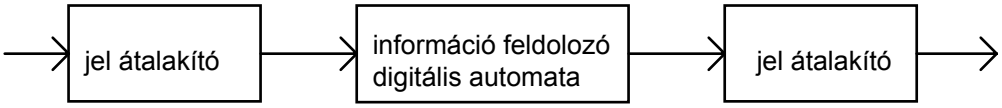
Kódoláselméleti alapfogalmak

Benesóczky Zoltán 2005

Ez összefoglaló digitális technika tantárgy kódolással foglalkozó anyagrészéhez készült, az informatika szakos hallgatók részére. Több-kevesebb részletességgel az előadásokon elhangzottakat foglalja össze.

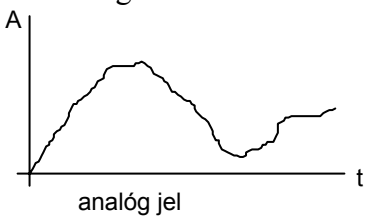
Bevezetés

Az információ átvitel ill. feldolgozás során az információt elektromos jellé kell alakítani, hogy az elektromossággal működő információ feldolgozó géppel feldolgozható legyen. Ezt az átalakítást analóg esetben analóg kódolásnak is nevezhetjük. Itt azonban digitális kódokkal foglalkozunk, ezért az alábbiakban a két kód közötti különbségre világítunk rá.



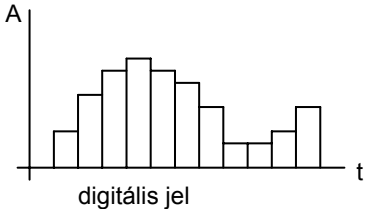
Az információ feldolgozó gép kódokkal dolgozik

analóg kód: amplitúdóban és időben folytonos (véges idő alatt végtelen információt hordoz, de a mindig jelenlevő zajok miatt csak véges használható fel)



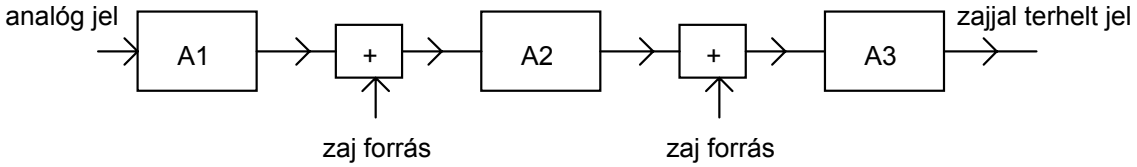
digitális kód: amplitúdóban és időben diszkréten értelmezett (véges idő alatt véges információt hordoz)

Analóg jelből digitális jel mintavételezéssel (adott időpontban mekkora a jel analóg értéke) majd kvantálással (a mintavett analóg érték *n* bites digitális számmá alakításával) nyerhető.



Pl. Egy hagyományos telefonnál a hangot (levegő nyomás változást) a mikrofon elektromos jellé (analóg kódolás), feszültség változássá alakítja. Ez erősítővel felerősítés után továbbhalad a telefonközponton keresztül egy másik telefonhoz, ahol a telefon hallagtóban újra hangnyomás változássá alakul (analóg dekódolás).

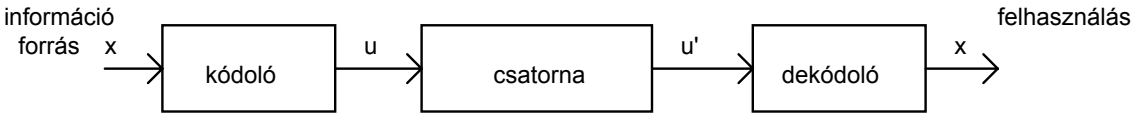
Az átvitel során a jelhez zaj adódik, amelyet szintén felerősít az erősítő. Mennél több analóg jelfeldolgozó áramkörön halad át az analóg jel, annál jobban romlik a jelnek a zajhoz képesti értéke (jel/zaj viszony.) Az analóg jelfeldolgozók méretét ez jelentősen korlátozza.



A zajforrások sokfélék lehetnek. Analóg áramköröknél a hálózati 50Hz az egyik nagy zajforrás. Rádiós átvitelnél a légköri zavarok okoznak jelentős zajt. De ha minden külső zajt sikerülne is kiküszöbölni, maguk az áramkörök akkor is termelnének valamekkora zajt (termikus zaj).

A digitalizált információ az analóghoz képest sok nagyságrenddel pontosabban vihető át, ill. dolgozható fel. A digitális információ helyes átvitelét egyrészt áramkörileg az analóghoz képest sokkal nagyobb biztonsággal meg lehet oldani, mert 1 bitnyi információ továbbításakor csak 2 érték fordulhat elő pl. a 0-át és az 1-et reprezentáló feszültség vagy áram érték, és ez viszonylag nagy zaj mellet is jól megkülönböztethető megfelelő áramköri megoldásokkal. Másrész megfelelő kódolással az előforduló hibákat jelezni, sőt javítani is lehet.

A digitális kódokat és kódolást az információ átvitel oldaláról közelítjük meg.



A kódolás magába foglalja az
ún. *forrás kódolást*, melynek célja az információ tömörítése,
a *titkosítást*, melynek célja, hogy illetéktelen ne tudja visszafejteni az információt,
a *csatorna kódolást*, melynek célja a digitális információ hibátlan átvitele

Egyszerű példa digitális információ átvitelre:

információ forrás:	pl: magyar szöveg
kódolás:	pl: a magyar szöveg betűnkénti kódolása bináris számokká (pl. ASCII kód)
csatorna:	pl: két állapotú, fizikailag két feszültségszint reprezentálja
dekódolás:	pl: a bináris számok magyar karakterekké alakítása

A csatorna lehet több állapotú is de itt csak a két állapotúval foglalkozunk

$A = \{a_1, a_2, \ldots a_n\}$ forrás ABC

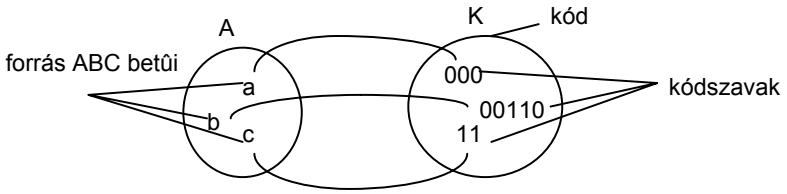
Bináris kód esetén a kód ABC 2 elemű: {0, 1}

$K = \{\alpha_1, \alpha_2, \ldots \alpha_n\}$ a kód ABC betűiből képzett véges hosszúságú sorozatok halmaza

Pl. bináris kód esetén K = {00, 010, 0111, …}

A 2-es számrendszrű számok egy számjegyét *bit*-nek nevezik, a *binary digit* (bináris számjegy) rövidítéseként

Betű szerinti kódolás:
Az *A* halmaz minden karakterének megfeleltetjük K egy-egy elemét
($a_1 \rightarrow \alpha_1$, $a_2 \rightarrow \alpha_2, \ldots$)



A K halmazt *kódnak* nevezzük.
A K halmaz elemei a *kódszavak*, de a kódszavakat is szokás kódnak nevezni.

Kódok osztályozása:

karakterkészlet alapján:	- bináris, {0,1} - nem bináris pl. {0, 1, 2}
a kódszavak hosszúsága alapján:	- fix hosszóságú, Pl. {100, 010, 111} - változó hosszúságú Pl. {01, 110,1111}
alkalmazási cél alapján:	- aritmetikai (1-es komlemens, offszet stb.), - pozició (Pl. Gray kód, Jophnson kód) - hibajavító (Pl. Hamming kód) stb...

***1. példa**:* Mekkora fix hosszúságú kód szükséges, ha a kódolandó halmaz számossága N=9, a kód ABC betűinek száma k=3?
Egy betűvel 3 elemet kódolhatunk, két betűvel 3*3-at, x betűvel 3^x -edikent.

$$k^x \geq N,\ x = \lceil \log_k N \rceil$$

x=2

Az információ mértékegysége
Mennyi információt hordoz *egy karakter*, ha a *k*-féle van belőle és mindegyik egyforma valószínűséggel fordul elő az átküldendő sorozatban? Az egyes karakterek azt az információt hordozzák, hogy a *k*-féle lehetőségből éppen melyiket választottuk. Pl. ha azt akarjuk átküldeni a csatornán, hogy egy kockával egymás után dobva éppen mely számok jöttek ki, akkor az aktuálisan vett karakter megmondja, hogy a 6 féle lehetőségből épp melyik jött ki. Gyakorlati esetekben a választási lehetőségek száma nagyon nagy lehet. Nem csak egy karakter által hordozott információ, hanem egymás után írt karakterek (karakter sorozat) által hordozott információ is érdekes lehet.

Mint az első példából is kiderült, *k*-féle karakterből *n* darabot egymás után írva k^n -féle lehetőség van (ennyi féle üzenet küldhető). Ha egy esemény valószínűsége p, annak bekövetkezését úgy is értelmezhetjük, hogy *1/p* féle lehetőségből pont az következett be. A dobókockánál egy konkrét szám pl. a 6-os dobásának valószínűsége *p = 1/6,* vagyis *1/(1/6)=6* féle esemény közül pont a 6-os jött ki.

Többek között ezért az információ mértékéül az választjuk, hogy az hány biten kódolható. Az előző gondolatmenet alapján egy *p* valószínűségű esemény tehát *m=1/p* féle lehetőség közüli 1 beközvetkezéseként fogható fel, így az bináris kódolással

$\log_2 1/p = -ld\ p$ biten kódolható vagyis ennyi információt hordoz (*ld*-vel a 2-es alapú logaritmust jelöljük).

Ez a kis példa talán érzékelteti, hogy miért igaz az, hogy a *p* valószínűségű esemény bekövetkezése által hordozott információ: $-ld\ p$ bit

Információ sebesség

Információ sebesség definíció szerint: $R = \lim_{t\to\infty} \frac{ld\, N(t)}{t}$, ahol *N(t)* a *t* idő alatt átvihető üzenetek száma, így *ld N(t)* a *t* idő alatt átvihető információ. Mivel az idő a határértékképzésben a végtelen felé tart, az információ sebesség annak a mérőszáma, hogy egységnyi idő alatt, *hosszú idő átlagában* hány bit információ megy át a csatornán (bit/időegység).

2. példa: Ha adott az egyes karakterek átviteli költsége ($a_i : t_i$), mekkora a *t* költséggel (idő alatt) még éppen átvihető üzenetek száma és az információ sebesség? Hogy egyszerűbb legyen a feladat tételezzük fel, hogy az összes karakter átviteli költsége egyformán 1.

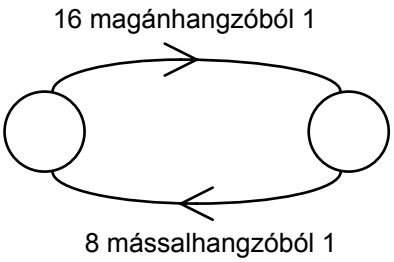
a. $\{s_1, s_2, \ldots s_D\}$ $t_i = 1$

- minden karakter átviteli költsége egyformán 1,
- D féle karakterből választhatunk,
- egységnyi idő alatt D féle üzenet lehetséges
- 2 egységnyi idő alatt D*D (a második karakter is D féle lehet)...

$N(t) = D^{\lfloor t \rfloor}$ (a kitevő *t* egész része)

Az információ sebesség: $R = \lim_{t\to\infty} \frac{t\, ld\, D}{t} = ld\, D$

b. Ha *szintaktikai megkötés* van, bonyolultabb a feladat. Legyen $t_i = 1$ és a szintaktikai megkötést az alábbi *gráffal* adjuk meg:



A gráf azt fejezi ki, hogy minden magánhangzót mássalhangzónak kell követni és fordítva.

Mivel $k2t_i$ időnként a kezdő állapotba kerül, és hosszú ídö távlatában $k2t_i$ helyett 2t-t írhatunk, ezért, jó közelítés, hogy

$$N(2t) = (8*16)^t$$

Amiből adódik, hogy

$$R = \lim_{t\to\infty} \frac{ld\, N(2t)}{2t} = \lim_{t\to\infty} \frac{t\, ld\,(16*8)}{2t} = \frac{ld\,16 + ld\,8}{2}$$

Bonyolultabb gráf esetén nehéz kiszámítani.

Változó hosszúságú kódolás (forrás kódolás)

Itt a kódolás célja az információ tömörítése. A változó hosszúságú kódolás esetén a kódszavak hossza különböző lehet, de ügyelni kell a megfejthetőségre.

Megfejthetőség
Egy kód megfejhető, ha a kódszavaiból előállított tetszőleges üzenet egyértelműen felbontható a kód kódszavaira.

A következő kód nem megfejthető {a: 00, b:01, c:11, d:0001}, ugyanis az abd kódolásával adódó 00010001 üzenet abd,dab, abab és dd üzenetként is értelmezhető. A problémát az okozta, hogy voltak kódszavak, amelyek más kódszó után írt karakterek segítségével generálhatók.

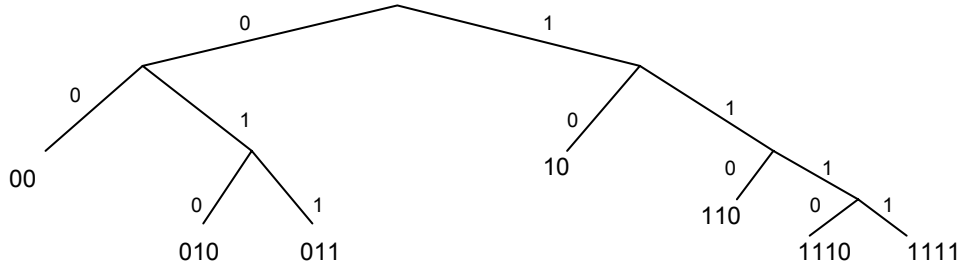
Prefix tulajdonság*:* egyik kódszó sem folytatása egy másiknak.
pl: a {01, 001, 100, 1100} kód prefix

Ha a kódszavak hossza egyforma, akkor a kódolt üzenet biztosan megfejthető, hiszen prefix.

A prefix tulajdonság a megfejthetőség elégséges, de nem szükséges feltétele.

Feladat: Próbáljon nem prefix, de megfejthető változó hosszú kódot generálni, 4 eseményhez!
A {10, 100, 1000, 10000} kód nem prefix, de megfejhető, mert a kódszavak határát jelzi az első karakter.

A prefix tulajdonságú kód fagráf segítségével generálható. Pl. bináris prefix kódot bináris fával generálhatunk. A gyökértől egy-egy levélig haladva megkapjuk a kódokat.



Így konstruálva egy k2 kód csak akkor folytatása egy k1-nek, ha k2-höz k1-en keresztül lehet eljutni, de akkor k1 nem levele a gráfnak, hanem egy belső pontja..

Ha *az információs csatorna zajmentes*, akkor a minél rövidebb (olcsóbb, gyorsabb, tömörebb) üzenet a cél.

Kód költsége

Egy információ átvitelének (vagy tárolásának) költsége annál nagyobb, minél hosszabb. Ezért zajmentes esetben a kódolás elsődleges célja, az *átlagos kódhossz* csökkentése.

Ha adott egy-egy karakter (a_i) előfordulási valószínűsége, p_i, az a_i hez tartozó kódszó hossza l_i és $\sum p_i = 1$ (teljes eseményrendszer, vagyis minden esemény előfordul 0-nál nagyobb valószínűséggel), akkor az M számú karakterből álló üzenet kódjának átlagos hossza a következő gondolatmenettel számítható:

- az M karaktert kódoló üzenetben átlagosan $n_{a_i} = Mp_i$ szer fordul elő az a_i karakter,

mert $p_i \cong \frac{n_{a_i}}{M}$, $M \to \infty$ *eseté n*

- a üzenetben előforduló a_i karakterek kódjának együttes hossza: $Mp_i l_i$

- a teljes kódolt üzenet várható hossza: $M\sum_i p_i l_i$

- a *kód költsége:* $\lim_{M\to\infty} \frac{\textit{Kódolt üzenet hossza}}{M} = \sum_i p_i l_i$ ennek létezik alsó korlátja, s ez bizonyíthatóan a következőkben definiált entrópia.

Definíció: Ha egy X valószínűségi változó eloszlása {p*1*,p*2*,p*3*,..p*n*} akkor a következő kifejezés az X entrópiája (Shannon formula)

$$H(X) = \sum_i p_i ld \frac{1}{p_i} = -\sum_i p_i ld\, p_i$$

Összehasonlítva az átlagos kódszóhosszat az alsó korlátját adó entrópiával, azt mondhatjuk, hogy ideális kódolás esetén (ami többnyire nem valósítható meg), a p*i* valószínűségű eseményt

$ld \frac{1}{p_i}$ számú biten kellene kódolni (ekkor érnénk el a költség elvi minimumát).

Ezt úgy is értelmezhetjük, hogy a p*i* valószínűségű esemény ennyi bit információt hordoz, mint ahogy ezt már említettük. Tehát egy esemény annál több információt hordoz, mennél valószínűtlenebb. (Ha a lottón 5 találatunk van, az sokkal több információ, mint ha megtudjuk, hogy egyáltalán nincs.)

Általános elv, hogy a ritkábban előforduló eseményt (karaktert) kódoljuk hosszabb kóddal.

Feladat:

Próbáljon mennél jobb változó hosszúságú kódolást készíteni az alábbi esetben! Ügyeljen a megfejthetőségre is!

{a1:0.3, a2:0.3,a3:0.2, a4:0.1, a5:0.05, a6: 0.05}

Az elvi ideális hossz eseményenként:

l1:1.73 l2:1.73 l3:2.32 l4:3.32 l5:4.32 l6:4.32

Elvileg elérhető átlagos kódszó hossz: 2.27

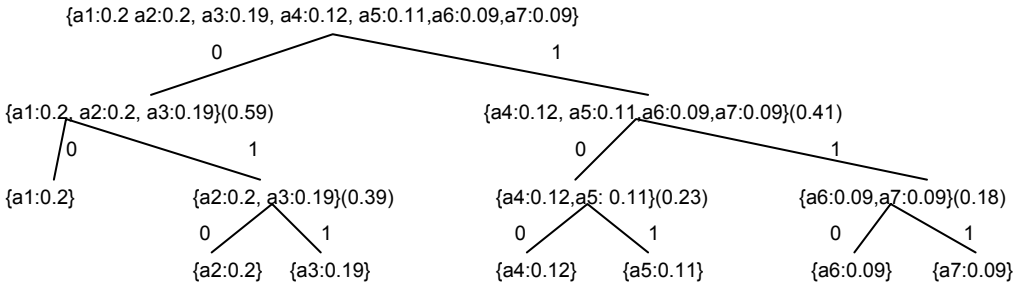
Shannon kód

Ez közel optimális hosszúságú prefix kódot eredményez. A kódolás algoritmusa a következő:

Állítsuk valószínűség alapján sorrendbe az eseményeket.

Pl. {a1:0.2 a2:0.2, a3:0.19, a4:0.12, a5:0.11,a6:0.09,a7:0.09}

Osszuk két olyan részre az így kapott halmazt, melyekben a valószínűségek összege közel egyforma, különböztessük meg őket egy bittel, majd az így kapott részhalmazokra is végezzük ezt el, amíg lehetésges. A végén az így kapott bináris fagráf gyökerétől az egyes levelekig (egyedi események) haladva egymás után írva a biteket, megkapjuk az adott eseményhez tartozó kódszót.



A kapott kód:

a1: 00, a2: 010, a3: 011, a4: 100, a5:101, a6: 110, a7:111

{a1:0.2 a2:0.2, a3:0.19, a4:0.12, a5:0.11,a6:0.09, a7:0.09}

Ennek költsége: 2*0.2+3*0.2+3*0.19+3*0.12+3*0.11+3*0.09+3*0.09=2.8

Az elvi ideális kódszó hosszak:

l1:2.32, l2:2.32, l3:2.4, l4:3.06, l5:3.18, l6:3.47, l7:3.47

Elvi alsó költséghatár: 2.73

Láthatóan elég jól megközelítette a kapott kódolás.

Ha minimális bitszámon, azonos hosszúságban kódoltunk volna, akkor 3 lett volna az átlagos kódszó hossz.

Feladat:

Készítsen Shannon kódolást az alábbi esetben, számítsa ki az átlagos kódszó hosszat is!

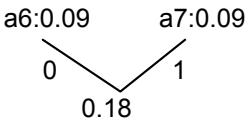
{a1:0.3, a2:0.3,a3:0.2, a4:0.1, a5:0.05, a6: 0.05}

Huffmann kód

A Huffmann kód változó hosszúságú optimális költségű prefix kód. A kódolás algoritmusát a következő példán mutatjuk be:
Adott az alábbi kód és valószínűségek:

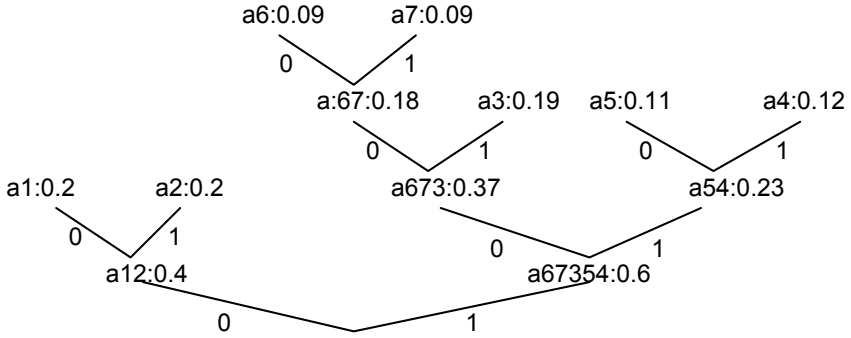
{a1:0.2 a2:0.2, a3:0.19, a4:0.12, a5:0.11,a6:0.09,a7:0.09} $\sum_i p_i = 1$ teljesül

a. Vegyük a két legkisebb valószínűségű eseményt és különböztessük meg őket egy bittel, s utána vonjuk őket össze egyetlen olyan eseménnyé, melyek valószínűsége a két esemény valószínűségének összege.



Ezután az összevont eseménnyel helyettesítve azokat, amelyek összevonásából keletkezett, folytassuk az előző pont szerint, amíg lehetséges.

Az összes lépés elvégzése után a következő adódik:



Az egyes események kódolása a kiadódó fa gyökerétől kiindulva egy-egy levélig (kódolandó karakterek) található 0-kat ill. 1-eket egymásután írva adódik:
a1:00, a2:01, a3:101, a4:111, a5:110, a6:1000, a7:1001

Az átlagos kódszó hossz: 2*0.2+2*0.2+3*0.19+3*0.12+3*0.11+4*0.09+4*0.09=2.78

Feladat:
Készítsen Huffman kódolást az alábbi esetben, és hasonlítsa össze a Shannon kódolással!
{a1:0.3, a2:0.3,a3:0.2, a4:0.1, a5:0.05, a6: 0.05}

Láthatóan, még a Huffman kóddal sem értük el az elvi határt. Ennek oka, hogy diszkrét számú eseményünk van, a valószínűségek pedig tipikusan nem $1/2^i$ értékűek, ezért általában csak megközelíteni tudjuk az elvi minimumot. Mennél több eseményünk van, annál jobban megközelíthetjük a minimumot. Hogyan lehetne növelni a kódolandó események számát?

Forráskiterjesztés

Ha nem az eseményeket, hanem pl. esemény párokat kódolunk, akkor n eseményből nxn eseményt csinálunk. Az esemény párok kódolása azt jelenti, hogy ha át akarjuk vinni pl. az alma szót, akkor az eredeti karakterek helyett az al és a ma karakterpárokat kódoljuk. Az nxn lehetséges esemény (karakter pár) kódolásához több bit kell, mint egy karakter kódolásához, de a hozzá tartozó valószínüségek is kisebbek, független események esetén az egyes valószínűségek szorzata.

Ha az eredeti eseményrendszer: {a1:0.3, a2:0.7} akkor a kiterjesztett:

	a1:0.3	a2:0.7
a1:0.3	a1a1:0.09	a1a2:0.21
a2:0.7	a2a1:0.21	a2a2:0.49

Az ez alapján számolt átlagos kódszóhossz az 2 karakterre vonatkozik, ezért 2-vel osztani kell, ha össze akarjuk hasonlítani az eredetivel. Az eredeti eseményeket 1 biten lehet kódolni, így az átlagos kódszóhossz is 1. A kiterjesztett eseményrendszer Huffman kódolással: {a1a1:000, a1a2:001, a2a1:01, a2a2:1} az egy karakterre jutó átlagos kódszóhossz 1.81/2=0.905, az eredeti eseményekre vonatkozó elvi minimum pedig:0.881, amit így jobban megközelítettünk.

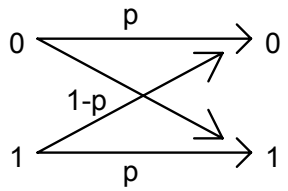
Természetesen a forráskiterjesztés nem csak 2, hanem több esemény alapján is elvégezhető.

Felhasználási példák

A változó hosszúságú kódolás felhasználására gyakorlati példa, a file tömörítés. A fileban levő karakterekről statisztikát készítve megállapítható a karakterek előfordulási valószínűsége. Ez alapján pedig elvégezhető a tömörítés. Természetesen a tömörített file-ban el kell tárolni az egyes karakterek kódját is.
Fekete fehér képek tömörítésére (pl. FAX) futamhossz kódolást alkalmaznak. Mivel a fekete fehér képen sok egymást követő 1-es ill. 0-van, egy-egy képsor kódjában azt adják meg, hogy éppen 1-es vagy 0 következik, s hogy hány darab van belőle.

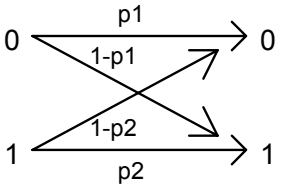
Fix hosszúságú kódolás (csatorna kódolás)

Zajos csatorna esetén a cél, a mennél kisebb hibával történő átvitel. A különböző zajos csatornákat különféleképpen lehet modellezni:
A *bináris szimmetrikus* emlékezet nélküli zajos csatorna esetén a helyes átvitel (0-ból 0 lesz, 1-ből 1 lesz) valószínűsége *p*, a hibásé pedig (1-ből 0 lesz 0-ból 1 lesz) *1-p*



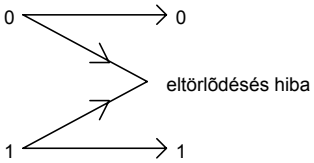
(p>0.5, ha ez nem teljesül akkor a csatorna invertál...)

Aszimmetrikus csatorna esetén a 0 ill az 1 helyes átvitelének valószínűsége eltér:



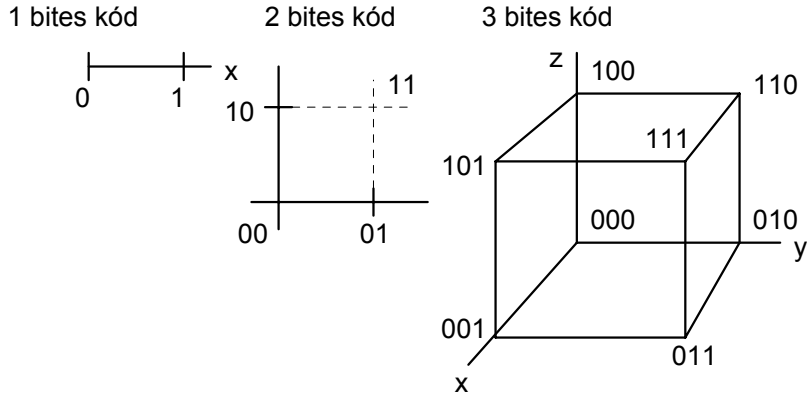
A fenti a hibamodellekben un. *átállítódásos hibák* szerepelnek, vagyis hiba esetén az információs bit negáltját érzékeli a vevő logika.

Olyan eset is lehetséges, amikor a vevő érzékelni képes, hogy se nem 0, se nem 1 jött, hanem "valami más". Ekkor ismerjük a hibás bit(ek) helyét, de annak értéke ismeretlen, eltörlődött, ezért ezt *eltörlődéses hibának* nevezzük



A hibák jelzésének ill. javíthatóságának érdekében
- fix hosszúságú kódolást alkalmazunk, és
- nem használjuk ki a kódtér összes elemét (az adott bit hosszúsággal lehetséges összes kódot), hanem csak ennek egy kisebb része megengedett az átvitel során
- Úgy kódolunk, hogy a feltételezett legnagyobb számú hiba esetén se fordulhasson elő, hogy a hiba hatására egy megengedett kód megengedett kóddá alakuljon.
- A maximálisan feltételezett számú vagy kevesebb hiba hatására tehát a megengedett kódból nem megengedett válik, ezért mindenképpen detektálni tudjuk a hibát, s ha a megengedett kódok "elég mesze" vannak egymástól, akkor javítani is tudjuk (a hibás kódhoz legközelebbi megengedett kódszóra javítunk).

A fix hosszúságú kódolásnál a kódteret egy hiperkockával ábrázolhatjuk. Itt minden bitnek egy koordinátát feleltetünk meg, minden koordinátatengelyen csak 2 érték lehetséges, 0 és 1. Minden kódnak egy pont felel meg a koordinátarendszerben. Az alábbi ábra 1, 2 és 3 bites kód esetén ábrázolja a hiperkockát. 4 bites kódnál már meglehetősen bonyolut ábra születne.



A kódok (a hiperkocka pontjai) között távolságot lehet értelmezni.
Két kódszó Hamming távolsága H(a,b): az eltérő bitek (koordináták) száma

Pl: H(101,010)=3

Bevezetjük a Kódszó súlyának W(a) fogalmát: az 1-es bitek száma a kódszóban
W(10110)=3
Ennek segítségével könnyen kiszámítható két kódszó Hamming távolsága, a kódszót bitenként EXOR (kizáró VAGY) kapcsolatba hozva, az eredmény súlya adja a Hamming távolságukat. (Két bit EXOR kapcsolata csak akkor ad 1-et, ha a bitek eltérők.)
Pl. legyen a két kód a: 1110, b: 01011

H(a,b)=W(a ⊕b) 10110
⊕01011

11101 W(11101)=4

Nem csak bináris kód esetén van értelme:
k1= BABUCI
k2= PAPUCI
eltérés: 101000 W(101000)=2, H(k1,k2)=2

Kód Hamming távolsága: a minimális Hamming távolság a kódszavak között

Hiba detektálása: a hibás kódszó nem eleme a kódtérnek

Hiba javítása: a kódtér legközelebbi elemére javítunk.
Példák fix hosszúságú kódokra:
1 hibát detektál a kétszer ismétlő kód (pl: 0000, 0101, 1010, 1111). Hiba van, ha a kód két fele nem megegyező.

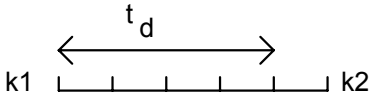
1 ill. páratlan hibát detektál a paritás kód. Az utolsó ún. *paritás bit* megegyezéstől függően párosra vagy páratlanra egészíti ki a többi bitet.
Páratlan számú hiba hatására a paritás bit ellenkezőjére változik, amit a vevő oldalon a paritás ellenőrzésekor észreveszünk. Paritás kódot használnak pl. a soros adatátvitel során a PC-ben is.
(pl: páros paritású paritás kód: (pl:000, 011, 101, 110)

1 hibát javít a 3-szor ismétlő kód (pl: 000000,010101,101010,111111). Amelyik két kódrész egyező, az a helyes.
1 hibát jelez a Berger kód, melyben az információs bitek után, az információs bitek között szereplő 0-ák számát is megadják. Ez a kód aszimmetrikus csatorna esetén hatékonyan jelzi a hibákat. 3 információs bitet tartalmazó Berger kód: {00000, 00110, 01010, 01101, 10010, 10101, 11001, 11100}

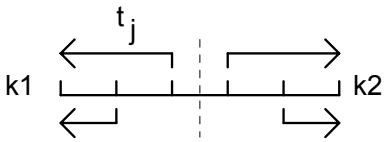
1 hibát jeleznek az N-ből k kódok, melyek N bitből k db 1-est tartalmaznak. Szintén hatékonyak aszimmetrikus csatorna esetén.

A Hamming távolság és a *hiba detektálás* ill. *hiba javítás* közötti összefüggés
Átállítódásásos hibák esete:

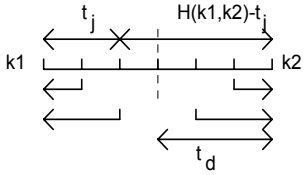
a. t_d hiba detektálható, ha $H \geq t_d + 1$



b. t_j hiba javítható, ha $H \geq 2t_j + 1$

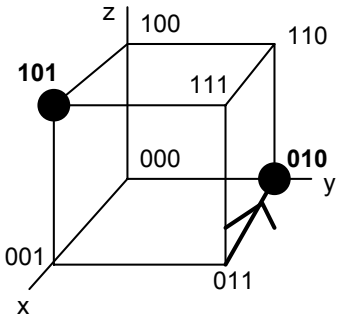


c. t_j hiba javítható és t_d hiba detektálható ha $H \geq t_d + t_j + 1$



A $t_d < H - t_j$ egyenlőlenségnek teljesülni kell, különben a detektálandó hiba az egyik kódtól t_j, vagy kisebb távolságra esik, tehát nem lesz eldönthető, hogy javítani, vagy detektálni kell. Tehát $t_d < H - t_j \Rightarrow H \geq t_d + t_j + 1$, emellett $t_d > t_j$
Az átállítódásos hibák mellet léteznek az *eltörlődéses hibák*, ahol a vevő azt érzékeli, hogy az átvitt bit se nem 0, se nem 1 szintű, hanem hibás. Ebben az esetben *tehát a hiba helyét ismerjük*, vagyis ideális esetben *minden eltörlődéses hiba érzékelhető*, jelezhető. Mivel *m* eltörlődésés hibát feltételezve egy *n* hosszúságú kódszóban a konkrét esetben tudjuk, hogy mely helyeken keletkezett a hiba, vagyis is ismerjük a kódszó *n-m* helyes bitjének helyét és értékét is. Az ettől *m+1* Hamming távolságra levő kódszavak biztosan különböznek az aktuálisan helyes bitek helyén 1 bitben, így *m+1* Hamming távoságú kód esetén *m* eltörlődéses hiba javítható, arra a kódszóra kell javítani, amelynek megfelelő bitjei megegyeznek a hibás kódszó hibátlan bitjeivel.
Pl. A kódszavak legyenek: 000, 110, 011, 101. Ez a kód 2 Hamming távolságú, így 1 eltörlődéses hibát javít. Ha a vevő 0x0-at érzékel (egy eltörlődéses hiba a megadott helyen), akkor tudjuk, hogy ez csak a 000-ból keletkezhetett. Átállítódásos hiba esetén, ha 000 helyett 010 megy át, akkor csak azt mondhatjuk, hogy hiba van, és ez az 101 kivételével bármelyik másikból keletkezhetett.

3. példa: Készítsünk minimális hosszúságú, 1 hiba javítására alkalmas kódot! Hány kódszót használunk ki a lehetséges kódok közül?



Ha egy hibát akarunk javítani, az előzőek szerint 3 Haming távolságú kód szükséges. 3 bites a legrövidebb kód, amelynél már hibát lehet javítani. Itt a 8 lehetséges kódból csak 2-őt lehet használni, ha 1 hibát akarunk javítani (a kocka szemközti csúcsainak megfelelő kódokat), mert csak ezek elégítik ki a minimálisan szükséges 3 Hamming

távolságot. Pl. az 010 és 101. A kódtér többi 6 elemét nem használjuk ki. Ha egy hiba bekövetkezik, akkor az így keletkezett kód minden más kódnál közelebb lesz az eredetihez.
Ha a 010 helyett 01**1** megy át a csatornán, akkor ahogy az alábbi ábra is mutatja, a feltéltelezett 1 hiba esetén ehhez az átvitelhez használt kódok közül (010, 101) a 010 kód van legközelebb.
Mivel a lehetséges kódok közül csak 2-őt használtunk ki, ezért ezzel a kóddal 1 bit információt tudunk - de azt viszonylag biztonságosan - kódolni. A többi bitre a biztonságosabb átvitel miatt van szükség.
Ha a csatornában az átvitel során feltételezett hibák *maximális* száma 2, ez a kód csak *hiba jelzésre* használható, javításra nem. Egy vagy két hiba hatására is biztosan a nem használt 6 másik kód valamelyikévé válik az átviendő kód, de a pontosan 2 hiba hatására keletkező kód a másik használt kódhoz lesz közelebb.

Redundacia

Mint az előző példában látható, a hiba jelzés ill. javítás miatt több bitet kell felhasználnunk az információ átvitelhez, mint ami minmálisan szükséges, vagyis *redundánsan* kódolunk. Akkor hatékony a kódolás, ha a célt (az adott számú hiba javítását ill. jelzését) a minimálisan szükséges számú bit felhasználásával érjük el, vagyis minimális a redundancia.

A Hamming kód

A Hamming kód *egy hiba javítására* képes, így 3 Hamming távolságú. A kódban a k darab információs bit között p darab páros paritás bit is el van helyezve oly módon, hogy a *feltételezett* 1 *hiba esetén* a paritásbiteket megfelelő sorrendben egymás mellé rakva és bináris számként értelmezve, azok megadják a hibás bit pozícióját, a *0 eredmény hibátlan átvitelt jelez*. Természetesen maguk a paritás bitek is elromolhatnak, így ekkor a hibás paritás bit pozícióját fogja adni ez a szám. Mindebből következik, hogy ha k információs bitünk van, akkor ennek a kód összes $k+p$ bitpozíciójára rá kell tudni mutatnia, vagyis

$$p \geq ld(k+p)$$

Ha az információs bitek száma pl. 4, akkor 3 paritás bit szükséges, hogy képes legyen rámutatni mind a 7 bitre.
A paritás biteket a 2 egész számú hatványának megfelelő bitpozíciókon célszerű elhelyezni: a7 a6 a5 p4 a3 p2 p1

Egy-egy paritás bit azon sorszámú információs bitek alapján számítandó, amelyek indexének bináris megfelelőjét a paritás bitekkel leírva, az adott paritás bit 1 értékű. Az előbbi példánál maradva, az alábbi táblázatból kiderül, hogy p4 az a7,a6,a5, paritása, p2 az a7, a6, a3 paritása, p1 pedig a7, a5, a3 paritása.

	p4	p2	p1
a7	1	1	1
a6	1	1	0
a5	1	0	1
a3	0	1	1

A kód dekódolása úgy történik, hogy a vevő oldalon szintén képezik a paritás biteket.
Ha 1 bit elromlik, akkor azok a paritás bitek, amelyek képzésében a bit résztvesz, szintén *megváltoznak* az ellenőrzés során.
Pl. Az átküldendő információs bitek: 1011
A paritásbitekkel kiegészített teljes kódszó:

	a7	a6	a5	p4	a3	p2	p1
Az eredeti kódszó:	1	0	1	**0**	1	**0**	**1**
Legyen a5 hibás, ez jön át:	1	0	0	**0**	1	**0**	**1**
Mi is számítsuk ki a paritásokat!				**1**		**0**	**0**
Változás a vett paritáshoz képest:				**1**		**0**	**1**

A táblázatból látható, hogy a p4, p2, p1 paritások vett és számított értékét összehasonlítva és 1-et írva, ahol különböznek (EXOR művelet) az eredmény 101b=5 megadja a hibás bit pozícióját. A hibajavításhoz tehát ezt a bitet kell invertálni.
Hamming kódot használnak pl. a teletext átvitelben is.

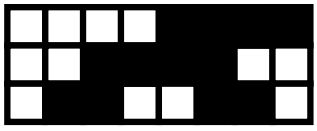
Néhány fontos kód

Pozíció kódok

Pozició (helyzet) kódolására használják (pl: a forgó szinpad éppen hogy áll). Az egymásután következő pozíciók kódja egy Hamming távolságú. Igy a pozíció érzékelők (pl. foto érzékelők) a pozíció határ átmenetnél nem adnak hibásan "távoli" pozíciót jelentő kódot, ahogy az több Hamming távolságú kód estén előfordulhatna.

Gray kód

Az alábbi ábra egy vízszintes szakaszt 8 részre osztva kódol, Gray kóddal.



A kombinációs hálózatok grafikus egyszerűsítésénél használt Karnaugh tábla peremezése is Gray kódú.
Tükrözéses módszerrel lehet kisebb bitszámú pozíció kódból nagyobbat készíteni.

Induljunk ki a 2 bites Gray kódból (most a kisebb helyfoglalás miatt az egymás alatti számok adják a kódot, 00,01,11,10):
0011
0110
Folytassuk a kódok felírását fordított sorrendben (tükrözés), majd a régi kódok elő írjunk 0-át, a tükrözöttek elé pedig 1-et:
0000 1111
0011 1100
0110 0110
Látható, hogy így megkaptuk az előbbi ábrának megfelelő Gray kódot.

Johnson kód
Ez szintén pozíció kód, vagyis az egymást követő kódok 1 Haming távolságúak. A 3 bites Johnson kód: 000, 001, 011, 111, 110, 100
A 4 bites Johnson kód: 0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000

Képzése során a csupa 0 kószóból kiindulva jobbról először 1-esekkel, majd 0-ákkal töltjük fel a bitpozíciókat. Ezért az N bites Johnson kód kódszavainak száma 2N.

Aritmetikai kódok
NBCD (Natural Binary Coded Decimal) kód: a decimális 0...9 számokhoz 4 bites bináris számokat (0000...1001) rendel.
Pl: 1997 NBCD kódban: 0001 1001 1001 0111
Kényelmes a decimális számok ábrázolására, de nehézkes számolni vele.
Számábrázolások

	előjeles abszolut értékes	egyes komplemens	kettes komplemens	offszet
3	011	011	011	111
2	010	010	010	110
1	001	001	001	101
0	000, 100	000, 111	000	100
-1	101	110	111	011
-2	110	101	110	010
-3	111	100	101	001
-4			100	000

Előjeles abszolut értékes:
A bináris számok abszolút értéke elé egy előjel bit kerül, mely 0, ha pozititív, 1, ha negatív a szám. Kényelmes előjeles számok ábrázolására, de nehézkes számolni vele. A 0-nak két kódja van, ezért nem használja ki az összes lehetőséget.

Egyes komplemens:
A pozitív bináris szám bitenkénti negáltja adja a negatív megfelelőjét. Könnyű egy pozitív szám -1 szeresét képezni, de nehézkes számolni. A 0-nak két kódja van, ezért nem használja ki az összes lehetőséget.

Kettes komplemens:
A pozitív számok ábrázolása azonos az előjeles abszolút értékessel. Egy szám -1 szeresét úgy képezzük, hogy bitenkénti negáltjához 1-et adunk hozzá. Könnyű számolni vele, a megszokott bináris összeadás az előjeles számok között helyes eredményt ad. A 0-nak egy kódja van.

Offszet:
A legnegatívabb számhoz rendeljük a csupa 0 kódot, majd a többihez egyre növekvőket. Az előjeles számok összehasonlítása könnyű, a megszokott bináris összehasonlítás helyes eredményt ad. A 0-nak egy kódja van.

Áttérés az egyes számábrázolások között
Egyes komplemens-> kettes komplemens: Ha MSB=1, akkor X=X+1.

Kettes komplemens -> egyes komplemens: Ha MSB=1, akkor X=X-1

Előjeles absz. értékes -> egyes komplemens: Ha MSB=1, akkor MSB kivételével invertálni.

Egyes komplemens -> előjeles absz. értékes: Ha MSB=1, akkor MSB kivételével invertálni.

Előjeles absz. értékes -> kettes komplemens: Ha MSB=1, akkor MSB kivételével invertálni, majd 1-et hozzáadni.

Kettes komplemens -> előjeles absz. értékes: Ha MSB=1, 1-et kivonni, majd MSB kivételével invertálni.

Offszet -> kettes komplemens: MSB-t invertálni.