveg az üzenet, akkor nyilvánvalóan más lesz az egyes szimbólumok előfordulási valószínűsége attól függően, hogy a korábbi betű mi volt. Például egy P után sokkal nagyobb valószínűséggel

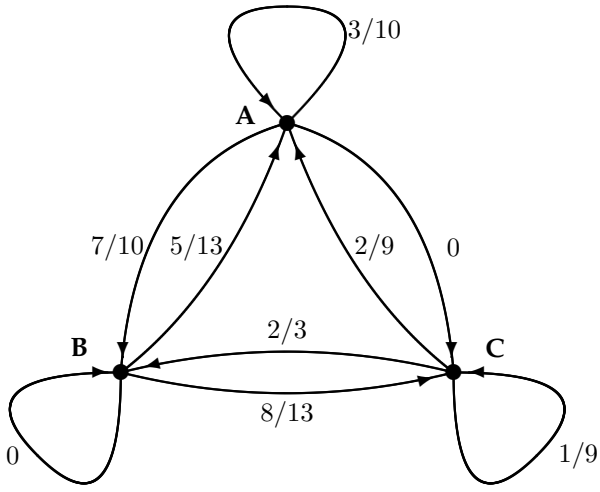
következik E, mint egy Ü után, azaz $p(E|P) > p(E|\text{Ü})$. Egy olyan stacionárius forrás, amelynél egy szimbólum kibocsátására mindig csak az azt megelőző egyetlen szimbólum van hatással, jól leírható a $p(A_{\text{aktuális}}|A_{\text{előző}})$ feltételes valószínűségekkel, vagy egy olyan gráffal, amelynek a csomópontjai a lehetséges előző szimbólumokat jelzik, az élei pedig az új szimbólumokat. Egy-egy él mindig abba a csomópontba mutat, amelyik új szimbólumot jelképezi, hiszen a következő lépésben az lesz a régi szimbólum. Az $A_{\text{előző}}$-ből az $A_{\text{aktuális}}$ pontba mutató élhez a $p(A_{\text{aktuális}}|A_{\text{előző}})$ valószínűséget rendeljük.

3.1. példa: Vizsgáljunk meg egy egyszerű, háromelemű ábécét használó forrást. Legyen a betűinek előfordulási valószínűsége $p_{\text{A}} = 5/16$, $p_{\text{B}} = 13/32$, és $p_{\text{C}} = 9/32$, az egyes szimbólumok feltételes valószínűségeit pedig tartalmazza a következő táblázat:

$p(a_i\|a_j)$		j A	B	C
	A	3/10	5/13	2/9
i	B	7/10	0	2/3
	C	0	8/13	1/9

Adjuk meg a forráshoz tartozó gráfot, és számoljuk ki az egyes betűkombinációk előfordulási valószínűségét.

Megoldás: A gráf csomópontjai az egyes szimbólumok, az „A", „B" és „C" betűk. Az A→A élhez tartozó valószínűség a $p(\text{A}|\text{A}) = 3/10$, az A→B élhez tartozó valószínűség a $p(\text{B}|\text{A}) = 7/10$, és így tovább, amint az ábra mutatja:



Az „AB” betűpáros együttes előfordulási valószínűsége felhasználva a (2.2) definíciót

$$p(AB) = p(B|A) \cdot p(A) = \frac{7}{10} \cdot \frac{5}{16} = \frac{7}{32},$$

a „BA” betűpáros együttes előfordulási valószínűsége

$$p(BA) = p(A|B) \cdot p(B) = \frac{5}{13} \cdot \frac{13}{32} = \frac{5}{32},$$

a „AA” betűpárosé pedig

$$p(AA) = p(A|A) \cdot p(A) = \frac{3}{10} \cdot \frac{5}{16} = \frac{3}{32}.$$

Hasonlóképpen a többi együttes valószínűség is kiszámítható, az eredményeket a következő táblázat tartalmazza:

$p(a_i a_j)$		j A	B	C
	A	3/32	7/32	0
i	B	5/32	0	1/4
	C	1/16	3/16	1/32

Látható, hogy a sorok, illetve az oszlopok összege kiadja az aktuális elemek előfordulási valószínűségét, mint azt a (2.4) egyenlőség is leírja, azaz

$$\sum_i p(a_i a_j) = \sum_i p(a_j a_i) = p(a_j).$$

Mivel az egyes betűk előfordulási valószínűségeinek összege (azaz $\sum_j p(a_j) = 1$), a táblázat összes elemét összeadva 1-et kapunk, ahogy a (2.5) is állítja. Figyeljük meg azt is, hogy a $p(a_i|a_j)$ táblázat oszlopaiban felsorolt valószínűségek összege 1, azaz

$$\sum_i p(a_i|a_j) = 1.$$

Ez a (2.3) állítást hivatott demonstrálni.

Ha egy folyamat során adott a rendszer lehetséges állapotainak egy $S = \{S_1, S_2, \ldots, S_n\}$ halmaza, valamint az S_i és S_j állapotok közötti $p(S_j|S_i)$ átmeneti valószínűségek, és a következő állapotba kerülést csak ezek az átmeneti valószínűségek befolyásolják, akkor egy **Markov-folyamatról** van szó. A Markov-folyamatokra vagy Markov-láncokra igaz, hogy

$$p(S_{\text{új}}|S_{\text{előző}} S_{\text{előző}-1} \ldots S_{\text{előző}-m}) = p(S_{\text{új}}|S_{\text{előző}})$$

bármely m-re, azaz csak a legutolsó állapot van hatással arra, hogy mi lesz a következő állapot. A Markov-folyamatokat a 3.1 példában látott gráfokkal szokták szemléltetni.

A forrásokat általában Markov-folyamatokkal lehet leírni.

Markov-folyamat volt az az eset is, amikor az egymást követő szimbólum-kibocsátások egymástól független események voltak: akkor egyetlen lehetséges S állapot volt és az egyes karaktereknek megfelelő élek abból indultak ki és oda is érkeztek, a hozzájuk rendelt valószínűség pedig a $p(A_i)$ előfordulási valószínűség.

A 3.1 példában az állapotok megegyeznek az aktuálist megelőző karakterrel, azaz S_1 =„A”, S_2 =„B” és S_3 =„C”; az átmeneti valószínűségek pedig a $p(a_i|a_j)$ feltételes valószínűségek.

Az olyan források, amelyeknél egy szimbólum kibocsátása az azt megelőző m szimbólumtól függ, szintén modellezhetők Markov-folyamattal, csak ekkor az állapotok a soron következő szimbólum előtti m hosszú szimbólumsorozatok, azaz $S_i = A_{\text{előző}_i} A_{\text{előző}_i-1} \dots A_{\text{előző}_i-m}$. Az átmeneti valószínűségek is $p(A_{\text{aktuális}}|A_{\text{előző}} A_{\text{előző}-1} \dots A_{\text{előző}-m})$ típusúak lesznek.

Pontosabb modell lehet egy forrásra, ha nem karaktereket, hanem szavakat veszünk az egységeinek, csak sokkal több lehetséges kibocsátott szó van, mint betű, így sokkal bonyolultabb modellt kapunk. Ez az eset is leírható Markov-folyamattal.

3.1.1. Forrásentrópia

Tegyük fel, hogy a forrásunk egy adott S_i állapotban van. Ha ismertek a $p(S_j|S_i)$ átmeneti valószínűségek minden j-re, akkor ki tudjuk számolni az i-edik állapotra a $H(S|S_i)$ feltételes entrópiát. A **forrásentrópia** ezeknek a $H(S|S_i)$ feltételes entrópiáknak a várható értéke, azaz ha P_i az S_i állapot előfordulási valószínűsége, akkor a forrásentrópia:

$$\mathcal{H} = \sum_i P_i H(S|S_i) = -\sum_{i,j} P_i \cdot p(S_j|S_i) \log_2 p(S_j|S_i) \tag{3.1}$$

Nyilván P_i-t, az S_i állapot előfordulási valószínűségét – bizonyos speciális eseteket kivéve – csak a forrás hosszú idejű megfigyelésével lehet megállapítani.

Ha a forrás stacionárius (és természetesen emlékezet nélküli), akkor az S_i állapotok maguk az i-edik szimbólumok lesznek, a P_i valószínűség pedig nyilván meg fog egyezni az A_i szimbólum előfordulási valószínűségével, p_i-vel. Stacionárius forrás esetén az egyes szimbólum-kibocsátások egymástól független események, így

$$p(S_j|S_i) = p(A_j|A_i) = p(A_j) = p_j.$$

A forrásentrópia tehát

$$\mathcal{H} = -\sum_i p_i \cdot \sum_j p_j \log_2 p_j = -\sum_j p_j \log_2 p_j$$

meg fog egyezni az egyetlen szimbólum kibocsátásának entrópiájával.

Egy alternatív megközelítés: A forrásentrópia a forrás által kibocsátott N darab egymás utáni szimbólum együttes entrópiájának N-ed része, ha az $N \to \infty$ határértéket vesszük, feltéve, hogy létezik ez a határérték. Stacionárius forrásoknál mindig létezik a határérték.

3.2. Egyértelműen dekódolható kódok

Legyen a forrásábécénk az $A = \{A_1, A_2, \ldots, A_n\}$ halmaz, a kódábécénk pedig a $B = \{B_1, B_2, \ldots, B_s\}$ halmaz. Az **üzenet**ek az A elemeiből képezett véges sorozatok. Az üzenetek halmazát $\mathcal{A}$-val jelöljük. A B elemeiből álló, véges hosszúságú sorozatokból, azaz a **kódszavak**ból álló halmaz pedig a $\mathcal{B}$. **Kód**nak nevezünk ekkor minden $f : A \mapsto \mathcal{B}$ függvényt, azaz az olyan leképezéseket, amelyek a forrásábécé elemeit kódszavakba transzformálják.

Egy $f : A \mapsto \mathcal{B}$ kód **egyértelműen dekódolható**, avagy megfejthető, ha segítségével minden B elemeiből álló véges sorozat csak egyféle $\mathcal{A}$-beli üzenetből állítható elő. (Ez több, mint ha csak azt követelnénk meg, hogy invertálható legyen a leképezés, lényegében azt követeljük meg, hogy az f-ekből felépített $F : \mathcal{A} \mapsto \mathcal{B}$ leképezés legyen invertálható.)

Egy $f : A \mapsto \mathcal{B}$ kódot **prefix**nek nevezünk, ha a lehetséges kódszavak közül egyik sem a másik folytatása. Ugyanazt jelenti, ha azt követeljük meg, hogy egyik kódszó végének megcsonkításával se kapjunk másik értelmes kódszót, bármekkora is legyen a levágott rész. Ha egy kód prefix, akkor egyben egyértelműen dekódolható is.

Például, ha $A = \{a, b, c\}$ a forrásábécénk, akkor prefix kód lehet, ha a három karakterhez a 0, az 10 és az 110 kódszavakat rendeljük, nem prefix, de megfejthető, azaz egyértelműen dekódolható viszont az, ha a 0, a 01 és a 011 lesznek az érvényes kódszavak. (Ez utóbbi kód posztfix: azaz, ha bármely kódszó elejéről vágunk le bármekkora darabot, nem kaphatunk másik értelmes kódszót.)

3.3. A kódszavak általagos hossza és a róluk szóló tételek

Természetesen, ha megállapodtunk, hogy minden kódszó egyforma hosszú, de különböző, akkor a kód egyértelműen dekódolható, sőt prefix lesz. *A forráskódolás fő célja viszont a minél rövidebb kódolt üzenet létrehozása*, amelyet az állandó kódszóhossz nem segít elő. Ha azonban a forrásábécé nagyobb valószínűséggel előforduló elemeihez rövidebb kódszavakat, a valószínűtlenebbekhez meg hosszabb kódszavakat rendelünk, akkor már tettünk egy nem elhanyagolható lépést az entrópianövelés (tömörítés) felé. Az olyan kódokat, amelyek a forrásábécé betűihez eltérő hosszúságú kódszavakat rendelnek, **változó szóhosszúságú** kódoknak nevezzük. Az A_i forrásábécébeli elemhez rendelt kódszó hosszát ℓ_i-vel jelöljük.

Egy $f : A \mapsto \mathcal{B}$ kód $L(A)$ **átlagos szóhosszán** ezen ℓ_i-k (2.6) szerinti várható értékét értjük, azaz

$$L(A) = \sum_{i=1}^{n} p_i \ell_i, \tag{3.2}$$

ahol p_i a forrásábécé A_i elemének előfordulási valószínűsége.

A **McMillan-féle egyenlőtlenség** kapcsolatot teremt a kódábécé elemeinek s száma és a forrásábécé A_i betűihez rendelt kódszavak ℓ_i hossza között a következőképpen: Minden egyértelműen dekódolható $f : A \mapsto \mathcal{B}$ kódra igaz, hogy

$$\sum_{i=1}^{n} s^{-\ell_i} \leq 1. \tag{3.3}$$

A **Kraft-egyenlőtlenség** lényegében a McMillan-egyenlőtlenség állításának megfordítása. Legyenek $\ell_1, \ell_2, \ \ldots, \ell_n$ természetes számok, $s > 1$ egész. Ha ezekre igaz, hogy

$$\sum_{i=1}^{n} s^{-\ell_i} \leq 1, \tag{3.4}$$

akkor létezik olyan prefix kód, amely s elemű kódábécével rendelkezik és egy n elemű forrásábécé i-edik eleméhez éppen ℓ_i hosszúságú kódszót rendel hozzá.

Az *egyértelműen dekódolható kódok átlagos kódszóhosszáról szóló tétel* a forrás szimbólumainak entrópiájától és a kódábécé elemeinek számától függő alsó korlátot ad meg a kódszavak átlagos hosszához. A következőt állítja:

Minden egyértelműen dekódolható $f : A \mapsto \mathcal{B}$ kódra

$$L(A) \geq \frac{H(A)}{\log_2 s}. \tag{3.5}$$

Bizonyítás: Először átszorzunk a jobb oldal nevezőjével és felhasználjuk az átlagos kódszóhossz (3.2) definícióját:

$$H(A) \leq \sum_{i=1}^{n} p_i \ell_i \cdot \log_2 s = -\sum_{i=1}^{n} p_i \log_2 s^{-\ell_i}. \tag{3.6}$$

Ezt az egyenlőtlenséget szeretnénk bebizonyítani. Bevezetjük a $q_i = s^{-\ell_i}/(\sum_{j=1}^{n} s^{-\ell_j})$ segédmennyiséget. Ezenkívül felhasználjuk azt, hogy minden $p_i \geq 0$, $q_i > 0$ $i = 1,2,\ldots,n$ számokra, ha $\sum_i p_i = 1$ és $\sum_i q_i = 1$, akkor

$$-\sum_{i=1}^{n} p_i \log_2 p_i \leq -\sum_{i=1}^{n} p_i \log_2 q_i. \tag{3.7}$$

Definíció szerint

$$H(A) = -\sum_{i=1}^{n} p_i \log_2 p_i. \tag{3.8}$$

A (3.7) egyenlőtlenség bal oldala tehát az entrópia, a jobb oldalába pedig beírhatjuk a q_i mennyiségeket, így

$$\begin{aligned} H(A) &\leq -\sum_{i=1}^{n} p_i \log_2 \frac{s^{-\ell_i}}{\sum_{j=1}^{n} s^{-\ell_j}} = \\ &= -\sum_{i=1}^{n} p_i \log_2 s^{-\ell_i} + \log_2 \left(\sum_{j=1}^{n} s^{-\ell_j} \right) \cdot \sum_{i=1}^{n} p_i. \end{aligned} \tag{3.9}$$

A második tagban a p_i-k összege 1. Ha felhasználjuk a (3.3) McMillan-egyenlőtlenséget, meg azt, hogy a logaritmus függvények monoton növekvők, kiderül, hogy a második tag nullánál kisebb, így az egész összeg felülről becsülhető az első tagjával. Ezzel beláttuk a tételt, igazoltuk, hogy (3.6) teljesül. (Q.E.D)

A prefix kódok között van olyan, amelyik a kódszóhossznak ezt az alsó korlátját elég jól megközelíti. Erről szól a következő tétel:

Létezik olyan $f : A \mapsto \mathcal{B}$ prefix kód, amelynek az átlagos kódszóhossza

$$L(A) < \frac{H(A)}{\log_2 s} + 1. \tag{3.10}$$

Bizonyítás: Legyenek $\ell_1,\ell_2,\ldots,\ell_n$ pozitív egész számok, amelyek mindegyikére igaz, hogy

$$-\frac{\log_2 p_i}{\log_2 s} \leq \ell_i < -\frac{\log_2 p_i}{\log_2 s} + 1. \tag{3.11}$$

Így a p_i-kből egyértelműen következnek az ℓ_i-k. Figyeljük meg, hogy a $\log_2 p_i / \log_2 s$ kifejezés tulajdonképpen p_i-nek az s alapú logaritmusa, $\log_s p_i$.

Vizsgáljuk először a *bal oldal*i egyenlőtlenségeket, pontosabban azoknak a -1-szeresét minden i-re: $\log_s p_i \geq -\ell_i$. Emeljük őket az s kitevőjébe, majd adjuk össze minden i-re:

$$\sum_{i=1}^{n} s^{-\ell_i} \leq \sum_{i=1}^{n} s^{\log_s p_i} = \sum_{i=1}^{n} p_i = 1. \tag{3.12}$$

Így azt kaptuk, hogy az ℓ_i számok kielégítik a (3.4) Kraft-egyenlőtlenség feltételeit. Így az ℓ_i számok lehetnek egy s elemű kódábécével rendelkező prefix kód n darab kódszavának a hosszai.

Ha megszorozzuk a *jobb oldal*i egyenlőtlenséget minden i-re p_i-vel, majd összegezzük őket, akkor a

$$\sum_{i=1}^{n} p_i \ell_i < -\sum_{i=1}^{n} p_i \frac{\log_2 p_i}{\log_2 s} + \sum_{i=1}^{n} p_i \tag{3.13}$$

kifejezésre jutunk, amelynek a bal oldala a kódszavak hosszának várható értéke, $L(A)$, a jobb oldal pedig pont $H(A)/(\log_2 s) + 1$. (Q.E.D.)

Ez utóbbi két tételt együttesen nevezik **Shannon első**, avagy **forráskódolási tételé**nek. Eszerint:

Egy emlékezet nélküli és stacionárius forrás $A_1,A_2,\ldots,A_n$ szimbólumaihoz lehet találni olyan s elemű kódábécével rendelkező $\ell_1,\ell_2,\ldots,\ell_n$ hosszúságú kódszavakat rendelő $f : A \mapsto \mathcal{B}$ kódot, melynek átlagos $L(A)$ szóhossza

$$\frac{H(A)}{\log_2 s} \leq L(A) < \frac{H(A)}{\log_2 s} + 1. \tag{3.14}$$

Bináris kódok esetén, azaz, ha $s = 2$, akkor ez az egyenlőtlenség a következő alakúra egyszerűsödik:

$$H(A) \leq L(A) < H(A) + 1. \tag{3.15}$$

Egy $f : A \mapsto \mathcal{B}$ s elemű kódábécével rendelkező kódot **optimális**nak nevezünk, ha a lehető legkisebb a kódszóhossza, azaz $L(A)$ a lehető legjobban megközelíti a elméleti minimumát, $H(A)/(\log_2 s)$-et. Az optimális kód nem feltétlenül egyértelmű.

4. Forráskódoló eljárások

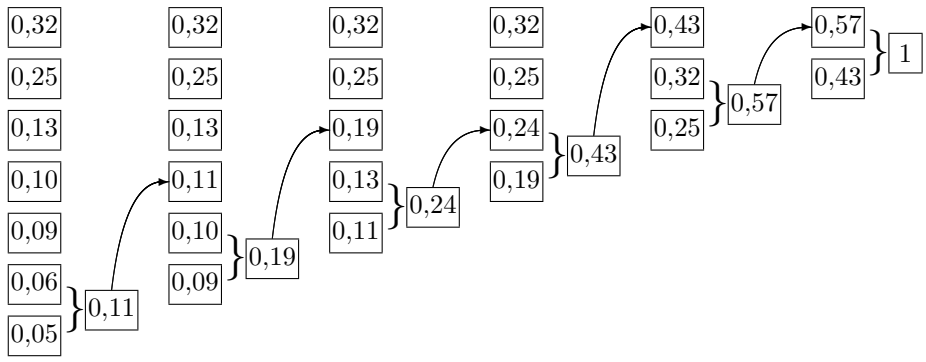
A következő oldalakon megismerkedhetünk néhány ma is használatos, illetve történelmi szempontból jelentős forráskódolási eljárással.

4.1. Huffman-kód

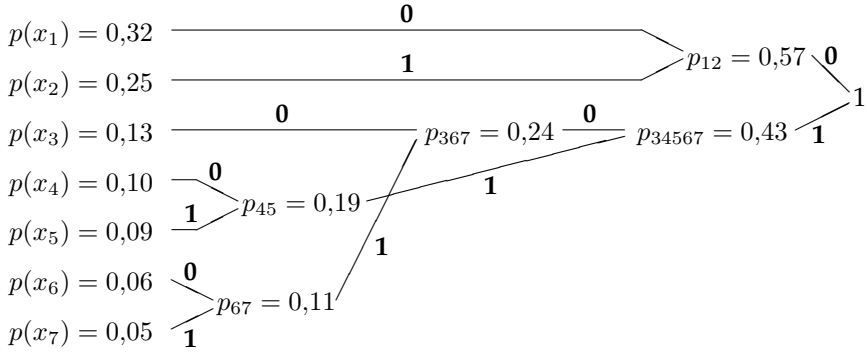
A Huffman-féle eljárás a legrövidebb átlagos szóhosszúságú prefix kódolás, 1952-ben találták ki, és ma is alkalmazzák például a fekete-fehér telefaxoknál és bizonyos képtömörítéseknél. Ahhoz, hogy a módszert alkalmazni lehessen, ismerni kell a forrás szimbólumainak előfordulási valószínűségét, melyet például a forrás hosszabb ideig tartó megfigyelésével lehet meghatározni. Az eljárás minden lépésében összevonjuk a két legkisebb előfordulási valószínűségű karaktert egy új összetett karakterré, míg végül egyetlen, 1 valószínűségű összetett szimbólumot nem kapunk. Az összevonások tulajdonképpen egy bináris fát alkotnak, amelynek a gyökere az 1-es valószínűségű, teljesen összetett szimbólum, az egyes csomópontjai a folyamat során előforduló összetett szimbólumok, míg a levelei a kiindulási karakterek. Az ágak jelképezik az összevonásokat. Az egyes szimbólumokhoz úgy rendelünk bináris kódszót, hogy minden összevonás során az egyik összevont karakterhez – a bináris fa csomópontjaiból elágazó két ág közül az egyikhez – 1-es, a másikhoz 0-s bitet rendelünk, és ezeket a biteket egymás elé írjuk. Az algoritmust a legegyszerűbben egy példa segítségével érthetjük meg.

4.1. példa: Legyen hét átvinni kívánt üzenetkarakterünk, x_1, x_2, x_3, x_4, x_5, x_6 és x_7. Az egyszerűség kedvéért tegyük fel, hogy már sorrendbe rendeztük őket, csökkenő előfordulási valószínűség szerint. Legyen például $p(x_1) = 0{,}32$, $p(x_2) = 0{,}25$, $p(x_3) = 0{,}13$, $p(x_4) = 0{,}10$, $p(x_5) = 0{,}09$, $p(x_6) = 0{,}06$ és $p(x_7) = 0{,}05$. (Ha nem csökkenő valószínűség szerint vannak sorban az elemek, első lépéskén sorba rendezhetjük őket.) Készítsük el a karakterekhez tartozó Huffman-kódot.

Megoldás: A kódolás minden lépésében a két legkisebb valószínűségű szimbólumból egy összetett szimbólumot képezünk. Az új elemhez a két eredeti szimbólum előfordulási valószínűségeinek összegét rendeljük valószínűségként. Az így kapott, eggyel kevesebb szimbólumot újra sorba rendezzük, s a két legkisebb valószínűségűt újra összevonjuk. Addig ismételjük ezt a lépést, amíg végül egy darab, 1 valószínűségű üzenetet nem kapunk. Lássuk:

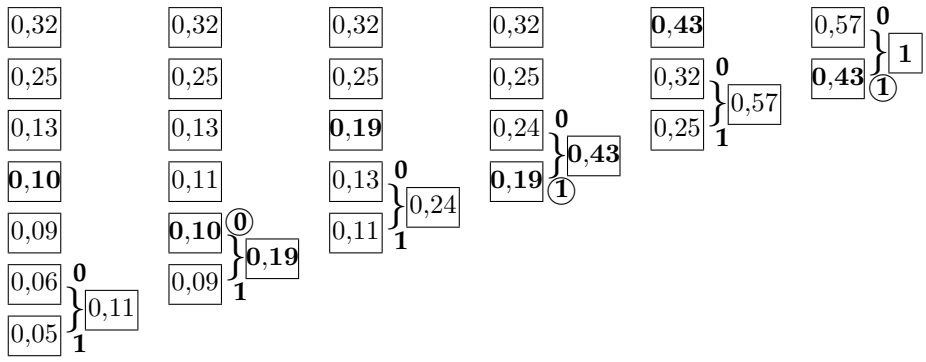


Az eljárás, ahogy a végső szimbólumot kaptuk, tulajdonképpen egy bináris fát rajzol ki, ahol minden egyes csomópont egy-egy az eljárás során kapott összetett vagy egyszerű szimbólum, mint azt a következő ábrán láthatjuk:



Rendeljünk a fa *gyökerétől visszafelé* elindulva minden, az ábrán lefelé fordulóághoz 1-es kódelemet, minden felső ághoz 0-t. Ezek a bitek találhatók vastagon szedve az egyes ágak fölött vagy alatt. Az x_1-es szimbólum kódja így 00 lesz, az x_2-é 01, az x_3-é 100, az x_4-é 110, az x_5-é 111, az x_6-é 1010, az x_7-é pedig 1011. A Huffman-kódolás nem egyértelmű, nem feltétlenül szükséges az, hogy a felső ágak legyenek a nullások, az alsók meg az egyesek, akár lépésenként változtathatjuk a szabályt.

Az egyes szimbólumokhoz a korábbi ábra segítségével is tudunk kódszavakat rendelni. Az egyes összevonásoknak a következő ábrán feltüntetett módon lehet biteket megfeleltetni: szintén vastagon szedett 0-k és 1-esek jelölik az aktuális biteket, a két ábra hozzárendelési szabálya ugyanaz. A kódszavakat ezesetben a következő módon határozzuk meg. Végigkövetjük a keresett kiindulási szimbólum előfordulási valószínűségét, illetve az abból összevonással keletkezett értékeket a diagram minden oszlopán. Példaként egy ilyen folyamat nyomon követhető a vastagon szedett számokkal: az x_4 szimbólum kódszavának a meghatározása.



Amelyik oszlopban az aktuális valószínűség egy másikkal összeadódik, ott a mellette bekarikázva található bitet az eddigi kódszóhoz írjuk, méghozzá a kódszó elejére. Így a negyedik szimbólumhoz tartozó kódszó 110 lesz.

4.2. példa: Vizsgáljuk meg, hogy mennyire lett optimális a kapott kód.

Megoldás: A kódábécé elemszáma $s = 2$, így a (3.14) Shannon-féle forráskódolási tétel a

$$H(x_1,x_2,x_3,x_4,x_5,x_6,x_7) \leq L \leq H(x_1,x_2,x_3,x_4,x_5,x_6,x_7) + 1 \tag{4.1}$$

egyenlőtlenséggé módosul. Ebből a forrásábécé entrópiája

$$\begin{aligned} H(x_1,x_2,x_3,x_4,x_5,x_6,x_7) \;=\; & -0{,}32\log_2 0{,}32 - 0{,}25\log_2 0{,}25 - 0{,}13\log_2 0{,}13 - \\ & -0{,}10\log_2 0{,}10 - 0{,}09\log_2 0{,}09 - 0{,}06\log_2 0{,}06 - \\ & -0{,}05\log_2 0{,}05 = 2{,}513, \end{aligned}$$

az átlagos kódszóhossz pedig

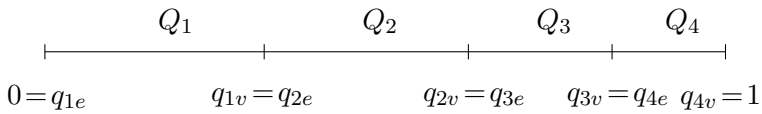
$$\begin{aligned} L \;&=\; 0{,}32\cdot 2 + 0{,}52\cdot 2 + 0{,}13\cdot 3 + 0{,}10\cdot 3 + 0{,}09\cdot 3 + 0{,}06\cdot 4 + 0{,}05\cdot 4 = \\ &=\; 2{,}54. \end{aligned}$$

Látható, hogy (4.1) teljesül, sőt az átlagos kódszóhossz elég jól megközelíti az elméletben elérhető minimális értéket, $H(x_1,x_2,x_3,x_4,x_5,x_6,x_7)$-t. Az, hogy mennyire sikerül a Huffman-féle eljárással kapott kódok átlagos szóhosszával megközelíteni az optimumot, attól függ, hogy milyenek a kódolandó szimbólumok előfordulási valószínűségei.

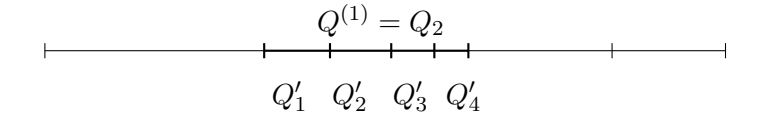
Gazdaságosabb, az optimális kódszóhosszt jobban megközelítő kódokat kaphatunk, ha az üzenetet nem betűnként kódoljuk, hanem blokkokat építünk a betűkből, és a blokkokat kódoljuk Huffman-eljárással. Általában fix számú karakterből álló blokkokat szoktak ilyen célból építeni.

4.2. Aritmetikai kódolás

Az aritmetikai kódolással eleve blokkokat lehet kódolni egy igen látványos módszerrel. Első lépésként az A forrásábécé minden A_i eleméhez hozzárendelünk egy $Q_i = [q_{ie},q_{iv})$ intervallumot a $[0;\ 1)$ intervallumon belül, úgy, hogy az intervallumok teljesen lefedjék a $[0;\ 1)$ intervallumot, viszont ne legyen közös tartományuk, azaz diszjunktak legyenek. (Diszjunkt halmazok metszete üres.) Négyelemű forrásábécére egy példát az alábbi ábra mutat:



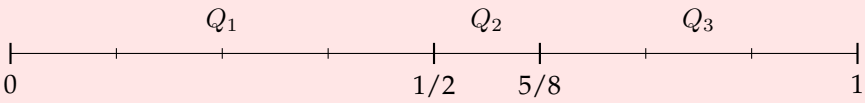
Egy m elemű blokkot a következőképpen kódolunk: kiválasztjuk a $[0;\ 1)$-ben az első elemnek megfelelő részintervallumot, ezt jelöljük $Q^{(1)} = [q_e^{(1)},q_v^{(1)})$-vel. Ezután $Q^{(1)}$ lesz az alapintervallumunk, azt osztjuk fel ugyanolyan arányban, mint az első lépésben a $[0;\ 1)$-est. Az ábrán az így kapott Q'_i új intervallumokat lehet látni az iménti négyelemű forrásábécé második intervallumában (azaz, ha a blokk első karaktere a forrásábécé második eleme volt):



Ezután kiválasztjuk a második karakternek megfelelő $Q^{(2)} = Q'_i$ intervallumot, és azt is felosztjuk ugyanolyan arányban, mint a $[0;\ 1)$ intervallumot, és így tovább, amíg a blokk utolsó, m-edik karakterét nem kódoltuk. Végül megkapjuk a már elég kicsi, $Q^{(m)} = [q_e^{(m)},q_v^{(m)})$ intervallumot. Ebből kiválasztunk egy elemet, és az lesz a kódszavunk. Mivel sehol sem fednek át az intervallumok, mindig vissza tudjuk kapni az eredeti blokkot, ha tudjuk a forrásábécét és az eredeti felosztási arányokat.

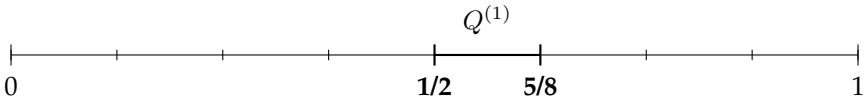
Ha binárisan kódolunk célszerű a végső intervallumbeli legrövidebb bináris törtet kiválasztani kódszónak. Szintén célszerű a gyakrabban előforduló karakterekhez hosszabb intervallumot rendelni, a ritkábbakhoz pedig rövid intervallumokat. Ez esetben ugyanis egy csupa gyakori karakterből álló blokkhoz hosszabb $Q^{(m)}$ intervallum keletkezik, amelyben nagyobb eséllyel találunk rövid kódszót. Ez azt eredményezi, hogy a gyakoribb blokkokhoz rövidebb kódszavakat fogunk rendelni, azaz jobban tudjuk tömöríteni az üzenetet. Nézzünk egy példát.

4.3. példa: Legyen az a_1 forrásábécébeli elem előfordulási valószínűsége $p_1 = 1/2$, az a_2-é $p_2 = 1/8$ az a_3-é pedig $p_3 = 3/8$. Osszuk fel úgy a $[0;\ 1)$ intervallumot, hogy a Q_i részintervallumok hossza azonos legyen a hozzájuk rendelt szimbólum előfordulási valószínűségével. Ekkor még mindig szabadon megválaszthatjuk a részintervallumok sorrendjét; tartozzon most a Q_1 első intervallum az a_1 karakterhez, a Q_2 második intervallum az a_2-höz, a Q_3 pedig az a_3-hoz. Az osztáspontok így a következő, vastag vonással jelölt pontok lesznek:

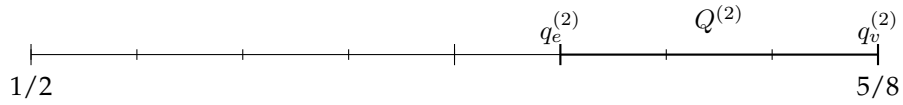


Kódoljuk az „$a_2a_3a_2a_1$" blokkot aritmetikai kódolással.

Megoldás: A blokk első karaktere az a_2, így a $Q^{(1)}$ első részintervallum az alábbi ábrán vastag vonallal kiemelt $[1/2;\ 5/8)$ lesz:



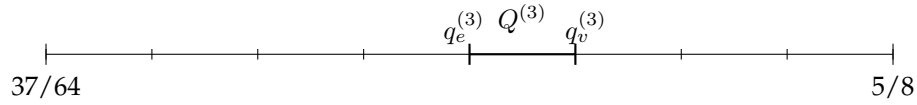
A $Q^{(1)}$ részintervallum hossza $l^{(1)} = 1/8$, ezt a kis intervallumot kell az eredeti $[0;\ 1)$ intervallum felosztásával megegyező arányban osztani. Hogy jobban tudjuk követni a folyamatot, nagyítsuk ki $Q^{(1)}$-et az eredeti $[0;\ 1)$ intervallummal azonos hosszúságúra:



Az ábrán vastag vonallal szerepel a blokk második karakterének, az a_3-nak megfelelő részintervallum, melynek végpontja jelen esetben megegyezik a $Q^{(1)}$ kiindulási intervalluméval, azaz $q_v^{(2)} = 5/8$, a kezdőpontja pedig

$$q_e^{(2)} = q_e^{(1)} + \frac{5}{8} \cdot l^{(1)} = \frac{1}{2} + \frac{5}{8} \cdot \frac{1}{8} = \frac{37}{64}.$$

A kapott $[37/64;\ 5/8)$ intervallum hossza $l^{(2)} = 3/64$ lesz, ezt megint kinagyítva ábrázoljuk. Az ábrán megvastagítottuk a blokk harmadik karakterének, az a_2-nek megfelelő $Q^{(3)}$ szakaszt.



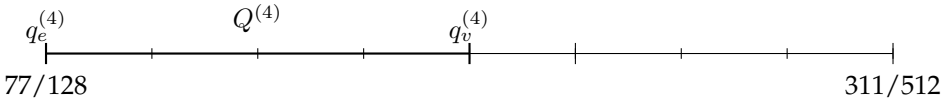
A harmadik szimbólumnak megfelelő részintervallum kezdőpontja

$$q_e^{(3)} = q_e^{(2)} + \frac{1}{2} \cdot l^{(2)} = \frac{37}{64} + \frac{1}{2} \cdot \frac{3}{64} = \frac{77}{128},$$

a végpontja pedig

$$q_v^{(3)} = q_e^{(2)} + \frac{5}{8} \cdot l^{(2)} = \frac{37}{64} + \frac{5}{8} \cdot \frac{3}{64} = \frac{311}{512}.$$

A blokk utolsó, a_1-es szimbólumához tehát az $l^{(3)} = 3/512$ hosszúságú $Q^{(3)}$ intervallum első része fog tartozni, amint az az alábbi ábrán kinagyítva látható.



A $Q^{(4)}$ intervallum kezdő és végpontjai a következők

$$\begin{aligned} q_e^{(4)} &= \frac{77}{128} \\ q_v^{(4)} &= \frac{77}{128} + \frac{1}{2} \cdot \frac{3}{512} = \frac{619}{1024}. \end{aligned}$$

Tehát a $[77/128;\ 619/1024)$ intervallumban keressük a legrövidebb bináris törtet. Írjuk át a kezdő és végpontokat bináris tört alakba:

$$[1001101;\ 101101011)$$

A két szám bináris alakja egy darabig megegyezik, utána a következő bit a kezdőpontban 0 (vagy semmi), a végpontban nyilván 1 lesz.

- Ha a kezdőpont a közös résszel befejeződik, akkor ő lesz a jó kódszó, ha nem, két eset lehetséges:

- Ha a végpont ezen az 1-esen kívül további számjegyeket is tartalmaz, akkor a közös rész egy 1-essel kiegészítve jó kódszó lesz.
- Mivel az intervallum végpontja nem tartozik az intervallumhoz, ha a szóban forgó 1-essel befejeződik a végpont bináris törtje, akkor egy 01 bitpárost kell a közös rész után írni, hogy a kódszót megkapjuk.

A mi esetünkben a 77/128 és a 619/1024 bináris alakjában az első két bit egyezik meg, utána a kezdőpont 0-t, a végpont 1-est és további nem csupa 0 számjegyeket tartalmaz, így a második eset áll fenn. Az „$a_2a_3a_2a_1$" blokkhoz rendelt kódszó tehát 101 lesz.

4.3. A Lempel–Ziv-kódolások

A Lempel–Ziv-kódolásokhoz nem szükséges ismerni a kódolni kívánt szimbólumok előfordulási valószínűségét. Egy adott, véges számú elemből felépülő bemeneti üzenetet képezünk egy véges számú elemből álló kódszóhalmazra, úgy hogy a kódolási eljárás során magából *az üzenetből dinamikusan generálódnak a kódszavak*. Az eljárások alapötlete az, hogy a bemeneti szimbólumsorozatot különböző blokkokra bontjuk, miközben folyamatosan elraktározzuk a már látott blokkokat egy szótárban. A még nem látott blokkoknak lényegében csak a már látottaktól való eltérését tároljuk.

Például az LZ78 algoritmus az alábbiak szerint épül fel. A szótár láncolt lista szerkezetű, minden sorában az egyik oszlopban a sor címkéjét (m), a másikban az előzmény sorcímkéjét (n), a harmadikban pedig az utolsó karaktert tároljuk. A kódolás elején a szótárban csak a nulladik sora van meg, az természetesen $n = 0$-ra mutat és üres a szimbólum mezője. Amíg el nem fogynak a szimbólumok, minden lépésben beolvassuk a következő karaktert.

- Ha az a karakter még nem szerepel a szótárban, akkor csinálunk egy új sort neki, amelynek a mutató része a nulladik sorra mutat, a címe pedig eggyel nagyobb, mint az eddig szerepelt maximális cím.
- Ha szerepel a beolvasott karakter azon sorok valamelyikében, amelyik a nulladik sorra mutató n-nel rendelkezik, megjegyezzük azt az m-et, amelyik sorban van. Ha már van megjegyzett m címünk, akkor természetesen azok között a sorok között keressük a beolvasott karaktert, amelyek a megjegyzett címre mutatnak, nem pedig az $n = 0$-s sorokban.

- Ha szerepel a karakter a szótárban, de nincs olyan, az utolsó megjegyzett m-re mutató sor, amelyiknek a bejegyzése az aktuális karakter, akkor nyitunk neki egy sort, amelyiknek a sorszáma eggyel nagyobb lesz, mint az utolsó sor száma, a mutatómezeje a megjegyzett m-edik sorra fog mutatni.

Minden szótársor nyitása után a megjegyzett sorszám 0-ra áll vissza. Ezeket a lépéseket addig ismételjük, amíg el nem fogy az üzenet, vagy be nem telik a szótár. A szótár maximális méretét ugyanis többnyire meg szokták adni. Vegyünk egy példát!

4.4. példa: Legyen három karakterünk, a, b és c. Kódoljuk LZ78-as algoritmussal az $abaabaabaccbabcabcaacba$ üzenetet.

Megoldás: A szimbólumsorozat alatti kapcsok jelzik az egyes lépéseket,

$$\overbrace{a}\ \overbrace{b}\ \overbrace{a\,a}\ \overbrace{b\,a}\ \overbrace{a\,b}\ \overbrace{a}\ \overbrace{c}\ \overbrace{c\,b}\ \overbrace{a\,b\,c}\ \overbrace{a\,b\,c\,a}\ \overbrace{a\,c}\ \overbrace{b\,a}$$

a kapott szótár:

m	n	szimbólum	sorozat
0	0	–	–
1	0	a	a
2	0	b	b
3	1	a	aa
4	2	a	ba
5	2	b	ab
–	1	–	a
6	0	c	c
7	6	b	cb
8	5	c	abc
9	8	a	$abca$
10	1	c	ac
–	4	–	ba

Ahhoz, hogy a szótárat és az üzenetet reprodukálni tudjuk, elegendő az „n” és a „szimbólum” oszlopok ismerete. Előfordulhat, hogy új karakter előtt, vagy az üzenet végén nem befejezett, frissen elraktározott karakterláncunk van, hanem egy olyan, amely már szerepel a szótárban. Ilyen esetekben nem kell új szótársort nyitni, csak az utolsó megjegyzett sorszámot kell a dekódoló tudtára adni, meg azt, hogy nem került bejegyzés a szótárba. Ugyanez a teendő, ha a szótár elérte az engedélyezett méretét; ekkor nem történik több bejegyzés, csak a meglévő sorok között válogatunk.

A továbbfejlesztett, LZW-eljárás nagyobb tömörítést tesz lehetővé. A szótár első soraiban felsorolják az összes, az üzenetben előforduló szimbólumot, ezeket a dekódoló is ismeri a folyamat elejétől kezdve. A tényleges szótárépítés során nem használják többé a tényleges karaktereket, csak az őket tartalmazó szótársor sorszámát. A módszer fő eltérése az LZ78-astól az, hogy egy-egy új szótársor megnyitása után nem nullára áll vissza az utolsó megjegyzett sor sorszáma, hanem a legutolsó karaktert rejtő sor sorszámára. Ez azért előnyös, mert az utolsó karakter a következő sorozat része lesz, így azt nem kell ismertetni a dekódolóval.

4.5. példa: Nézzük az előbbi példát az LZW-algoritmussal kódolva.

Megoldás: Az üzenet :

$$\overbrace{a\ b}\ \overbrace{a\ a}\ b\ \overbrace{a\ a}\ \overbrace{b\ a}\ \overbrace{c\ c}\ \overbrace{b\ a}\ \overbrace{b\ c}\ \overbrace{a\ b}\ \overbrace{c\ a}\ \overbrace{a\ c}\ b\ a$$

a kapott szótár:

m	n	szimbólum	sorozat	kimenet
0	0	–	–	
1	0	a	–	„a”
2	0	b	–	„b”
3	1	c	–	„c”
4	2	b	ab	1
5	1	a	ba	2
6	4	a	aa	1
7	6	a	aba	4
8	6	b	aab	6
9	5	c	bac	5
10	3	c	cc	3
11	3	b	cb	3
12	5	b	bab	5
13	2	c	bc	2
14	3	a	ca	3
15	4	c	abc	4
16	14	a	caa	14
17	1	c	ac	1
18	11	a	cba	11
–	–	–	a	1

A vízszintes vonal határolja a definiáló és az érdemi részt. Az üzeneten kapcsos zárójelek jelzik az egyes lépéseket.

Természetesen, mivel a szótár az üzenethez kötődik, s minden egyes szimbólumsorozatra más és más lesz. Látható, hogy ha tömörítésre szeretnénk ezt az algoritmust használni, rövid üzenetet nem érdemes így kódolni, hiszen a szótár felépítése az üzenet elején nagy mennyiségű plusz adatot generál, anélkül, hogy lényegesen csökkentené az üzenet hosszát. Nagyobb méretű üzenetet azonban igen jó arányban lehet tömöríteni ezzel a módszerrel. Ennek az az alapja, hogy a Lempel–Ziv-algoritmusokkal generált „kódszavak" körülbelül egyforma valószínűséggel fordulnak elő a szövegben.

A Lempel–Ziv-kódolások során a szótárakat nem engedik a végtelenségig nőni, egy idő után csak a már meglévő elemekből építkeznek. A szótár méretének csökkentését segíti elő az is, hogy a nagyon ritkán használt sorokat törlik, helyettük gyakoribbakat töltenek be.

5. Forráskódolás a gyakorlatban

5.1. A mintavételezésről és a kvantálásról

A hangok és képek időben és intenzitásban is folytonos függvényekkel írhatók le. Az analóg hírközlésben ezeket a jeleket – kisebb módosításokkal – valamilyen moduláció segítségével ráültetik egy adott frekvenciájú szinuszos vivőjelre.

Digitális hírközlés során ezeket a leíró függvényeket valamilyen módon időben és intenzitásban is diszkrét jelekké kell alakítani. Ez az átalakítás, ha ügyesen csináljuk, szintén lehetőséget biztosít tömörítésre, entrópianövelésre.

Az első lépés az időbeni **mintavételezés**. Vegyünk egy $f(t)$ függvényt. Ennek a T *mintavételi idő*vel vett mintavételezettje az

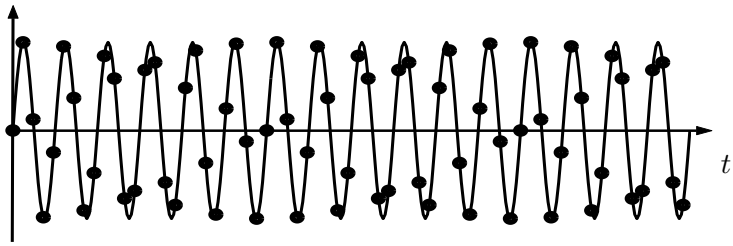
$$f(t_0), f(t_0+T), f(t_0+2T), f(t_0+3T), \ldots \tag{5.1}$$

számsorozat. Sok esetben $t_0 = 0$.

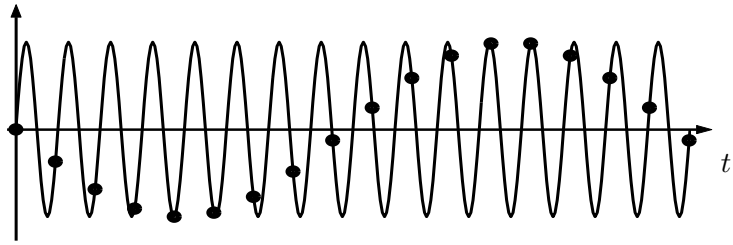
A mintavételi idő meghatározása alapvető feladat, hiszen ha túl nagy időközönként veszünk mintát, akkor sok információt veszíthetünk a jelről, és nem leszünk képesek azt visszaállítani. Ha igen kis időközönként veszünk mintát, akkor túl sok pontunk marad, kicsi lesz a (5.1) sorozat entrópiája, nem lesz elég nagy a tömörítés, pedig a mintavételezésnek ez is célja. A mintavételezési tétel a mintavételezési frekvenciának ad alsó korlátot – a mintavételezési időnek felső korlátot. Vegyünk egy olyan jelet, amely frekvenciatartományban korlátos, azaz amelynek egy $\frac{1}{2\pi}[-B,B]$ frekvenciaintervallumon kívüli összetevői (jó közelítéssel) nullák. Az ilyen jeleket B *sávra korlátozott*aknak nevezik. A **mintavételezési tétel** szerint egy B sávra korlátozott jelre a mintavételi idő

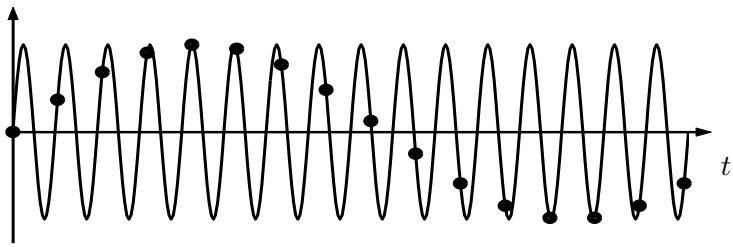
$$T < \frac{\pi}{B} \tag{5.2}$$

legyen. Ez azt jelenti, hogy a mintavételezés frekvenciája legalább a jelben szereplő maximális frekvencia kétszerese kell, hogy legyen. A tétel bizonyításával nem foglalkozunk, de néhány ábrával demonstráljuk érvényességét. Mindhárom ábrán a mintavételezett szinuszos jel folytonos vonallal, a mintavételezési pontokban vett értékei pedig sötét pöttyökkel szerepelnek. Az első ábrán kellően kicsi mintavételezési időt használtunk, a jelalak visszaállítására van esély:



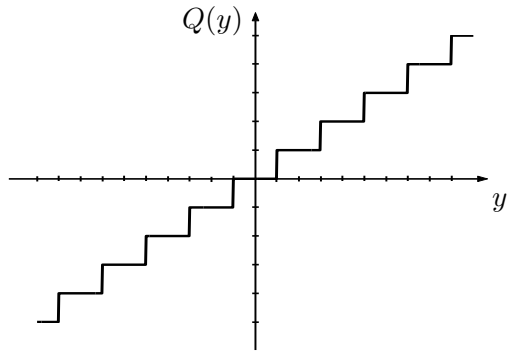
A második ábrán egy a szükséges mintavételezési frekvenciánál kicsit kisebb frekvenciával mintavételeztük a jelet, a harmadikon pedig a szükséges mintavételezési idő kétszeresénél is nagyobb időintervallumonként:





Látható, hogy a második és harmadik esetben a jelalak visszaállítása nem lehetséges – feltéve, hogy nem tudjuk eleve, hogy alul mintavételeztünk. Mindkét esetben, a mintavételezési pontok helytelen megválasztása miatt, egy sokkal kisebb frekvenciájú jelre következtethetünk a mintavételezési pontokban felvett értékekből. (Bizonyos esetekben ki szokták használni, hogy az alul mintavételezett periodikus jelek hasonlítanak az eredetijükre, csak a frekvenciájuk más. Ilyen eset, ha közelítőleg ismerik a jel frekvenciáját, és a mintavételező annál csak kisebb frekvenciákra képes.)

Ha digitálisan szeretnénk továbbítani vagy tárolni az információt, akkor a mintavételezés utáni folytonos (5.1) értékeket valamilyen módszerrel diszkretizálni kell. Azt az algoritmust, amellyel a jelünket leíró $f(t)$ függvény folytonos értékkészletét nem átfedő, de összességében a teljes értékkészletet lefedő intervallumokra bontjuk, majd ezekhez az intervallumokhoz egy-egy – többnyire binárisan jól ábrázolható – számot rendelünk, **kvantálás**nak hívjuk. A kvantálást többnyire egy $Q(y)$ függvénnyel reprezentáljuk, melynek értelmezési tartománya a lehetséges kvantálandó y_i jelek halmaza (általában egy zárt, de folytonos intervallum), értékkészlete pedig véges sok számból áll. Igen kézenfekvő például a következő lépcsősfüggvényt választani kvantálónak:



Az ilyen kvantálók a lineáris kvantálók. Természetesen ez az egyszerű függvény többnyire nem tömörít elég gazdaságosan, hiszen például az ember fül vagy szem nem egyformán érzékeny a különböző intenzitású hangra, illetve fényre. Ezért többnyire vagy bonyolultabb kvantálást alkalmaznak, vagy pedig kvantálás előtt megfelelőképpen transzformálják a jelet.

A kvantálást lehet jellemezni a **négyzetes torzítás**sal, ami tulajdonképpen az eredeti számsornak a kvantált számsortól vett eltérésnégyzeteinek várhatóértéke:

$$D(Q) = \frac{1}{N} \left\langle \sum_{i=1}^{N} (y_i - Q(y_i))^2 \right\rangle .$$

Egy jó, az alkalmazáshoz illeszkedő kvantáló megszerkesztése bonyolult feladat.

Ha a kvantálás végeredménye nem bináris szám, viszont bináris jeleket szeretnénk eredményül, egy újabb hozzárendeléssel kell megoldani a problémát.

5.2. A hang kódolása

A modernebb hangtömörítő eljárások figyelembe veszik az emberi hallás sajátosságait. Az emberi hallás leírására a frekvenciatartományban állnak rendelkezésre jó modellek, ezért általában a frekvenciatartományban analizálják a hangokat.

Jó közelítéssel 20 Hz és 20 kHz közötti frekvenciájú hangokat hallunk. Az igen alacsony és az igen magas frekvenciájú hangokra azonban sokkal kevésbé érzékeny a fülünk, mint a 2 és 4 kHz közötti tartományban, ezért a hallható frekvenciatartomány szélein sokkal nagyobb torzítást engedhetünk meg: ott sokkal nagyobb lépésközű kvantálónk lehet, mint a középső frekvenciatartományban. Ezen kívül egy nagy intenzitású hang a fülünk számára elfedi, *maszkol*ja a hozzá közeli frekvenciájú halkabb hangokat, ha azok vele egyidőben szólnak. A **maszkolás** tulajdonképpen a nagy intenzitású hang bekapcsolása előtti rövid, kb. 2 ms-os, és kikapcsolása utáni kb. 15 ms-os időintervallumra is kiterjed. Mindezek következményeként nem kell minden frekvenciatartománybeli hangösszetevőt egyforma felbontással kvantálni, illetve bizonyos összetevők el is hagyhatók.

A CD-k kevésbé használják ki ezeket a lehetőségeket, az egyes hangcsatornák (sztereó hangzás esetén például a jobb és a bal oldali hangcsatorna) jeleit általában $44{,}1$ kHz frekvenciával mintavételezik. A mintavételezett jelet lineáris kvantálóval 2 bájtra kvantálják, majd megfelelő eljárással rögzítik. A filmek hangtömörítő eljárásai – mint az MPEG Layer1, 2, és 3 elnevezésű algoritmusai – egyre jobban kihasználják a maszkolás jelensége által nyújtott lehetőségeket. A filmek hangjait 32, $44{,}1$, vagy 48 kHz frekvenciával mintavételezik, majd a teljes frekvenciasávot 32 részsávra bontják. Minden részsávban elvégeznek egy a maszkolásokat figyelembe vevő transzformációt – a különböző részsávokban különbözőket –, majd a részsávnak megfelelő finomsággal kvantálnak. A fejlettebb algoritmusok (például az MP3) a részsávokat tovább bontják, jobb, a maszkolást jobban figyelembe vevő transzformációkat használnak, a lineáris kvantáló helyett a feladathoz jobban illeszkedőt alkalmaznak, illetve a kimeneti jeleket még egy általános forráskódolási eljárással (például Huffman-kóddal) tömörítik is.

5.3. Állóképek fekete-fehérben és színesben

A képek digitális feldolgozásakor az első lépés a kép területének apró négyzetekre, **pixelekre**, való felbontása. Ha a képek mozgókép részei, akkor a pixelek száma meghatározott (720×480, az NTSC szabvány szerint, illetve

768×576 a PAL szerint), egyébként tetszőleges lehet. mozgóképeknél szükséges adat a képfrissítési frekvencia is. Fekete-fehér képeknél az egyes pixelekhez csak egy adatot rendelnek hozzá: a képpont világosságát, míg színes képeknél (néhány eset kivételével) több értéket is: a vörös, zöld és kék színek intenzitását az adott pontban, vagy a világosságot és még két színkoordináta értékét. A színtérben való kvantálást a **színmélység** jellemzi, amely azt mondja meg, hogy egy pixel leírására hány bitet használunk. Általában 8 vagy 24 bit szokott lenni a színmélység.

Az emberi szemben található receptorok (csapok és pálcikák) a 360 nm és 830 nm közötti hullámhosszú fényre érzékenyek. Általában a három különböző hullámhossz-tartományú fényre érzékeny háromféle csap alakítja ki főként a látást, mindegyiknek az intenzitásra való érzékenysége közel lineáris (legalábbis egy tartományon belül). A szem általában a fényességbeli eltérésekre jobban reagál, mint a szín eltéréseire. Könnyebben észreveszünk egy halványabb mintázatot, hogyha az ritkább, mintha sűrűbb, tehát a különböző térbeli frekvenciával rendelkező képösszetevőkre sem egyforma a szem érzékenysége. Sok helyen ezt ki is használják, a nagyobb térbeli frekvenciájú komponenseket elhagyják, főleg, ha például filmben csak rövid ideig jelenik meg.

A színeket két különböző, de egymásba transzformálható, három komponensű vektorral szokták leírni. A televíziókban általában egy pixelhez három különböző színű pont (például elektronok által besugárzott foszforpötty, világító tranzisztor) tartozik: piros, zöld és kék. Attól függően látjuk a képpontot valamilyen színűnek, hogy az egyes pöttyök milyen intenzitással világítanak. Az RGB (red, green, blue) paraméterek mindegyikét legtöbbször nyolc biten kvantálják, ez általában elegendően finom osztás, 2^{24}-féle színt képes megkülönböztetni. Ez a három paraméter helyettesíthető másik hárommal, az Y luminanciával, amely a képpont fényességét írja le, és a két krominanciával, C_r-rel és C_b-vel, amelyek a színérzetért felelősek. Mivel a szem a fény intenzitására érzékenyebb, mint a színre, az utóbbiak kvantálásakor nagyobb torzítást engedhetünk meg, mint Y-nál.

5.3.1. A GIF (Graphics Interchange Format) szabvány

Ez a szabvány nem hagy el részleteket a képből, a soronként letapogatott, mintavételezett, kvantált jelsorozatot tömöríti egy Lempel–Ziv-algoritmussal. A színeket indexelt tárolással kezeli. Ez azt jelenti, hogy a színértékekből (azok RGB koordinátáiból) táblázatot alakít ki, és az egyes képpontokhoz csak a megfelelő elem címkéjét kell eltárolni. Az így kapott szimbólumsorozatot tömörítik LZW algoritmussal. Mivel a táblázatot –

melyet ez esetben *palettá*nak hívnak – és a Lempel–Ziv-kód szótárát el kell raktározni, egy ilyen tömörítés csak kellően nagy képeknél tud kifizetődni.

Akkor célszerű ezt az algoritmust használni, ha a képen csak kevés szín van, így kicsi lesz a paletta. Hatékony a módszer akkor is, ha egy-egy szín nagyobb felületeket tölt ki, hiszen ilyenkor a Lempel–Ziv-tömörítés a hosszan ismétlődő szimbólumokat egy rövid kódszóba viszi. Ilyenek például a számítógépeken szereplő ikonok: kevés színből építkeznek és egy-egy színt nagy, kiterjedt felületen használnak. A palettát lehet mesterségesen is csökkenteni: egy olyan szín helyett, amely még nem szerepel a táblázatban, lehet egy hozzá közeli, már szereplő színre mutatni. A fényképek tömörítésére a módszer nem igazán megfelelő.

A tömörített kép elején közlik a kép méretét, és a paletta elemeinek megcímkézésére használt bitek számát. A paletta a kódolás során folyamatosan nőhet, ha alulbecsültük az elején a szükséges színek számát, és tele lesz a táblázatunk, a program megduplázza annak méretét, és eggyel megnöveli a címkézésre felhasznált bitek számat. Csak addig dupláz, amíg a felhasznált színek száma el nem ér egy korlátot (2^{12}), utána már csak a palettán szereplő színekkel közelít.

5.3.2. A JPEG (Joint Photographic Experts Group) szabvány

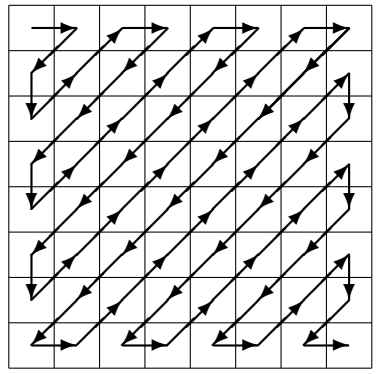
A nyolcvanas évek közepén összefogott az International Telecommunication Union (ITU) és az International Organisation for Standardization (az ISO) néhány hozzáértő embere, hogy létrehozzon egy szabványrendszert a színes és fekete-fehér állóképek tömörítésére. A két szervezetből alakult csoport nevezte magát, illetve az általuk kidolgozott és összegyűjtött kódolásokat JPEG-nek. A szabvány keretet kínál veszteségmentes és veszteséges tömörítésre is.

A **veszteségmentes** tömörítési eljárásuk egy úgynevezett *prediktív kódolás* (predikció – jóslás), amely az egy képponthoz tartozó intenzitásértékek helyett csak azoknak egy az intenzitást becslő értéktől való kis eltérését tárolja el. A becsült értéket mindig már meglévő szomszédos képpontok intenzitásaiból állítják elő. A képnél egy pixel már meglévő három szomszédos képpontja a fölötte, az előtte és az előtte átlósan felfelé lévő pont. Ha elég jó az ezekből – például ezek számtani közepeként – származtatott becslés, akkor igen kicsi eltérést kell eltárolni, ami természetesen kevesebb tárat igényel. A kapott eltéréseket egyfajta aritmetikai kódolással tömörítik.

A veszteséges kódolásoknál először előállítják és különválasztják a három képsíkot (az Y luminanciát és a két C_r és C_b krominanciát). A JPEG teljesen elszeparálva kezeli a három képsíkot; egy színes kép helyett három

egyszínűt használ. Azt megengedi, hogy a három képsík más és más térbeli felbontást használjon. Mindhárom szint $8 - 8$ biten tárolják képpontonként, de egy-egy képpont a két krominancia-képben általában nagyobb, mint a finomabb térbeli felbontású luminancia síkban. Ezek után a kódoló a következő lépéseket hajtja végre:

- Felbontja a képeket 8×8 pixeles négyzetekre, úgynevezett *csempé*kre.
- minden csempét diszkrét koszinusz *transzformáció*nak vet alá. Így kap a frekvenciatartományban egy valós számokból álló sorozatot.
- A kapott valós számsorozatot újfent *kvantál*ja, hogy egész értékei legyenek. A kvantáló egyenletes, de a lépésköze a csempe minden elemére (a csempében található különböző frekvenciájú tagokra) más és más lehet. A csempe elemei úgy helyezkednek el, hogy kisebb frekvenciás tagok – amelyekre a szem érzékenyebb – a bal felső sarokba kerülnek, őket szokás kisebb lépésközzel kvantálni, a nagyobb frekvenciás, jobb alsó elemeket nagy lépésközzel. Az egyes elemekre vonatkozó *kvantálási lépésköző*et is el kell tárolni egy táblázatban.
- A nagyfrekvenciás, majdnem nulla elemeket a csempében elhagyja a tömörítő eljárás. Így tehát a csempe jobb alsó sarkában szinte csak nulla van. Annak érdekében, hogy ezek egy hosszú nullákból álló sorozatot alkossanak, amelyet könnyű kezelni, a JPEG a következő *kiolvasási sorrend*et használja:



- Az úgynevezett *futamhossz-kódolás* során a kapott, sok nullát tartalmazó sorozatot úgy bontják részekre, hogy minden részsorozat valamennyi (lehet nulla is) nullával kezdődjön, és egyetlen nem nulla elemmel végződjön. Egy ilyen részsorozathoz hozzárendel a kód három számot: a nullák számát, a nem nulla elem leírásához használt bitek számát

és a nem nulla elem értékét. Látható, hogy a különböző értékeket különböző bit pontossággal lehet a JPEG-en belül tárolni.

- Végül *Huffman-kód*dal tömöríti a kapott számhármasokból az első kétkét elemet. A Huffman-kódot nem lehet megválasztani, az rögzített a szabványban. Későbbi verziókban aritmetikai kódolás is lehetséges.

Ha egy 24 bit/képpontos képet JPEG-gel tömörítenek képpontonként 2 bitesre, a különbség alig észrevehető.

A JPEG2000 szabvány alapvető újdonsága az, hogy diszkrét koszinusz transzformáció helyett wavelet-transzformációt alkalmaz. A wavelet-transzformáció képes arra, hogy helyről helyre különböző frekvenciájú felbontást használjon, az éles határvonalakon lokálisan finomabb felbontás lehetséges, mint a homogénebb háttéren. Ha azonos tömörítési arányt használunk, az éles vonalak kevésbé elmosódottak JPEG2000-rel, mint az eredeti JPEG szabvány esetén, amely az egész blokkot egyféle felbontási szinten kezelő koszinusz transzformációt alkalmaz.

5.4. Mozgóképek

A hagyományos videoszalagra minden képkockát egyforma számú bittel írtak le, így a folytonos szalagolvasási sebesség folytonos filmnézést tett lehetővé. Lehetetlen volt azonban a – mintavételezés és a kvantálás finomságának megválasztásán túl – a mozgóképet tömöríteni, így a filmeket leíró fájlok igen nagyok voltak. A videoszalag hosszát a film hosszának megfelelőre vágták, azt oda-vissza le lehet játszani, és odébb is lehet tekerni. A CD-k majd DVD-k megjelenése tette szükségessé a mozgókép-tömörítést, hiszen azokat nem kell feltétlenül állandó sebességgel forgatni.

Az **MPEG**, a Moving Picture Experts Group az ISO egy csoportja, amely a veszteséges mozgókép-tömörítés szabványosításával foglalkozik. Ezen szabványokat mind a digitális televíziózásban, mind a filmeknél, mind pedig az interaktív multimédiás alkalmazásokban használják. Mivel egy mozgókép mintavételezése és kvantálása rendkívül nagyméretű adatfolyamokat eredményez, igen nagy tömörítésre van szükség.

Az MPEG-gel kódolt bitfolyam különböző részekből tevődik össze.

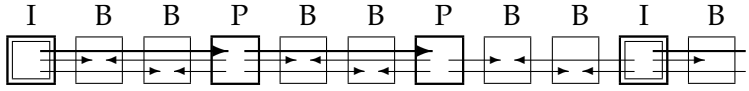
- A legbővebb halmaz a *videoszekvencia*, amelynek a fejléce tartalmazza a képméretet, képsebességet és a képsorozat fontos paramétereit.
- A videoszekvencia *képcsoport*okból áll. Egy képcsoport több, egymás utáni kép. Minden képcsoportot külön, egymástól függetlenül kódolnak.

- A *kép*eknek három típusát különbözteti meg a szabvány, ezeket a típusokat I-vel, P-vel és B-vel jelöli.
- A képek *sáv*okból állnak, és a sávok elején mindig ugyanaz a bitsorozat található, hogy ha valamilyen adatátviteli hiba lép fel, akkor is tudja a dekódoló, hogy hol vagyunk, azaz képes legyen a jellel szinkronizálódni.
- Egy sáv 16 *makroblokk*ból épül fel, egy-egy makroblokk 16×16 képpont leírására szolgál.
- A makroblokkok mindegyike 6 *blokk* együttese. Minden blokk 8×8-as mátrix. Két blokk írja le a két krominanciát, a maradék négy a luminanciát, amelyet kétszer finomabb térbeli felbontással kezel a kód.

Egy mozgóképben általában az egymást követő képek csak kis mértékben térnek el egymástól, így elvileg szükségtelen lenne minden képkockát tárolni, elegendő lenne csak a hasonló képek közül az elsőt, és aztán a többinek csak az ettől való eltérését. A *különbség* kódolása a gyakorlatban úgy történik meg, hogy az egyes makroblokkokhoz megkeresik az előző és esetleg a következő képeken az adott makroblokkra leginkább hasonlító részletet, megjegyzik, hogy a vizsgált makroblokk mennyire van eltolva azokhoz képest, és hogy a makroblokk mennyiben tér el az azokból becsült értéktől. (Az azokból becsült érték lehet egyszerűen a megelőző képen a hasonló részlet értéke, de lehet több környező képből képzett átlag is.) Minden – nem vágás utáni – kép egy makroblokkjának kódolásához tehát hat különbségekből álló kis méretű blokk és két mozgásvektor tartozik. A blokkokat aztán lehet a JPEG-hez hasonló módon tömöríteni.

Ha így kódolnánk, nem lenne lehetőség a filmbe akárhol bekapcsolódni, mindig csak a vágások elején, mert az egész képsorozat dekódolásához kellene a legelső kép. Hogy mégis bárhol el lehessen kezdeni nézni a mozgóképet, bizonyos lépésközönként a teljes képkockát meghagyják, és azt tömörítik a JPEG-hez hasonló módon. Az ilyen képkockákat jelölik I-vel, és a két I típusú filmkocka közötti képek alkotják a *képcsoport*okat. Az egyes képcsoportok további részekre bomlanak. Minden harmadik elemet csak az azt hárommal megelőző képből származtatják. Ezek a P típusú képek, és csak egyetlen, (vagy P, vagy I típusú) képből eredeztethetők. A többi képet az időben eggyel előttük és az eggyel utánuk lévő P vagy I filmkockából származtatják, ezek a B képek. A B típust nem használják fel másik B-kép kódolásakor, mert úgy előfordulhatna, hogy oda-vissza utal a két kocka egymásra, és nem lehetne dekódolni. Az alábbi ábra szemlélteti az egyes képkockák egymásból származtatását. A duplán bekeretezett kockák az I

típusúak, a vastag vonallal rajzolt üresek a P típusúak, a vékony vonalasok B-képek. A nyilak az előállított képkockákba mutatnak a becsléshez felhasznált kockákból.



Nem kötött, hogy két I-kép között hány P-típusú helyezkedik el, így lehet a vágásokat s a hosszabb állóképeket gazdaságosan tömöríteni. Gyakran használják az ábrán látható két P-s sémát. Látható, hogy egyetlen kép sem hivatkozik a két I által meghatározott zárt intervallumon kívülre, így igen minimális számolással bárhová be lehet csatlakozni, bárhonnan el lehet kezdeni nézni a filmet. Ezt szolgálja az is, hogy a B képeket a kódolt adatfolyamban hátrébb tolják, közvetlen az előállításukhoz szükséges I vagy P típusú kocka mögé úgy, hogy ha csak a B-k sorrendjét nézzük, ne legyen keveredés. Ha így rendezzük a filmkockákat, akkor mindig csak egy irányba (visszafelé) kell keresni a előzményeket, így az adatfolyam könnyebben kezelhető. Kiolvasáskor természetesen a dekódoló tudja, hogy az adott B képet hamarabb kell levetíteni.

6. Út a csatornán át – a digitális modulációról és a döntésről

6.1. A digitális modulációról

Eddig igen absztrakt szimbólumok hordozták az információt, de nyilván kevés csatorna képes például 0-s és 1-es számokat, vagy a, b, c betűket továbbítani. A csatornák általában valamilyen típusú jelek átvitelére alkalmasak, így az absztrakt szimbólumoknak különböző paraméterekkel jellemzett elektromos áramot, mágnesezettséget, elektromágneses hullámot, vagy például kristályszerkezeti állapotokat kell megfeleltetni. Ha például egy időintervallumban folyt áram a vezetőben, azt vehetjük 1-es bitnek, ha nem – illetve csak egy referencia értéknél kisebb folyt –, 0-nak. Hasonlóképpen, egy mágneses lemez vagy szalag adott pontján a mágnesezettség két iránya, míg az optikai lemezek (CD, DVD) pontjaiban az anyag kristályos vagy amorf szerkezete, és így két különböző fénytörési jellemzője feleltethető meg a két bitnek.

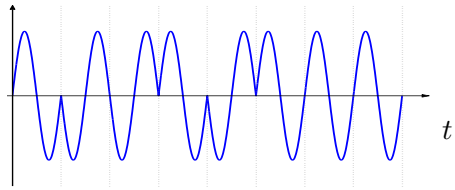
Az elektromágneses hullámok sokkal több lehetőséget tartogatnak annál, hogy van hullám, vagy nincs. A fény- és rádióhullámok három fő jellemzővel írhatók le: a **frekvenciá**jukkal, az **amplitúdó**jukkal és a **fázisszög**ükkel. Mindhárom hordozhat mind digitális, mind pedig analóg információt, melyek közül az előbbivel fogunk foglalkozni. A modulált jelalakok leírhatók egy adott ω_0 körfrekvenciájú, A_0 amplitúdójú, φ_0 fázisszögű harmonikus jel ($A_0 \sin(\omega_0 t + \varphi)$), az úgynevezett **vivőjel** különböző módosításaiként. A digitális moduláció alapötlete az, hogy az időt diszkrét, nem átfedő, T hosszúságú intervallumokra bontjuk, a szimbólumainknak megfeleltetünk egy-egy rendelkezésre álló T hosszúságú jelszakaszt, és az i-edik időintervallumban az i-edik szimbólumhoz hozzárendelt jelalakot adjuk le. Az idő intervallumokra való felbontásához szükség van egy órajelre, s az órajel frekvenciáját a demodulátorban is ismerni kell. Ezen kívül a demodulátor szinkronban működik a modulátorral, ismeri a rendelkezésre álló jelalakokat és a vivőjelet, ezek alapján dönt arról, hogy melyikhez hasonlít legjobban a vett jel, melyik szimbólumnak megfelelő jelalakot adhatták le.

6.1.1. Impulzus amplitúdómoduláció – PAM

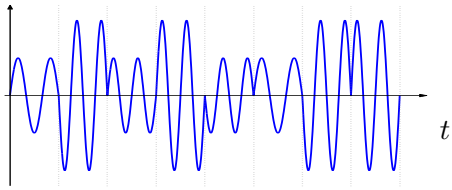
Az amplitúdómoduláció során az $A_0 \sin(\omega_0 t + \varphi)$ vivőjel amplitúdóját változtatják meg a

$$y_i(t) = a_i \cdot A_0 \sin(\omega_0 t + \varphi)$$

formula szerint, és az i-edik bithez $y_i(t)$-nek a $t \in [0,T)$ intervallumbeli szakaszát feleltetik meg. A bináris amplitúdómodulációnál lehet például $a_0 = 0$ és $a_1 = 1$, vagy $a_0 = -1$ és $a_1 = 1$, a négyes AM esetén a négy különböző érték $a_0 = -2$, $a_1 = -1$, $a_2 = 1$ és $a_3 = 2$. Illusztrációként álljon itt az 10010111 bitsorozatból második típusú modulációval létrejött jel (ha a T éppen a vivőjel periódusideje):



és a 20231203 szimbólumsorozatból a fent leírt módon négyes amplitúdómoduláció segítségével létrehozott jel:



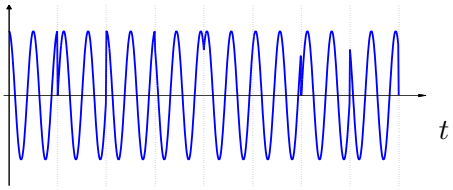
Az ábrákon két függőleges szaggatott vonal határolja az egyes időintervallumokat. Az amlitúdófaktorok lehetnek komplex számok is, ekkor a jelnek nem csak az amplitúdója, hanem a fázisa is módosul.

6.1.2. Fázismoduláció – PSK

Az fázistolásos moduláció a harmonikus $A_0 \sin(\omega_0 t + \varphi)$ vivőjel fázisát változtatja meg, így a modulált jel $[iT,(i+1)T)$ szakasza

$$y_i(t) = A_0 \sin(\omega_0 t + \varphi + \psi_i)$$

lesz aszerint, hogy az i-edik pozícióban melyik szimbólum volt. Tipikusan a $360°$-ot 2^n egyenlő részre osztják, ezeknek felelnek meg a ψ_i-k. A bináris fázismodulációnál (BPSK) $\psi_0 = 0°$, $\psi_1 = 180°$, a kvadratúra fázismodulációnál (QPSK, vagy 4PSK) $\psi_0 = 0°$, $\psi_1 = 90°$, $\psi_2 = 180°$ és $\psi_3 = 270°$, a 8PSK-nál pedig $\psi_0 = 0°$, $\psi_1 = 45°$, $\psi_2 = 90°$, $\psi_3 = 135°$, $\psi_4 = 180°$, $\psi_5 = 225°$, $\psi_6 = 270°$ és $\psi_7 = 315°$. Illusztrációként a 20231103 szimbólumsorozatból QPSK-val létrehozott jel:

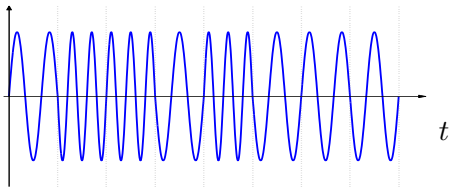


6.1.3. Frekvenciamoduláció – FSK

Ezek után könnyű rájönni, hogy az i-edik szimbólumhoz rendelt, frekvenciamodulált jelalak

$$y_i(t) = A_0 \sin((\omega_0 + \alpha_i \cdot \omega_c)t + \varphi)$$

lesz. A bináris frekvenciamodulációra például $\alpha_0 = 1$, $\alpha_1 = -1$. Ha $\omega_c = \omega_0/4$, akkor az 10010111 bitsorozatból a



jelet kapjuk.

6.2. A csatorna megosztásáról

A hírközlési csatornák véges sávszélességűek, így mivel igen sok felhasználó szeretne egyszerre egy csatornát használni, azt valamilyen módszerrel meg kell osztani. Azokat az eljárásokat, amelyek lehetővé teszik azt, hogy a jelátvitel csoportosan, a csatorna megosztásával történjen, **nyalábolási** vagy **multiplexelési technikák**nak hívjuk.

Egy közös csatornát frekvenciában és időben is meg lehet osztani. A **frekvenciaosztású multiplexelés**, vagy FDM (Frequency Division Multiplexing) a csatorna frekvenciasávját több alsávra bontja. Olyan modulációkat alkalmaz, amelyek az átvinni kívánt több adatfolyamból egy-egy szimbólumsorozatot csak egy alsávba visznek. Az egyes vevők tudják, hogy az őket érdeklő információ milyen frekvenciatartományban található,

és annak megfelelő sávszűrőre vezetik először a vett jelet. Ha több felhasználó egy csatornát használhat, a csatorna frekvenciaosztással többszörösen hozzáférhető, vagy FDMA (Frequency Division Multiple Access).

Az **időosztással többszörösen hozzáférhető** TDMA (Time Division Multiple Access) csatornákon működő rendszerek, egy-egy felhasználónak csak bizonyos időintervallumokban engedik az információ adását és vételét.

A fenti két módszer lényegében azt eredményezi, hogy hiába találkoznak és adódnak össze a különböző jelek a csatornán belül, a vevők nem fogják azt érzékelni, mert a sávszűrőjük elnyomja az idegen jeleket, illetve időosztásos multiplexelésnél az ő időintervallumaikban mások nem adhatnak. Lehetséges azonban az is, hogy az egyes felhasználópárok a 0 és 1 bitjeiket csak rájuk jellemző és általuk ismert, hosszabb (16, 32 vagy 64 bites) sorozat és annak ellentettje formájában közlik egymással. Úgy választják meg ezeket a sorozatokat, hogy azok a *többi felhasználó sorozatával összeszorozva nullát adjanak*, míg a sajátjukkal 1-et (illetve az ellentett sorozat esetén -1-et). Ez a csatornafelosztás a **közvetlen sorozatú kódosztásos többszörös hozzáférés**, vagy angol rövidítéssel a DSSSMA (Direct Sequence Spread Spectrum Multiple Access). *Kódosztás*nak azért hívják, mert az egyes felhasználók kódjaik – a csak rá jellemző sorozataik – segítségével úgy osztják fel egymás között a csatornát, hogy azt frekvenciában és időben korlátozatlanul igénybe vehetik. Elképzelhető a kiosztott kód másféle felhasználása is. A következő két bekezdés ezeket a lehetőségeket foglalja össze.

Igen elterjedt a **frekvenciaugratásos spektrumszórás** (FH – Frequency Hopping) is. Ezesetben a csatorna több rész-frekvenciasávra van bontva, és az adó az előre elküldött sorozat elemeinek megfelelően T időnként másik alsávban ad. Ha például az adó és a vevő a $241528\ldots$ sorozatot kapta, hogy az adás kezdetétől számított T ideig a második frekvenciasávban kell adnia, illetve vennie, T-től $2T$-ig a negyedikben, és így tovább. A jó kódoknál ritkán fordul az elő, hogy két felhasználó is ad egy frekvenciasávban, azaz hogy **ütközés** következik be. Attól függően nevezik a rendszert *lassú vagy gyors frekvenciaugratásúnak*, hogy a T idő hosszabb-e annál, ameddig egy bitnyi információ áthalad a csatornán vagy rövidebb. Lassú frekvenciaugratást használ például a GSM.

Az előre kiosztott kódok felhasználásának harmadik módja az, ha a frekvenciaugratásnál felvázolt eljárást az időintervallum további osztásával kapott idősávok közötti ugratásra használják.

Nagy előnye a *kódosztásos nyalábolás*nak (CDMA – Code Division Mul-

tiple Access), hogy a rendelkezésre álló részsávoknál nagyobb számú felhasználót képes kezelni (persze nem akar például mindenki egyszerre telefonálni, így sok a passzív tag). Nem szükséges minden egyes berendezést egy központhoz szinkronizálni, ami megkönnyíti a kezelhetőséget. Minden felhasználó tetszőleges időpontban kezdhet el kommunikálni.

Ahhoz, hogy ezt lehetővé tegyék, a kiosztott kódoknak – akár közvetlen sorozatú, akár ugratásos CDMA-ról beszélünk – igen speciálisaknak kell lenniük.

- Ahhoz, hogy a sok felhasználót kezelni tudják, a különböző felhasználóknak kiosztott sorozatoknak nagyon különbözőeknek kell lenniük, hogy a csatornában való ütközések számát minimalizálják. Mivel többnyire nem egyidőben kezdik el használni a csatornát a különböző adó-vevő párosok, szükséges az is, hogy a kódok időbeli eltoltja kellően eltérő legyen. Két, időben egymáshoz képest tetszőlegesen eltolt függvény, illetve sorozat hasonlóságát **keresztkorreláció**val mérik. A jó kódoknak tehát kicsi a keresztkorrelációjuk.
- Ahhoz, hogy a központi szinkronizáció elkerülhető legyen az egyes kódoknak és az időben eltoltjuknak is kellő mértékben különbözniük kell. Egy függvény vagy sorozat önmagának időbeli eltoltjához való hasonlóságát a függvény, illetve sorozat **autokorreláció**ja jellemzi. A jó kódoknak tehát az autokorrelációja is kicsi.

6.3. A döntésről

Az információátviteli csatornák többnyire bizonyos mértékben torzítják a rajtuk áthaladó jeleket. Ezt a torzítást úgy szokták modellezni, hogy a bemeneti jelhez valamiféle zajt adnak hozzá, s így kapják meg a csatorna kimenetén a vett jelet.

Ha visszagondolunk a digitális modulációk demodulátoraira, azok képesek voltak a vett jelet megfelelőképpen szakaszokra bontani, s a szakaszokról *eldönteni*, hogy azok milyen szimbólumhoz tartozó jelalakok lehettek a csatorna bemenetén. Hogy a rendszerbe ne vigyünk be újabb bizonytalanságot, egy bizonyos vett jelalakhoz mindig ugyanazt a szimbólumot kell választani.

Tegyük fel, hogy a csatorna bemenetén a $C_i \in C$ elemek fordulhatnak elő, a kimenetén pedig az X halmaz elemei. Ha a megfigyelt X_j jelalakból szeretnénk arra következtetni, mik voltak a megfelelő C_i-k, a folyamatot le tudjuk írni egy $g : X \mapsto C$ függvénnyel. A $g(X)$ függvényt **döntés**nek nevezzük. (Ha a g értékkészlete, azaz a C halmaz nem véges sok elemből

áll, akkor a függvényt nem döntésnek, hanem becslésnek hívjuk.) Az adott döntés jóságát általában valamilyen költségfüggvénnyel szokták jellemezni. A döntés során aztán arra törekednek, hogy e költségfüggvény várható értéke minimális legyen. A következő két szakasz két különböző lehetséges költségfüggvényről – az a posteriori valószínűségeknek, illetve a likelihoodnak a reciprokáról – fog szólni.

Az az esemény, ha a C_i mellett döntünk, az *i-edik hipotézis*. Az X halmaz azon $D^{(i)}$ részhalmaza, amelynek vétele esetén mindig az i-edik hipotézist tesszük (azaz mindig a C_i elem mellett döntünk), a g függvény i-edik **döntési tartomány**a. Ha ismerjük a g döntés összes döntési tartományát, azzal tulajdonképpen megadjuk az egész g függvényt. Ha tudjuk, hogy az adott C_i elemek $p(C_i)$ valószínűséggel fordulnak elő, a $p(C_1)$, $p(C_2)$, . . . ,$p(C_m)$ valószínűségeket *a priori valószínűségek*nek nevezzük. A $p(C_i|X_j)$ valószínűségekre pedig, mint *a posteriori valószínűségek*re szoktak hivatkozni.

6.3.1. Bayes-döntés

Válasszuk meg úgy az i-edik döntési tartományt, hogy minden $x \in D_B^{(i)}$-re $p(C_i|x) \leq p(C_j|x)$ legyen, minden j-re, ha $i \neq j$. Ez azt jelenti, hogy $D_B^{(i)}$-be csak olyan elemeket válogattunk, amelyeknek a vétele esetén az i-edik C-beli elem adása volt a legvalószínűbb. (Ha esetleg több „legvalószínűbb" C_i is van, akkor valahogy választunk közülük, például a legkisebb indexűt. Nem engedünk meg átfedést a döntési tartományok között.) A

$$g_B(x) = C_i, \qquad \text{ha } x \in D_B^{(i)} \tag{6.1}$$

döntést *Bayes-döntés*nek, vagy maximum a posteriori döntésnek nevezzük.

A Bayes-döntés optimális, neki a legkisebb a hibavalószínűsége, de csak akkor alkalmazható, hogy ha ismerjük az a posteriori valószínűségeket. Ha valamilyen okból azok ismeretlenek, akkor más döntési stratégiát kell bevezetnünk. (Az a ritkább, ha ismertek az a posteriori valószínűségek.)

6.3.2. Maximum likelihood döntés

Egy a Bayes-döntés helyett alkalmazható megoldás a maximum likelihood döntés. Elképzelhető, hogy a $p(C_i|X_j)$ valószínűségeket ugyan nem ismerjük, de azt tudjuk, hogy ha C_i volt a csatorna bemenetén, akkor milyen valószínűséggel lesz a kimeneten X_j, azaz, mik a $p(X_j|C_i)$ valószínűségek. A $p(X_j|C_i)$ mennyiséget *likelihood*nak nevezünk a *maximum likelihood döntés* során pedig a $D_{ML}^{(i)}$ döntési tartományokba azon $x \in X$-ek tartoznak,

melyekre a $p(x|C_i)$ maximális. Ez formulákkal a következőképpen írható le:

$$g_B(x) = C_i, \qquad \text{ha } x \in D_{ML}^{(i)}$$

és

$$x \in D_{ML}^{(i)}, \qquad \text{ha } p\left(x|C_i\right) = \max_k p\left(x|C_k\right). \tag{6.2}$$

Ha az a posteriori valószínűségek egyenlők, akkor a maximum likelihood döntés megegyezik a Bayes-döntéssel.

Fontos látni a két döntési folyamat közötti különbséget. A Bayes-döntésnél például egy X_i szimbólum vétele esetén amellett a leadott szimbólum mellett döntünk, amelynek a legnagyobb a valószínűsége, feltéve, hogy a megadott szimbólumot vettük. A maximum likelihood döntésnél viszont az összes leadható jel közül amellett aszimbólum mellett döntünk, amelynek az adása esetén a legnagyobb a valószínűsége annak, hogy az X_i-t vesszük.

Például egy csatornát általában a $p(X_j|C_i)$ feltételes valószínűséggel szoktak megadni, így a maximum likelihood döntés meghozása igen egyszerű. Ahhoz viszont, hogy Bayes-döntést tudjunk hozni, ismerni kell a bemeneti és a kimeneti szimbólumok előfordulási statisztikáját, hiszen ha felhasználjuk $p(X_j|C_i)$ és $p(C_i|X_j)$ (2.2) definíciós összefüggéseit,

$$p(C_i|X_j) = \frac{p(X_j|C_i) \cdot p(C_i)}{p(X_j)}$$

Ez azt jelenti, hogy nem csak a csatornáról, hanem a forrásról: a csatorna használatáról is adatokkal kell rendelkeznünk, és ez nem túl gyakori eset.

7. Csatornakódolás

A Shannon-féle hírközlési modell szerint csatorna minden olyan közeg vagy eszköz, amely képes arra, hogy információt szállítson két különböző térbeli és időbeli pont: a forrás és a nyelő között. A csatornán való áthaladás során azonban az üzenet módosulhat, átalakulhatnak bizonyos jelek másokká, vagy akár törlődhetnek is. Az ilyen csatornák a zajos csatornák. Zajmentes csatornák csak igen ritkán fordulnak elő, ilyenek például (egy ideig) a nyomtatott könyvek, ha nem károsodnak – nem áznak, szakadnak el, vagy égnek meg, nem esnek bele egy nagy adag festékbe. Az elektronikus csatornák többnyire zajosak, legyen szó akár a levegőről, s a benne terjedő elektromágneses hullámokról, akár egy koax kábelről, akár pedig a számítógép merevlemezéről. A csatorna zajosságának elfogadott mértéke a jel-zaj arány, avagy az SNR (signal to noise ratio), amelyet a következőképpen szoktak definiálni: Legyen a jel átlagos teljesítménye S, a zajé N, ekkor

$$\mathrm{SNR} = 20 \lg \left(\frac{S}{N} \right).$$

A mértékegység decibel.

A decibelt általában az erősítés jellemzésére szokták alkalmazni, az erősítési arány tízes alapú logaritmusát – pontosabban annak húszszorosát – lehet decibelben mérni. A hangerő esetében a jellemezni kívánt hang intenzitásának egy adott, egyezményben fixált hangintenzitáshoz (20 mPa hangnyomáshoz) viszonyított arányát fejezik ki decibelben.

Fel fogjuk tenni a következőkben, hogy a csatornánk *diszkrét jeleket visz át*, továbbá nem nyel el és nem bocsát ki új jeleket: egy bemeneti szimbólum hatására egy szimbólum jelenik meg a kimeneten; persze nem feltétlenül mindig ugyanaz a szimbólum. Az ilyen csatornákat **szinkron csatornák**nak nevezik.

7.1. A csatorna jellemzése, csatornamátrix

Egy csatornát úgy tudunk megadni, ha megadjuk a $C = \{c_1, c_2, \ldots, c_r\}$ bemeneti és a $X = \{x_1, x_2, \ldots, x_s\}$ kimenti jelkészletét, valamint a $p(x_j|c_i)$ feltételes valószínűséget, azaz minden bemeneti jelre megadjuk, milyen lehetséges kimeneti jeleket vonhat maga után, s mekkora valószínűséggel.

Akkor **memóriamentes** a diszkrét csatornánk, ha a csatorna kimenetén megjelenő jelek mindig csak az aktuális bementtől függenek, azaz

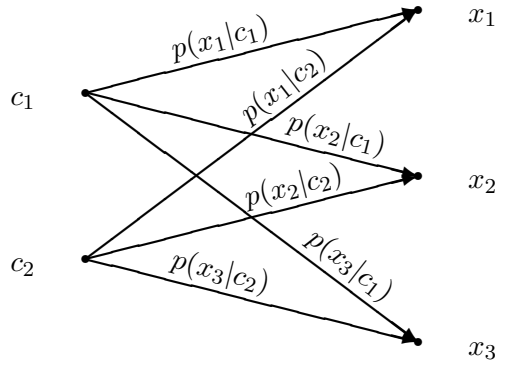
az egymást követő szimbólumoknak a csatornán való áthaladása független esemény. Adjunk le a csatornánkon egymás után n darab szimbólumot, $c^{(1)},c^{(2)},\ldots,c^{(n)}$-et, ez eredményezze a kimeneten az $x^{(1)},x^{(2)},\ldots,x^{(n)}$ jelsorozatot. Ennek az eseménynek a valószínűsége $p(x^{(1)},x^{(2)},\ldots,x^{(n)}|c^{(1)},c^{(2)},\ldots,c^{(n)})$. A memóriamentesség követelménye megfogalmazható, mint

$$p(x^{(1)},x^{(2)},\ldots,x^{(n)}|c^{(1)},c^{(2)},\ldots,c^{(n)}) = \prod_{i=1}^{n} p(x^{(i)}|c^{(i)}). \tag{7.1}$$

A $p(x_j|c_i)$ valószínűségeket **mátrix**ba szokták rendezni a következőképpen

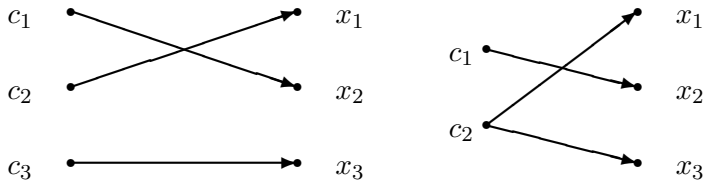
$$P = \begin{pmatrix} p(x_1|c_1) & p(x_2|c_1) & \ldots & p(x_s|c_1) \\ p(x_1|c_2) & p(x_2|c_2) & \ldots & p(x_s|c_2) \\ \vdots & \vdots & \ddots & \vdots \\ p(x_1|c_r) & p(x_2|c_r) & \ldots & p(x_s|c_r) \end{pmatrix}. \tag{7.2}$$

Szokás ezenkívül a csatornát bal oldalon a bemeneti jelekkel, jobb oldalon a kimeneti jelekkel, közöttük pedig nyilakkal ábrázolni. A nyilakon fel lehet tüntetni a megfelelő feltételes valószínűségeket:

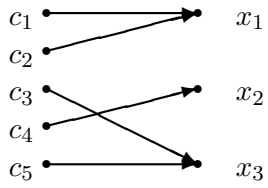


Az ilyen ábrázolás a csatorna **gráf**ja.

Egy csatorna **zajmentes**, ha a mátrixának minden oszlopában pontosan egy nem nulla elem van. Ez azt jelenti, hogy egy kimenet mindig csak egy bizonyos bemenetből állhat elő. Nem következik belőle viszont az, hogy egy bemeneti jel csak egy kimenetet generálhat. Például mindkét következő csatorna zajmentes:

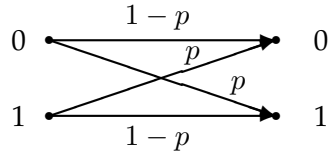


*Determinisztikus*nak nevezzük azokat a csatornákat, amelyekre egy adott bemeneti jel mindig ugyanazt a kimenetet eredményezi, azaz a csatornamátrixának egy sorában csak egy nem nulla elem van, és ez az elem 1. Az előző ábra első csatornája determinisztikus és zajmentes is, míg az itt következő csatorna csak determinisztikus:



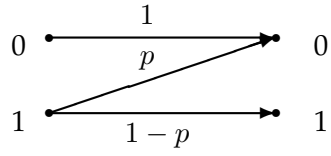
Példaképpen álljon itt néhány, gyakorlati szempontból is jelentős bináris csatorna. A *bináris szimmetrikus csatorná*nak (BSC – Binary Symmetric Channel) mind a bemeneti, mind pedig a kimeneti szimbólumkészlete 0-ból és 1-ből áll, és mind a 0, mind pedig az 1 bemenet esetén p valószínűséggel lesz rossz a kimenet. A csatornamátrix és a csatornagráf a következő:

$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$



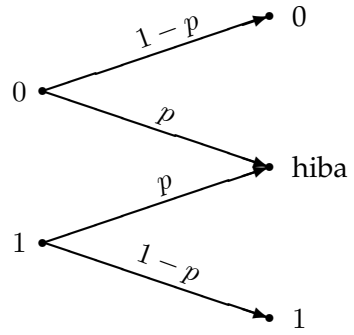
A *bináris Z-csatorna* annyiban különbözik a BSC-től, hogy csak az 1-es bit esetén hibázik, a 0-t minden esetben 0-ba viszi át. Csatornamátrix és -gráf az alábbi:

$$\begin{pmatrix} 1 & 0 \\ p & 1-p \end{pmatrix}$$



A *bináris törléses csatorna* bemeneti szimbólumkészlete is 0-ból és 1-ből áll, de a kimeneten a 0-n és 1-en kívül egy hibaszimbólumot is tartalmaz. Mind a 0, mind pedig az 1 bemeneti jel esetén p valószínűséggel a hibaszimbólumot tudjuk venni. A csatornamátrixa és a gráfja így néz ki:

$$\begin{pmatrix} 1-p & p & 0 \\ 0 & p & 1-p \end{pmatrix}$$



7.2. A csatorna vesztesége és az átvitt információ

A csatornán folyó információátvitel során a C_i forráselemek $p(C_i) = p_i$ előfordulási valószínűsége azt írja le, hogy hogyan használjuk a csatornát, a $p(X_j|C_i)$ valószínűségek pedig a csatornát magát jellemzik. Hasonlóképpen, az entrópia a $H(C) = -\sum_{i=1}^{r} p_i \log_2 p_i$ alakjában a csatorna forrását írja le, míg a $H(X|C) = -\sum_{i=1}^{r}\sum_{j=1}^{s} p(X_j \cdot C_i) \log_2 p(X_j|C_i)$ feltételes entrópia a csatornát.

A csatorna kimenetén csak az $X_j \in X$ szimbólumokat lehet megfigyelni. Miután megfigyeltük a kimenetet, marad még bizonytalanság arra nézve, hogy a bemeneti oldalon milyen szimbólum szerepelt, azaz a közlés során információt veszítettünk. Ennek a bizonytalanságnak a várható értéke

$$H(C|X_j) = \sum_{i=1}^{r} p(C_i|X_j) \log_2 p(C_i|X_j).$$

A csatorna **vesztesége** ennek az elvesztett információnak a várható értéke, a

$$H(C|X) = \sum_{j=1}^{s} p(X_j) \cdot H(C|X_j) = -\sum_{j=1}^{s}\sum_{i=1}^{r} p(C_i \cdot X_j) \log_2 p(C_i|X_j)$$

entrópia. Magától értetődik, hogy a veszteség nem lehet nagyobb, mint a teljes csatornára adott információ várható értéke, a forrás $H(C)$ entrópiája.

Zajmentes csatorna vesztesége 0, mivel $p(C_i|X_j)$ vagy 1, vagy pedig 0, attól függően, hogy összetartoznak-e a C_i és X_j szimbólumok vagy nem. Ennek következtében vagy a $\log_2 p$ vagy pedig p és a teljes $p \log_2 p$ lesz 0 az összeg minden tagjában.

Teljesen zajos csatorna vesztesége a teljes $H(C)$, hiszen ekkor a leadott és a vett jelek egymástól függetlenek, így $p(C_i \cdot X_j) = p(C_i)p(X_j)$ és $p(C_i|X_j) = \frac{p(C_i \cdot X_j)}{p(X_j)} = p(C_i)$. Ennek következményeként

$$H(C|X) = -\left(\sum_{j=1}^{s} p(X_j)\right) \sum_{i=1}^{r} p(C_i) \log_2 p(C_i) = 1 \cdot H(C).$$

Az **átvitt információ** a csatornára adott információ várhatóértékének és a veszteségnek a különbsége, azaz a

$$H(C) - H(C|X) = I(C \cdot X) \tag{7.3}$$

kölcsönös információ. A csatornát jól le lehet írni a fenti kölcsönös információval, ez adja ugyanis meg azt az átlagos információt, amelyet egy X_j vételekor nyerünk az őt előidéző C_i-ről. Az átvitt információ felírható a feltételes és együttes valószínűségek felhasználásával a következőképpen:

$$\begin{aligned} I(C \cdot X) &= -\sum_{i=1}^{r} p(C_i) \log_2 p(C_i) + \\ &\quad + \sum_{i=1}^{r} \sum_{j=1}^{s} p(C_i \cdot X_j) \log_2 p(C_i|X_j) = \\ &= -\sum_{i=1}^{r} \sum_{j=1}^{s} p(C_i \cdot X_j) \log_2 p(C_i) + \\ &\quad + \sum_{i=1}^{r} \sum_{j=1}^{s} p(C_i \cdot X_j) \log_2 p(C_i|X_j), \end{aligned} \tag{7.4}$$

mivel $p(C_i) = \sum_j p(C_i \cdot X_j)$. Ha kiemeljük páronként a $p(C_i \cdot X_j)$ feltételes valószínűségeket és átírjuk a logaritmusok különbségét a hányados logaritmusára, a

$$\begin{aligned} I(C \cdot X) &= \sum_{i=1}^{r} \sum_{j=1}^{s} p(C_i \cdot X_j) \log_2 \frac{p(C_i|X_j)}{p(C_i)} = \\ &= \sum_{i=1}^{r} \sum_{j=1}^{s} p(C_i \cdot X_j) \log_2 \frac{p(C_i \cdot X_j)}{p(C_i)p(X_j)} \end{aligned} \tag{7.5}$$

kifejezést kapjuk.

A **csatorna kapacitása** a rajta maximálisan átvihető információ, a

$$\mathcal{C} = \max I(C \cdot X) \tag{7.6}$$

Felhasználva a (7.5) egyenletet,

$$\mathcal{C} = \max \left(\sum_{i=1}^{r} \sum_{j=1}^{s} p(C_i \cdot X_j) \log_2 \frac{p(C_i \cdot X_j)}{p(C_i)p(X_j)} \right). \tag{7.7}$$

Shannon eredetileg a következő határértéket nevezte csatornakapacitásnak:

$$C = \lim_{T \to \infty} \frac{\log N(T)}{T},$$

ahol $N(T)$ az olyan T ideig tartó jeleknek a száma, amelyek létrejöhetnek. A két definíció lényegében ekvivalens, ami belátható úgy, hogy vesszük a C_1,C_2,...,C_r szimbólumokból – a csatorna bemeneti szimbólumkészletéből – alkotott olyan üzeneteket, amelyek T ideig tartanak, megszámoljuk őket, és vizsgáljuk a darabszámuknak $T \to \infty$ viselkedését. Tudni kell azt, hogy az egyes szimbólumnak a csatornán való áthaladása rendre T_1,T_2,...,T_r ideig tart.

7.3. Csatornakódok jellemzése

A csatornakódolási eljárások célja az, hogy az információátvitel hibáját minimálisra csökkentsük, illetve, hogy a zajos csatorna által okozott hibákat korrigálni tudjuk. A Shannon-féle csatornakódolási tétel óta tudták, hogy lehetséges zajos csatornán is nagy biztonsággal, hatékonyan üzenetet továbbítani. Mivel azonban a tétel nem konstruktív – azaz nem ad meg konkrét módszert arra, hogyan kódoljuk az üzenetet –, egy jó ideig nem tudták kihasználni a hírközlési csatornáknak a tétel által leírt korlátait. Eleinte nem is volt szükség a hibajavító kódok polgári alkalmazására, legfeljebb egy-egy berendezés irányítási rendszerében. Az első felhasználók a katonai hírközlő rendszerek voltak, és az eredmények nem mindig jutottak nyilvánosságra. A mikroprocesszorok és a személyi számítógépek kifejlesztése azonban maga után vonta a hibajavító kódolások alkalmazásának széles körű elterjedését.

Legyen a csatorna bemeneti ábécéje $C = \{C_1,C_2,\ldots,C_r\}$ a kimeneti ábécéje $X = \{X_1,X_2,\ldots,X_s\}$. A csatorna bemenetére adjunk egy n

hosszúságú $\mathbf{c} = (c^{(1)},c^{(2)},\ldots,c^{(n)})$ kódszót. Minden $c^{(i)} \in C$. A kimeneten ekkor egy szintén n elemű jelsorozat fog megjelenni, $\mathbf{x} = (x^{(1)},x^{(2)},\ldots,x^{(n)})$, $x^{(i)} \in X$. Ekkor el kell dönteni minden egyes $x^{(i)}$ szimbólumról, hogy miből származik, azaz hogy a bemeneti ábécé melyik C_j elemét adhatták az i-edik helyen. Ezt valamilyen $g : X \mapsto C$ döntési algoritmussal eldöntjük, s így kapunk egy $\mathbf{v} = (v^{(1)},v^{(2)},\ldots,v^{(n)})$ szimbólumsorozatot. Minden i-re $v^{(i)} \in C$ bemeneti ábécének, de nem feltétlenül igaz, hogy $v^{(i)} = c^{(i)}$. Az n hosszúságú, C-beli elemekből felépülő szimbólumsorozatok halmazát jelöljük C^n-nel, így $\mathbf{c} \in C^n$ és $\mathbf{v} \in C^n$.

Matematikai kitérő - Vektorterekről. A C^n halmaz tulajdonképpen a szó matematikai értelmében vett vektortér, így logikus az elemeit a vektorokra jellemző vastag kisbetűkkel jelölni. A **vektortér** matematikai definíciója a következő: Vegyünk egy V halmazt, értelmezzünk az elemein egy műveletet, a *számmal való szorzás*t, és az elemek között egy másik műveletet, az *összeadás*t. A $\mathbf{v} \in V$ vektor λ számmal való szorzását $\lambda \cdot \mathbf{v}$-vel jelölhetjük, egy másik, $\mathbf{w} \in V$ vektorral vett összegét pedig $\mathbf{v} + \mathbf{w}$-vel. Alapvető, hogy sem a számmal való szorzás, sem pedig két vektor összeadása ne vezessen ki a V térből, azaz $\lambda \cdot \mathbf{v} \in V$ és $\mathbf{v}+\mathbf{w} \in V$. Ezenkívül a számmal való szorzás rendelkezik a következő tulajdonságokkal:

1. $1 \cdot \mathbf{v} = \mathbf{v}$
2. $\lambda \cdot (\kappa \cdot \mathbf{v}) = \lambda\kappa \cdot (\mathbf{v})$ (asszociativitás, azaz csoportosíthatóság)
3. $(\lambda+\kappa)\cdot\mathbf{v} = \lambda\cdot\mathbf{v}+\kappa\cdot\mathbf{v}$ (disztributivitás, azaz kb. szétterjeszthetőség)

A vektorok összeadása pedig az alábbi feltételeket teljesíti:

1. $\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$ (kommutativitás, azaz felcserélhetőség)
2. $\mathbf{v} + (\mathbf{w} + \mathbf{u}) = (\mathbf{v} + \mathbf{w}) + \mathbf{u}$ (asszociativitás)
3. létezik olyan $\mathbf{0}$ nullelem, amelyre minden $\mathbf{v} \in V$ vektorra $\mathbf{v} + \mathbf{0} = \mathbf{v}$
4. minden $\mathbf{v} \in V$-nek létezik ellentettje, $-\mathbf{v} \in V$, mellyel $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$

A vektortereket szokás *lineáris terek*nek is nevezni. Vektortér például a két vagy háromdimenziós (euklideszi) tér vektoraiból (irányított szakaszaiból) álló tér, de a fogalom ennél sokkal általánosabb. A legfeljebb n-edfokú polinomok tere is vektortér.

Ahhoz, hogy a vektorokat le tudjuk írni, be kell vezetni néhány fogalmat. *Lineárisan független*ek a $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_N$ vektorok, ha a

$$\lambda_1 \cdot \mathbf{v}_1 + \lambda_2 \cdot \mathbf{v}_2 + \ldots + \lambda_N \cdot \mathbf{v}_N = 0 \tag{7.8}$$

egyenlőség akkor és csak akkor teljesül, ha minden i-re $\lambda_i = 0$. Ha van olyan N darab λ_i szám, amelyek közül néhány nem nulla, és teljesítik a (7.8) egyenletet, akkor a vektoraink összefüggők. A $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_N$ vektorok **lineáris kombinációján** a $\lambda_1 \cdot \mathbf{v}_1 + \lambda_2 \cdot \mathbf{v}_2 + \ldots + \lambda_N \cdot \mathbf{v}_N$ összeget értjük.

Egy V vektortér *N-dimenziós*, ha található N darab független vektora, azonban $N+1$ darab független vektora már nincs. Ekkor a vektortér minden egyes eleme kifejezhető, mint az iménti N elem lineáris kombinációja. Az N dimenziós vektortér N független vektora tehát teljesen meghatározza a teret, ezért az ilyen vektorokat alapvektornak, vagy **bázisvektor**nak nevezzük, a bázisvektorok összességét pedig bázisrendszernek, vagy egyszerűen *bázis*nak. Ekkor tehát minden $\mathbf{v}$ felírható, mint $\alpha_1 \cdot \mathbf{e}_1 + \alpha_2 \cdot \mathbf{e}_2 + \ldots + \alpha_N \cdot \mathbf{e}_N$, ha $\mathbf{e}_i$-k a bázisvektorok. Ha $\mathbf{e}_i$-k adottak, akkor ez a felírás (sorrendtől eltekintve) egyértelmű, azaz csak egy $\{\alpha_1,\alpha_2,\ldots,\alpha_N\}$ szám-N-esre igaz, hogy $\mathbf{v} = \sum_{i=1}^{N} \alpha_i \cdot \mathbf{e}_i$. Ha megállapodtunk a bázisvektorok mibenlétében (például az xyz Descartes-koordinátarendszer három tengelye irányába mutató egységvektorok), akkor a vektorokat elég az α_i kifejtési együtthatóikkal jellemezni. Ezt a jelölést alkalmazva $\mathbf{v} = (\alpha_1,\alpha_2,\ldots,\alpha_N)$ egy sorvektor, ami a koordinátageometriában tanult $(\alpha_x,\alpha_y,\alpha_z)$ jelölés általánosítása. Szokás még a következő, oszlopvektoros jelölést alkalmazni:

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_i \end{pmatrix}.$$

Egy tér bázisa nem egyértelmű. Gondoljunk csak arra, hogy ha a koordinátarendszert elforgatjuk, akkor az új x', y' és z' tengelyek irányba mutató $\mathbf{e}'_x,\mathbf{e}'_y,\mathbf{e}'_z$ egységvektorok egymástól függetlenek maradnak. Így alkalmasak lesznek bázisnak, csak a vektorok $(\alpha'_x,\alpha'_y,\alpha'_z)$ alakja lesz más.

7.1. példa: Adjunk meg egy bázist a már említett, legfeljebb n-edfokú polinomok terében.

Megoldás: Bázisnak kiválóan alkalmas az $\{x^0,x^1,\ldots,x^n\}$ rendszer, hiszen minden (legfeljebb) n-edfokú polinom felírható ezen függvények lineáris kombinációjaként, azaz $\alpha_0 \cdot x^0 + \alpha_1 \cdot x^1 + \ldots + \alpha_N \cdot x^N = \sum_{i=0}^{N} \alpha_i \cdot x^i$ alakban.

Egy V vektortér **alteré**nek nevezzük az olyan W részhalmazát, amely szintén vektortér, azaz a számmal való szorzás és a vektorok összeadása nem vezet ki belőle. Altér például a háromdimenziós terünkben a kétdimenziós sík, vagy a legfeljebb n-edfokú polinomok terében a legfeljebb k-adfokú polinomok tere, ha $k < n$.

7.3.1. Kódtávolság és a javítható hibák száma

Két C^n-beli elem, **c** és **v** **Hamming-távolsága** azon i pozíciók száma, ahol a csatorna hibázott, azaz ahol $c^{(i)} \neq v^{(i)}$. A Hamming-távolság jele $d(\mathbf{c},\mathbf{v})$. A Hamming-távolság teljesíti a matematikai távolságfogalom követelményeit, azaz

$$\begin{aligned} d(\mathbf{c},\mathbf{v}) &\geq 0 \\ d(\mathbf{c},\mathbf{v}) &= d(\mathbf{v},\mathbf{c}) \\ d(\mathbf{c},\mathbf{v}) &\leq d(\mathbf{c},\mathbf{w}) + d(\mathbf{w},\mathbf{v}) \end{aligned}$$

Az utolsó egyenlőtlenség a háromszög-egyenlőtlenség.

A csatorna torzítja a rajta áthaladó jeleket, így a kimeneti oldalon az üzenetbe hibákat visz bele. Egyszerű hibázásnak nevezzük azt, ha a hibák helye és értéke nem ismert, csak maga a vett **v** szimbólumsorozat. Ha tudjuk, hogy melyik pozíció(k)ban lehet hiba, törléses hibáról beszélünk.

A C^n halmaznak a $K = \{\mathbf{c}_0,\mathbf{c}_1,\ldots,\mathbf{c}_{M-1}\}$ részhalmazát *kód*nak nevezzük, a K-beli elemeket kódszavaknak. Ha a már tömörített B-beli elemekből álló üzenet l hosszúságú szakaszainak egy-egy K-beli kódszót feleltetünk meg, akkor az üzeneten végrehajtott

$$F : B^l \mapsto K \tag{7.9}$$

egy-egyértelmű leképezés a **csatornakódolás**.

A dekódolás során két műveletet hajtunk végre: Először a vett C^n-beli **v** vektorokról el kell dönteni, hogy milyen K-beli kódszavakból keletkezhettek, majd alkalmazni kell az F inverzét. Ez formulákkal a következőképpen néz ki:

$$G : C^n \mapsto K, \qquad \text{és} \qquad F^{-1} : K \mapsto B^l. \tag{7.10}$$

Itt az F^{-1} lépés az F ismeretében triviális. A G függvény végzi a tulajdonképpeni dekódolást. A G függvény lehet táblázatban megadott, vagy például választhatjuk azt a $\mathbf{c}'$ kódszót, amelynek a legkisebb a vett $\mathbf{v}$-től való Hamming-távolsága, azaz amelyre $d(\mathbf{c}',\mathbf{v})$ minimális. (Nem szokták azonban minden lépés során kiszámítani a vett $\mathbf{v}$-nek minden K-beli elemtől vett a távolságát, inkább itt is táblázatban kezelik minden lehetséges $\mathbf{v} \in C^n$-hez a hozzárendelendő kódszót.) BSC-re ez a dekódolás optimális.

Egy kódolás fontos paramétere a **kódtávolság**, amelyet a

$$d_{\min} = \min_{\mathbf{c}\neq\mathbf{c}',\mathbf{c},\mathbf{c}'\in K} d(\mathbf{c},\mathbf{c}') \tag{7.11}$$

formulával definiálhatunk. A kódtávolság lényegében a kódszavak közötti minimális távolság.

Ha a vevő oldalon csak azt szeretnénk jelezni, hogy a vett sorozat hibás – például azért, hogy újra elküldessük a kódszót –, akkor *hibajelzés*ről beszélünk. Csak akkor tudjuk jelezni, hogy a kódszavunkban hiba van, ha nem egy másik érvényes kódszóba transzformálódott, azaz, ha $\mathbf{v} \notin K$. Ez akkor lehetséges, ha a keletkezett $\mathbf{v}$ és az eredeti $\mathbf{c}$ távolsága kisebb, mint a kódtávolság, $d(\mathbf{c},\mathbf{v}) < d_{\min}$.

Ha ν-vel jelöljük az adott kódszóban elforduló hibás szimbólumok számát, akkor egy $d_{\min}$ kódtávolságú kódolással $\nu < d_{\min}$ hibát tudunk jelezni.

Mivel a hiba jelzése után a kódszót újból el szokták küldeni, és ez igen sok időt vesz igénybe, általában olyan csatornakódolási eljárásokat alkalmaznak, amelyek lehetővé teszik a hiba kijavítását. Tegyük fel, hogy $\mathbf{c}$-ből a csatornán való áthaladás során $\mathbf{v}$ vektor keletkezik, mégpedig ν darab egyszerű hibával. Ekkor csak úgy lehet visszaállítani az eredeti $\mathbf{c}$ kódszót, ha a $\mathbf{v}$-nek minden más lehetséges $\mathbf{c}' \in C$ kódszótól való távolsága nagyobb, mint $d(\mathbf{c},\mathbf{v})$. Írjuk fel a háromszög-egyenlőtlenséget a kódszavak távolságára:

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

Ha ezt a formulát átrendezzük úgy, hogy az egyik oldalon csak $d(\mathbf{c}',\mathbf{v})$ maradjon, majd behelyettesítjük a hibák javíthatóságának

$$d(\mathbf{c}',\mathbf{v}) > d(\mathbf{c},\mathbf{v}) \tag{7.12}$$

feltételébe, akkor azt kapjuk, hogy

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

Újabb átrendezéssel ebből a

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

feltételt kapjuk, minden $\mathbf{c}' \in C$-re. A $\mathbf{c}'$-k egyike sem nagyobb, mint $d_{\min}$, így

$$d(\mathbf{c},\mathbf{v}) < \frac{1}{2} d_{\min}, \tag{7.13}$$

tehát

ha egyszerű hibázásról van szó, egy $d_{\min}$ kódtávolságú kóddal legfeljebb $(d_{\min}-1)/2$ hibát tudunk kijavítani.

(Ha $d_{\min}$ páros, akkor a javítható hibák száma $(d_{\min}-2)/2$.)

Törléses hibák esetében tudjuk, hogy melyik pozíciókban van a hiba. Ekkor $d_{\min}$ azt a számot takarja, amennyi szimbólumnak a megfelelő pozíciókról való törlésével a két legközelebbi kódszó maradéka azonos lesz. Ha ennél kevesebb törléses hibát generál a csatorna, akkor azok javíthatók lesznek.

Törléses hibák esetén tehát maximálisan $d_{\min}-1$ hiba javítható.

Látható, hogy minél nagyobb a kódtávolság, annál több hibát tudunk javítani, illetve jelezni.

A **Singleton-korlát** összefüggést ad egy kód ábécéjének r elemszáma, a kódszavak n hossza és M száma, valamint a kódtávolság között, mégpedig az

$$M \le r^{n-d_{\min}+1} \tag{7.14}$$

egyenlőtlenséggel.

Bizonyítás: Az r elemből felépülő, k hosszúságú különböző sorozatok száma r^k (l. ismétléses variáció). Legyen k egy olyan természetes szám, amelyre $r^{k-1} < M \le r^k$. Mivel több a kódszavak M száma, mint ahány különböző $k-1$ hosszúságú sorozatot tudunk a kódábécé r eleméből felépíteni, biztos, hogy van legalább két olyan kódszó, $\mathbf{c}_1$ és $\mathbf{c}_2$, amelyeknek az első $k-1$ pozíciójában azonos elemek vannak, azaz amelyeknek a távolsága kisebb, mint $n-(k-1)$. Így a kódtávolság

$$d_{\min} \le n-k+1 \tag{7.15}$$

Ha átrendezzük az egyenlőtlenséget k-ra, majd minkét oldalt az r kitevőjébe emeljük, és figyelembe vesszük, hogy $M \le r^k$, a tételt bebizonyítottuk. (Q.E.D)

A kódszavak M száma egyértelműen meghatározza k-t, így a Singleton-korlát lényegében egy az M-től függő felső korlátja a kódtávolságnak,

amely a (7.15) alakot ölti. Egy kódot **maximális távolságú**nak, vagy **MDS**-nek hívunk (maximum distance separable), ha a (7.14) formulában az egyenlőség érvényes, azaz

$$d_{\min} = n - k + 1 = n - \log_r M + 1.$$

Az M-nek az r alapú logaritmusa ugyan nem mindig egész szám, de mindig találhatunk az M-hez egy k természetes számot, amelyre $r^{k-1} < M \leq r^k$ igaz. Az n és a k számokat szokták a kód paramétereinek nevezni.

A **Hamming-korlát** vagy gömbpakolási korlát megadja annak a feltételét, hogy egy (n,k) paraméterű kóddal ν egyszerű hibát ki lehessen javítani. Szemléletesen, a C^n térben – az n hosszúságú szimbólumsorozatok terében – a $\mathbf{c} \in K$ kódszavak pontok, méghozzá lehetőleg egymástól minél távolabb elhelyezkedő pontok. A javítás alapötlete az, hogy ha a $\mathbf{v}$ vektor egy adott $\mathbf{c}$ kódszó körüli ν sugarú gömbben van, azt a $\mathbf{c}$-be javítjuk. Természetesen a gömbök nem fedhetnek át, különben nem tudnánk, melyik kódszóba javítsunk. Ha r elemű a kódábécé, akkor az olyan vektorok száma, amelyek a $\mathbf{c}$-től pontosan i elemben térnek el (méghozzá meghatározott pozíciókban),

$$(r-1)^i.$$

Azt, hogy melyek legyenek azok a pozíciók, amelyen eltérés mutatkozik az n hosszú kódszótól,

$$\binom{n}{i}\text{-féleképpen}$$

választhatjuk ki, így azon vektorok száma, amelyek pontosan i elemben különböznek $\mathbf{c}$-től

$$\binom{n}{i}(r-1)^i. \tag{7.16}$$

Hogy megkapjuk a ν sugarú gömbön belüli vektorok számát, összegezni kell a (7.16) tagokat $i = 1,\ldots,\nu$-re. Az összes gömbben elhelyezkedő vektorok száma nem haladhatja meg a C^n tér elemeinek a számát, r^n-t, így ha a kódszavak száma r^k, akkor

$$r^k \sum_{i=0}^{\nu} \binom{n}{i}(r-1)^i \leq r^n. \tag{7.17}$$

Az olyan kódokat, amelyekre a (7.17) Hamming-korlátban az egyenlőség igaz, **perfekt kódoknak** nevezzük.

7.4. Csatornakódolási tétel és megfordítása – a Shannon–Hartley-tétel

Az információátvitel gyorsaságának jellemzésére bevezethetjük a *jelsebesség*et vagy más néven *kódsebesség*et a következő formula segítségével:

$$R = \frac{H(K)}{n}. \tag{7.18}$$

Jelen esetben a $H(K)$ a kódolt forrás entrópiája, az n pedig továbbra is a kódszavak hossza. A jelsebesség tehát az egy kódszóval átlagosan átvitt információ és a kódszó hosszának a hányadosa. Ha nem ismerjük a kódszavak előfordulási valószínűségét, általában egyenletes eloszlással szoktuk közelíteni, azaz $p(\mathbf{c}_i) = 1/M$ minden i-re, ha M a kódszavak száma. Ekkor az entrópia a $-\sum_{i=1}^{M} \frac{1}{M} \log_2 \frac{1}{M} = \log_2 M$ alakot ölti, a kódsebesség pedig

$$R = \frac{\log_2 M}{n} \tag{7.19}$$

lesz. Ez a kódsebesség tulajdonképpen egy a csatornakódnak csak a legáltalánosabb paramétereitől (kódszóhossz és kódszószám) függő, a többi specifikációjától független *átlagos kódsebesség*nek is felfogható.

A **Shannon–Hartley-tétel** vagy csatornakódolási tétel arra ad választ, hogy az adott $\mathcal{C}$ csatornakapacitású csatornán milyen sebességgel lehet megfelelő biztonsággal, hibamentesen információt továbbítani. A tételt a következőképpen lehet összefoglalni: Egy $\mathcal{C}$ kapacitású, diszkrét, memóriamentes csatornán, ha a kódsebesség kisebb, mint a csatornakapacitás, akkor lehet olyan n hosszúságú kódszavakból álló csatornakódot találni, amelynél a hibás dekódolás valószínűsége tetszőlegesen kicsi. Ha a csatornakapacitásnál nagyobb jelsebességgel szeretnénk a csatornán információt továbbítani, a hibás dekódolás valószínűsége nem csökkenthető tetszőlegesen kicsivé, még az n kódszóhossz növelésével sem. Matematikai formulákkal:

- Ha $R < \mathcal{C}$, akkor lehet olyan n kódszóhosszt találni, hogy a hibás dekódolás valószínűsége bármilyen kis $\varepsilon > 0$ számnál kisebb legyen.
- Ha $R > \mathcal{C}$, akkor a hibás dekódolás valószínűsége mindig nagyobb, mint $1 - \frac{\mathcal{C}}{R} - \frac{1}{nR} > 0$. Ez az alsó korlát n növelésével nő.

A tétel nem konstruktív, azaz nem ad meg módszert az ideális jelsebesség elérésére, csak azt adja meg, hogy a hibamentes információ-

továbbítás maximális jelsebessége a csatornakapacitásnál mindenképpen kisebb.

8. Lineáris blokk-kódok

Az (n,k) paraméterű lineáris blokk-kódok k elemű tömörített üzenetvektorokból képeznek n elemű kódszóvektorokat.

8.1. Generátormátrix

A lineáris csatornakódok $K \subset C^n$ tere egy vektortér, avagy lineáris tér. Létezik tehát a K vektortéren belül egy bázisrendszer, amelynek az elemeit $\mathbf{g}_0,\mathbf{g}_1,\ldots,\mathbf{g}_{k-1}$-gyel jelöljük, ahol k a K tér dimenziószáma. A $\mathbf{c}_i$ kódszavak egyértelműen kifejthetők a $\mathbf{g}_j$ bázisvektorok szerint:

$$\mathbf{c}_i = \sum_{j=0}^{k-1} \alpha_{ij}\mathbf{g}_j. \tag{8.1}$$

A $\mathbf{c}_i$-t tehát lehet reprezentálni az $\vec{\alpha}_i = (\alpha_{i0},\alpha_{i1},\ldots,\alpha_{i(k-1)})$ sorvektorral. A $\mathbf{c}_i$ kódszóvektort visszakapjuk, ha $\vec{\alpha}_i$-t egy olyan mátrixszal szorozzuk meg, amelyet a $\mathbf{g}_j$ bázisvektorok egymás alá írásával kapunk meg:

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{pmatrix} = \begin{pmatrix} g_{00} & g_{01} & \cdots & g_{0\,(n-1)} \\ g_{10} & g_{11} & \cdots & g_{1\,(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{(k-1)\,0} & g_{(k-1)\,1} & \cdots & g_{(k-1)\,(n-1)} \end{pmatrix}.$$

A $\mathbf{G}$ mátrixot a kód **generátormátrix**ának nevezik. A mátrix k sorból és n oszlopból áll.

Generátor úgy, mint a kódot generáló rendszer, amely az n hosszúságú szimbólumsorozatok C^n teréből előállítja a kódok K alterét.

Mivel a bázisrendszer választása nem egyértelmű, a generátormátrix sem az, minden lehetséges bázisrendszerhez más és más $\mathbf{G}$ mátrix tartozik. Van azonban egy olyan bázisrendszer, amelyre a kifejtési együtthatókból képezett $\vec{\alpha}_i$ vektor pont a kódolandó tömörített (forráskódolt) $\mathbf{b}_i$ üzenetvektor. Az is igaz, hogy ha a $\{\mathbf{b}_i | i = 0,1,\ldots,M-1\}$ üzenetvektorok mindegyikét megszorozzuk egy $\mathbf{G}$ generátormátrixsszal, akkor jó lineáris blokk-kódot fogunk kapni.

Figyeljük meg, hogy az n dimenziós C^n vektorterünk n elemű vektorait használjuk a k dimenziós K altér leírására, azaz a kódolásra. A k dimenziós

tereket viszont jól le lehet írni k elemű sorvektorokkal, tehát a maradék $n-k$ elem nem hordoz lényeges, új információt, azaz *redundáns*. Ezt a redundanciát használjuk arra, hogy megnöveljük a kódszavak Hamming-távolságát: arra, hogy még zajos csatorna esetén is nagy biztonsággal visszafejthető legyen az üzenetünk.

A csatornakódolás tehát a lényeges információ mennyiségét nem változtatja, csak a felhasznált szimbólumok számát növeli, így az egy szimbólumra jutó információ várható értékét, azaz az entrópiát csökkenti.

Nézzünk egy egyszerű számpéldát.

8.1. példa: Vegyünk kétdimenziós bináris vektorokat, mint üzeneteket, a 2×5-ös generátormátrix pedig legyen

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

Adjuk meg a kódszavakat és a kódtávolságot.

Megoldás: A generátormátrixból a bázisvektorok

$$\begin{aligned} \mathbf{g}_0 &= (1\,0\,1\,0\,1) \\ \mathbf{g}_1 &= (0\,1\,1\,1\,0). \end{aligned}$$

A $\mathbf{b}_0 = (0\ 0)$ üzenethez tartozó kódszó $\mathbf{c}_0 = b_{00} \cdot \mathbf{g}_0 + b_{01} \cdot \mathbf{g}_1 = 0 \cdot (1\ 0\ 1\ 0\ 1) + 0 \cdot (0\ 1\ 1\ 1\ 0) = (0\ 0\ 0\ 0\ 0)$, ahol b_{00} a $\mathbf{b}_0$ vektor nulladik, b_{01} pedig az első komponense. Hasonlóképpen a többi lehetséges üzenetvektorra, $\mathbf{b}_1 = (0\ 1)$-re, $\mathbf{b}_2 = (1\ 0)$-ra és $\mathbf{b}_3 = (1\ 1)$-re a kapott kódszóvektorok:

$$\begin{aligned} \mathbf{c}_1 &= b_{10} \cdot \mathbf{g}_0 + b_{11} \cdot \mathbf{g}_1 = 0 \cdot (1\,0\,1\,0\,1) + 1 \cdot (0\,1\,1\,1\,0) = (0\,1\,1\,1\,0) \\ \mathbf{c}_2 &= b_{20} \cdot \mathbf{g}_0 + b_{21} \cdot \mathbf{g}_1 = 1 \cdot (1\,0\,1\,0\,1) + 0 \cdot (0\,1\,1\,1\,0) = (1\,0\,1\,0\,1) \\ \mathbf{c}_3 &= b_{30} \cdot \mathbf{g}_0 + b_{31} \cdot \mathbf{g}_1 = 1 \cdot (1\,0\,1\,0\,1) + 1 \cdot (0\,1\,1\,1\,0) = (1\,1\,0\,1\,1). \end{aligned}$$

A kód valóban lineáris, vektorainak összeadására és a 0, illetve 1 számmal való szorzására érvényesek a vektortér-axiómák. A kapott kód minimális kódtávolságának kiszámításához nézzük meg a kódszavak Hamming-távolságát:

$$\begin{aligned} &d(\mathbf{c}_0,\mathbf{c}_1) = 3, \quad d(\mathbf{c}_0,\mathbf{c}_2) = 3, \\ &d(\mathbf{c}_0,\mathbf{c}_3) = 4, \quad d(\mathbf{c}_1,\mathbf{c}_2) = 4, \\ &d(\mathbf{c}_1,\mathbf{c}_2) = 3, \quad d(\mathbf{c}_2,\mathbf{c}_3) = 3. \end{aligned}$$

Ezek közül a legkisebb a 3, így $d_{\min} = 3$.

8.1.1. Szisztematikus kódok generátormátrixa

Ha az (n,k) paraméterű kódunk $\mathbf{c}_i \in K$ vektorai olyan szerkezetűek, hogy azok végéről az utolsó $n-k$ elemet elhagyva, az eredeti üzenetet kapjuk

vissza, akkor **szisztematikus kód**ról beszélünk. A szisztematikus kódok c_i vektorainak első k elemét *üzenetszegmens*nek hívják, a maradékot *paritásszegmens*nek.

Az első alkalmazásokkor csak egy egyszerű paritásbitet tettek az üzenetek után a hibás üzenetátvitel detektálására, azaz az üzenetek után megadták, hogy páros vagy páratlan számú egyest továbbítottak. Innen eredeztethető a név.

A 8.1 példában szereplő kód szisztematikus volt. A szisztematikus kódok generátormátrixa jellegzetes, az első k oszlopa egységmátrixot alkot – ez adja a mátrixszorzás során az üzenetszegmenst –, a további $n-k$ oszlop közül pedig egyik sem áll csupa nullából; ez a $k \times (n-k)$-s mátrix a paritásszegmenst hozza létre.

Matematikai kitérő – Mátrixszorzásról Vegyünk először egy j elemű $\mathbf{a}$ sorvektort, és egy szintén j elemű $\mathbf{b}^T$ oszlopvektort. Az $\mathbf{a} \cdot \mathbf{b}^T$ skalárszorzaton a következő számot értjük:

$$\mathbf{a} \cdot \mathbf{b}^T = \begin{pmatrix} a_1 & a_2 & \dots & a_j \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_j \end{pmatrix} = a_1 b_1 + a_2 b_2 + \ldots + a_j b_j. \quad (8.2)$$

Legyen egy $\mathbf{A}$ és egy $\mathbf{B}$ mátrixunk, az első álljon ugyanannyi oszlopból (j darabból), mint ahány sorból áll a második:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} \\ a_{21} & a_{22} & \dots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{j1} & b_{j2} & \dots & b_{jk} \end{pmatrix}.$$

Az $\mathbf{A} \cdot \mathbf{B}$ szorzaton azt az i sorból és k oszlopból álló mátrixot értjük, amelynek az m-edik sorának n-edik elemét a következőképpen kapjuk meg:

$$(\mathbf{A} \cdot \mathbf{B})_{m,n} = a_{m1} b_{1n} + a_{m2} b_{2n} + \ldots + a_{mj} b_{jn}$$

Ez tulajdonképpen az A mátrix m-edik sorvektorának és a B mátrix n-edik oszlopvektorának a skalárszorzata.

Ha $i \neq k$, akkor a $\mathbf{B} \cdot \mathbf{A}$ szorzás nem értelmezhető, mivel a $\mathbf{B}$ mátrix sorvektorai nem ugyanolyan hosszúak, mint az A mátrix oszlopvektorai. Ha $i = k$, akkor lehet értelmezni a $\mathbf{B} \cdot \mathbf{A}$ szorzatot, de roppant kevés kivételtől eltekintve $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$. A mátrixszorzás tehát többnyire nem felcserélhető, nem kommutatív.

Ha valamelyik négyzetes mátrix

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

szerkezetű, akkor a vele való szorzás – akár jobbról, akár balról – az eredeti mátrixot adja, azaz:

$$\mathbf{IA} = \mathbf{A}, \qquad \mathbf{AI} = \mathbf{A}.$$

Az $\mathbf{I}$ mátrixokat ezért *egységmátrix*nak nevezzük.

A vektorok skalárszorzatánál mindig sorvektort szorzunk oszlopvektorral, ha fordított sorrendet alkalmaznánk, azaz egy n elemből álló oszlopvektort szoroznánk ugyanolyan elemszámú sorvektorral, az eredmény egy $n \times n$-es mátrix lenne, amelynek az i-edik sorában a j-edik elem az oszlopvektor j-edik és a sorvektor i-edik elemének szorzata. A vektorok ilyen szorzását diadikus szorzásnak nevezzük.

Megfigyelhetjük, hogy a 8.1 példában szereplő generátormátrixának az első két oszlopa egységmátrixot alkot, és minden kódszónak az első két eleme megegyezik a kódolt üzenettel. Ezt a két elemet hozza létre az egységmátrix.

8.2. A paritásellenőrző mátrix és a szindróma

Ha a csatorna kimenetén kaptunk egy $\mathbf{v} \in C^n$ vektort, azt valahogy ellenőrizni kell, hogy jó kódszó-e. Az ellenőrzést úgy lehet elvégezni, hogy a $\mathbf{v}$ vektort egy **paritásellenőrzési mátrix**szal szorozzuk meg, és vizsgáljuk a kapott $\mathbf{s}$ vektort, amelyet **szindróma**nak nevezünk. Jelöljük $\mathbf{H}^T$-vel a

paritásellenőrző mátrixot (vagy röviden paritásmátrixot), amely n sorból és $n-k$ oszlopból áll. Akkor és csak akkor érvényes kódszó a $\mathbf{v}$ vektor, ha

$$\mathbf{s} = \mathbf{v} \cdot \mathbf{H} = \mathbf{0}. \tag{8.3}$$

A $\mathbf{c_i} = \vec{\alpha}_i \mathbf{G}$ kódszavak szindrómája tehát $\mathbf{0}$, azaz

$$\mathbf{c_i}\mathbf{H}^T = \vec{\alpha}_i \mathbf{G}\mathbf{H}^T = \mathbf{0}, \tag{8.4}$$

minden i-re.

Ha csak a $\mathbf{G}\mathbf{H}^T$ mátrixszorzásával előállt $k \times (n-k)$ elemű mátrixot vizsgáljuk, annak is minden eleme 0 lesz, különben nem teljesülhetne (8.4) minden lehetséges $\vec{\alpha}_i$-re. Így

$$\mathbf{G}\mathbf{H}^T = 0. \tag{8.5}$$

Ezt az egyenlőséget használják fel $\mathbf{H}^T$ előállítására $\mathbf{G}$-ből, illetve $\mathbf{G}$ előállítására $\mathbf{H}^T$-ből.

8.2. példa: Készítsük el a 8.1 példában szereplő generátormátrixhoz tartozó paritásellenőrző mátrixot. Számoljuk ki vele a $\mathbf{v} = (1\ 0\ 1\ 1\ 1)$ vektor szindrómáját.

Megoldás: Tudjuk, hogy ha a generátormátrix 2×5-ös volt, akkor a kód két paramétere $n = 5$ és $k = 2$, a paritásmátrix mérete pedig $n \times (n-k) = 5 \times 3$. Ha felhasználjuk azt, hogy a generátor- és paritásmátrixok szorzata nullmátrix (8.5), a következő mátrix-egyenletet kell megoldani:

$$\mathbf{G}\mathbf{H}^T = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \\ h_{41} & h_{42} & h_{43} \\ h_{51} & h_{52} & h_{53} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Eszerint a $\mathbf{H}^T$ első oszlopainak az elemeire igaz lesz, hogy

$$\begin{aligned} 1 \cdot h_{11} + 0 \cdot h_{21} + 1 \cdot h_{31} + 0 \cdot h_{41} + 1 \cdot h_{51} &= 0 \\ 0 \cdot h_{11} + 1 \cdot h_{21} + 1 \cdot h_{31} + 1 \cdot h_{41} + 0 \cdot h_{51} &= 0. \end{aligned}$$

Mivel a paritásmátrixnak nem lehet tiszta nulla oszlopa, az első egyenletből következik, hogy a h_{11}, h_{31}, h_{51} hármasból pontosan 2 db 1 a harmadik 0, a második egyenletből pedig a h_{21}, h_{31}, h_{41} hármasra következik hasonló állítás. Az egyik számhármas lehet tiszta 0 is, ha a másik nem az. Látható, hogy a $\mathbf{H}^T$ mátrix második és harmadik oszlopára is igaz lesz az, hogy a $\{h_{1i}, h_{3i}, h_{5i}\}$ és a $\{h_{2i}, h_{3i}, h_{4i}\}$ számhármasok egyikének pontosan kettő eleme lesz 1, a harmadik

nulla, a másiknak pedig vagy 2, vagy 0 eleme lesz 1. Írjuk fel az összes olyan oszlopot, amelyre ez teljesül:

$$\begin{array}{cccccc} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}$$

Ebből a hat oszlopból kell hármat kiválasztani úgy, hogy a belőlük felépített mátrixnak ne legyen sem ismétlődő, sem pedig tiszta nullából álló sora. Egy megoldás:

$$\mathbf{H}^T = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Az oszlopok tetszőlegesen felcserélhetők, és máshogy is megválaszthatók.

A $\mathbf{v}$ vektor szindrómája:

$$\mathbf{s}(\mathbf{v}) = \mathbf{v} \cdot \mathbf{H}^T = (1\,0\,1\,1\,1) \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (0\,1\,0).$$

A $\mathbf{v}$ vektor nem kódszó, a szindrómája tényleg nem 0.

8.2.1. Szisztematikus kódok paritásellenőrző mátrixa

Ha *szisztematikus* a kódunk, akkor A paritásellenőrző mátrix előállítása igen egyszerű. Szisztematikus kódok esetén ugyanis nem csak a $\mathbf{G}$ generátormátrix első k oszlopa alkot egységmátrixot, hanem a $\mathbf{H}^T$ paritásellenőrző mátrix utolsó $n-k$ sora is. A maradék $k \times (n-k)$-s mátrixok a G végén, illetve H^T felső részén egymás ellentettjei lesznek. Ennek oka a következő: $\mathbf{G}$ és $\mathbf{H}^T$ felírható, mint

$$\mathbf{G} = \left(I_{k\times k}, \mathbf{P}_{k\times(n-k)}\right), \quad \text{illetve} \quad \mathbf{H}^T = \begin{pmatrix} \mathbf{P}'_{k\times(n-k)} \\ I_{(n-k)\times(n-k)} \end{pmatrix}.$$

A $\mathbf{G}\mathbf{H}^T$ mátrixszorzás során az i-edik sor j-edik elemének előállításakor a $\mathbf{G}$ mátrix i-edik sorát szorozzuk a $\mathbf{H}^T$ mátrix j-edik oszlopával. A $\mathbf{G}$ mátrixból vett vektor első k eleme közül csak pont az i-edik lesz 1, a

többi 0. A fennmaradó $n-k$ elem $\mathbf{P}$ i-edik sora. A $\mathbf{H}^T$-ből származó vektorra, hasonlóképpen, az első k elem lesz a $\mathbf{P}'$ j-edik oszlopa, a maradék k közül pedig csak a j-edik lesz 1, a többi nulla. Szorzásukkor a sorvektor egységmátrixból származó része az oszlopvektor $\mathbf{P}'$-ből származó részével kerül össze, s mivel az első vektorból csak az i-edik elem nem nulla, csak az ad nem nulla szorzatot a (8.2) összeg első k tagja közül. Ennek a résznek az eredménye tehát P'_{ij} ($\mathbf{P}'$ j-edik sorvektorának i-edik eleme). A második $n-k$ tagnál hasonló helyzet következik be, csak a második vektorból származik az egyetlen 1-est tartalmazó rész, az elsőből a P, az eredmény pedig P_{ij} ($\mathbf{P}$ i-edik sorvektorának j-edik eleme). A teljes összeg nullát kell hogy adjon, mivel $\mathbf{G}\mathbf{H}^T = 0$, tehát $P_{ij} = -P'_{ij}$. Ez $n=5$, $k=3$ esetre például az alábbi szerint alakul:

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{P_{11}} & \mathbf{P_{12}} \\ 0 & 1 & 0 & P_{21} & P_{22} \\ 0 & 0 & 1 & P_{31} & P_{32} \end{pmatrix} \cdot \begin{pmatrix} P'_{11} & \mathbf{P'_{12}} \\ P'_{21} & \mathbf{P'_{22}} \\ P'_{31} & \mathbf{P'_{32}} \\ 1 & \mathbf{0} \\ 0 & \mathbf{1} \end{pmatrix}$$

A vastag szedéssel kiemelt első sor, illetve második oszlop szorzásakor az eredmény $1 \cdot P'_{12} + P_{12} \cdot 1$.

8.2.2. Egy szindróma által generált mellékosztály és a hibajavítás

Térjünk vissza a vett $\mathbf{v}$ szimbólumsorozat hibáit kijavító dekódolás lehetőségeihez. Vegyük észre, hogy a $\mathbf{v}$ vektor szindrómája tulajdonképpen csak a $\mathbf{v}$-nek a $\mathbf{c}$-től való eltérésének és a paritásmátrixnak a szorzata, azaz, ha $\mathbf{v} = \mathbf{c} + \Delta\mathbf{c}$, akkor

$$\mathbf{s} = \mathbf{v} \cdot \mathbf{H}^T = (\mathbf{c} + \Delta\mathbf{c}) \cdot \mathbf{H}^T = \mathbf{0} + \Delta\mathbf{c} \cdot \mathbf{H}^T,$$

mivel a mátrixokkal való szorzás is disztributív, és $\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0}$.

8.3. példa: Adjuk meg a 8.2 példában szereplő $\mathbf{v} = (1\ 0\ 1\ 1\ 1)$ vektor lehetséges hibavektorainak a szindrómáját.

Megoldás: A 8.1 példa alapján a lehetséges kódszavakat, és azok $\mathbf{v}$-től vett eltérését a következő táblázat tartalmazza:

$\mathbf{c}_i$	$\Delta\mathbf{c}_i = \mathbf{c}_i - \mathbf{v}$
(0 0 0 0 0)	(1 0 1 1 1)
(0 1 1 1 0)	(1 1 0 0 1)
(1 0 1 0 1)	(0 0 0 1 0)
(1 1 0 1 1)	(0 1 1 0 0)

Ha bármely $\Delta\mathbf{c}_i$ vektor szorzatát kiszámoljuk a

$$\mathbf{H}^T = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

mátrixszal, akkor a szindrómának $\mathbf{s}(\Delta\mathbf{c}_i) = (0\ 1\ 0)$-t kapunk.

Dekódolás során legtöbbször a vett $\mathbf{v}$ vektor szindrómája alapján megbecsülik a $\Delta\mathbf{c}$ hibavektort, és ezt levonva a $\mathbf{v}$-ből, megkapják az üzenetet, vagy legalábbis egy becslést arra, mi lehetett az üzenet. Mivel a vektoraink elemszáma n véges, véges sokféle $\Delta\mathbf{c}$ hibavektor fordulhat elő. Ha az egyes $\Delta\mathbf{c}_i$-khez hozzáadjuk a K kód elemeit, akkor megkapjuk a belőlük létrejövő $\mathbf{v}$ vektorokat. Az így létrejött $M_i \subset C^n$ halmazt a $\Delta\mathbf{c}_i$ hiba által generált **mellékosztály**nak nevezzük.

Bizonyos $\Delta\mathbf{c}_j$ hibamintázatok előállnak egy másik $\Delta\mathbf{c}_i$ hibából egy kódszóvektor hozzáadásával, azaz $\Delta\mathbf{c}_i + \mathbf{c}_k$ alakban. Ezek a hibamintázatok tehát nem generálnak külön-külön mellékosztályt, hiszen ugyanazt a halmazt kapjuk belőlük, csak az elemek sorrendje lesz más. Egy-egy mellékosztály elemei így nem csak a lehetséges vett $\mathbf{v}$ vektorok, hanem a kódelemeket az adott $\mathbf{v}$-kbe vivő $\Delta\mathbf{c}_i$ hibavektorok is egyszerre. (Hiszen lineáris kódokról van szó, azaz a nullvektor is érvényes kódszó, az meg minden hibát önmagába visz.)

A mellékosztályok elemeit súlyuk szerint sorrendbe szokták rakni. Egy $\mathbf{v} = (v_1, \ldots, v_n) \in C^n$ **vektor** $w(\mathbf{v})$ **súlya** a nem nulla v_i szimbólumainak száma. A **mellékosztály vezető elemé**nek a legkisebb súlyú elemét nevezik.

Egy kód $w_{\min}$ *minimális súlyán* a nem nulla kódszavak súlyai közül a legkisebbet értik. Ez egyben a minimális kódtávolság is.

Természetesen egy $\mathbf{v}$-t többféle hibavektorból más és más $\mathbf{c} \in K$-val is megkaphatunk, de ugyanaz a $\mathbf{v}$ csak egy M_i mellékosztálynak lesz eleme. Egy M_i mellékosztály elemei mindig ugyanazt az $\mathbf{s}_i$ szindrómát adják. El kell tehát dönteni, hogy dekódolás során az azonos mellékosztálybeli $\Delta\mathbf{c}$ hibavektorok közül melyikkel számoljuk ki az üzenetet. Vegyük észre, hogy a legkisebb súlyú elem, azaz a mellékosztály vezető eleme adott szindróma mellett a csatorna legkisebb hibáját tételezi fel. (A $\Delta\mathbf{c}_i$ hibavektorban található nulla szimbólumok helyén nincs hiba, csak a nem nulla

szimbólumok helyén.) Ezért, ha a $\mathbf{v}$ vett vektor szindrómája $\mathbf{s}_i$, a legegyszerűbb döntési séma arra nézve, hogy melyik M_i-beli elem legyen a hibánk, egyszerűen kiválasztani a mellékosztály vezető elemét, $\Delta\mathbf{c}_{i,0}$-t. Így a vett üzenet $\mathbf{c}' = \mathbf{v} - \Delta\mathbf{c}_{i,0}$ lesz. Az egyes szindrómákhoz tartozó minimális súlyú hibákat táblázatban lehet tárolni.

8.4. példa: Adjuk meg a $\mathbf{v} = (1\ 0\ 1\ 1\ 1)$ vektor, mint lehetséges hibamintázat által a a 8.1 példában szereplő kódszavakból generált mellékosztályt. Javítsuk ki a $\mathbf{v} = (1\ 0\ 1\ 1\ 1)$ vektort a mellékosztály vezető elemével.

Megoldás: A $\mathbf{v} = (1\ 0\ 1\ 1\ 1) = \Delta\mathbf{c}_\mathbf{v}$ hibamintázat által generált mellékosztályt úgy kapjuk meg, hogy minden lehetséges kódszóhoz hozzáadjuk a szóban forgó hibavektort. Lássuk:

i	$\mathbf{c}_i$	$\Delta\mathbf{c}_i = \mathbf{c}_i + \Delta\mathbf{c}_\mathbf{v}$
0	$(0\ 0\ 0\ 0\ 0)$	$(1\ 0\ 1\ 1\ 1)$
1	$(0\ 1\ 1\ 1\ 0)$	$(1\ 1\ 0\ 0\ 1)$
2	$(1\ 0\ 1\ 0\ 1)$	$(\mathbf{0\ 0\ 0\ 1\ 0})$
3	$(1\ 1\ 0\ 1\ 1)$	$(0\ 1\ 1\ 0\ 0)$

A táblázat harmadik oszlopa tartalmazza a mellékosztály elemeit. A legkisebb súlyú elem nyilvánvalóan a második, amelyik vastag szedéssel ki van emelve, hiszen az csak egyetlen nem nulla komponenst tartalmaz. Ezzel kijavítva a $\mathbf{v}$ vektort a

$$\mathbf{v} - \Delta\mathbf{c}_2 = (1\ 0\ 1\ 1\ 1) - (0\ 0\ 0\ 1\ 0) = (1\ 0\ 1\ 0\ 1) = \mathbf{c}_2$$

kódszót kapjuk.

Bármely $\Delta\mathbf{c}_i$-ből kiindulva ugyanezt a mellékosztályt kapjuk. Nézzük például a $\Delta\mathbf{c}_1$-et, mint kiindulási elemet:

i	$\mathbf{c}'_i$	$\Delta\mathbf{c}'_i = \mathbf{c}_i + \Delta\mathbf{c}_1$
0	$(0\ 0\ 0\ 0\ 0)$	$(1\ 1\ 0\ 0\ 1)$
1	$(0\ 1\ 1\ 1\ 0)$	$(1\ 0\ 1\ 1\ 1)$
2	$(1\ 0\ 1\ 0\ 1)$	$(0\ 1\ 1\ 0\ 0)$
3	$(1\ 1\ 0\ 1\ 1)$	$(0\ 0\ 0\ 1\ 0)$

A másik két hibamintázatból kiindulva hasonlóan ellenőrizhető az állítás igazsága.

9. Hamming-kód

Hamming-kódoknak nevezzük azokat a perfekt hibajavító kódokat, amelyek egy hibát képesek kijavítani. A Hamming-kódok konstrukciójakor az alapvető feladat tehát adott $n - k$ számú paritásbithez megtalálni a maximális k üzenethosszat – s egyben a maximális n kódszóhosszt – úgy, hogy egy hibát javítani tudjunk.

Matematikai kitérő – Véges számtestekről avagy a Galois-testekről. Vegyünk egy számokból álló véges elemszámú $GF(N) = \{t_1, t_2, \ldots, t_N\}$ halmazt. Értelmezzünk a halmaz elemei között két műveletet, a „+"-szal jelölt összeadást és a „·"-tal jelölt szorzást úgy, hogy egyik művelet se vezessen ki a halmazból – azaz ha $t \in GF(N)$ és $u \in GF(N)$, akkor $t+u \in GF(N)$ és $t \cdot u \in GF(N)$. Akkor nevezzük a halmazunkat véges számtestnek, vagy Galois-testnek (Galois Field), ha ezen kívül

1. az összeadásra igaz, hogy
   (a) $t + u = u + t$, azaz az összeadás kommutatív,
   (b) ha $s \in GF(N)$, $(s+t)+u = s+(t+u)$, az összeadás asszociatív is,
   (c) létezik egy **nullelem**, amely minden $t \in GF(N)$-nel összeadva az eredeti t-t adja: ha a nullelemet 0-val jelöljük, $t + 0 = t$,
   (d) minden $t \in GF(N)$-nek létezik egy ellentett eleme, amellyel összeadva 0-t ad eredményül. Ha t ellentettjét $(-t)$-vel jelöljük, akkor $t + (-t) = 0$;
2. illetve a szorzásra teljesül, hogy
   (a) $t \cdot u = u \cdot t$, azaz a szorzás kommutatív,
   (b) ha $s \in GF(N)$, $(s \cdot t) \cdot u = s \cdot (t \cdot u)$, a szorzás asszociatív,
   (c) létezik egy egységelem, amely minden $t \in GF(N)$-nel megszorozva az eredeti t-t adja: ha az egységelemet 1-gyel jelöljük, $t \cdot 1 = t$,
   (d) minden $t \in GF(N)$-nek $(t \neq 0)$ létezik egy inverz eleme, amellyel megszorozva 1-et ad eredményül. Ha t inverzét (t^{-1})-nel jelöljük, akkor $t \cdot (t^{-1}) = 1$,
3. továbbá a két művelet együttes alkalmazására a következők igazak:
   (a) ha $s \in GF(N)$ (és $s \neq 0$), akkor $s \cdot (t + u) = s \cdot t + s \cdot u$, tehát igaz a disztributivitás,

(b) $0 \cdot t = 0$. Emiatt a nullelemnek nincsen inverze.

A Galois-test definiálásakor nem elég csak a $GF(N)$ alaphalmazt definiálni, hanem a halmazon értelmezett összeadás és szorzás értelmezése is szükséges.

Véges test például a $GF(2) = \{0,1\}$, ha az összeadásra igaz, hogy $1 + 1 = 0$, azaz ha az eredmény kivezetne $GF(2)$-ből, akkor a kettővel való osztásának a maradékát vesszük eredménynek.

Ez a módszer általánosítható kettőnél nagyobb N **prímszámokra** is: Vegyük a $GF(N) = \{0,1,\ldots N-1\}$ halmazt. Ha az összeadást a szokásos algebrai összeadásként értelmezzük (illetve a szorzást is a szokásos szorzásként) azzal a megkötéssel, hogy *ha az eredmény kivezetne $GF(N)$-ből, azaz nagyobb lenne, mint $N-1$, akkor annak az N-nel való osztása utáni* **maradéka** *lesz az eredmény*. Egy α szám β-val való osztása után keletkezett δ maradékára az

$$\alpha \equiv \delta \mod \beta$$

jelölést szokás alkalmazni (α ekvivalens δ-val moduló β-ként olvasandó). Ha egy β szám maradék nélküli osztója egy másik α-nak, azt az alábbi módon lehet jelezni:

$$\alpha \equiv 0 \mod \beta.$$

Szükséges az *ellentett* és az *inverz* elemek definiálására, a többi pont teljesülése egyszerűen ellenőrizhető.

- Vegyünk egy $t \in GF(N)$ elemet, ennek a hagyományos algebra szerinti **ellentett**je, $-t < 0$, tehát nincs a halmazunkban. Az $(N - t)$ azonban $GF(N)$ eleme, és az N-nel adott maradéka ugyanannyi, mint $-t$-nek, így ő lesz t ellentettje. Tehát az ellentett elem kereséséhez a

$$-t \equiv N - t \bmod N \tag{9.1}$$

azonosságot kell használni.

- Az **inverz** létezésének belátásához van szükség arra, hogy N prímszám legyen. Ekkor ugyanis egyik $t \in GF(N)$ elem sem osztója N-nek, tehát két elem szorzata csak akkor lehet nulla, ha az egyik elem a 0. Ebből az következik, hogy ha egy tetszőleges

$t \in GF(N)$-nel megszorozzuk $GF(N)$ minden elemét, az így kapott $t \cdot GF(N) = \{t \cdot 0, t \cdot 1, \ldots, t \cdot N\}$ halmaz minden eleme más és más.
(Ha ugyanis $s \neq u$ és $s,u \in GF(N)$ lenne, és fennállna, hogy $t \cdot s = t \cdot u$, akkor $t \cdot (s-u) = 0$ lenne, így $s - u = 0$, ami ellentmond a kezdőfeltevésnek. Lehet még $t = 0$, de az érdektelen, hiszen a nullelemnek nincsen inverze.)
Így $t \cdot GF(N) = GF(N)$, csak az elemek sorrendje cserélődhet fel, tehát valamelyik elemmel való szorzata t-nek feltétlen 1 lesz. Ez az elem lesz a t inverze. (Hogy melyik, azt ki kell próbálni.) Tehát az inverz elem kereséséhez a

$$t \cdot t^{-1} \equiv 1 \bmod N \tag{9.2}$$

egyenletet kell megoldani.

9.1. példa: Adjuk meg a $GF(5)$ véges test elemeinek az ellentettjeit és inverzeit.

Megoldás: A 0, 1, 2, 3 és 4 számok ellentettje $GF(5)$-ben:

$$\begin{aligned} -0 &\equiv (5-0) \equiv 0 \bmod 5 \\ -1 &\equiv 5-1 \equiv 4 \bmod 5 \\ -2 &\equiv 5-2 \equiv 3 \bmod 5 \\ -3 &\equiv 5-3 \equiv 2 \bmod 5 \\ -4 &\equiv 5-4 \equiv 1 \bmod 5 \end{aligned}$$

Látszik, hogy az ellentettek párban vannak: 1 ellentettje 4, és 4 ellentettje 1... Az iverzelemeket számolásához a legegyszerűbb, hogyha elkészítjük a $GF(5)$-beli szorzótáblát:

$\times$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	$6 \equiv \mathbf{1} \bmod 5$	$8 \equiv 3 \bmod 5$
3	0	3	$6 \equiv \mathbf{1} \bmod 5$	$9 \equiv 4 \bmod 5$	$12 \equiv 2 \bmod 5$
4	0	4	$8 \equiv 3 \bmod 5$	$12 \equiv 2 \bmod 5$	$16 \equiv \mathbf{1} \bmod 5$

A táblázatból elolvasható, hogy a 0 elemnek nincs inverze, az 1-é önmaga, a 2-é a 3, a 3-é a 2 a 4-é pedig szintén önmaga. Természetesen, ha csak egyetlen elem inverzét keressük, akkor nem kell az egész táblázatot megcsinálni, egyszerűbb elkezdeni az elemet beszorozni az 1-nél nagyobb $GF(N)$-beli elemekkel és vizsgálni a kapott szorzatok n-nel való osztás utáni maradékát. Ha 1-et kaptunk, megtaláltuk az elem inverzét.

Egy $t \in GF(N)$ elem **hatvány**ait is értelmezhetjük önmagával vett szorzatainak egymásutánjaként, azaz ha ismert $t^1 = t$, akkor

$$t^n = t^{n-1} \cdot t. \tag{9.3}$$

A $t \neq 0$ elem **rendjé**nek azt a legkisebb $\varrho > 0$ egész számot értjük, amelyre

$$t^\varrho = 1.$$

Az 1 elem rendje értelemszerűen 1.

Ha egy $t \neq 0$ elem rendje $N - 1$, akkor annak a nulladik és az első $N - 1$ hatványa mind különböző, azaz az összes $GF(N)$-beli elem előáll t valamelyik egész hatványaként. Az ilyen elemeket $GF(N)$ **primitívelemei**nek nevezik. Ha N prím, mindig van primitívelem.

9.2. példa: Vegyük az $N = 5$ esetet. Adjuk meg a $GF(5)$ elemeinek a hatványait és keressünk egy primitívelemet a számtesten belül.

Megoldás: Az 1-nél nagyobb elemeket, hatványaikat és rendjeiket a következő táblázat mutatja:

t	t^2	t^3	t^4	rend
2	4	3	1	4
3	4	2	1	4
4	1	4	1	2

A primitív elem itt például a 3.

9.1. Bináris Hamming-kód

A bináris kódoknál mind a $\mathbf{b} \in B^k$ üzenetek, mind pedig a $\mathbf{c} \in K$ kódszavak bináris vektorok, azaz csak nullákból és egyesekből állnak. Elegendő ekkor azt megtudnunk, hogy a vett szimbólumsorozat hányadik helyén van a hiba, a javítás ennek ismeretében könnyen elvégezhető.

Írjuk fel a kódunk paritásmátrixát a következő módon: Jelöljük $\mathbf{H}^T$ sorvektorait $\mathbf{h}_i$-vel, $i = 1,2,\ldots,n$, azaz legyen

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_n \end{pmatrix}. \tag{9.4}$$

A sorvektorok hossza $n-k$. Ha csak egy hiba fordul elő a kapott v vektorban, akkor annak a Δc hibavektora legfeljebb egy darab 1-est tartalmazhat, a többi eleme nulla lesz. Tegyük fel, hogy a j-edik elem az 1. Egy ilyen sorvektorral megszorozva a paritásmátrixot, megkapjuk $\mathbf{H}^T$ j-edik sorát, $\mathbf{h}_j$-t. Így ahhoz, hogy megkapjuk a H^T paritásmátrixot, egymás alá kell írnunk az összes lehetséges nem nulla szindrómát, azaz az egyesekből és nullákból összeállítható megfelelő hosszúságú sorvektorokat.

$\mathbf{H}$ oszlopainak száma, így az s szindrómák hossza $n-k$. Az összes lehetséges (nem nulla) szindróma száma $2^{n-k}-1$ (két elem $n-k$ helyen való ismétléses variációja, a tiszta nullából álló elem kivételével). Ez azt jelenti, hogy nem minden n kódszóhosszhoz s nem minden k üzenethosszhoz lehet bináris Hamming-kódot találni. Az első néhány lehetséges számhármast a következő táblázatban láthatjuk.

$n-k$	$n=2^{n-k}-1$	k
2	3	1
3	7	4
4	15	11
5	31	26
6	63	57

Ha szisztematikus kódot szeretnénk, akkor a paritásellenőrzési mátrix utolsó $n-k$ sorába hagyjuk az egy darab 1-est tartalmazó vektorokat, és úgy rendezzük el őket, hogy H^T alján egységmátrix legyen. A G generátormátrix ebből a (8.5) egyenlet utáni megjegyzés alapján könnyen elkészíthető. Egy $n=7$, $k=4$-es Hamming-kód paritásellenőrző mátrixa lehet például:

$$\mathbf{H}^T = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{9.5}$$

Az ehhez tartozó generátormátrix

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 0 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}. \tag{9.6}$$

Az említett megjegyzéssel összhangban a két mátrix vastag szedéssel kiemelt része egymás ellentettje kell, hogy legyen. Bináris esetben – mivel $1+1=0$, azaz az 1 ellentettje önmaga – ez tulajdonképpen azt jelenti, hogy a két mátrixrész azonos.

Nézzünk egy számpéldát.

9.3. példa: A kódunk generátor- és paritásellenőrző mátrixa legyen a (9.6), illetve (9.5) szerinti. Kódoljuk a $\mathbf{b} = (0\ 1\ 0\ 1)$ üzenetvektort.

Megoldás: A kapott kódszó:

$$\mathbf{c} = \mathbf{b} \cdot \mathbf{G} = (0\ 1\ 0\ 1) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = (0\ 1\ 0\ 1\ 1\ 0\ 1).$$

9.4. példa: Tegyük fel, hogy a csatorna a 9.3 példában előállított $\mathbf{c}$ vektor továbbításakor a második pozícióban lévő elemet elrontotta, így a vett vektor $\mathbf{v} = (0\ 0\ 0\ 1\ 1\ 0\ 1)$. A kérdés, hogy mivé dekódoljuk a $\mathbf{v}$ vett szimbólumsorozatot.

Megoldás: Számoljuk ki v szindrómáját.

$$\mathbf{s} = \mathbf{v} \cdot \mathbf{H}^T = (0\ 0\ 0\ 1\ 1\ 0\ 1) \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (1\ 1\ 0),$$

ez a H^T paritásmátrix második sora, eszerint a csatorna a második helyen rontott. A v vektort a dekódolás során a $(0\ 1\ 0\ 1\ 1\ 0\ 1)$ érvényes kódszóba javítjuk. Mivel a kód szisztematikus, a dekódolt üzenetet úgy kapjuk, hogy elhagyjuk a javított kódszó utolsó 3 elemét (a paritásszegmensét), a dekódolt üzenet tehát $(0\ 1\ 0\ 1)$ lesz.

9.5. példa: A Hamming-kódok egyetlen hiba kijavítására alkalmasak. Mit történik, ha mégis többet hibázik a csatorna? Torzuljon a 9.3 példabeli $(0\ 1\ 0\ 1\ 1\ 0\ 1)$ kódszó a második és az ötödik helyen, így a $\mathbf{v}' = (0\ 0\ 0\ 1\ 0\ 0\ 1)$ vektort próbáljuk dekódolni.

Megoldás: A szindrómája:

$$\mathbf{s} = \mathbf{v} \cdot \mathbf{H}^T = (0\,0\,0\,1\,0\,0\,1) \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (0\,1\,0),$$

ami a paritásmátrix hatodik sora, tehát a dekódolás során a hatodik bitet cseréljük ki, a kapott kódszó $(0\,0\,0\,1\,0\,1\,1)$, a dekódolt üzenet $(0\,0\,0\,1)$ lesz, ami nem az eredeti üzenet.

9.2. Nembináris Hamming-kódok

Legyen a rendelkezésre álló szimbólumok száma N. Mivel a kódábécé nem csak 0-ból és 1-ből áll, a hiba nagysága is lehet 1-nél nagyobb. Legyen a hibánk az i-edik pozícióban, Δc nagyságú. Ekkor a szindrómája, azaz a (9.4) alakú H^T paritásellenőrző mátrixszal vett szorzata $\mathbf{s} = \Delta c \cdot \mathbf{h}_i$ lesz. Ha minden $\mathbf{h}_i$ olyan, hogy az első nem nulla eleme 1, akkor a hiba $\mathbf{s}$ szindrómájának első nem nulla eleme pont a hiba nagysága, Δc lesz. A hiba pozícióját úgy határozzuk meg, hogy a szindrómát osztjuk első nem nulla elemével – tehát vesszük $\mathbf{s}/\Delta c$-t –, ami megadja $\mathbf{h}_i$-t. (A $\Delta c \in GF(N)$ számmal való osztás helyett természetesen a $(\Delta c)^{-1} \in GF(N)$ számmal szorozzuk a szindrómát.) $\mathbf{H}^T$ ismeretében i megkereshető. Így mind a hiba nagysága, mind pedig a helyzete ismert lesz.

A H^T paritásellenőrző mátrixnak tehát olyannak kell lennie, hogy tartalmazza az összes olyan $n-k$ hosszúságú sorvektort, amely nem csupa nullából áll, a $\{0,1,\ldots,N-1\}$ halmaz elemeiből építkezik, és az első nem nulla eleme 1. Adott $n-k$ számú paritásbit mellet az n kódszóhossz – azaz a fenti tulajdonságú sorvektorok száma – a következő:

$$n = \frac{N^{n-k}-1}{N-1}.$$

(Az eredményt úgy kapjuk, hogy az N elem $n-k$ helyen vett ismétléses variációjából, N^{n-k}-ból, levonunk egyet (a tiszta nullából álló vektort), így megkapjuk az összes lehetséges $n-k$ elemű, nem nulla vektor számát. Mivel az első nem nulla elem az $\{1,2,\ldots,N-1\}$ halmazból bármi lehetett, a csak ebben az elemben eltérő vektorok száma pont $N-1$. Ezek közül minket csak egy érdekel – amelyiknek 1 az első nem nulla eleme –, ezért kell

az előbbi eredményt $N-1$-gyel osztani.) Ha a fenti eredményt átrendezzük, az

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

egyenlőséget kapjuk. Ha a (7.17) gömbpakolási korlátot vagy Hamming-korlátot $r = N$, $\nu = 1$-re kiírjuk (két tag lesz az összegben) a következő egyenlőtlenséget kapjuk:

$$N^k \cdot (1 + n(N-1)) \leq N^n$$

Összevetve ezt a fenti egyenlőséggel és azzal, hogy az olyan kódokat, amelyek a Hamming-korlátot elérik, azaz az egyenlőség érvényes, perfekt kódnak nevezzük, megállapíthatjuk, hogy *a Hamming-kódok perfekt kódok*.

Példaként nézzünk egy olyan szisztematikus Hamming-kódot, amelyben $n-k = 2$, azaz a paritásszegmens hossza 2. Ekkor, mivel $\mathbf{H}^T$ utolsó két sora az *egységmátrix*nak kell, a maradék $n-2$ sort kell különböző vektorokkal feltölteni. Ezen vektorok első eleme mindenképpen 1, hiszen az első nem nulla elem 1, az egyetlen olyan vektor pedig, amelynek az első eleme 0, és csak a második 1, az utolsó sorban szerepel. Szintén az egységmátrixban szerepel az $(1\ 0)$ vektor is, tehát $\mathbf{H}^T$ felső sorainak második elemei csak $\{1,2,\ldots,N-1\}$-ből kerülhetnek ki, tetszőleges sorrendben. (Ilyen sorvektorból tehát $N-1$ van, így $N-1 = k$ és $n = k+2 = N+1$.) Ha tehát t a kódábécé *primitív elem*e, akkor egy $(N+1,N-1)$ paraméterű nembináris Hamming-kódhoz jó paritásmátrix a

$$\mathbf{H}^T = \begin{pmatrix} 1 & 1 \\ 1 & t \\ 1 & t^2 \\ \vdots & \vdots \\ 1 & t^{N-2} \\ 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{9.7}$$

A generátormátrix a bináris esethez hasonlóan állítható elő, csak az ellentett elemekre kell figyelni.

9.6. példa: Konkrétabb számokkal, ha $N = 5$ (elemei a fejezet elején, a matematikai kitérőben fel vannak sorolva, rendjükkel egyetemben), a primitív elem a 3. Készítsük el a $GF(5)$-ben értelmezett, $(6,4)$ paraméterű nembináris Hamming-kód paritásellenőrző és generátormátrixait.

Megoldás: Az ellentett elemek párokba rendezve a következők: (1,4) és (2,3). Ekkor a paritásellenőrző mátrix

$$\mathbf{H}^T = \begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 4 \\ 1 & 2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{9.8}$$

a generátormátrix pedig

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & -(1) & -(1) \\ 0 & 1 & 0 & 0 & -(1) & -(3) \\ 0 & 0 & 1 & 0 & -(1) & -(4) \\ 0 & 0 & 0 & 1 & -(1) & -(2) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 4 & 4 \\ 0 & 1 & 0 & 0 & 4 & 2 \\ 0 & 0 & 1 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 & 4 & 3 \end{pmatrix}. \tag{9.9}$$

Az első mátrix megmutatja, hogyan kaptuk meg **G**-t.

A (9.8) $\mathbf{H}^T$ mátrix sorait felcserélve szintén jó paritásmátrixot kapunk, de a generátormátrix is más lesz, nem (9.9).

9.7. példa: Kódoljuk a $\mathbf{b} = (3\ 0\ 4\ 1)$ üzenetet a (9.9) generátormátrixú (6,4) paraméterű nembináris Hamming-kóddal.

Megoldás:

$$\mathbf{c} = \mathbf{b} \cdot \mathbf{G} = (3\,0\,4\,1) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 4 & 4 \\ 0 & 1 & 0 & 0 & 4 & 2 \\ 0 & 0 & 1 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 & 4 & 3 \end{pmatrix} = (3\,0\,4\,1\,2\,4).$$

9.8. példa: Dekódoljuk a (9.8) paritásellenőrző mátrixszal rendelkező kódoló által a csatornára bocsátott kódszóból torzult $v = (3\ 0\ 2\ 1\ 2\ 4)$ $GF(5)$ feletti vektort.

Megoldás: Első lépésként kiszámítjuk a vektor szindrómáját:

$$\mathbf{s} = \mathbf{s} \cdot \mathbf{H}^T = (3\,0\,2\,1\,2\,4) \cdot \begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 4 \\ 1 & 2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = (3\ 2)$$

Felhasználtuk, hogy $8 \equiv 3 \bmod 5$, és $17 \equiv 2 \bmod 5$.

A hibanagyság tehát a szindróma első komponense, azaz $\Delta c = 3$. Elosztjuk a szindrómát a hibanagysággal:

$$\frac{\mathbf{s}}{\Delta c} = \mathbf{s} \cdot (\Delta c)^{-1} = (3\ \ 2) \cdot 2 \equiv (1\ \ 4) \bmod 5$$

Itt azt használtuk, hogy a véges testekben egy szám inverze az a véges testbeli elem, amellyel a számot megszorozva 1-et kapunk. Jelen esetben $3 \cdot 2 = 6 \equiv 1 \bmod 5$. Az így kapott vektor a H^T mátrix egyik sora kell, hogy legyen: az is, a harmadik sor. Ez azt jelenti, hogy a harmadik helyen van hiba, így a v vektor harmadik komponenséből vonunk le $\Delta c = 3$-at. A kapott kódszó:

$$\mathbf{c_v} = (3\ 0\ 2\ 1\ 2\ 4) - (0\ 0\ 3\ 0\ 0\ 0) = (3\ 0\ 4\ 1\ 2\ 4),$$

így, mivel (2 hosszúságú paritásszegmenssel rendelkező) szisztematikus kódról van szó, a dekódolt üzenet: $\mathbf{b_v} = (3\ 0\ 4\ 1)$.

10. Ciklikus kódolás

Egy $\mathbf{c} = (c_0,c_1,\ldots,c_{n-1})$ vektor ciklikus eltoltján egy olyan vektort értünk, melynek az elemei ugyanazok, mint **c**-nek, csak eggyel jobbra tolódva, c_{n-1} pedig – mivel jobbra már nem tudott tolódni – a vektor elejére került. Ha a ciklikus eltolás operátorát S-sel jelöljük, akkor matematikai formulákkal **c** ciklikus eltoltja az

$$S\mathbf{c} = (c_{n-1},c_0,c_1,\ldots,c_{n-2}). \tag{10.1}$$

Egy K kódot **ciklikus**nak nevezünk, ha minden $\mathbf{c} \in K$ kódszóra $S\mathbf{c} \in K$.

Abból, hogy egy kód ciklikus, nem következik az, hogy lineáris is, lássuk:

10.1. példa: Legyen $K = \{000,100,010,001,111\}$. Vajon lineáris-e ez a ciklikus kód?

Megoldás: Ha K lineáris, érvényesek rá a vektortér-axiómák. Ezek közül vizsgáljuk azt, hogy a K elemei közötti összeadás nem visz ki a K halmazból. Nézzük a második és harmadik elem összegét, az 110, ami nem eleme K-nak, így nem kódszó, tehát ez a kód nem lehet lineáris.

Mivel a ciklikus kódokat a legegyszerűbben a kódszavakhoz rendelt polinomok segítségével kezelhetjük, szükség lesz egy újabb matematikai kitérőre.

Matematikai kitérő – A véges testeken értelmezett polinomokról és a polinom-véges testekről. Vegyünk egy $GF(N)$ Galois-testet. A $t \in GF(N)$ elem n-edik hatványát értelmezzük a (9.3) szabály szerint. A

$$p(t) = p_0 + p_1 \cdot t + p_2 \cdot t^2 + \ldots + p_m \cdot t^m \tag{10.2}$$

kifejezést a $GF(N)$ **véges test feletti polinom**nak nevezzük, ha minden p_i, $i = 0,1,\ldots,m$ **együttható** a Galois-testünk eleme, azaz $p_i \in GF(N)$.

Egy $p(t)$ polinom egyenlő egy másik, $p'(t)$ polinommal, akkor és csak akkor, ha minden i-re $p_i = p'_i$, azaz az együtthatóik megegyeznek. A $p(t)$ polinomban szereplő legmagasabb m kitevő a **polinom fokszáma**, melyre az alábbi jelölést alkalmazzuk:

$$\deg p(t) = m.$$

A Galois-testek polinomjai között is lehet összeadást és szorzást definiálni

- A $p(t)$ és $q(t)$, $GF(N)$ Galois-test feletti polinomok $r(t) = p(t) + q(t)$ **összegé**n azt a polinomot értjük, melynek együtthatói az $r_i = p_i + q_i$ $GF(N)$-en értelmezett (moduló N) összeadás szerint állnak elő minden i-re.
- A $p(t)$ és $q(t)$, $GF(N)$ Galois-test feletti polinomok $r(t) = p(t) \cdot q(t)$ **szorzatá**n azt a polinomot értjük, melynek együtthatói az

$$r_i = \sum_{j \leq i} p_j \cdot q_{i-j} \tag{10.3}$$

  $GF(N)$-en értelmezett szorzásokból álló moduló N összegként állnak elő minden i-re.

A polinomokat úgy szorozzuk össze, hogy a $p(t) = p_0 + p_1 t + p_2 t^2 + \ldots + p_m t^m$ összeg minden tagját a $q(t) = q_0 + q_1 t + q_2 t^2 + \ldots + q_n t^n$ összegnek minden tagjával összeszorozzuk, majd rendszerezzük az így kapott tagokat fokszámuk szerint, és az azonos hatványú t-vel rendelkező tagok együtthatóit összevonjuk. Így keletkezett a fenti összeg. A két polinom fokszáma lehet különböző, és a szorzás eredményének a fokszáma meghaladhatja mindkét polinom fokszámát (m-et és n-et is), de nem lehet nagyobb $m + n$-nél.

Vegyünk egy $p(t)$ és egy $q(t)$ $GF(N)$ feletti polinomot úgy, hogy $\deg\ p > \deg\ q$. A p-nek a q-val való osztását a következőképpen lehet elvégezni:

I. Ha $\deg\ p = m$ és $\deg\ q = n$, illetve a $p(t)$-ben t^m együtthatója p_m, $q(t)$ legmagasabb fokú tagjának együtthatója pedig q_n, akkor szorozzuk meg $q(t)$-t

$$r_{m-n} t^{n-m} = \frac{p_m}{q_n} t^{n-m}\text{-nel.}$$

II. Vonjuk le az így kapott polinomot $p(t)$-ből. A különbség m-edik fokú tagjának nulla az együtthatója, tehát a különbség fokszáma legfeljebb $m - 1$. Jelöljük most a különbséget $s^{(m-1)}(t)$-vel.

III. Ismételjük meg az I. és II. lépést úgy, hogy p helyére mindig az előző lépésből maradt $s^m(t)$-t tegyük, egészen addig, amíg az új maradék fokszáma kisebb nem lesz, mint n, a $q(t)$ fokszáma.

IV. Az eredmény

$$p(t) = q(t)r(t) + s^{(n-1)}(t). \tag{10.4}$$

Nézzünk egy konkrét példát:

10.2. példa: Legyen $p(t) = 4t^5 + 2t^4 + 4t^2 + t + 3$ és $q(t) = 2t - 3$. Adjuk meg a két polinom hányadosát és maradékát.

Megoldás:

1. $\deg p = 5$ és $\deg q = 1$, így $q(t)$-t

   $$r_4 t^4 = \frac{4}{2}t^4\text{-nel}$$

   szorozzuk meg. Az eredmény $4t^5 - 6t^4$. Ezt $p(t)$-ből levonva $s^{(4)}(t) = 8t^4 + 4t^2 + t + 3$-at kapunk.

2. $s^{(4)}(t)$ fokszáma 4, a negyedfokú (legmagasabb fokszámú) tagjának együtthatója 8, így

   $$r_3 t^3 = \frac{8}{2}t^3.$$

   Ezt a kifejezést $q(t)$-vel megszorozva, majd $s^{(4)}(t)$-ből levonva $s^{(3)}(t) = 12t^3 + 4t^2 + t + 3$ marad.

3. $s^{(3)}(t)$ fokszáma 3, a köbös tagjának együtthatója 12, tehát $q(t)$-t

   $$r_2 t^2 = \frac{12}{2}t^2\text{-tel}$$

   szorozzuk. A $p(t)$-től való eltérés $s^{(2)}(t) = 22t^2 + t + 3$.

4. Ebben a lépésben

   $$r_1 t^1 = \frac{22}{2}t,$$

   a különbség pedig $s^{(1)}(t) = 34t + 3$.

5. Végül

   $$r_0 t^0 = \frac{34}{2},$$

   $q(t)$-t ezzel megszorozva $34t - 51$-et kapunk, $s^{(1)}(t) = 34t + 3$-ból ezt levonva 54 marad.

6. Az osztás eredménye:

   $$4t^5 + 2t^4 + 4t^2 + t + 3 = (2t - 3)(2t^4 + 4t^3 + 6t^2 + 11t + 17) + 54$$

Ezen megoldásnál feltettük, hogy N elég nagy prímszám (54-nél nagyobb).

10.3. példa: A 10.2 példa $GF(7)$-ben a következőképpen néz ki

Megoldás: Először is a -3 nem eleme $GF(7)$-nek, így az osztópolinom $(2t+4)$ lesz, mivel $-3 \equiv 4 \bmod 7$. Ezen kívül, a 2 inverzeleme az a szám, amellyel megszorozva 1-et kapunk, (moduló 7), tehát $2^{-1} = 4$. Ezek szerint, ahol az iménti példában 2-vel osztottunk, most 4-gyel szoroznunk kell.

$$
\begin{array}{rcl}
4t^5+2t^4+0t^3+4t^2+1t+3 & = & (2t+4)\cdot\left(2t^4+4t^3+6t^2+4t+3\right)+5. \\
\underline{-4t^5-1t^4} & & (4\cdot 2 = 8 \equiv 1 \bmod 7) \\
1t^4+0t^3+4t^2+1t+3 & & (-1+2 \equiv 1 \bmod 7) \\
\underline{-1t^4-2t^3} & & (1\cdot 2^{-1} \equiv 4; 4\cdot 4 = 16 \equiv 2 \bmod 7) \\
5t^3+4t^2+1t+3 & & (-2 \equiv 5 \bmod 7) \\
\underline{-5t^3-3t^2} & & (5\cdot 2^{-1} = 20 \equiv 6; 6\cdot 4 = 24 \equiv 3 \bmod 7) \\
1t^2+1t+3 & & (-3+4 \equiv 1 \bmod 7) \\
\underline{-1t^2-2t} & & (1\cdot 2^{-1} \equiv 4; 4\cdot 4 = 16 \equiv 2 \bmod 7) \\
6t+3 & & (-2+1 \equiv 5+1 = 6 \bmod 7) \\
\underline{-6t-5} & & (3\cdot 4 = 12 \equiv 5 \bmod 7) \\
5 & & (-5+3 \equiv 2+3 = 5 \bmod 7)
\end{array}
$$

A sorok jobb szélén zárójelben az alkalmazott mod 7 műveletek láthatók.

A számok maradékos osztásának példájára a polinomok maradékos osztását a következőképpen jelölik:

$$p(t) \equiv s^{(n-1)}(t) \mod q(t)$$

ha a (10.4) egyenlőség jelöléseit alkalmazzuk.

Egy $p(t)$ polinom **gyökei**, vagy más szóval zérushelyei azok a $t_i \in GF(N)$ véges testbeli elemek, amelyekre

$$p(t_i) = 0.$$

Egy polinom *gyökeinek a száma nem lehet nagyobb a fokszámánál*. Tudvalevő, hogy ha t_i a $p(t)$ polinom gyöke, akkor $(t - t_i)$ maradék nélküli osztója $p(t)$-nek. A polinomokat szokás gyöktényezős alakjukban is megadni. Ha t_0,t_1, . . . ,t_m egy m-edfokú $p(t)$ polinom gyökei, akkor a polinom felírható a

$$p(t) = (t - t_0) \cdot (t - t_1) \cdot \ldots \cdot (t - t_m) \tag{10.5}$$

formula szerint is. A (10.5) a $p(t)$ polinom **gyöktényezős felbontása**.

Egy $GF(N)$-en értelmezett $p(t)$ polinom **irreducíbilis**, hogy ha nem található nála kisebb fokszámú két $GF(N)$ feletti polinom, $q(t)$ és $r(t)$, amelyekre

$$p(t) = q(t) \cdot r(t)$$

azaz ha az azonosan 1 függvényen és önmagán kívül nincs semmiféle $GF(N)$ feletti polinomosztója. Ez a definíció, a $GF(N)$ polinomjai közül az irreducíbilisek kiválasztása emlékeztet a valós számok közül a *prímszámok* kiválasztására. Ugyanúgy, mint a számok között, a polinomok között is létezik összeadás és szorzás, vonjunk tehát analógiát egy $GF(N)$ test felett értelmezett, tetszőleges fokszámú polinomok és az egész számok között. Prímszámok voltak szükségesek a véges testek definíciójához, analóg módon, *irreducíbilis polinomok lesznek szükségesek az új,* ***polinom-Galois-testek*** *létrehozásához.*

Legyen $P(t)$ egy M-edfokú irreducíbilis polinom $GF(N)$ felett. A $GF(P(t))$ halmaz tartalmazza a $GF(N)$ feletti, legfeljebb $M-1$-edfokú polinomokat. A $GF(P(t))$ halmaz elemei között a következőképpen kell az összeadást és a szorzást definiálni ahhoz, hogy $GF(P(t))$ is véges test legyen, (teljesítse a Galois-testek előző kitérőben felsorolt axiómáit):

- A $q(t) \in GF(P(t))$ és $r(t) \in GF(P(t))$ összegén azt az $s(t) \in GF(P(t))$ polinomot értjük, amelyre

$$s(t) \equiv q(t) + r(t) \mod P(t).$$

- A $q(t) \in GF(P(t))$ és $r(t) \in GF(P(t))$ szorzata az az $s(t) \in GF(P(t))$ polinom, amelyre

$$s(t) \equiv q(t) \cdot r(t) \mod P(t).$$

Az egységelem, nullelem, inverz és ellentett elemek a $GF(N)$ esetből egyszerűen általánosíthatók.

10.4. példa: Legyen a $P(t) = 1 + t + t^2$ másodfokú irreducíbilis polinom a $GF(P(t))$ legfeljebb elsőfokú $GF(2)$-beli polinomokat tartalmazó polinom-Galois-test definiáló polinomja. Adjuk meg a lehetséges polinomok összeadó- és szorzótábláját.

Megoldás: Az összeadások, mivel a $GF(2)$ véges testben vagyunk:

$$\begin{aligned}
&(0t+0)+(0t+0)=0t+0 && (0t+1)+(0t+1)=0t+2\equiv 0t+0 \mod 2\\
&(0t+1)+(0t+0)=0t+1 && (1t+1)+(0t+1)=1t+2\equiv 1t+0 \mod 2\\
&(1t+0)+(0t+0)=1t+0 && (1t+0)+(1t+0)=2t+0\equiv 0t+0 \mod 2\\
&(1t+1)+(0t+0)=1t+1 && (1t+1)+(1t+0)=2t+1\equiv 0t+1 \mod 2\\
&(1t+0)+(0t+1)=1t+1 && (1t+1)+(1t+1)=2t+2\equiv 0t+0 \mod 2.
\end{aligned} \tag{10.6}$$

Ebből az összeadótábla:

	0t+0	0t+1	1t+0	1t+1
0t+0	0t+0	0t+1	1t+0	1t+1
0t+1	0t+1	0t+0	1t+1	1t+0
1t+0	1t+0	1t+1	0t+0	0t+1
1t+1	1t+1	1t+0	0t+1	0t+0

A szorzások:

$$\begin{aligned}
&(0t+0)\cdot(0t+0)=0t+0 && (0t+1)\cdot(0t+1)=0t+1\\
&(0t+1)\cdot(0t+0)=0t+0 && (0t+1)\cdot(1t+1)=1t+1\\
&(1t+0)\cdot(0t+0)=0t+0 && (1t+0)\cdot(1t+0)=t^2\\
&&& \equiv 1t+1 \bmod (1+t+t^2)\\
&(1t+1)\cdot(0t+0)=0t+0 && (1t+0)\cdot(1t+1)=t^2+t\\
&&& \equiv 0t+1 \bmod (1+t+t^2)\\
&(1t+0)\cdot(0t+1)=1t+0 && (1t+1)\cdot(1t+1)=t^2+2t+1\\
&&& \equiv t^2+0t+1 \bmod 2\\
&&& \equiv 1t+0 \bmod (1+t+t^2).
\end{aligned} \tag{10.7}$$

Az utolsó három sor második elemeihez használt $P(t)=1+t+t^2$ polinommal való maradékos osztások során felhasználtuk, hogy $GF(2)$-ben vagyunk: itt ugyanis $-1\equiv 1$. A szorzótábla:

	0t+0	0t+1	1t+0	1t+1
0t+0	0t+0	0t+0	0t+0	0t+0
0t+1	0t+0	0t+1	1t+0	1t+1
1t+0	0t+0	1t+0	1t+1	0t+1
1t+1	0t+0	1t+1	0t+1	1t+0

Az összeadótáblából látszik, hogy a $GF(P(t))$ polinom-Galois-test nulleleme a $0t + 0$ polinom, hiszen azt bármelyik $p(t) \in GF(P(t))$ polinomhoz hozzáadva $p(t)$-t ad eredményül. Hasonlóképpen, a szorzótáblából az adódik, hogy az egységelem a $0t + 1$ polinom, az, amelyikkel bármely $p(t) \in GF(P(t))$ polinomot megszorozva önmagát kapjuk. Látszanak továbbá az ellentett- és inverz polinompárok is. Például az összeadótábla megmutatja, hogy az $1t + 0$ elem ellentettje önmaga, mivel az önmagával alkotott összeg a nullelem. Hasonlóképpen, a szorzótáblából kiderül, hogy mondjuk az $1t + 0$ polinom inverze az $1t + 1$, mivel a kettő szorzata az egységelem.

10.1. Generátorpolinom és paritásellenőrző polinom

Térjünk vissza a ciklikus kódjainkra. A $\mathbf{c} \in K$ kódszavakhoz rendelhetünk polinomot az alábbi szabály szerint:

$$\begin{array}{rcl} \mathbf{c} & = & (c_0,c_1,\ldots,c_{n-1}) \\ & \Updownarrow & \\ c(t) & = & c_0 + c_1 t + c_2 t^2 + \ldots + c_{n-1} t^{n-1} \end{array} \tag{10.8}$$

A kódszavakhoz rendelt polinomok, vagy röviden kódszópolinomok halmazát jelöljük $K(t)$-vel. Ezzel a reprezentációval

a kódszavak ciklikus eltolása t-vel való $\bmod\ (t^n - 1)$ szorzássá egyszerűsödik,

tehát ha $S\mathbf{c} = \mathbf{c}'$, és a $\mathbf{c}'$ vektorhoz rendelt polinom $c'(t)$, akkor

$$c'(t) = t \cdot c(t) \mod (t^n - 1).$$

A polinomos leírással könnyű feltételt találni arra, hogy a kód *lineáris* legyen. Ha létezik egyetlen olyan nem nulla fokszámú kódszópolinom, melynek a fokszáma minimális és a legmagasabb kitevőjű tagjának az együtthatója 1, akkor a kód lineáris. Ezt a polinomot a kód **generátorpolinomjá**nak hívják, és $g(t)$-vel jelölik. A generátorpolinom fokszáma megegyezik a paritásszegmens hosszával: $n - k$-val. A $g(t)$ minden $c_i(t) \in K(t)$ kódszópolinomnak osztója, tehát minden $c_i(t) \in K(t)$ felírható, mint

$$c_i(t) = \alpha_i(t) \cdot g(t) \tag{10.9}$$

ahol az $\alpha_i(t)$ polinom együtthatói a (8.1) egyenlőségben található $\vec{\alpha}_i$ vektor komponensei: $\alpha(t) = \alpha_{i0}t^0 + \alpha_{i1}t^1 + \ldots + \alpha_{i(k-1)}t^{k-1}$.

Ha a generátorpolinomot legfeljebb n-edfokú polinomként fogjuk fel, akkor együtthatói g_0,g_1,$\ldots$,g_{n-k-1},1,0,$\ldots$,0 alakúak (a végén k db nullával). A kód generátormátrixa a generátorpolinom együtthatóiból felírható:

$$\mathbf{G} = \begin{pmatrix} g_0 & g_1 & \ldots & g_{n-k-1} & 1 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & \ldots & g_{n-k-1} & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & g_0 & g_1 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 0 & g_0 & \ldots & g_{n-k-1} & 1 \end{pmatrix} \tag{10.10}$$

A generátormátrixnak van polinom megfelelője, vajon a paritásellenőrző mátrixnak is lesz-e? A paritásellenőrző polinom megkonstruálásához nyújt segítséget a következő állítás:

A generátorpoliom osztója $(t^n - 1)$-nek.

Bizonyítás: Az állítást úgy lehet belátni, hogy vesszük $g(t)$-nek a t^{k-1}-nel, illetve t^k-nal való szorzatát (az eltoltjait) és kivonjuk őket egymásból. Mivel mindkettő maradék nélkül osztható $g(t)$-vel, a különbségük is az lesz. Mivel $t^{k-1}g(t)$ utolsó, $n-1$-edfokú tagjának az együtthatója 1, a $t^k g(t)$-nek az első, nulladrendű tagjának lesz 1 az együtthatója, az összes többi tag pedig $t^{k-1}g(t)$-nek lesz a t-szerese. Így

$$t^k g(t) = -t^n + 1 + t \cdot t^{k-1}g(t),$$

és ennek a kifejezésnek oszthatónak kell lennie $g(t)$-vel. Az utolsó tag osztható, tehát az első kettő összegének is annak kell lennie, tehát $t^n - 1$ valóban osztható lesz $g(t)$-vel. (Q.E.D.)

A $t^n - 1$ minden irreducíbilis osztópolinomja egy-egy kódnak a generátorpolinomja.

10.5. példa: A $GF(5)$ véges számtest felett a $t^6 - 1$ polinomnak a $g(t) = t^2 + 4t + 1$ polinom irreducíbilis osztója. Adjuk meg a $g(t)$ generátormpolinomú, $GF(5)$ feletti (6,4) paraméterű ciklikus kód által a $b = (2\ 3\ 0\ 1)$ üzenetből létrehozott kódszót

Megoldás: Az üzenethez rendelt polinom: $b(t) = 2 + 3t + t^3$. A kódszópolinom ennek a $g(t)$ generátorpolinommal vett szorzata a $GF(5)$ számtest felett:

$$\begin{aligned} b(t) \cdot g(t) &= \left(2 + 3t + t^3\right) \cdot \left(1 + 4t + t^2\right) \equiv \\ &\equiv 2 + 3t + 2t^2 + 3t + 2t^2 + 3t^3 + t^3 + 4t^4 + t^5 \equiv \\ &\equiv 2 + t + 4t^2 + 4t^3 + 4t^4 + t^5 \mod 5. \end{aligned}$$

A kapott kódszó: $c = (2\ 1\ 4\ 4\ 4\ 1)$.

Alternatív megoldás: a kód generátormátrixa:

$$\mathbf{G} = \begin{pmatrix} 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

A kódszót a $\mathbf{b} \cdot \mathbf{G}$ képlettel is megkephatjuk.

A $g(t)$ generátorpolinommal rendelkező n kódszóhosszú kód **paritásellenőrző polinom**ja a

$$h(t) = \frac{t^n - 1}{g(t)} \tag{10.11}$$

kifejezéssel adható meg. Ezzel lesz ugyanis

$$g(t) \cdot h(t) = t^n - 1 \equiv 0 \mod t^n - 1,$$

ami maga után vonja azt, hogy érvényes kódszavakra a szindrómapolinom $s(t) = c(t) \cdot h(t) \equiv 0 \mod t^n - 1$. Egy érvényes kódszópolinom ugyanis $c(t) = \alpha(t)g(t)$ alakú, így ha megszorozzuk a $h(t)$ paritásellenőrző polinommal,

$$c(t)h(t) = \alpha(t)g(t)h(t) = \alpha(t)(t^n - 1) \equiv 0 \mod t^n - 1.$$

10.6. példa: Adjuk meg a 10.5 példában szereplő ciklikus kód paritásellenőrző polinomját. Számoljuk ki vele a $\mathbf{v} = (4\ 2\ 0\ 0\ 3\ 1)$ vett vektor szindrómáját

Megoldás: A paritásellenőrző polinom a $t^6 - 1$ polinomnak és a generátorpolinomnak a hányadosa. A $GF(5)$ számtesten $t^6 - 1 \equiv t^6 + 4 \mod 5$, így a polinomosztás:

$$\begin{array}{rrr} 1t^6 +0t^5 +0t^4 +0t^3 +0t^2 +0t +4 & = (t^2+4t+1)(t^4+t^3+4t+4) & \\ -t^6 -4t^5 -1t^4 & & \\ \hline +1t^5 +4t^4 +0t^3 +0t^2 +0t +4 & & (-4 \equiv 1 \quad -1 \equiv 4) \\ -1t^5 -4t^4 -1t^3 & & \\ \hline +4t^3 +0t^2 +0t +4 & & (-1 \equiv 4) \\ -4t^3 -1t^2 -4t & & (4 \cdot 4 = 16 \equiv 1) \\ \hline +4t^2 +1t +4 & & \\ -4t^2 -1t -4 & & \\ \hline 0 & & \end{array}$$

A sorok mellet zárójelben a felhasznált mod 5 műveletek láthatók. Tehát $h(t) = 4 + 4t + t^3 + t^4$.

A $\mathbf{v}$ vektorhoz rendelhető polinom

$$v(t) = 4 + 2t + 3t^4 + t^5.$$

A szindrómapolinom ennek a $h(t)$ polinommal vett szorzata. A szorzás során figyelembe kell venni nem csak azt, hogy $GF(5)$ számtest felett vagyunk, hanem azt is, hogy a ciklikus kódoknál minden polinomot modulo $(t^n - 1)$ kell értelmezni. A szorzás eredménye:

$$\begin{aligned} v(t) \cdot h(t) &= \left(4 + 2t + 3t^4 + t^5\right) \cdot \left(4 + 4t + t^3 + t^4\right) = \\ &= 1 + t + 4t^3 + 4t^4 + 3t + 3t^2 + 2t^4 + 2t^5 + \\ &\quad + 2t^4 + 2t^5 + 3t^7 + 3t^8 + 4t^5 + 4t^6 + t^8 + t^9 \equiv \\ &\equiv 1 + 4t + 3t^2 + 4t^3 + 3t^4 + 3t^5 + 4t^6 + 3t^7 + 4t^8 + t^9 \bmod 5 \end{aligned}$$

Ennek a polinomnak kell a $t^6 - 1 \equiv t^6 + 4$ polinommal való osztása utáni maradékát venni:

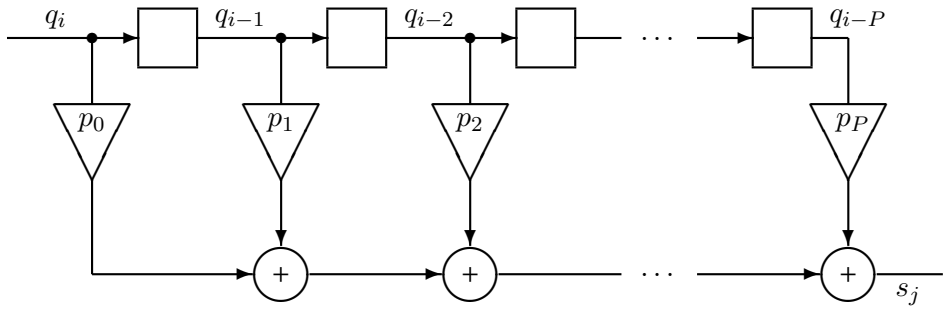
$$\begin{array}{rrrrrrrrrl} t^9 & +4t^8 & +3t^7 & +4t^6 & +3t^5 & +3t^4 & +4t^3 & +3t^2 & +4t & +1 = (t^6+4)(t^3+4t^2+3t+4) \\ -t^9 & -0t^8 & -0t^7 & -0t^6 & -0t^5 & -1t^4 & -4t^3 & & & \\ \hline & 4t^8 & +3t^7 & +4t^6 & +3t^5 & +3t^4 & +0t^3 & +3t^2 & +4t & +1 \\ & -4t^8 & -0t^7 & -0t^6 & -0t^5 & -1t^4 & -0t^3 & -4t^2 & & \\ \hline & & 3t^7 & +4t^6 & +3t^5 & +3t^4 & +0t^3 & +4t^2 & +4t & +1 \\ & & -3t^7 & -0t^6 & -0t^5 & -1t^4 & -0t^3 & -0t^2 & -3t & \\ \hline & & & 4t^6 & +3t^5 & +3t^4 & +0t^3 & +4t^2 & +1t & +1 \\ & & & -4t^6 & -0t^5 & -1t^4 & -0t^3 & -0t^2 & -0t & -4 \\ \hline & & & & 3t^5 & +3t^4 & +0t^3 & +4t^2 & +4t & +2 \end{array}$$

A szindrómapolinom tehát: $s(t) = 2 + 4t + 4t^2 + 3t^4 + 3t^5$, azaz $\mathbf{v}$ nem kódszó.

10.2. Ciklikus kódok a gyakorlatban

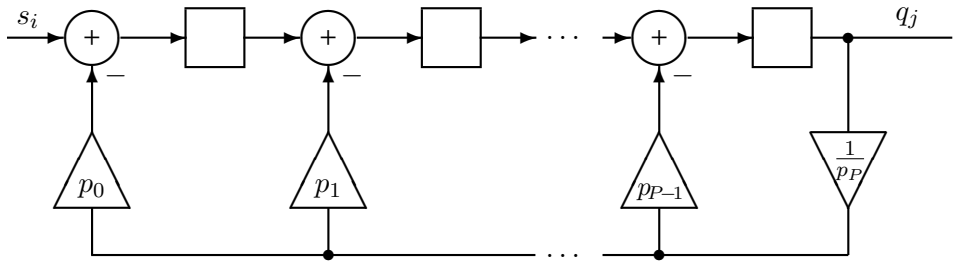
A szabványokban a ciklikus kódokat *generátorpolinomjukkal* szokták megadni.

Azért szeretik a polinomos megadást, mert a polinomok szorzása és osztása áramkörökkel egyszerűen megoldható. A következő ábra azt mutatja, hogy egy $p(t) = p_0 + p_1 + p_2t^2 + \ldots p_Pt^P$ polinomnak a $q(t) = q_0 + q_1 + q_2t^2 + \ldots q_Qt^Q$-nal vett **szorzat**át, $s(t) = s_0 + s_1 + s_2t^2 + \ldots s_{P+Q}t^{P+Q}$-t, illetve annak együtthatóit milyen áramkörrel tudjuk létrehozni. A háromszögek erősítők, vagy konstansszoros szorzók, a négyzetek tárolók, a körök +-jellel pedig összeadók.



Tegyük fel, hogy kezdetben minden tároló 0-n áll, majd adjuk q_0-t az áramkörre. A kimeneten megjelenik p_0q_0, az $s(t)$ nulladrendű együtthatója. A következő lépésben az első tárolón q_0 van, a bemeneten q_1, a kimeneten $q_0p_1 + q_1p_0$, ami pont $s(t)$ elsőrendű együtthatója, és így tovább. Ha az u_i-k elfogytak, 0-kat kell a bemenetre adni addig, amíg minden tároló újra ki nem ürül.

A polinomok **osztás**át az alábbi visszacsatolt léptetőregiszteres áramkörrel lehet megvalósítani:



Ha fokszám szerint csökkenő sorrendben beadjuk az osztani kívánt $s(t)$ polinom együtthatóit, akkor a kimeneten a $q(t) = s(t)/p(t)$ hányadospolinom együtthatói jelennek meg, a tárolókban pedig a maradék.

Ciklikus kódolásra és dekódolásra kész integrált áramkörök kaphatók különböző fokszámú generátorpolinomokkal. Mind a kódoláshoz, mind a paritásellenőrzéshez alkalmazhatók lineárisan előrecsatolt, illetve lineárisan visszacsatolt léptetőregiszterek.

A műholdas műsorszórás során alkalmazott (32,12) paraméterű *Golay-kód* is ciklikus kód, generátorpolinomja

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

E kód minimális súlya 7, és maximálisan 3 egyszerű hibát tud javítani.

Az úgynevezett **CRC-kódok** (Cyclic Redundancy Check) szintén ciklikus kódok. Csak hibajelzésre alkalmasak, általában egy igen nagy méretű (tipikusan több ezres, több tízezres nagyságrendű elemszámú) blokkhoz egy viszonylag kicsi (néhány tíz elemből álló) paritásszegmenst tesznek. Mivel csak hibajelzésre alkalmasak, két fő felhasználási lehetőségük van. Alkalmazzák egy másik, hibajavító kóddal kombinálva, hogy annak biztonságát növeljék viszonylag kis kódsebesség csökkenés árán. Használják zajos visszacsatolt csatornákon való üzenetátvitel során is, ahol a hiba detektálása után automatikusan az üzenet megismétlését kérik az adótól. A CRC-kódok alkalmasak hibacsomók detektálására, illetve ha a CRC-kódolás után egy hibajavító kódolást is elvégzünk, akkor az a hibás dekódolást is nagy valószínűséggel felfedi.

11. Reed–Solomon-kódok

Legyen t_0,t_1,...,t_{n-1} a $GF(N)$ véges test $n < N$ darab, különböző eleme, továbbá $\mathbf{b} = (b_0,b_1,\ldots,b_{k-1})$ egy továbbítandó üzenet. Rendeljük az üzenethez a

$$b(t) = b_0 + b_1 t + \ldots + b_{k-1}t^{k-1} \tag{11.1}$$

$GF(N)$ feletti polinomot. A Reed–Solomon-kód a $\mathbf{b}$ üzenethez azt a $\mathbf{c} = (c_0,c_1,\ldots,c_{n-1})$ kódszóvektort rendeli hozzá, amelynek c_i komponenseit a

$$c_i = b(t_i) \tag{11.2}$$

képlet határozza meg. Látható, hogy a t_i elemek kiválasztása adja meg a kódot.

11.1. példa: A $GF(5)$ feletti Reed–Solomon-kódunk generáló elemei legyenek a $t_0 = 1$, $t_1 = 4$, $t_2 = 3$ és a $t_3 = 2$. Adjuk meg a $\mathbf{b} = (1\ 2\ 4)$ üzenetből kapott kódszót.

Megoldás: Az üzenethez rendelhető polinom a $b(t) = 1 + 2t + 4t^2$. A kódoló ebbe a polinomba helyettesíti be rendre a t_0, t_1, t_2 és t_3 számokat, hogy megkapja az $n = 4$ elemű kódszó nulladik, első, második és harmadik komponensét:

$$b(t_0) = 1 + 2\cdot 1 + 4\cdot 1^2 = 7 \equiv 2 \bmod 5 \tag{11.3}$$
$$b(t_1) = 1 + 2\cdot 4 + 4\cdot 4^2 = 73 \equiv 3 \bmod 5 \tag{11.4}$$
$$b(t_2) = 1 + 2\cdot 3 + 4\cdot 3^2 = 43 \equiv 3 \bmod 5 \tag{11.5}$$
$$b(t_3) = 1 + 2\cdot 2 + 4\cdot 2^2 = 21 \equiv 1 \bmod 5 \tag{11.6}$$

A kapott kódszó tehát $\mathbf{c} = (2\ 3\ 3\ 1)$.

A Reed–Solomon-kódok lineárisak, maximális távolságúak (MDS-ek).

Bizonyítás: Azt szeretnénk belátni, hogy a Reed–Solomon-kódokra teljesül, hogy $d_{\min} = n - k + 1$. A (7.15) Singleton-korlát szerint

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

így azt kell belátnunk, hogy a $\mathbf{c} \neq \mathbf{0}$ kódszavakra igaz, hogy

$$d_{\min} = \min_{\mathbf{c}} w(\mathbf{c}) \geq n - k + 1.$$

Minden kódszó súlya a nem nulla komponenseinek a száma. A kódszavak komponenseit viszont úgy kaptuk, hogy a megfelelő t_i számokat helyettesítettük be az üzenetpolinomba. A csupa nulla együtthatós polinom kivételével a $b(t_i)$ eredménye csak akkor lesz 0, ha t_i a $b(t)$ polinom

gyöke. Mivel az üzenet k komponensű, a hozzárendelt polinom legfeljebb $k-1$-edfokú, egy $k-1$-edfokú polinomnak viszont legfeljebb $k-1$ gyöke van. Az n darab különböző t_i számból tehát legfeljebb $k-1$ adhat 0-t az üzenetpolinomba behelyettesítve, így a kódszónak legfeljebb $k-1$ nulla komponense lesz, a maradék, legalább $n-(k-1)$, komponens nem lesz 0. Így

$$w(c) \geq n-k+1.$$

(Q.E.D.)

Mivel a Reed–Solomon-kódok MDS-ek, $n-k$ hiba jelzésére képesek, legfeljebb $(n-k)/2$ egyszerű és $n-k$ törléses hibát tudnak javítani.

11.1. A generátormátrix és a paritásellenőrző mátrix

A $GF(N)$ véges test feletti, t_0,t_1,...,t_{n-1} generáló elemekkel rendelkező Reed–Solomon-kódok generátormátrixa

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ t_0 & t_1 & \dots & t_{n-1} \\ t_0^2 & t_1^2 & \dots & t_{n-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ t_0^{k-1} & t_1^{k-1} & \dots & t_{n-1}^{k-1} \end{pmatrix} \tag{11.7}$$

11.2. példa: Készítsük el a 11.1 példában megadott (4,3) paraméterű Reed–Solomon-kód generátormátrixát. A generátormátrix használatával adjuk meg a $\mathbf{b} = (1\ 2\ 4)$ üzenetből kapott kódszót.

Megoldás: Mivel az üzenetszegmens hossza $k=3$, a mátrix 3 soros lesz, és mivel $n=4$, négy oszlopos. Ahhoz, hogy megadhassuk a $\mathbf{G}$ mátrixot, szükség van a t_0, t_1, t_2 és t_3 elemek első két hatványára. A t_0 minden hatványa 1, a $t_1=4$ négyzete $16 \equiv 1 \bmod 5$; a három második hatványa $GF(5)$-ben 4; a 2-é pedig szintén 4. A generátormátrix tehát a következő:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 3 & 2 \\ 1 & 1 & 4 & 4 \end{pmatrix}.$$

A kódszó

$$\mathbf{b} \cdot \mathbf{G} = (1\ 2\ 4) \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 3 & 2 \\ 1 & 1 & 4 & 4 \end{pmatrix} \equiv (2\ 3\ 3\ 1) \mod 5.$$

Ez azonos a 11.1 példában kapott eredménnyel.

11.2. Egyetlen generálóelemmel definiált Reed–Solomon-kód

Lássuk, hogyan érdemes a t_i kódot meghatározó elemeket megválasztani. Legyen $\vartheta \in GF(N)$, nem nulla, és a rendje legyen legalább n. (Mivel $n < N$, a $GF(N)$ egy primitíveleme mindig jó választás). Ekkor $\vartheta^0 = 1, \vartheta^1, \vartheta^2, \ldots, \vartheta^{n-1}$ mind különböző $GF(N)$ beli szám lesz, tehát a $t_i = \vartheta^i$ választás jó. A $\vartheta^0, \vartheta^1, \ldots, \vartheta^{n-1}$ Galois-testbeli elemek által generált Reed–Solomon-kód **generátormátrixa**

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \vartheta & \vartheta^2 & \ldots & \vartheta^{n-1} \\ 1 & \vartheta^2 & \vartheta^4 & \ldots & \vartheta^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \vartheta^{k-1} & \vartheta^{2(k-1)} & \ldots & \vartheta^{(n-1)(k-1)} \end{pmatrix}. \tag{11.8}$$

A kód **paritásellenőrző mátrixa**

$$\mathbf{H}^T = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \vartheta & \vartheta^2 & \ldots & \vartheta^{n-k} \\ \vartheta^2 & \vartheta^4 & \ldots & \vartheta^{2(n-k)} \\ \vdots & \vdots & \ddots & \vdots \\ \vartheta^{n-1} & \vartheta^{2(n-1)} & \ldots & \vartheta^{(n-1)(n-k)} \end{pmatrix}. \tag{11.9}$$

A fenti konstrukció tulajdonképpen azt jelenti, hogy a (11.2) képlet

$$c_i = b\left(\vartheta^i\right) = \sum_{j=0}^{k-1} b_j \left(\vartheta^i\right)^j \tag{11.10}$$

alakra módosul.

11.3. példa: A $GF(7)$ Galois-testnek a 3 hatodrendű eleme. Adjuk meg a $\vartheta = 3$ generálóelemű $(6,4)$ paraméterű Reed–Solomon-kód t_i generálóit. Számoljuk ki a $b(t) = 1 + 5t + 4t^3$ üzenetpolinomból létrehozott kódszót.

Megoldás: A 3 első hat hatványa különböző lesz, mivel a 3 hatodrendű elem $GF(7)$-ben. Így a kódot generáló hat szám:

$$\begin{aligned} t_0 &= \vartheta^0 = 3^0 = 1 \\ t_1 &= \vartheta^1 = 3^1 = 3 \\ t_2 &= \vartheta^2 = 3^2 = 9 \equiv 2 \mod 7 \\ t_3 &= \vartheta^3 = 3^3 = 27 \equiv 6 \mod 7 \\ t_4 &= \vartheta^4 = 3^4 = 81 \equiv 4 \mod 7 \\ t_5 &= \vartheta^5 = 3^5 = 243 \equiv 5 \mod 7 \end{aligned}$$

A $b(t)$ üzenetpolinomba behelyettesítve ezeket a számokat megkapjuk a kódszót:

$$\begin{aligned} b(\vartheta^0) &= 1+5\cdot 1+4\cdot 1^3 = 10 \equiv 3 \mod 7, \\ b(\vartheta^1) &= 1+5\cdot 3+4\cdot 3^3 = 124 \equiv 5 \mod 7, \\ b(\vartheta^2) &= 1+5\cdot 2+4\cdot 2^3 = 43 \equiv 1 \mod 7, \\ b(\vartheta^3) &= 1+5\cdot 6+4\cdot 6^3 = 895 \equiv 6 \mod 7, \\ b(\vartheta^4) &= 1+5\cdot 4+4\cdot 4^3 = 277 \equiv 4 \mod 7, \\ b(\vartheta^5) &= 1+5\cdot 5+4\cdot 5^3 = 526 \equiv 1 \mod 7, \end{aligned}$$

így $\mathbf{c} = (3\ 5\ 1\ 6\ 4\ 1)$.

11.4. példa: Készítsük el a 11.3 példabeli Reed–Solomon-kód generátormátrixát és paritásellenőrző mátrixát.

Megoldás: A generátormátrix a (11.8) képlet szerint:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & 6 & 4 & 5 \\ 1 & 2 & 4 & 1 & 2 & 4 \\ 1 & 6 & 1 & 6 & 1 & 6 \end{pmatrix}$$

Itt felhasználtuk a következő moduló 7 ekvivalenciákat: $9 \equiv 2$, $27 \equiv 6$, $8 \equiv 1$, $36 \equiv 1$, $216 \equiv 6$, $16 \equiv 2$, $64 \equiv 1$, $25 \equiv 4$ és $125 \equiv 6$.

A paritásmátrix a (11.9) képlet alapján, mivel $n-k = 6-4 = 2$, két oszlopos és $n = 6$ soros lesz:

$$H = \begin{pmatrix} 1 & 1 \\ 3 & 2 \\ 2 & 4 \\ 6 & 1 \\ 4 & 2 \\ 5 & 4 \end{pmatrix}. \tag{11.11}$$

Leellenőrizhetjük, hogy a két mátrix szorzata $GF(7)$-ben egy 4 soros, 2 oszlopos nullmátrix.

11.3. A Reed–Solomon-kód, mint ciklikus kód

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

$$c(t) = c_0 + c_1 t + c_2 t^2 + \ldots + c_{n-1} t^{n-1} = \sum_{i=0}^{n-1} c_i \cdot t^i$$

polinomot. Helyettesítsük be a kapott kódszópolinomba a ϑ generáló elem első $n-k$ hatványát, az elsőtől kezdve:

$$c\left(\vartheta^l\right) = \sum_{i=0}^{n-1} c_i \cdot \left(\vartheta^l\right)^i$$

Ha behelyettesítjük a c_i együtthatók (11.10) definíciós összefüggését, az előbbi egyenlőség a következő alakot ölti:

$$c\left(\vartheta^l\right) = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{k-1} b_j \vartheta^{ij} \right) \cdot \left(\vartheta^l\right)^i = \sum_{i=0}^{n-1} \sum_{j=0}^{k-1} b_j \vartheta^{i(j+l)}. \tag{11.12}$$

Mivel a j index 0 és $k-1$ között van, l pedig 1 és $n-k$ között, a $j+l$-re mindenképpen igaz lesz, hogy $1 \leq j+l \leq n-1$. Eszerint, mivel ϑ-nak csak a nulladik és az n-edik hatványa lehet 1, a $\vartheta^{j+l} \neq 1$. a (11.12) így a következő alakúra írható át:

$$c\left(\vartheta^l\right) = \sum_{j=0}^{k-1} b_j \left(\sum_{i=0}^{n-1} \left(\vartheta^{(j+l)}\right)^i \right),$$

ahol a nagy zárójelben szereplő kifejezés tulajdonképpen egy mértani sor első n elemének az összege, csak a mértani sor $GF(N)$-en van értelmezve. A mértani sor n-edik részösszegének a képlete véges számtesteken is érvényes (feltéve, hogy a sor kvóciense nem 1), így

$$\sum_{i=0}^{n-1} \left(\vartheta^{(j+l)}\right)^i = \frac{\left(\vartheta^{(j+l)}\right)^n - 1}{\vartheta^{(j+l)} - 1},$$

ami nullát ad eredményül, mivel $\vartheta^n = 1$. Visszahelyettesítve az eredményt (11.12)-be, a Reed–Solomon-kód **paritásegyenlet**eit kapjuk:

$$c\left(\vartheta^l\right) = 0 \qquad \text{ha } 1 \leq l \leq n-k. \tag{11.13}$$

A paritásegyenletek szerint a ϑ generátorelem első $n-k$ hatványa minden kódszópolinomnak a gyöke.

Így létezik egy olyan polinom, amely minden kódszónak osztója: az a polinom, melynek a gyöktényezői a $(t-\vartheta^l)$-ek, $l = 1,2,\ldots,n-k$-ra.

A szóban forgó Reed–Solomon-kód tehát egyben egy a $GF(N)$ véges test feletti (n,k) paraméterű **ciklikus kód** is, melynek a generátorpolinomja (gyöktényezős felbontásban)

$$g(t) = (t-\vartheta) \cdot (t-\vartheta^2) \cdot \ldots \cdot (t-\vartheta^{n-k}),$$

Belátható, hogy a $t^n - 1$ polinomnak az összes $(t - \vartheta^i)$ polinom a osztója, így a kód paritásellenőrző polinomja előáll

$$t^n - 1 = \prod_{i=0}^{n-1} (t - \vartheta^i)$$

polinom $g(t)$-ben fel nem használt gyöktényezőkből, azaz

$$h(t) = (t - \vartheta^{n-k+1}) \cdot (t - \vartheta^{n-k+2}) \cdot \ldots \cdot (t - \vartheta^n).$$

11.5. példa: Adjuk meg a 11.3 példában szereplő $GF(7)$ feletti, $(6,4)$ paraméterű Reed–Solomon-kód generátorpolinomját és paritásellenőrző polinomját.

Megoldás: A kódot generáló $\vartheta = 3$ szám első 6 hatványa 3, 2, 6, 4, 5 és 1 voltak. A generátorpolinom a $\vartheta = 3$ generáló elem első $n - k = 2$ hatványát tartalmazza gyöktényezőként:

$$g(t) = (t - 3) \cdot (t - 2).$$

A 3 további négy darab hatványa a paritásellenőrző polinomot alkotja:

$$h(t) = (t - 6) \cdot (t - 4) \cdot (t - 5) \cdot (t - 1).$$

A különböző szabványokban a Reed–Solomon-kódoknak is a generátorpolinomját szokták megadni, többnyire gyöktényezős alakban.

11.4. Reed–Solomon-kódok nem prím elemszámú véges testeken

Matematikai kitérő – A nem prím elemszámú véges testekről. Igen érdekes következménye a polinom-Galois-testek létezésének az, hogy az egész számok körében lehet $GF(N^M)$ típusú véges testeket is definiálni, azaz nemcsak prím elemszámú, hanem *egy prím egész hatványaival megegyező elemszámú Galois-testeket is létre lehet hozni*. A gyakorlatban a számítástechnikában a *bit* egységen kívül a *bájt* is használatos, ami 2 egész hatványa (2^8), így a véges testeknek ez az általánosítása igen hasznos.

Egy $\{0,1,\ldots,N^M - 1\}$ halmazhoz tudunk úgy összeadást és szorzást definiálni, hogy véges testet alkosson. Minden $p \in \{0,1,\ldots,N^M - 1\}$ elemhez rendeljünk hozzá egy-egyértelműen egy $p(t)$ $GF(N)$ feletti, legfeljebb $M - 1$-edfokú polinomot. Ha a $p \in \{0,1,\ldots,N^M - 1\}$ számokat úgy adjuk és szorozzuk össze, mint a hozzájuk rendelt

polinomokat, akkor $\{0,1,\ldots,N^M-1\} = GF(N^M)$, véges test, vagy Galois-test lesz. Ez a hozzárendelés egy-egyértelmű megfeleltetést jelent $GF(N^M)$ és $GF(P(t))$ között, ha $P(t)$ $GF(N)$-nek egy M-edfokú irreducíbilis polinomja.

Például a $GF(2^8)$ testhez lehet $P(t) = t^8 + t^4 + t^3 + t^2 + 1$. Az egyes $p \in GF(2^8)$ elemekhez a következőképen rendelhetjük a $GF(t^8 + t^4 + t^3 + t^2 + 1)$-beli polinomokat: Vegyük p kettes számrendszerbeli alakját

$$p = p_8 \cdot 2^8 + p_7 \cdot 2^7 + \ldots + p_0 \cdot 2^0.$$

A p-hez rendelt $p(t)$ polinom kézenfekvő módon a

$$p(t) = p_8 \cdot t^8 + p_7 \cdot t^7 + \ldots + p_0 \cdot t^0. \tag{11.14}$$

11.6. példa: Készítsük el a $GF(2^2) = GF(4)$ véges testet.

Megoldás: Ha a (11.14) hozzárendelési szabályhoz hasonlót használunk, akkor a 0, 1, 2 és 3 számokhoz a következő polinomok fognak tartozni (az egyenlőségjel után a számok bináris alakja található):

$$\begin{array}{lcl} 0 = 0\,0 & \Leftrightarrow & 0t + 0 \\ 1 = 0\,1 & \Leftrightarrow & 0t + 1 \\ 2 = 1\,0 & \Leftrightarrow & 1t + 0 \\ 3 = 1\,1 & \Leftrightarrow & 1t + 1 \end{array}$$

Ezek pont a 10.4. példában szereplő polinomok, így tudjuk a szorzó- és összeadótáblájukat. Behelyettesítve a 10.4. példa táblázataiba a polinomoknak megfelelő számokat, a következő összeadó, illetve szorzótáblát kapjuk:

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Eszerint például $1 + 3 = 2$, vagy $2 \cdot 3 = 1$, ami eléggé ellentmond a hagyományos elképzeléseknek, de még a $GF(N)$ típusú Galois-testekből örökölt elképzeléseinknek is.

Természetesen nem csak a (11.14) hozzárendelési szabály alkalmazható, a lényeg az, hogy egy-egyértelmű szabály legyen.

A nem prím elemszámú véges testeken is lehet hatványozást értelmezni, a sima $GF(N)$ esethez hasonlóan: Egy $t \in GF(N^M)$

elem **hatvány**ait önmagával vett szorzatainak egymásutánjaként értelmezhetjük, azaz ha ismert $t^1 = t$, akkor

$$t^n = t^{n-1} \cdot t. \tag{11.15}$$

A szorzás természetesen a véges test szorzótáblájának megfelelően történik. A $t \neq 0$ elem **rendjé**nek itt is azt a legkisebb $\varrho > 0$ egész számot értjük, amelyre

$$t^\varrho = 1.$$

Az egységelem rendje 1, a nullelemnek továbbra sincsen rendje.

11.7. példa: Adjuk meg a $GF(4)$ véges test elemeinek a hatványait és az elemek rendjét. Használjuk a 11.6 példa szorzótábláját.

Megoldás: Az 1 elem minden hatványa 1, rendje így 1 lesz. A 2 szám első hatványa 2, a második a szorzótábla szerint

$$2^2 = 2 \cdot 2 = 3,$$

a harmadik pedig

$$2^3 = 2^2 \cdot 2 = 3 \cdot 2 = 1.$$

A 2 tehát harmadrendű elem $GF(4)$-ben. A 3 hatványai:

$$\begin{aligned} 3^1 &= 3 \\ 3^2 &= 3 \cdot 3 = 2 \\ 3^3 &= 3^2 \cdot 3 = 2 \cdot 3 = 1, \end{aligned}$$

tehát a 3 is harmadrendű.

A hatványozás ismeretében a $GF(N^M)$ feletti polinomok definiálása egyértelmű. A polinomok összeadásánál és szorzásánál az együtthatókon végzett műveletek esetén a véges test összeadó- és szorzótábláját kell használni, nem a $GF(N)$ műveleteket.

Egy $\vartheta \in GF(N^M)$ n-edrendű elemmel lehet Reed–Solomon-kódokat definiálni nem prím elemszámú véges testekre is, hasonlóan a prím elemszámú véges testekhez. Az összes korábban felírt állítás igaz marad, azzal a kitétellel, hogy az összeadás és a szorzás nem modulo N művelet lesz, hanem a $GF(N^M)$ test összeadó-, illetve szorzótáblájának megfelelően kell őket elvégezni.

12. A Reed–Solomon-kódok spektruma és dekódolása

12.1. A Reed–Solomon-kódok spektruma

Matematikai kitérő – A véges testeken értelmezett Fourier-transzformációról és a ciklikus konvolúcióról. A spektrum kiszámolásához szükség van a Fourier-transzformációnak az n darab $GF(N^M)$-beli elemből álló $\mathbf{c}$ vektorokra való kiterjesztésére. A Fourier-transzformálás eredménye egy szintén n elemű $\mathbf{C}$ vektor, amely szintén a $GF(N^M)$ test elemeiből épül fel.

Egy $(c_0,c_1,\ldots,c_{n-1})$ komponensű $\mathbf{c} \in GF(N^M)^n$ vektor **Fourier-transzformáltja** egy olyan $\mathbf{C}=(C_0,C_1,\ldots,C_{n-1})$ vektor, melyre

$$C_j = \sum_{i=0}^{n-1} c_i \vartheta^{ij}, \tag{12.1}$$

ha ϑ a $GF(N^M)$ véges test n-edrendű eleme. (Azaz, ha $\vartheta^n = 1$.) A $\mathbf{C}$ vektor a $\mathbf{c}$-nek a **spektrum**a. A $\mathbf{c}$ Fourier-transzformáltját szokás $\mathcal{F}\{\mathbf{c}\}$-ként írni.

A véges testeken értelmezett Fourier-transzformáció is invertálható, azaz a $\mathbf{C}$ ismeretében $\mathbf{c}$ egyértelműen megadható, méghozzá a

$$c_i = f(n) \sum_{j=0}^{n-1} C_j \vartheta^{-ij} \tag{12.2}$$

formulával, ahol az $f(n)$ szám az n-nek a véges testen belüli (moduló N) inverzét jelöli. Ha $N = 2$, akkor természetesen $f(N) = 1$.

Legyen két, egyenként n darab $GF(N^M)$-beli elemből álló vektorunk, $\mathbf{t}$ és $\mathbf{u}$. Ekkor a két vektor **ciklikus konvolúciój**án azt az $\mathbf{s} \in GF(N^M)^n$ vektort értjük, amelynek komponensei az

$$s_j = f(n) \sum_{k=0}^{n-1} t_{(j-k) \bmod n} \cdot u_k \tag{12.3}$$

képlettel állíthatók elő. A ciklikus konvolúció jelölése hasonló a hagyományos, nem véges testeken vett konvolúcióéhoz: $\mathbf{s} = \mathbf{t} * \mathbf{u}$.

A véges testeken is érvényes a **konvolúciós tétel**, amely a következőt állítja: Legyen $\mathbf{s}$, $\mathbf{t}$, $\mathbf{u} \in GF(N^M)^n$ vektorok spektruma rendre $\mathbf{S}$, $\mathbf{T}$ és

U. Ha a vektorok komponenseire fennáll, hogy

$$s_i = t_i \cdot u_i \tag{12.4}$$

akkor a spektrumaik között az

$$\mathbf{S} = \mathbf{T} * \mathbf{U} \tag{12.5}$$

összefüggés lesz érvényben.
Bizonyítás: Az s a Fourier-transzformáltja a (12.1) definíció szerint:

$$S_j = \sum_{i=0}^{n-1} s_i \vartheta^{ij} =$$

a (12.4) képletet behelyettesítve

$$\begin{aligned} &= \sum_{i=0}^{n-1} t_i \cdot u_i \vartheta^{ij} = \\ &= \sum_{i=0}^{n-1} t_i \cdot \left(f(n) \sum_{k=0}^{n-1} U_k \vartheta^{-ik} \right) \vartheta^{ij} = \\ &= f(n) \sum_{k=0}^{n-1} U_k \cdot \left(\sum_{i=0}^{n-1} t_i \vartheta^{i(j-k)} \right). \end{aligned}$$

Az utolsó sorban a nagy zárójelben szereplő kifejezés lényegében **t** Fourier-transzformáltjának $(j-k)$-adik komponense. Természetesen előfordulhat, hogy $j-k<0$, ekkor helyette az n-nel adott maradékát kell használni, $(j-k) \bmod n$-t. (Q.E.D.)

12.1.1. A spektrum jellemzői

Ahogy a $\mathbf{c} \in K$ kódszóvektorhoz lehetett rendelni polinomot, a $\mathcal{F}\{\mathbf{c}\} = \mathbf{C} = (C_0, C_1, \ldots, C_{n-1})$ spektrumához is lehet egy polinomot rendelni a

$$C(t) = C_0 + C_1 \cdot t + C_2 \cdot t^2 + \ldots + C_{n-1} \cdot t^{n-1} \tag{12.6}$$

képlettel. A $C(t)$ polinomot a $c(t)$-hez rendelt **spektrumpolinom**nak nevezzük.

Legyen a Reed–Solomon-kódunk generátorpolinomja $g(t)$, a kódolni kívánt tömörített üzenetünk $b = (b_0, b_1, \ldots, b_{k-1})$, a hozzá rendelt polinom $b(t) = b_0 + b_1 \cdot t + \ldots + b_{k-1} t^{k-1}$. Vegyük észre, hogy ha a $c(t)$ polinomjaink

$(n-1)$-edfokúak, s ezért az $(n-1)$-edfokú polinomok között szeretnénk dolgozni. A $b(t)$ együtthatói, ha $b(t)$-t $n-1$-edfokú polinomként kezeljük, a következők lesznek: *0-tól* $(k-1)$*-ig a* $\mathbf{b}$ *vektor komponensei,* k*-tól* $(n-1)$*-ig pedig csupa 0.*

A $c(t)$ kódszópolinom a $g(t) \cdot b(t)$ képlettel állítható elő. Vizsgáljuk meg a polinomszorzás (10.3) definíciójában szereplő együtthatókat. Ezt alkalmazva a $c(t)$ együtthatóira

$$c_i = \sum_{j=0}^{n-1} g_{i-j} b_j,$$

ami pont a $\mathbf{g}$ és a nullákkal kibővített $\mathbf{b}$ (12.3) szerinti konvolúciójának felel meg.

A $g(t)$ generátorpolinomú Reed–Solomon-kóddal a $\mathbf{b}$ üzenetből létrehozott $\mathbf{c}$ kódszó tehát

$$\mathbf{c} = \mathbf{g} * \mathbf{b}.$$

A kódszó spektrumára ekkor a konvolúciós tétel szerint igaz, hogy

$$C_j = G_j \cdot B_j,$$

ha B_j az üzenet Fourier-transzformáltjának j-edik komponense.

A $c(t)$ polinomnak akkor és csak akkor lehet gyöke ϑ^i, ha a $C(t)$ spektrumpolinomban az i-edik együttható:

$$C_i = 0 \tag{12.7}$$

Bizonyítás: Ez az állítás könnyen belátható, hiszen ha ϑ a Fourier-transzformálásban használt n-edrendű $GF(N^M)$-beli elem, akkor

$$c(\vartheta^i) = c_0 + c_1\vartheta^i + c_2\vartheta^{2i} + \ldots + c_{n-1}\vartheta^{(n-1)i} = \sum_{j=0}^{n-1} c_j \vartheta^{ji},$$

ami az (12.1) definíció szerint pont C_i. A $c(t)$-nek pedig akkor gyöke ϑ^i, ha $c(\vartheta^i) = 0$. (Q.E.D.)

Az állítás fordítva is igaz, a $C(t)$ spektrumpolinomnak akkor és csak akkor gyöke ϑ^i, ha $c_i = 0$.

Legyen a Reed–Solomon-kód generátorpolinomjának gyöktényezős felbontása

$$g(t) = (t-\vartheta^1)(t-\vartheta^2)\cdot\ldots\cdot(t-\vartheta^{n-k}).$$

Mivel ϑ^i gyöke $g(t)$-nek $i = 1,2,\ldots,n-k$-ra, a (12.7) szerint azt eredményezi, hogy

$$C_i = 0, \qquad i = 1,2,\ldots,n-k\text{-ra.}$$

Az (n,k) paraméterű Reed–Solomon-kódszavak spektrumának első $n-k$ komponense nulla.

Ez a tény lehetőséget biztosít a Reed–Solomon-kódok spektrumukon keresztül való generálására. Ha ugyanis a $\mathbf{C}$ kódszóspektrum 0-tól $(n-k)$-ig terjedő komponenseit nullára választjuk, a maradék k helyre pedig betöltjük az üzenetet, majd C-t inverz Fourier-transzformáljuk, akkor biztos, hogy jó kódszót fogunk kapni.

12.1. példa: Számítsuk ki a 11.3 példában szereplő $GF(7)$ feletti, $(6,4)$ paraméterű Reed–Solomon-kód által generált $\mathbf{c} = (3\ 5\ 1\ 6\ 4\ 1)$ kódszó spektrumát. (A Fourier-transzformációban szereplő ϑ szám azonos a kódot generáló $\vartheta = 3$-mal.)

Megoldás: Alkalmazzuk a Fourier-transzformáció (12.1) definíciós összefüggését, figyelembe véve, hogy $\vartheta^0 = 1$, $\vartheta^1 = 3$, $\vartheta^2 = 2$, $\vartheta^3 = 6$, $\vartheta^4 = 4$ és $\vartheta^5 = 5$:

$$\begin{aligned}
C_0 &= 3\cdot 1 + 5\cdot 1 + 1\cdot 1^2 + 6\cdot 1^3 + 4\cdot 1^4 + 1\cdot 1^5 = 20 \equiv 6 \mod 7,\\
C_1 &= 3\cdot 1 + 5\cdot 3 + 1\cdot 3^2 + 6\cdot 3^3 + 4\cdot 3^4 + 1\cdot 3^5 = 756 \equiv 0 \mod 7,\\
C_2 &= 3\cdot 1 + 5\cdot 2 + 1\cdot 2^2 + 6\cdot 2^3 + 4\cdot 2^4 + 1\cdot 2^5 = 161 \equiv 0 \mod 7,\\
C_3 &= 3\cdot 1 + 5\cdot 6 + 1\cdot 6^2 + 6\cdot 6^3 + 4\cdot 6^4 + 1\cdot 6^5 = 14325 \equiv 3 \mod 7,\\
C_4 &= 3\cdot 1 + 5\cdot 4 + 1\cdot 4^2 + 6\cdot 4^3 + 4\cdot 4^4 + 1\cdot 4^5 = 2471 \equiv 0 \mod 7,\\
C_5 &= 3\cdot 1 + 5\cdot 5 + 1\cdot 5^2 + 6\cdot 5^3 + 4\cdot 5^4 + 1\cdot 5^5 = 6428 \equiv 2 \mod 7.
\end{aligned}$$

A spektrum tehát $\mathbf{C} = (6\ 0\ 0\ 3\ 0\ 2)$, ahol a C_1 és C_2 komponensek valóban nullák.

12.2. A Reed–Solomon-kódok dekódolása

Legyen a vett vektorunk $\mathbf{v} = (v_0,v_1,\ldots,v_{n-1})$, és tegyük fel, hogy egy $\mathbf{c} = (c_0,c_1,\ldots,c_{n-1})$ kódszóból keletkezett, $\Delta\mathbf{c} = (\Delta c_0,\Delta c_1,\ldots,\Delta c_{n-1})$ hibavektor hozzáadásával. Emlékeztetőül: a Reed–Solomon-kódok paritásellenőrző mátrixa a (11.9) szerint:

$$\mathbf{H}^T = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \vartheta & \vartheta^2 & \ldots & \vartheta^{n-k} \\ \vartheta^2 & \vartheta^4 & \ldots & \vartheta^{2(n-k)} \\ \vdots & \vdots & \ddots & \vdots \\ \vartheta^{n-1} & \vartheta^{2(n-1)} & \ldots & \vartheta^{(n-1)(n-k)} \end{pmatrix},$$

ahol ϑ a kód generáló eleme. Minden lineáris blokk-kódra igaz, hogy a szindróma a vett vektornak csak a hiba-részéből keletkezik, hiszen az érvényes kódszavak szindrómája 0, azaz:

$$\mathbf{s}(= \mathbf{v} \cdot \mathbf{H}^T) = \Delta\mathbf{c} \cdot \mathbf{H}^T.$$

Ennek az egyenletnek a megoldása a hibavektor, amellyel ki tudjuk javítani a vett vektorunkat. Ha elvégezzük ezt a mátrixszorzást, akkor a szindróma j-edik elemére a következő képletet kapjuk:

$$s_j = \sum_{i=0}^{n-1} \Delta c_i \cdot \vartheta^{ij} \tag{12.8}$$

12.3. Törléses hibák javítása

Ha ismerjük a hibák helyét, azaz törléses hibákról beszélünk, akkor legfeljebb $n-k$ hibát tudunk kijavítani egy (n,k) paraméterű Reed–Solomon-kóddal. Ez azt jelenti, hogy a Δc hibavektornak legfeljebb $n-k$ komponense lesz 0, és azt is tudjuk, hogy melyek ezek a komponensek, csak a nagyságukat nem ismerjük.

Tegyük fel, hogy pontosan $n-k$ törléses hibánk van. (Ha ennél kevesebb, akkor néhány helyen nulla lesz a törléses hiba nagysága.) Így a (12.8) egyenlőségek mindegyikében csak jól meghatározott $n-k$ darab tag adhat nem nulla eredményt az összegben. Tulajdonképpen a $\mathbf{H}^T$ mátrixnak csak bizonyos sorai vesznek részt a műveletekben, mivel ha $\Delta c_i = 0$, akkor a $\mathbf{H}^T$ mátrix i-edik sorában minden elemnek nulla lesz a szorzója a (12.8) egyenletek mindegyikében. Töröljük gondolatban ezeket a sorokat a mátrixból, így kapunk egy csonkolt $\tilde{\mathbf{H}}^T$ mátrixot, amelynek $n-k$ sora (és ugyanannyi oszlopa) van. Töröljük a hibavektorból is a nulla elemeket, így $\Delta\tilde{\mathbf{c}}$ vektort kapunk $n-k$ komponenssel. Mivel a szindróma $n-k$ elemből áll, egy $n-k$ darab egyenletből álló egyenletrendszert kapunk $\Delta\tilde{\mathbf{c}}$-re, ami teljesen meghatározza $\Delta\tilde{\mathbf{c}}$-t, azaz a

$$\begin{aligned}
\tilde{H}_{00} \cdot \Delta\tilde{c}_0 + \tilde{H}_{10} \cdot \Delta\tilde{c}_1 + \ldots + \tilde{H}_{n-k-1\,0} \cdot \Delta\tilde{c}_{n-k-1} &= s_0 \\
\tilde{H}_{01} \cdot \Delta\tilde{c}_0 + \tilde{H}_{11} \cdot \Delta\tilde{c}_1 + \ldots + \tilde{H}_{n-k-1\,1} \cdot \Delta\tilde{c}_{n-k-1} &= s_1 \\
&\vdots \\
\tilde{H}_{0\,n-k-1} \cdot \Delta\tilde{c}_0 + \tilde{H}_{1\,n-k-1} \cdot \Delta\tilde{c}_1 + \ldots + \tilde{H}_{n-k-1\,n-k-1} \cdot \Delta\tilde{c}_{n-k-1} &= s_{n-k-1}
\end{aligned}$$

egyenletrendszer egyértelműen megoldható.

Az, hogy egy $n-k$ egyenletből álló egyenletrendszernek van-e egyértelmű megoldása, attól függ, hogy az egyenletrendszert meghatározó mátrix milyen. Ha a mátrixnak nincs tiszta 0 sora, sem oszlopa és egyik sora(oszlopa) sem áll elő másik sorok(oszlopok) lineáris kombinációjaként, akkor az egyenletrendszer megoldható.

12.2. példa: Tegyük fel, hogy a 11.3 példában keletkezett kódszó a második és a negyedik helyen meghibásodott a csatornán való áthaladás során, így a $\mathbf{v} = (3\ 5\ 4\ 6\ 5\ 1)$ vektort vettük. Dekódoljuk a vektort a 11.11 paritásellenőrző mátrix segítségével.

Megoldás: Számoljuk ki a vektor szindrómáját:

$$\mathbf{s} = \mathbf{v} \cdot \mathbf{H}^T = (3 \quad 5 \quad 4 \quad 6 \quad 5 \quad 1) \cdot \begin{pmatrix} 1 & 1 \\ 3 & 2 \\ 2 & 4 \\ 6 & 1 \\ 4 & 2 \\ 5 & 4 \end{pmatrix} = (87 \quad 49) \equiv (3 \quad 0) \mod 7.$$

A hiba a 2. és a 4. pozícióban van, így a csonkolt paritásellenőrző mátrix a következő (a sorok számozása nullánál kezdődik)

$$\tilde{\mathbf{H}}^T = \begin{pmatrix} 2 & 4 \\ 4 & 2 \end{pmatrix}.$$

E mátrixnak a csonkolt $\Delta\tilde{\mathbf{c}} = (\Delta c_2\ \Delta c_4)$ hibavektorral vett szorzata megadja a szindrómát:

$$\begin{aligned} 2 \cdot \Delta c_2 + 4 \cdot \Delta c_4 &\equiv 3 \\ 4 \cdot \Delta c_2 + 2 \cdot \Delta c_4 &\equiv 0 \end{aligned}$$

A második egyenlőségből kifejezve Δc_2-t, felhasználva, hogy $-4 \equiv 3 \bmod 7$ és $3^{-1} \equiv 5 \bmod 7$:

$$c_2 = 2 \cdot 5c_4 \equiv 3c_4 \bmod 7.$$

Az első egyenlet szerint

$$(6+4)c_4 \equiv 3c_4 = 3,$$

így $c_4 = 1$ és $c_2 = 3$, tehát a második helyen a hiba 3, a negyediken 1. A javított kódszó $\mathbf{v}_{\text{jav}} = (3\ 5\ 1\ 6\ 4\ 1)$, ami a kiindulási $\mathbf{c}$-vel megegyezik.

12.4. Egyszerű hibák javítása

Ha az egyszerű hibák száma ν, akkor a Δc vektornak $n-\nu$ komponense 0 lesz, ν darab pedig nem nulla. A probléma csak az, hogy nem tudjuk,

melyik helyeken vannak a nem nulla Δc_i elemek, és azt sem tudjuk, azok mekkorák.

Tegyük fel, hogy az l-edik helyen van hiba. Ekkor a (12.8) egyenletek mindegyikében az összegekben szerepel egy

$$\Delta c_l \cdot \left(\vartheta^l\right)^j$$

típusú tag, ahol j az egyenlet sorszáma. A ϑ^l mennyiségből egyértelműen ki lehet számítani a hiba helyét, hiszen ϑ minden n-nél kisebb hatványa különböző, ezért a ϑ^l mennyiséget **hibahely lokátornak** nevezik.

Vezessünk be egy **hibahelypolinomot**:

$$L(t) = \prod_{i=1}^{\nu}(1 - \vartheta^{l_i} \cdot t) = 1 + L_1 t + L_2 t^2 + \ldots + L_\nu t^\nu, \qquad (12.9)$$

ahol a l_i index jelöli azokat a pozíciókat, ahol hiba van. Ezeket a pozíciókat természetesen nem ismerjük. Figyeljük meg, hogy az $L(t)$ polinomnak a gyökei pont a hibahely lokátorok inverzei. Ha tehát meg tudjuk határozni az $L(t)$ polinomot, akkor ki tudjuk számolni azt, hogy melyik pozíciókban van hiba úgy, hogy

- megkeressük a polinom gyökeit,
- invertáljuk azokat a megfelelő véges test felett,
- majd az inverzről kiderítjük, hogy ϑ-nak hányadik hatványa.

A ϑ^l szám inverze az $L(t)$ polinom gyöke, azaz

$$L\left(\vartheta^{-l}\right) = 0.$$

Szorozzuk meg az egyenlet mindkét oldalát $\Delta c_l \cdot \left(\vartheta^l\right)^{\nu+j}$-vel, így a következő egyenletet kapjuk:

$$\Delta c_l \cdot \left(\vartheta^l\right)^{\nu+j} \cdot L\left(\vartheta^{-l}\right) = 0.$$

Összegezzük a fenti kifejezéseket $l = 1,2,\ldots\nu$-re, így az egyenlőség a

$$\sum_{l=1}^{\nu} \Delta c_l \cdot \left(\vartheta^l\right)^{\nu+j} \cdot L\left(\vartheta^{-l}\right) = 0 \qquad (12.10)$$

alakú lesz. Ez az egyenlet minden j-re igaz. Fejtsük ki az $L\left(\vartheta^{-l}\right)$ polinomot:

$$L\left(\vartheta^{-l}\right) = 1 + L_1 \cdot \vartheta^{-l} + L_2 \cdot \left(\vartheta^{-l}\right)^2 + \ldots + L_\nu \cdot \left(\vartheta^{-l}\right)^\nu .$$

Helyettesítsük be ezt a felírást a (12.10) egyenletbe. vegyük figyelembe azt, hogy a polinom minden tagja meg van szorozva $\left(\vartheta^l\right)^{\nu+j}$-nel, így a

$$\sum_{l=1}^{\nu} \Delta c_l \cdot \left(1 \cdot \left(\vartheta^l\right)^{\nu+j} + L_1 \cdot \left(\vartheta^l\right)^{\nu+j-1} + \ldots + L_\nu \cdot \left(\vartheta^l\right)^j\right) = 0.$$

egyenlőséget fogjuk kapni, minden j-re. Összegezzük a kifejezést tagonként és emeljük ki minden tagból az $L(t)$ polinom együtthatóját:

$$\sum_{l=1}^{\nu} \Delta c_l \cdot \left(\vartheta^l\right)^{\nu+j} + L_1 \cdot \sum_{l=1}^{\nu} \Delta c_l \left(\vartheta^l\right)^{\nu+j-1} + \ldots + L_\nu \cdot \sum_{l=1}^{\nu} \Delta c_l \left(\vartheta^l\right)^j = 0.$$

Hasonlítsuk össze az egyes összegeket a (12.8) egyenlettel. Vegyük észre azt, hogy minden összeg egy-egy szindrómakomponens, így az előbbi egyenlet felírható, mint

$$s_{\nu+j} + L_1 \cdot s_{\nu+j-1} + L_2 \cdot s_{\nu+j-2} + \ldots + L_\nu \cdot s_j = 0, \tag{12.11}$$

a $j = 0,1,2,\ldots\nu - 1$ indexekre. Kaptunk tehát ν darab egyenletből álló egyenletrendszert a ν darab L_i együtthatóra. (Belátható, hogy az egyenletrendszer megoldható.)

Az egyenletrendszerben a legnagyobb előforduló szindróma-index $j = \nu - 1$ esetén az első tag $j + \nu = 2\nu - 1$ indexe. Mivel az (n,k) paraméterű lineáris blokk-kódokkal legfeljebb $\nu = (n-k)/2$ egyszerű hibát lehet kijavítani, ez a maximális index nem haladja meg a szindróma komponenseinek $n - k$ számát.

A (12.11) egyenletrendszer megoldásával megkapjuk a hibahelypolinomot, amelynek meg tudjuk keresni a gyökeit, és ki tudjuk számolni belőlük a hibák helyét. Ha a hibahelyek ismertek, visszaértünk a törléses hibák javításának problémájához.

12.3. példa: A $GF(11)$ véges számtest feletti Reed–Solomon-kód paritásellenőrző mátrixa

$$\mathbf{H}^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 7 & 5 & 2 & 3 \\ 5 & 3 & 4 & 9 \\ 2 & 4 & 8 & 5 \\ 3 & 9 & 5 & 4 \\ 10 & 1 & 10 & 1 \\ 4 & 5 & 9 & 3 \\ 6 & 3 & 7 & 9 \\ 9 & 4 & 3 & 5 \\ 8 & 9 & 6 & 4 \end{pmatrix}.$$

(Ez a $GF(11)$ feletti, $\vartheta = 7$ generálóelemű, (6,10) paraméterű R–S-kód paritásellenőrző mátrixa.) Dekódoljuk a $\mathbf{v} = (8\,3\,1\,9\,5\,3\,7\,0\,0\,8)$ vett vektort.

Megoldás: A kóddal legfeljebb $\nu = 2$ egyszerű hiba javítható. A vektor szindrómája:

$$\mathbf{s} = \mathbf{v} \cdot \mathbf{H}^T = (8\ \ 3\ \ 1\ \ 9\ \ 5\ \ 3\ \ 7\ \ 0\ \ 0\ \ 8) \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 7 & 5 & 2 & 3 \\ 5 & 3 & 4 & 9 \\ 2 & 4 & 8 & 5 \\ 3 & 9 & 5 & 4 \\ 10 & 1 & 10 & 1 \\ 4 & 5 & 9 & 3 \\ 6 & 3 & 7 & 9 \\ 9 & 4 & 3 & 5 \\ 8 & 9 & 6 & 4 \end{pmatrix} =$$

$$= (189\quad 217\quad 256\quad 147) \equiv (2\quad 8\quad 3\quad 4) \mod 11.$$

A hibahelypolinom feltételezett alakja:

$$L(t) = 1 + L_1 t + L_2 t^2.$$

A szindróma komponenseit felhasználva felírhatjuk a hibahelypolinom együtthatóira a 12.11 egyenletrendszert $j = 0$-ra és $j = 1$-re:

$$\begin{aligned} s_2 + L_1 \cdot s_1 + L_2 \cdot s_0 &= 3 + 8L_1 + 2L_2 = 0 \\ s_3 + L_1 \cdot s_2 + L_2 \cdot s_1 &= 4 + 3L_1 + 8L_2 = 0 \end{aligned}$$

A második egyenletből fejezzük ki L_2-t: mivel $-8 \equiv 3 \mod 11$, és $3^{-1} \equiv 4 \mod 11$,

$$L_2 = 4 \cdot 4 + 4 \cdot 3L_1 \equiv 5 + 1L_1 \mod 11,$$

ami behelyettesítve az első egyenletbe

$$0 = 3 + 8L_1 + 2(5 + L_1) \equiv 2 + 10L_1 \mod 11.$$

A 10 ellentettje 1 a $GF(11)$-ben ($11 - 10 = 1$), így $L_1 = 2$. Ebből $L_2 = 5 + 2 \equiv 7 \bmod 11$. A hibahelypolinom tehát:

$$L(t) = 7t^2 + 2t + 1.$$

A polinom gyökeit meghatározhatjuk próbálgatással: az 1 behelyettesítésével 10-et kapunk, tehát az 1 nem gyök. A 2 behelyettesítésével 33-at kapunk (ami $\equiv 0 \bmod 11$), így a hibahelypolinom egyik gyöke a 2. A másik gyök meghatározható az $L(t)/(t-2)$ polinomosztással vagy a további számok behelyettesítésével. A másik gyök a 4.

A hibahely lokátorokat a hibahelypolinom gyökeinek inverzeként kapjuk meg:

$$\begin{aligned} \vartheta^{l_1} &= 2^{-1} \equiv 6 \bmod 11 \\ \vartheta^{l_2} &= 4^{-1} \equiv 3 \bmod 11. \end{aligned}$$

Az utolsó lépésként meg kell keresni, hogy a $\vartheta = 7$-nek hányadik hatványa a 6, illetve a 3. A 7 hatványai a nulladiktól kezdve a $\mathbf{H}^T$ mátrix első oszlopában láthatók (emlékezzünk a mátrix (11.9) definíciójára). Ezek szerint

$$\begin{aligned} \vartheta^7 &= 6 \\ \vartheta^4 &= 3, \end{aligned}$$

a hibák tehát a 4. és 7. helyen fordulhatnak elő.

13. Hibacsomók elleni védekezés

Egy csatornán nem csak elszórtan jelentkeznek hibák, előfordul, hogy egy egész sor egymás után következő szimbólum elromlik a csatornán való áthaladás során, például megkarcolódik a CD felülete. A sok egymást követő hibás szimbólumot szokás **hibacsomó**nak nevezni. Mivel egymás utáni hosszú hibasorokat csak nehezen, igen hosszú paritásszegmensekkel lehetne javítani, vagy még úgy sem, ezeket a hibacsomókat célszerű valahogy szétbontani. Ha a kódolás során tehát szétválasztják a szimbólumsorozatot, hogy majd a csatornán való áthaladás után újra összefésüljék, akkor ha volt is a csatornán való áthaladáskor hibacsomó, az szétosztódik. A hibacsomók elleni védekezést **kódátfűzés**nek, vagy angol szóval „interleaving"-nek nevezik.

13.1. Többutas kódátfűzés

E módszer használatakor több párhuzamos szálon kódolják a b_1,b_2,... üzenetet, több kódoló berendezést alkalmaznak. Legyen összesen Λ águnk, Λ nem feltétlenül azonos csatornakódoló eljárással. A tömörített (forráskódolt) üzenet első szimbólumát az első ágra vezetjük, másodikat a másodikra, és így tovább a Λ-adikat a Λ-adikra, a $\Lambda + 1$-ediket megint az elsőre. Az i-edik ágon tehát a b_i,$b_{i+\Lambda}$,$b_{i+2\Lambda}$,...,$b_{i+(k-1)\Lambda}$ üzenetet kódoljuk. Ezután az egyes ágakon keletkezett kódszavakat összefésüljük, és úgy engedjük a csatornára. A csatorna bemenetén tehát először az első ág kódszavának első szimbóluma jelenik meg, majd a második ág első szimbóluma, és így tovább. Az első ág második szimbólumát csak az utolsó ág első szimbóluma után adjuk le. Vétel után szintén szétválogatjuk a szimbólumokat Λ ágra, Λ darab dekódoló berendezésre vezetjük őket, majd újra összefésüljük őket.

A többutas interleavinghez Λ darab kódoló-dekódoló pár szükséges, azonban azok lehetnek lassú berendezések is (bonyolultabb kódolási algoritmusokkal), hiszen csak minden Λ-adik szimbólummal kell elbánniuk, elég az órajel frekvenciájának Λ-adrészén működniük.

A hibacsomót így egy-egy kódszóban Λ-adrészére csökkenthetjük.

13.2. Blokkos kódátfűzés

Ez a módszer nem igényel több kódoló-dekódoló párt, viszont több tároló szükséges hozzá. Először a kívánt csatornakóddal kódoljuk az információfolyamunkat, majd az így kapott szimbólumokkal (például) oszloponként

feltöltünk egy $D \times D$ méretű mátrixot. Amikor a mátrix tele van, elkezdjük kiolvasni soronként, és így adjuk a csatornára. A csatorna kimenetén a $D \times D$-s mátrixot soronként töltjük fel, majd ha betelt, oszloponként olvassuk ki. A kapott szimbólumsorozatot ezután dekódoljuk.

Itt egy-egy kódszóban a hibacsomónak csak a D-edrészére jelenik meg.

Hátránya ennek a módszernek, hogy egy-egy mátrix feltöltésére D^2 időegységet várni kell. Úgy szokták javítani a módszer időfelhasználását, hogy amíg egy mátrixot töltenek, az előzőt éppen kiolvassák.

13.3. A digitális hangrögzítésben alkalmazott kódolások és kódátfűzés elve

A CD-k kódolásakor egyfajta blokkos interleavinget használnak. Az egyes hangcsatornák jeleit $44{,}1$ kHz frekvenciával mintavételezik és két bájtba kvantálják. Vizsgáljuk most csak a két hangcsatornás (jobb és bal) sztereó rendszert. Legyen a jobb csatorna mintavételezett, kvantált jele az i-edik időegységben x_{i1},x_{i2} (első és második bájt), a bal oldalé y_{i1},y_{i2}. Mind x_{i1} és x_{i2}, mind pedig y_{i1} és y_{i2} a $GF(2^8)$ véges test eleme, hiszen bájtokról van szó. A két hangcsatorna két-két szimbólumsorozatát úgy fésüljük össze egy adatfolyammá, hogy először a jobb csatorna első bájtjait vesszük, majd a bal csatornáét, eztán a jobb csatorna második bájtjait, majd a balét és így tovább, végül az adatfolyamunk x_{11}, x_{12}, y_{11}, y_{12}, x_{21}, x_{22}, y_{21}, y_{22}, x_{31}, x_{32}, y_{31}, y_{32}, ... lesz.

A feltöltendő mátrix 28×28-as, melynek a bal felső 24×24-es blokkját töltjük fel az adatokkal oszlopfolytonosan. Minden oszlopot (28,24) paraméterű $GF(2^8)$ feletti Reed–Solomon-kóddal kódolunk, így tele lesz az első 24 oszlop. Ezután minden sort is kódolunk ugyanezzel a kóddal, így tele lesz a mátrix. Ezt a mátrixot olvassák ki soronként, és így rögzítik a CD-re.

Az alkalmazott Reed–Solomon-kód kódtávolsága 5, így 4 hibát tud jelezni, 4 törléses és 2 egyszerű hibát tud javítani. A dekódolás során azonban nem használják ki maximálisan ezeket a lehetőségeket.

- Első lépésként a beolvasott mátrix minden sorának kiszámolják a szindrómáját. Ha 0 a szindróma, a sort békén hagyják. Ha egyetlen egyszerű hibát detektáltak, azt kijavítják, ha kettőt, akkor azok helyére törléses hibákat tesznek, ha pedig 2-nél több hibát detektáltak, az egész sort törléses hibákkal helyettesítik.
- Ezután megvizsgálják oszloponként a mátrixot. Ha az adott oszlopban legfeljebb két törléses hiba van, azokat kijavítják (nem mind a 4

javítható törléses hibát). Ha ennél több törléses hiba van, azok helyén a jel értékét a környező hibátlan szimbólumok felhasználásával közelítik.

Azért nem használják ki a kódból eredő összes javítási lehetőséget, mert gyors dekódoló algoritmusra van szükség, és a közelítés gyors, és – mivel a mintavételezett jel viszonylag folytonos volt, az emberi fül pedig elég nagy tehetetlenségű – elég pontos is.

14. Konvolúciós kódolás

A konvolúciós kódolás egészen más módszereket használ, mint a blokkkódolás.

Konvolúciós kódolás során forrásból származó, már tömörített b_1,b_2,b_3,... bitsorozatot k bites szakaszokra, **üzenetszegmens**ekre bontjuk. A kódolóban mindig m darab üzenetszegmens tárolódik. Egy tárolt k-bites üzenetszegmenst **keret**nek vagy üzenetkeretnek nevezünk. Egy időegység alatt a következőket hajtja végre a kódoló:

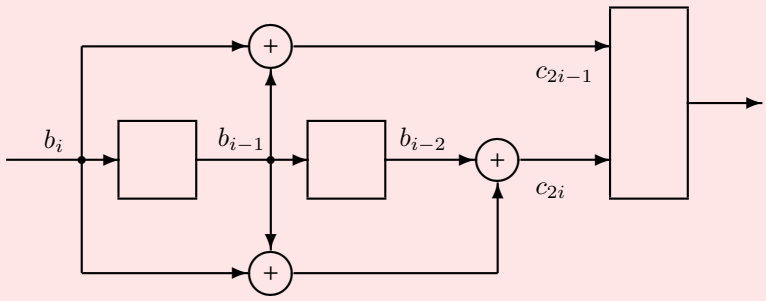
- Beolvas egy új üzenetszegmenst.
- Ebből és a tárolt m darab keretből kiszámít egy úgynevezett **kódszókeret**et, amelyet megjelenít a kimenetén. Ha a kódszókeret n bites, az
$$R = \frac{k}{n}$$
kifejezés a **kódsebesség**.
- eldobja a legrégebben tárolt keretet, elraktározza az újonnan beolvasottat, így az új ciklus elejére újra m üzenetszegmenst fog tárolni.

Látató, hogy egy üzenetszegmens $(m+1)$ lépéssel a megjelenése után tűnik el a kódolóból, így a $K = (m+1)k$ kifejezés a kódoló **kényszerhossz**a bitekben. Az $N = (m+1)n$ mennyiséget **blokkhossz**nak nevezik, és azt mutatja meg, hogy egy bit a forráson hány bitet befolyásol a kimeneten.

A kódoló berendezést *léptetőregiszter*ekkel valósítják meg. (N,K) **paraméterű konvolúciós kód**nak, vagy trellis-kódnak nevezzük a lineáris, időinvariáns, véges kényszerhosszú kódokat, melyeknek a blokkhossza N és a kényszerhossza K.

A kapott kódot egy úgynevezett **fa-kód** formájában lehet megadni. Ha a kódoló bementére ráeresztünk egy a nullával kezdődő, végtelen bitsorozatot, akkor kapunk egy – szintén végtelen – kimeneti bitsorozatot. A fa-kód lényegében e sorozatok közötti hozzárendelést mutatja meg. Azért a „fa" elnevezés, mert az ilyen hozzárendelést sokszor bináris fa formájában adják meg, a fa gyökere az az állapot, amikor a léptetőregiszterek minden tárolója nullával van feltöltve, és minden elágazás a fában egy-egy állapotból a lehetséges bemeneti bitkombinációk hatására létrejövő változást jelképez. Mivel a bináris fa ábrázolás eléggé nehézkes, helyette alternatív reprezentálási módszereket – az állapotátmenet-gráfot és a trellis-diagramot – fogunk tanulni.

14.1. példa: A kódoló áramkörünk blokkvázlata legyen a következő



Adjuk meg a kódoló paramétereit és a bemeneti és kimeneti bitek közötti összefüggést.

Megoldás: Mivel egyetlen bemeneti bit van, $k = 1$. A kimenetet a kódoló két bitből fésüli össze, tehát $n = 2$. Kettő darab tároló van a kódolóban, így $m = 2$, ebből pedig $K = (m + 1) \cdot k = 3$ és $N = (m + 1) \cdot n = 6$.

A kimenet páros, illetve páratlan bitjei a fenti kódoló esetén:

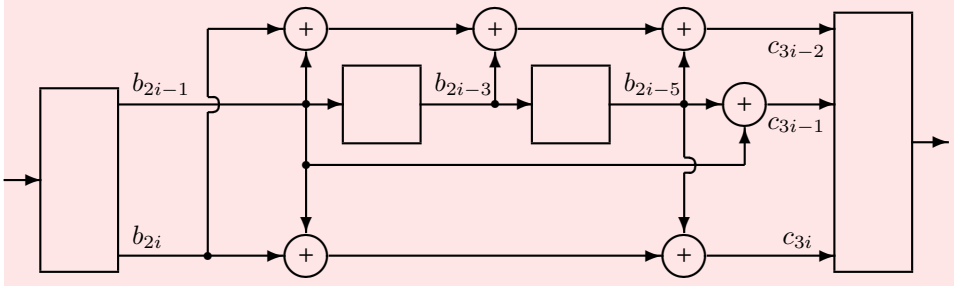
$$c_{2i-1} = b_i + b_{i-1}$$

és

$$c_{2i} = b_i + b_{i-1} + b_{i-2}$$

A bitjeink a $GF(2)$ véges test elemei, tehát minden műveletet $\mod 2$ kell nézni.

14.2. példa: Adjuk meg a következő blokkvázlattal rendelkező kódoló paramétereit, kódsebességét és a bemeneti és kimeneti bitek közötti összefüggést:



Megoldás: A kódoló a bemenetet két bitfolyamra választja szét és a bemenetet három bitsorozatból fésüli össze, így $k = 2$ az üzenetkeret, és $n = 3$ a kódszókeret hossza. A két ág közül a felsőben van több tároló, szám szerint 2, így $m = 2$. A kényszerhossz ebből $K = 6$ a blokkhossz pedig $N = 9$. A kódsebesség $R = k/n = 2/3$.

A hárommal osztva 1, 2, illetve 0 maradékot adó kimenetek a következőképpen függenek a páros és páratlan sorszámú bemeneti bitektől:

$$\begin{aligned} c_{3i-2} &= b_{2i-1} + b_{2i-3} + b_{2i-5} + b_{2i}, \\ c_{3i-1} &= b_{2i-1} + b_{2i-5} \end{aligned}$$

és

$$c_{3i} = b_{2i-1} + b_{2i-5} + b_{2i}.$$

Egy $k > 1$ üzenetkeret-hosszú konvolúciós kódolónak k darab bemenete van. Első lépésként a bemenő jelet úgy választja szét, hogy az i-edik bemenetre az i-edik, $i + k$-adik, $i + 2k$-adik,... bit kerüljön. Minden bemenethez tartozik egy a tárolósor. A k darab tárolósor kimeneteiből állítja elő a kódoló saját kimenetének megfelelő bitjeit, amelyeket úgy fésül össze, hogy először az első kimenet első bitjét adja le, majd a második kimenet első bitjét, a harmadik kimenet első bitjét, és így tovább. Az első kimenet második bitje az n-edik kimenet első bitje után következik, őt a második kimenet második bitje követi...

Az egyes ágakat lehet egy-egy bitsorozattal jellemezni a következőképpen: A sorozat j-edik eleme akkor legyen 1, ha azon az ágon megjelenik az üzenet j-edik eltoltja, egyébként legyen a j-edik bit 0. Azért nevezik konvolúciós kódoknak az ilyeneket, mert a kimenet bitjeit a bemeneti bitsorozatnak a megfelelő ágat jelképező sorozattal vett moduló 2 konvolúciójaként kaphatjuk meg. A 14.1 példához tartozó két bitsorozat: a páratlan ághoz $(1,1,0,0,0,\ldots)$, a pároshoz pedig $(1,1,1,0,0,\ldots)$. A 14.2 példa esetén a helyzet kissé bonyolultabb, ott ugyanis két bemenet van, így az egyes bemenetek hatását külön-külön kell figyelembe venni, s a végén összeadni őket. Az első bemenet hatása az első kimenetre az $1,1,1,0,0,\ldots$ bitsorozattal jellemezhető, a második kimenetre az $1,0,1,0,0,\ldots$ sorozattal, a harmadikra szintén az $1,0,1,0,0,\ldots$ bitsorozattal. A második bemeneti ág hatása az első és a harmadik kimenetre az $1,0,0,0,0\ldots$ sorozattal írható le, míg a középső kimenetre tiszta nulla sorozattal.

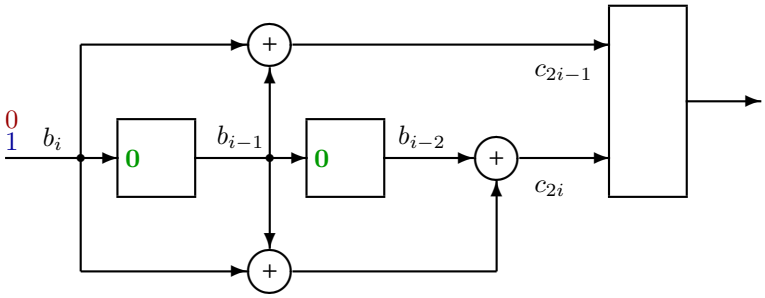
14.1. Állapotátmeneti gráf és trellis

A kódolót lehet az **állapotátmenet-gráf**jával jellemezni. Az állapotátmenet-gráf minden csomópontja a kódoló tárolóiban szereplő különböző értékeknek (mind a 14.1, mind pedig a 14.2 példa esetében 00, 01, 10 és 11)

feleltethető meg, az élek az ezek közti átmeneteket jellemzik. Az éleken fel szokták tüntetni a bementre kerülő bitkombinációt, és az adott éllel jellemzett átmenetkor a kimeneten megjelenő biteket. Minden állapotból 2^k nyíl vezet egy-egy másik állapotba vagy önmagába, ez a 2^k-féle bemenő jelnek megfelelő 2^k lehetséges átmenetet jelképezi.

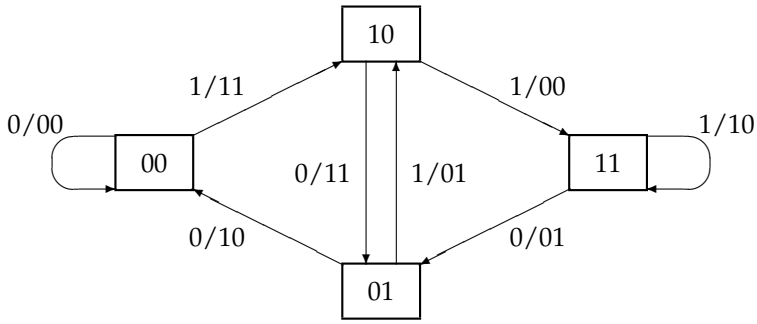
14.3. példa: Rajzoljuk fel a 14.1 példában látható kódoló állapotátmeneti gráfját.

Megoldás: Tegyük fel, hogy a kódoló a 00 állapotban van, azaz mindkét tárolója 0-n áll, mint azt a következő ábrán a zöld számok mutatják



Ha a bemenetre 0 kerül, akkor a következő lépésben ez az új 0-s lesz az első tárolón, az ott levő 0-s érték pedig átkerül a másodikra. Így a következő állapot megint a 00 lesz. Ezt az átmenetet jelképezi az alábbi ábra bal szélén található nyíl. Mivel minden tárolt keret és a bemenet is 0 volt, a kimenet csak 00 lehet. Az átmenetet leíró nyílon feltüntetett számok közül az első a bemeneti bit, a „/”-jel utáni kettő pedig a kimeneti bitpáros.

Ha a 00 állapotból kiindulva a bemenetre 1-es bit kerül, akkor a következő lépésben az az 1-es bit már az első tárolón lesz, az első tárolón levő 0 a második tárolóra lép, s mivel harmadik tároló nincsen a sorban, az második tároló nullását eldobjuk. 1 bemeneti bit hatására tehát a 00 állapotból az 10 állapotba kerülünk, ezt az átmenetet jelképezi az alsó ábra bal oldalán lévő, 00 felől 10-ba mutató nyíl. Ha a bemenet 1, akkor mindkét kódoló ágon 1 lesz a kimenet, mivel ahhoz az 1-hez a tárolókon levő nullák adódnak csak hozzá. Ezért van a nyílon „1/11” felirat.

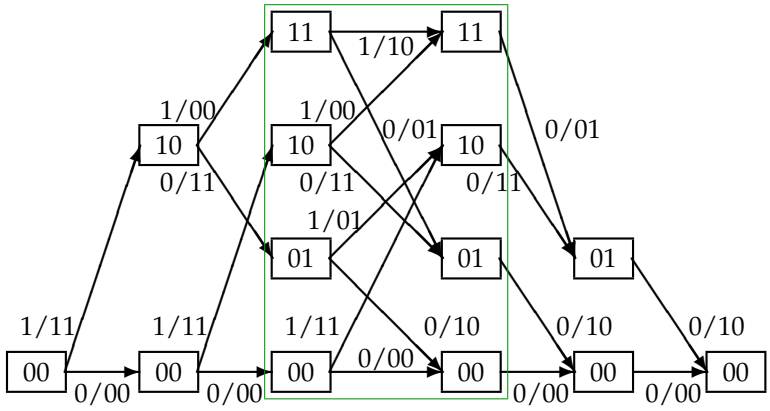


A többi állapotátmenetet hasonlóan kell megvizsgálni, csak a kiindulási állapot nem a 00 lesz, hanem az 10, a 01, illetve az 11. Az eredményt a fenti ábra tartalmazza.

Az állapotátmeneti gráfból könnyen fel lehet rajzolni a kódoló úgynevezett **trellis reprezentáció**ját. A trellisen a lehetséges állapotok egymás fölött helyezkednek el, egy-egy ilyen állapot-oszlop jelképez egy-egy lépést a kódolásban. Annyi oszlopot rajzolunk egymás mellé, ahány lépést követni szándékozunk. Az alap trellis N lépést tüntet fel, ahol N továbbra is a blokkhossz. Az első csomópontja az az állapot, amikor csupa nullával vannak feltöltve a tárolók. Az első K lépés során felrajzolja, hogy különböző bemeneti jelekre milyen állapotokba juthat a rendszer, majd a maradék lépésekben csupa nulla bemenetet tételezve fel kiüríti a tárolókat. Az egyes állapotok közötti átmeneteket jelképező nyilakra feltüntethető a bemeneti bit(sorozat) és a kimeneti bitsorozat, az állapotátmeneti gráfhoz hasonló alakban. A trellis-diagram megértését segíti a következő példa.

14.4. példa: Készítsük el a 14.3 példa állapotátmeneti gráfjából a szóban forgó kódoló alap trellisét

Megoldás: Az alap trellis kiindulási állapota a 00-s tárolóállapot, amely az állapotátmeneti gráf bal szélén látható. A 00 állapotból 2 nyíl indul ki, az egyik szintén 00 állapotba visz, míg a másik az 10-ba. Az állapotátmenetekhez tartozó bemeneti és kimeneti bitek a gráfról leolvashatók. Ezeket az átmeneteket jelképezik az alábbi trellis bal oldali nyilai. A nyilakon ugyanaz a „bemeneti bit/kimeneti bitpáros" felirat van, mint az állapotátmeneti gráf 00 állapotából kiinduló két nyilán:



A trellis következő lépésében már két lehetséges kiindulási állapottal rendelkezik, a 00-val és az 10-val. A 00-ból ugyanolyan állapotátmenetekkel

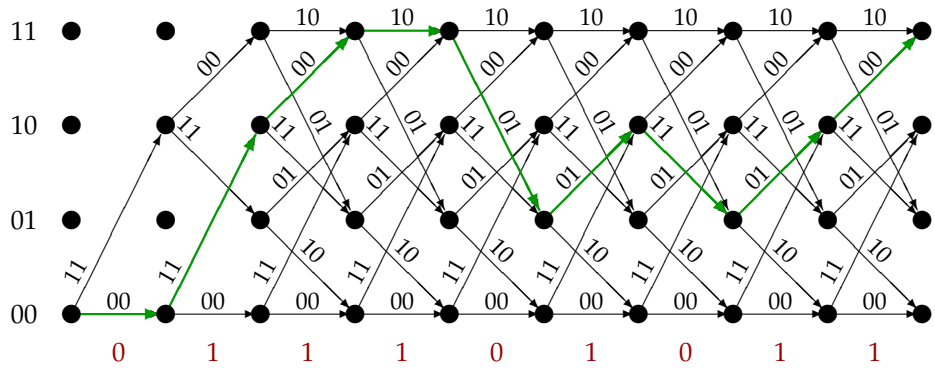
ugyanolyan állapotokba juthatunk, mint az előző lépésben. Az 10 állapotból az állapotátmeneti gráf szerint az 11, illetve a 01 állapotokba kerülhetünk, 00, illetve 11 kimenettel. Ezeket az átmeneteket a trellis második és harmadik oszlopa közötti nyílsorozat mutatja.

A harmadik lépés során már mind a négy állapot lehet kiindulási állapot, így az állapotátmeneti gráfnak mind a nyolc nyilát el kell helyezni a harmadik és a negyedik állapotoszlop között. Innentől kezdve már csak nulla bemenetet tételezünk fel, és csak azokat az állapotátmenteket tüntetjük fel a gráfról, amelyeken „0/xx" a felirat.

Az14.4 példában felrajzolt alap trellis közepén a vékony zöld vonallal bekeretezett részt akárhányszor megismételhetjük, így nem csak a kényszerhossznak megfelelő hosszúságú üzeneteket, hanem annál sokkal hosszabbakat is nyomon lehet követni trellis diagramon. A trellisben az egymást folytonosan követő, a csupa nulla tartalmú tárolókkal induló, csupa nullából álló állapotba visszatérő élsorozatokat **utak**nak nevezzük. Egy-egy út egy-egy kódszónak felel meg. Nézzünk egy példát.

14.5. példa: Vegyük a 14.4 példában feltüntetett alap-trellisszel rendelkező konvolúciós kódolót. Legyen a kódolni kívánt üzenet a 011101011.

Megoldás: A következő ábrán kicsit egyszerűsített trellis diagramon fogjuk nyomon követni az üzenethez tartozó utat. Az egyes állapotokat csak egy-egy pont jelöli; azt, hogy melyik állapotról van szó, az ábra bal széléről lehet leolvasni. Az átmeneteket jellemző nyilakra csak a kimenet került rá, a lefelé mutató nyilak 0 bemeneti bitet, a felfelé mutatók 1-et jelentenek. (A trellis végét elhagytuk, nem tudható ugyanis, hogy három 0-s karakter jön-e valóban.)



Az első bit 0, mint az az ábra alján az első és második pontoszlop között láthatjuk. Az 0 bemenetnek a vízszintesen mutató, vastagon kihúzott zöld nyíl felel meg az első és második állapotoszlop között. A kimenet látszik a nyíl feliratából: 00. A

második, 1-es bemeneti bit újra a 00 állapotban találja a rendszert, és átviszi az 10-ba, a kimeneten pedig két egyes jelenik meg. A harmadik, újfent 1-es bemeneti bit érkezésekor tehát az 10-s állapotban vannak a tárolóink, és az onnan kiinduló, felfelé mutató nyíl mentén átkerülnek az 11 állapotba. Ez alatt a lépés alatt a kimeneten 00 bitek generálódnak. Ezek alapján a többi lépés követhető: a vastag zöld vonal a bemeneti bitsorozatnak megfelelő útvonalat mutatja, a rajta látható felirat pedig a kimeneti biteket. A 011101011 bemenet hatására tehát a kimeneten a 00 11 00 10 01 01 11 01 00 bitpár-sorozat fog megjelenni (vagy egyszerűen a 001100100101110100 bitsorozat).

Ebből az ábrázolásmódból látszik, honnan van e diagramok elnevezése: a trellis angol szó, jelentése rácsozat (esetleg lugas).

A rácsozat szabályos struktúrájú, egy-egy sora egy-egy állapotot jelent, az $i+1$-edik oszlopa meg az i-edik bit beolvasása után a lehetséges állapotokat: **az i-edik mélységbeli csomópontokat**.

14.2. A konvolúciós kódok polinom-reprezentációja

A konvolúciós kódolást is lehet polinomokkal reprezentálni, gondoljunk csak arra, hogy a 14.1 példánkban szereplő kódoló áramkör tulajdonképpen két polinomszorzó eredményét fésüli össze. A példánkban szereplő bináris polinomok:

$$\begin{aligned} g_{11}(t) &= 1+t \\ g_{12}(t) &= 1+t+t^2. \end{aligned}$$

Azért van két indexe a $g_{ij}(t)$ polinomoknak, mert az első index jelöli az üzenetkeret i-edik bitjét, a második pedig a kódszókeret j-edik bitjét. Általános esetben tehát $k \times n$ polinommal írható le egy (N, K, n, k) paraméterű konvolúciós kód üzenetkeretei és kódszókeretei közötti kapcsolat. Ezeket a polinomokat egy mátrixba rendezhetjük a

$$\mathbf{G}(t) = \begin{pmatrix} g_{11}(t) & g_{12}(t) & \dots & g_{1n}(t) \\ g_{21}(t) & g_{22}(t) & \dots & g_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1}(t) & g_{k1}(t) & \dots & g_{kn}(t) \end{pmatrix} \tag{14.1}$$

szabály szerint.

14.6. példa: Adjuk meg a 14.2 példában szereplő konvolúciós kódolót leíró polinom-mátrixot.

Megoldás: A kódoló áramkör blokkvázlatából kitűnik, hogy az első (3-mal osztva 2 maradékot adó sorszámú) kimeneti biteket az első (páratlan sorszámú) bemeneti bitek, azok egy és két ütemmel eltoltjai hozzák létre. Ha ezt az ágat nézzük, az azt leíró polinom:

$$g_{11}(t) = 1 + t + t^2.$$

Ugyanezt a kimenetet a második (páros sorszámú) bemeneti bitek hozzák létre, ezeknek semmiféle eltoltjai nem befolyásolják az első kimenetet, így

$$g_{21}(t) = 1.$$

A második kimeneti ágon az első bemeneti ág bitjei és azok két ütemmel késleltetett verziói jelennek meg, tehát:

$$g_{12}(t) = 1 + t^2.$$

A második kimeneti biteket egyáltalán nem befolyásolják a második bemeneti bitek, így a $g_{22}(t)$ polinom 0 lesz. A harmadik kimenet ezek alapján egyértelmű, a $\mathbf{G}(t)$ polinom-mátrix pedig

$$\mathbf{G}(t) = \begin{pmatrix} 1+t+t^2 & 1+t^2 & 1+t^2 \\ 1 & 0 & 1 \end{pmatrix}. \tag{14.2}$$

Az üzenetkeretek sorozatát is lehet polinomokkal reprezentálni: az i-edik bemeneten megjelenő b_i,b_{k+i},b_{2k+i},... sorozathoz hozzá lehet rendelni a

$$b^{(i)}(t) = b_i + b_{k+i}t + b_{2k+i}t^2 + \ldots$$

polinomot minden $i = 1,2,\ldots,k$-ra. Az említett sorozatot úgy képezzük, hogy vesszük minden üzenetkeretből az i-edik bitet. E polinomokból tudunk egy sorvektort alkotni:

$$\mathbf{b}(t) = \begin{pmatrix} b^{(1)}(t) & b^{(2)}(t) & \ldots & b^{(k)}(t) \end{pmatrix} \tag{14.3}$$

Ezzel teljesen analóg módon a kódszókeretekhez is rendelhető egy polinom-vektor, jelöljük azt $\mathbf{c}(t)$-vel. Ekkor a kódolási szabály felírható a

$$\mathbf{c}(t) = \mathbf{b}(t) \cdot \mathbf{G}(t) \tag{14.4}$$

alakban.

14.7. példa: Kódoljuk az 1001110101 üzenetet a 14.2 példában szereplő kódoló áramkörrel. Használjuk fel a 14.6 példában felírt (14.2) polinom-mátrixot.

Megoldás: Először készítsük el a kódoló két bemeneti ágának polinomjait. Az első bemeneti ágra az üzenet első, harmadik, ötödik, hetedik és kilencedik bitje kerül, azaz 10100. A második ágon van a többi bemeneti bit: 01111. Így

$$\begin{aligned} b_1(t) &= 1 + 0t + 1t^2 + 0t^3 + 0t^4, \\ b_2(t) &= 0 + 1t + 1t^2 + 1t^3 + 1t^4. \end{aligned}$$

A (14.3) polinom-vektor tehát:

$$\mathbf{b}(t) = \Big(\big(1 + 1t^2\big) \qquad \big(1t + 1t^2 + 1t^3 + 1t^4\big) \Big),$$

amelyet a következőképpen szorzunk össze a (14.2) polinom-mátrixszal:

$$\begin{aligned} \big((1 + 1t^2) \quad (1t + 1t^2 + 1t^3 + 1t^4)\big) \quad \cdot \quad & \begin{pmatrix} 1 + t + t^2 & 1 + t^2 & 1 + t^2 \\ 1 & 0 & 1 \end{pmatrix} = \\ = \quad & (c_1(t) \quad c_2(t) \quad c_3(t)), \end{aligned}$$

ahol

$$\begin{aligned} c_1(t) &= (1 + 1t^2) \cdot (1 + 1t + 1t^2) + (1t + 1t^2 + 1t^3 + 1t^4) \cdot (1) = \\ &= 1 + 1t + 1t^2 + 1t^2 + 1t^3 + 1t^4 + 1t + 1t^2 + 1t^3 + 1t^4 = 1 + 1t^2 \end{aligned}$$

$$\begin{aligned} c_2(t) &= (1 + 1t^2) \cdot (1 + 1t^2) + (1t + 1t^2 + 1t^3 + 1t^4) \cdot (0) = \\ &= 1 + 1t^2 + 1t^2 + 1t^4 + 0 = 1 + 1t^4 \end{aligned}$$

$$\begin{aligned} c_3(t) &= (1 + 1t^2) \cdot (1 + 1t^2) + (1t + 1t^2 + 1t^3 + 1t^4) \cdot (1) = \\ &= 1 + 1t^2 + 1t^2 + 1t^4 + 1t + 1t^2 + 1t^3 + 1t^4 = 1 + 1t + 1t^2 + 1t^3 \end{aligned}$$

A kimenetek első 5 üteme tehát:

$$\begin{aligned} \mathbf{c}_1 &= 1\,0\,1\,0\,0, \\ \mathbf{c}_2 &= 1\,0\,0\,0\,1, \\ \mathbf{c}_3 &= 1\,1\,1\,1\,0, \end{aligned}$$

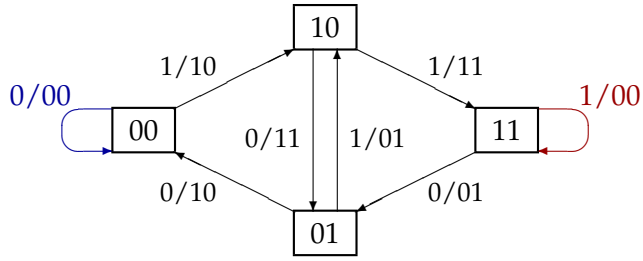
a kimenet ezekből összefésülve, $\mathbf{c}_1$ első bitje után c_2 első bitje, majd c_3-é, ezután c_1 második bitje, c_2 második bitje, és így tovább: 111 001 101 001 010. Szóközöket az egyes ütemek határán csak a jobb átláthatóság kedvéért hagytunk.

14.3. Katasztrofális kódolók

Nem minden lehetséges kódoló lesz jó, előfordulhat az, hogy egy idő után csupa nullából álló sorozat jön ki a kódolóból akkor is, ha nem csupa 0

kerül a bemenetére, hanem egy ciklikusan ismétlődő, 1-eseket is tartalmazó bitsorozat. Az ilyen kódolókat **katasztrofális**nak nevezik.

Az, hogy a konvolúciós kód katasztrofális, látszik az állapotátmenet-gráfján:



Ha ki tudunk jelölni a gráfban egy olyan zárt hurkot, amelynek az élein végig csupa nulla kimenet keletkezik, a bemenet mégsem csupa 0, akkor a kódoló katasztrofális. A fenti gráfon egy ilyen hurok van pirossal jelölve. Nem számít a keresett hurkok közé az ábrán kékkel húzott hurok: az, amelyik a 0-kkal feltöltött tárolókkal jellemzett állapotból 0 bemeneti bitek hatására önmagába mutat vissza. Ilyen hurok minden kódoló gráfján szerepel.

Ha $k = 1$, azaz mindig csak egy bitből áll az üzenetkeret, akkor arra, hogy a kódoló ne legyen katasztrofális, létezik egy szükséges és elégséges feltétel: A $k = 1$ üzenetkeret-hosszú konvolúciós kódok közül nem katasztrofális az, amelynek a $g_{11}(t)$,$g_{12}(t)$,...,$g_{1n}(t)$ generátorpolinomjainak a legnagyobb közös osztója az 1.

14.4. Távolságprofil, szabad távolság

Mivel a blokk-kódoknál a kódszavak közötti minimális távolság (kódtávolság) fontos volt a javítható hibák számának megbecsüléséhez, a konvolúciós kódoknál is szeretnénk egy hozzá hasonló mennyiséget bevezetni. Itt nem beszélhetünk kódszavakról, mivel a kódszókeretek nem függetlenek egymástól, s nem lehet egyértelműen azt mondani, hogy az üzenetnek egy adott szakasza hozza létre a kód egy meghatározott szakaszát.

Tegyük fel, hogy a kódszókeretek hossza n. Legyen d_i^* a lehetséges kimeneti jelsorozatok első i darab kódszókeretéből képezett vektorok Hamming-távolságai közül a minimális. A d_1^*,d_2^*,... mennyiségeket a konvolúciós kódunk **távolságprofil**jának nevezik. Ahogy növeljük a figyelembe vett kódszókeretek számát – így a vizsgált vektorok hosszát –, a

köztük lévő Hamming-távolság nem csökkenhet:

$$d_1^* \leq d_2^* \leq d_3^* \leq \dots .$$

Mivel a konvolúciós kódolás lineáris folyamat, az egyes d_i^* távolságok kiszámításakor nem kell minden lehetséges, i hosszúságú kimeneti bitsorozatnak minden másik i hosszúságú kimeneti bitsorozattól vett Hamming-távolságát vizsgálni, elég csak a tiszta 0-ból álló kimenettől vett távolságokat nézni.

A d_i^*-okból álló monoton növekvő sorozat határértékét hívják a konvolúciós kód **szabad távolság**ának, d_∞-nel jelölik, és a

$$d_\infty = \max_i d_i^*$$

formulával definiálják.

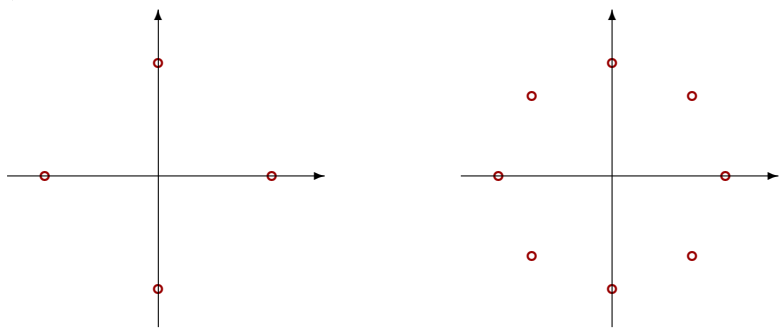
A csatornakódolás során az a cél, hogy minél nagyobb legyen a kódszavak közötti távolság. A **kódolási nyereség**en a kódolatlan és a kódolt jelsorozat szabad távolságainak arányát értjük, decibel skálában:

$$\mathcal{G} = 20 \lg \frac{d_\infty}{d_{\text{ref}}} \tag{14.5}$$

A kódolatlan sorozat a referencia, innen a nevező indexe. (A $\mathcal{G}$ az angol gain–nyereség szóból származik.)

14.4.1. Ungerboeck-kódok: komplex kódábécé használata

Az Ungerboeck-kódok a (0,1) bináris számok helyett 2^n darab komplex számot használnak jelkészletként. Látványos ha az ábécé elemei a komplex egységvektor $i \cdot 360^\circ/2^n$ szöggel való elforgatottjai, $i = 0,1,\dots,2^n-1$-re. Az ilyen kódábécé elemeit fázismodulációval (2^nPSK) könnyen a csatornára tudjuk bocsátani: minden elemnek $i \cdot 360^\circ/2^n$ szögű fázistolás felel meg. Példaként álljon itt a 4PSK és a 8PSK jelkészlete (a piros karikák a komplex számok):



A kódoló a k bitből álló üzenetkereteket képezi komplex számokból álló sorozatokba. A $\mathbf{c}_1$, $\mathbf{c}_2$ (esetleg végtelen) kódszavak távolsága a hozzájuk rendelt komplex vektorok hagyományos, euklideszi távolsága:

$$d(\mathbf{c}_1,\mathbf{c}_2) = \left(\sum_i |c_{1i} - c_{2i}|^2 \right)^{1/2}$$

Például a 4PSK ábécé elemeinek távolsága $\sqrt{2}$, a 8PSK-é növekvő sorrendben $\sqrt{2-\sqrt{2}}$, $\sqrt{2}$, $\sqrt{2+\sqrt{2}}$ és 2.

4PSK jelkészletet lehet például két kimenetű konvolúciós kódolókkal előállítani, ha a $00 \mapsto 0°$, $01 \mapsto 90°$, $10 \mapsto 180°$ és $11 \mapsto 270°$ hozzárendeléseket alkalmazzuk a kódoló kimenete és a fázistolások között. Hasonlóképpen 8PSK-t három kimenetelű kódolóval lehet megfeleltetni.

Ha kilépünk a számegyenesről a komplex számsíkra, akkor amellett, hogy a kapott jelsorozat mindenféle köztes transzformációk nélkül alkalmas modulációra, a kódolási nyereséget is tudjuk növelni.

15. A Viterbi-dekódolás

A konvolúciós kódok azért is előnyösek, mert van egy jól algoritmizálható, látványos dekódoló algoritmusuk, amely nem köti meg a lehetséges hibák számát, nem tartalmaz számításigényes mátrix-invertálásokat, de még lineáris egyenletrendszereket sem kell megoldani az üzenet reprodukálásához. Egyetlen dolog kell csak, hogy a dekódoló ismerje a trellist, amely a kódot létrehozta.

Tegyük fel, hogy a csatorna bemenetére a $\mathbf{c} = (c_1,c_2,\ldots,c_N)$ (bináris) vektort adtuk, a $\mathbf{v} = (v_1,v_2,\ldots,v_N)$-t pedig a kimeneten vettük. A vevő ismeri a kódoló trellisét, de nyilván nem tudja azt, hogy a trellis melyik útján jött létre $\mathbf{c}$.

A Viterbi-algoritmus a trellis minden éléhez egy súlyozó faktort rendel, mintha minden él más és más hosszúságú lenne. A különböző élekhez rendelt hosszakat nevezik mértéknek, vagy **metrikán**ak. Így a lehetséges utaknak is tudunk hosszát definiálni, ha összeadjuk az éleinek a metrikáját. Az élek metrikáját úgy választják meg, hogy arányos legyen a

$$\sum_{i\in\mathcal{D}} \log_2 p(v_i|c_i)$$

kifejezéssel, ahol az összegzés az adott él által létrehozott kódszókeretben található c_i bitekre hajtandó végre. Az összegzés az aktuális kódszókeret minden indexére vonatkozik, ezt az indexhalmazt jelöljük $\mathcal{D}$-vel. A formulában szereplő feltételes valószínűség azt takarja, hogy mekkora a v_i bit vételének a valószínűsége, ha a c_i-t adtuk a csatornára. A Viterbi-algoritmus a lehetséges utak közül keresi a maximális valószínűségűt. A kódszavak ilyen dekódolása ekvivalens azzal, hogy ha a lehetséges kódszavak közül *maximum likelihood* döntéssel választunk a vett jelek alapján.

Bináris szimmetrikus csatorna esetén belátható, hogy a *maximális metrikájú út* helyett lehet azt keresni, amelyik a vett vektortól *minimális Hamming-távolság*ra van.

Az egész gondolatmenet feltételezi azt, hogy a csatornánk szinkron és az egyes szimbólumoknak a csatornán való áthaladása egymástól független esemény. Ekkor lehet ugyanis a $p(v_{i_1}v_{i_2}\ldots v_{i_k}|c_{i_1}c_{i_2}\ldots c_{i_k})$ valószínűséget $p(v_{i_j}|c_{i_j})$ valószínűségek szorzataként felírni, így az összvalószínűség logaritmusa a részvalószínűségek logaritmusainak összegeként.

A Viterbi-algoritmus a következő ciklust hajtja végre:

- Veszi az i-edik kódszókeretet.
- Ennek ismeretében a trellis $i-1$-edik és i-edik oszlopa közötti ágaknak kiszámolja a súlyát.
- előhívja a memóriából az $i-1$-edik mélységbeli csomópontokhoz tartozó összesített súlyt. Egy-egy ilyen állapot összesített súlyához hozzáadja a belőle kiinduló ágak súlyait. A kapott értékeket azokhoz az i-edik mélységbeli állapotokhoz rendeli hozzá, amelyekbe mutatnak. Ekkor minden i-edik mélységbeli állapothoz annyi új összesített súly tartozik, ahány él fut belé. (A refpelda:konvTR példánkban, és minden 1 üzenetkeret-hosszú, bináris kódolónál két érték fog egy csomópontra jutni.)
- Az állapotokhoz rendelt súlyok közül kiválasztja a maximálisat, ez lesz a csomópont új összesített súlya. A súlyt és a hozzá vezető útvonalat (az elejétől kezdve) eltárolja. Ezek az útvonalak a *túlélők*. A nem használt össz-súlyokat a dekódoló eldobja. Azokat a trellis $i-1$-edik és i-edik mélységi szintje közötti éleket, amelyek nem tartoznak a túlélő útvonalba, törli az ábrából.

15.1. példa: Tegyük fel, hogy a 14.4 példában vizsgált kódolónk 011101011 bemenetére adott választ adjuk a csatornára, és ott két helyen, a harmadik és az ötödik pozícióban hiba történik. Ekkor a

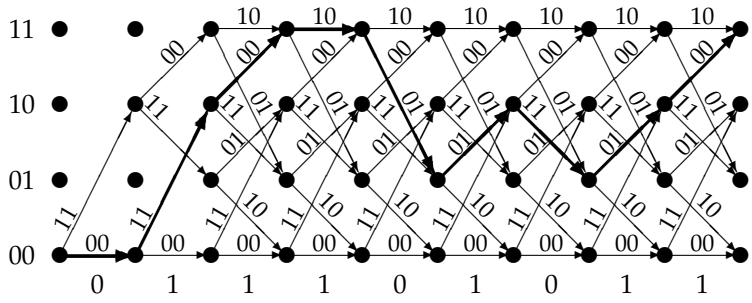
00 11 00 10 01 01 11 01 00

bitsorozat helyett a

00 01 10 10 01 01 11 01 00

lesz a csatorna kimenetén.

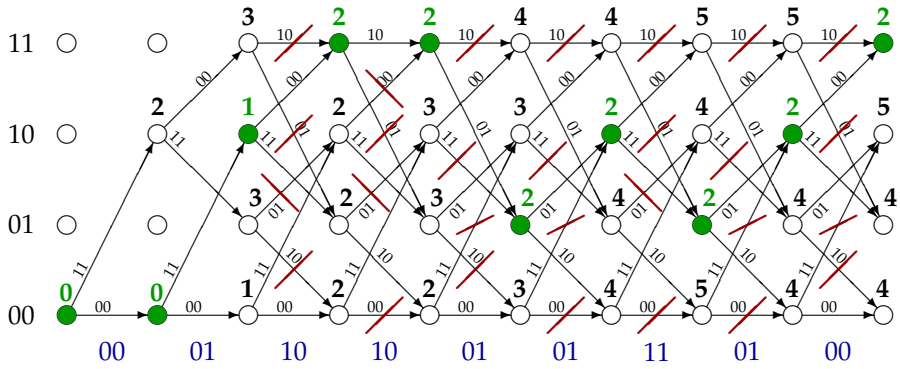
Megoldás: Az alábbi ábrán – emlékeztetőül – kódoló trellise és a kódszó útvonala található.



A dekódolás menete a következő. A metrika itt a vett bitpár és az adott élhez tartozó bitpár közötti Hamming-távolság, így nem a maximumot, hanem a minimumot kell keresni. Nézzük a lépéseket. A kiindulási állapot a 00 állapot, az ő össz-súlya 0, ez látszik a következő ábrán az első mélységi csomópont felett vastagon (és zölden) szedve.

Az első bemeneti bitpáros a 00, amely a trellis első és második pontoszlopa között alul fel is van tüntetve. A vett bitpáros az alsó állapotátmenetet jellemző él 00 kimenetétől nem tér el, így a kettő Hamming-távolsága 0, ezt hozzáadva a kiindulási 0 össz-súlyhoz 0-t kapunk, ez lesz a 00 állapothoz tartozó második mélységi csomópont összsúlya. Az első bemenetnek a másik lehetséges állapotátmenet kimenetétől – az 11-től – való Hamming-távolsága 2, így a második mélységbeli 01 állapot össz-súlya 2, ami az állapot felett vastag szedéssel fel is van írva. Mivel az első lépésekben még csak egy-egy él fut be az egyes állapotokba, a túlélő él egyértelmű.

A második és harmadik állapotoszlop között már négyféle átmenet lehetséges, ezek mindegyikének ki kell számolni a 01 bemenettől vett Hamming-távolságát, és hozzáadni a kiindulási állapot össz-súlyához, hogy megkapjuk a végállapotok össz-metrikáját. Az eredmények a harmadik mélységi csomópontok felett vastag szedéssel láthatók.



A következő, 10 vett bitpáros újabb problémát vet fel. A harmadik és negyedik mélységbeli csomópontok között ugyanis 8-féle állapotátmenet lehetséges, így minden állapotba 2 él fut be, ezek közül el kell dönteni, melyik a túlélő. Nézzük azokat az átmeneteket, amelyek a legalsó, 00-s állapotba érkeznek:

- Az első ilyen átmenet kiindulási állapota az előző oszlopbeli 00 állapot, melynek addigi össz-súlya 1 volt. Az átmenethez tartozó 00 kódoló-kimenetnek az 10 bemenettől vett Hamming-távolsága 1, így ezen az útvonalon az össz-súly $1 + 1 = 2$ lesz.
- A másik lehetséges, 00-ba torkolló állapotátmenet kiindulási pontja a 01 állapot, melynek össz-súlya az előző lépésben 3 volt. Ehhez kell hozzáadni az él 10 kimenetének az 10 bemenettől vett Hamming-távolságát, azaz 0-t. Így ezen az útvonalon a 00 állapot össz-metrikája $3 + 0 = 3$ lenne.

A két össz-súly közül nyilván az első a kisebb, így a túlélő él a $00 \to 00$ él lesz, a másikat törölni kell: pirossal áthúztuk.

A többi állapotra ezek a lépések egyszerűen általánosíthatók, az eredmények az ábráról leolvashatók. A diagramon nem tüntettük fel az élekhez tartozó mértéket, csak áthúztuk pirossal a nem túlélőket. A csomópontok fölött vastagon szedve a hozzájuk vezető túlélő útvonal összesített metrikája látható. Ha mindkét befutó él azonos értéket hozott, akkor véletlenszerűen döntöttünk. Az ábra alján látszanak az adott lépésben bejövő bitpárok.

Az utolsó lépés után az 11 állapotnak van a legkisebb össz-súlya – méghozzá 2 – az lesz a végső túlélő él végpontja. Ebből a pontból kiindulva visszafelé meg tudjuk találni a túlélő útvonalat, hiszen minden csomópontba csak egyetlen túlélő él fut be. Ez a végső túlélő útvonal, ennek a csomópontjai vannak zöld pöttyökkel kiemelve. Figyeljük meg, hogy az ideális útvonal mentén csak azokban a lépésekben nőtt a – zöld színnel kiemelt – metrika, ahol a csatornán hiba keletkezett, vagyis a második és a harmadik lépésben. Az össz-súly, 2 is a keletkezett hibák számát adja meg.

Ha követjük a minimális metrikájú végállapothoz vezető utat, megkapjuk a csatorna bemenetére adott bitsorozatot 011101011-et. Ehhez persze tudni kell a kódoló trellisének éleit létrehozó bemeneti biteket.

A Viterbi-dekódolásnál is előfordulhat, hogy rossz üzenetté dekódoljuk a vett jeleket, különösen akkor, ha a csatorna többet hibázik. Gondoljunk csak bele, hogy ha a 15.1 példában dekódolt 00 01 10 10 01 01 11 01 00 bitsorozat nem a 00 11 00 10 01 01 11 01 00 bitpáros -sorozatból keletkezett két hibával, hanem az 11 00 10 10 01 01 11 01 00 kódból jött létre a csatorna 3 hibája után, akkor is ugyanazzá a 011101011 üzenetté dekódolnánk, holott az 111101011-ből keletkezett.

Az, hogy a Viterbi-algoritmussal milyen bithiba-arányú átvitel mellett tudunk hiba nélkül dekódolni, attól függ, hogy mekkora az alkalmazott konvolúciós kódolás szabad távolsága.

Hivatkozások

[1] Shannon, C. E., „A Mathematical Theory of Communication", *Bell System Technical Journal*, 1948.

[2] Nyquist, H., „Certain Factors Affecting Telegraph Speed",*Bell System Technical Journal*, 324, 1924.

[3] Nyquist, H., „Certain Topics in Telegraph Transmission Theory",*A.I.E.E.Trans.* **42**, 617, 1928.

[4] Hartley, R. V. L., „Transmission of Information", *Bell System Technical Journal*, 535, 1928.

[5] Gordos G., Varga A., Adatátvitel és adatfeldolgozás, Tankönyvkiadó, Budapest, 1971.

[6] Shannon, C. E., Weaver, W., „The Mathematical Theory of Communication", University of Illinois Press, Urbana, 1949, és magyar fordítása, „A kommunikáció matematikai elmélete", OMIKK, Budapest, 1986.

[7] Györfi L., Győri S. és Vajda I., „Információ- és kódelmélet", Typotex, Budapest, 2005.

[8] Linder T., Lugosi G., „Bevezetés az információelméletbe", Tankönyvkiadó, Budapest, 1990.

[9] „Híradástechnika", főszerkesztő Géher K., Műszaki Könyvkiadó, Budapest, 1993.

[10] Ferenczy P., Hírközléselmélet, Tankönyvkiadó, Budapest, 1972.

[11] Ferenczy P., „Video- és hangrendszerek", Műszaki Könyvkiadó, Budapest, 1986.

[12] www.cableworld.hu.

[13] Fegyverneki S., „Információelmélet", Összefoglaló – Segédlet, Miskolci Egyetem, 2002.

[14] Vassányi I., „Információelmélet", Kivonatos jegyzet,
Veszprémi Egyetem, 2002-2003.

[15] Csurgay Á., Simonyi K., „Az információtechnika fizikai alapjai – Elektronfizika",
Mérnöktovábbképző intézet, 1997.

[16] Bronstejn, I. N., Szemengyajev, K. A., Musiol, G. és Mühlig, H., „Matematikai kézikönyv", Typotex, Budapest, 2000.

[17] „Matematikai kislexikon", főszerkesztő: Farkas M., Műszaki Könyvkiadó, Budapest, 1972.

[18] Rózsa P., „Lineáris algebra és alkalmazásai", Műszaki Könyvkiadó, Budapest, 1976.

[19] Christopoulos, C., Skodras, A., Ebrahimi, T., „The JPEG2000 Still Image Coding System: an Overview", IEEE Trans. Cons. El., **46** 1103-27, 2000.

Tárgymutató