# Informatika alapjai
# Kódolás

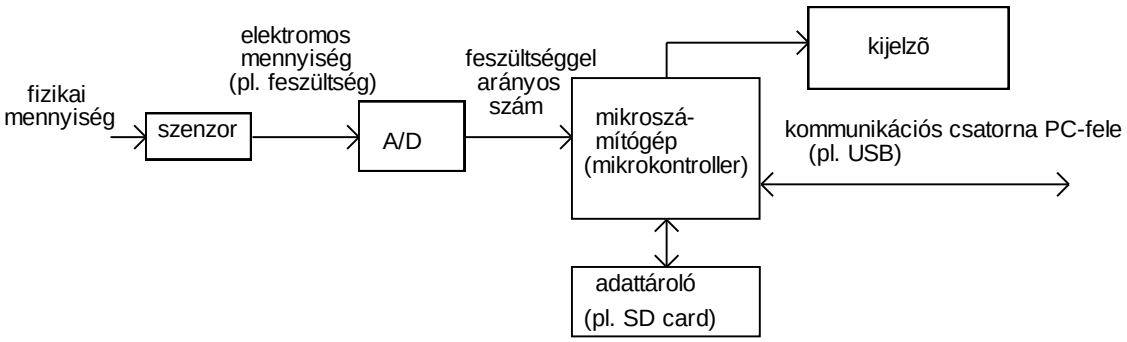
Mi az információ?

- valaminek az ismerete (pl. fizikai mennyiség)
- ahhoz, hogy feldolgozni, tárolni, továbbítani tudjuk, számmá kell alakítanunk (kódolni kell)

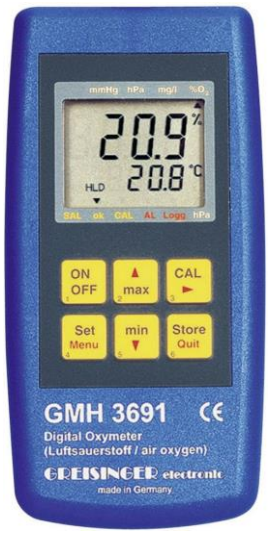
A mérnök gyakran  műszereket használ információ szerzésre

A környezetmérnök műszerekkel méri (adatokat, információt gyűjt) a
- légszennyezetséget
- vízszennyezettséget
- talajszennyezettséget
- zajszennyezettséget
  stb.

A műszer szenzora a fizikai mennyiséget azzal ismert függvénykapcsolatban levő (sokszor lineáris) elektromos mennyiséggé (feszültség, áram) alakítja.
Az elektromos mennyiséget analóg erősítővel felelrősítik, analóg-digitál konverter (A/D) számmá alakítja.
A műszer mikroszámítógépe az így kapott információt feldolgozza és kijelzi a mért fizikai mennyiséget a megfelelő mértékegységben.




oxigénmérő
oxigén: 0 ... 100 % $O_2$



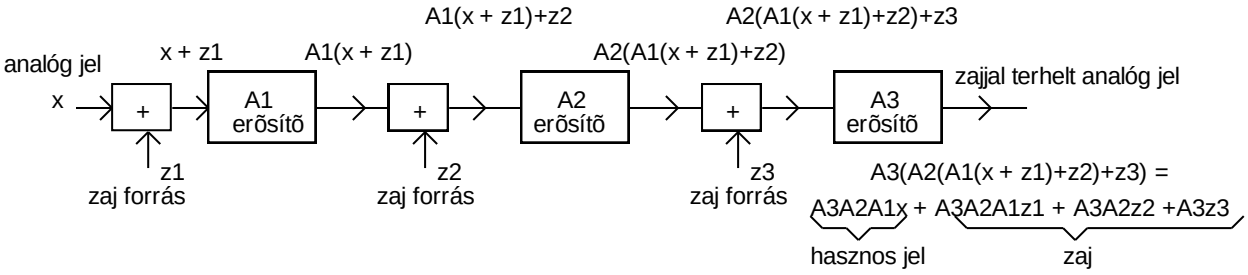

Az A/D konvertálás után a digitális automata digitális kódokkal dolgozik. A mikroszámítógép aritmetikai kódokkal számol, az információt kódolva küldi tovább a kommunikációs csatornán a PC-nek.

**Kódoláselmélet alapjai I.**

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. Az információ feldolgozó gép kódokkal dolgozik.

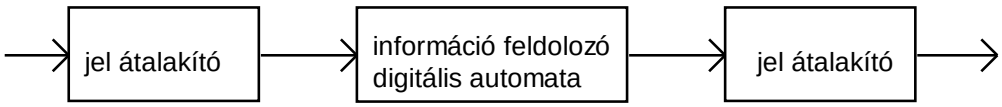
**Analóg kód:** ***amplitúdóban és időben folytonos*** (véges idő alatt végtelen információt hordoz, de a mindig jelenlevő zajok miatt csak véges használható fel)

Egy múlt századi analóg telefonnál a hangot (levegő nyomás változást) a mikrofon elektromos jellé (analóg kódolás), feszültség változássá alakítja. Ez az ***analóg jel*** erősítővel felerősítés után továbbhalad a telefonközponton keresztül egy másik telefonhoz, ahol a telefon hallagtóban újra hangnyomás változássá alakul (analóg dekódolás).
Az átvitel során ***a hasznos jelhez zaj adódik***, amelyet szintén felerősít az erősítő. Mennél több analóg jelfeldolgozó áramkörön halad át az analóg jel, annál jobban romlik a hasznos jelnek a zajhoz képesti értéke (jel/zaj viszony.) ***Az analóg jelfeldolgozók méretét a zaj jelentősen korlátozza***.



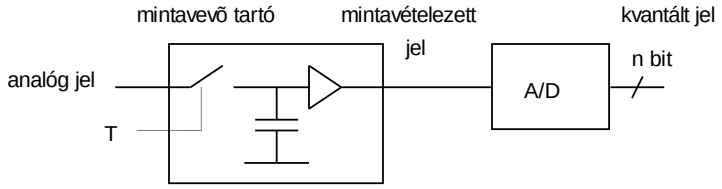

A zajforrások sokfélék lehetnek. Analóg áramköröknél a hálózati 50Hz az egyik nagy zajforrás. Rádiós átvitelnél a légköri zavarok okoznak jelentős zajt. De ha minden külső zajt sikerülne is kiküszöbölni, maguk az áramkörök akkor is termelnének valamekkora zajt (termikus zaj).

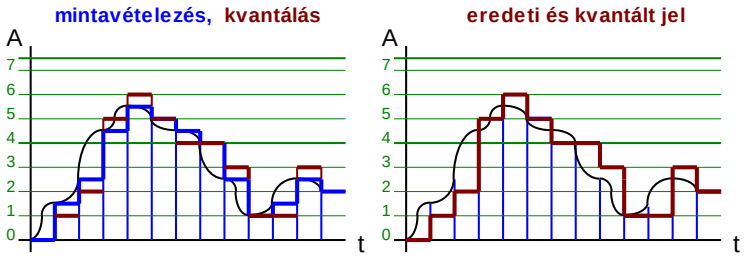
**Mai módszer:** analóg digitális alakítás (A/D konverzió), digitális feldolgozás, digitális analóg átalakítás (D/A konverzió). Előtte és utána szűrés.




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

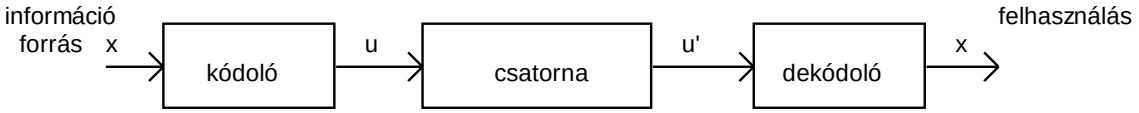
- mintavételezés: T időnként mintát vesznek az analóg jelből és tárolják (ez még analóg jel)
- kvantálás: analóg jellel arányos n bites számmá alakítás (ez már digitális jel)




***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. Ezek ***viszonylag nagy zaj mellet*** is jól megkülönböztethetők 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
a ***forrás kódolást*** (ha szükség van rá), melynek célja az információ ***tömörítés*****e**,
a ***titkosítást*** (ha szükség van rá), melynek célja, hogy ***illetéktelen ne tudja visszafejteni*** az információt,
a ***csatorna kódolást*** (mindenképpen), melynek célja a digitális *információ hibátlan átvitele*

Egyszerű példa digitális információ átvitelre:
információ forrás: pl: angol szöveg
kódolás: pl: az angol szöveg betűnkénti kódolása bináris számokká (pl. ASCII kód, pl. A 2 ASCII kódja bináris számként felírva 00110010)
csatorna: pl: két állapotú, fizikailag két feszültségszint reprezentálja
dekódolás: pl: a bináris számok angol karakterekké alakítása

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

***Forrás ABC*** az információ forrás karaktereinek a halmaza. $A = \{a_1, a_2, \ldots a_n\}$

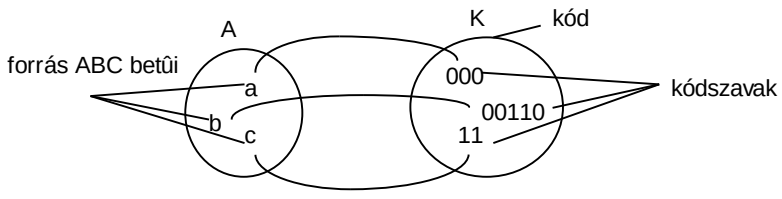
***Kód ABC*** a kódolásra használt karakterek halmaza.
***Bináris kód*** esetén a *kód ABC* 2 elemű: {0, 1}

A ***kód***, a kód ABC betűiből képzett véges hosszúságú sorozatok (kód szavak) halmaza. $K = \{\alpha_1, \alpha_2, \ldots \alpha_n\}$

Pl. ***bináris kód*** esetén egy lehetséges kód K = {00, 010, 0111, …} ($\alpha_1$ kódja 00, $\alpha_2$ kódja 010, stb.)

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

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




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

Kódok osztályozása:

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

Az információ mértékegysége

Játékoknál gyakran hasznájják a körforgó mutatós véletlenszám generátort. Egy N részre osztott kör közepén a mutatót megforgatva mind az N szám azonos eséllyel jön ki. Hány bites számmal lehet átküldeni azt az információt, hogy melyik szám jött ki?

$$bits = \log_2 N$$

Mekkora annak a valószínűsége, hogy az N közül pontosan az i-edik szám jön ki?

p = 1/N

A fenti képletbe N helyett 1/p-t helyettesítve: $bits = \log_2 1/p = -\log_2 p = -ld\ p$

Mennél kisebb a p valószínűség, annál több bit szükséges az információ kódolásához. Mennél kisebb valószínűségű egy esemény, annál több információt hordoz. Az az információ sokkal értékesebb, ha megtudom, hogy a lottón kihúzták a számaimat, mint az, hogy valaki azt üzeni, hogy holnap felkel a nap…

Ha egy esemény valószínűsége p, annak bekövetkezését úgy is értelmezhetjük, hogy *1/p* féle lehetőségből pont az következett be. Többek között ezért az információ mértékéül azt választjuk, hogy az hány biten kódolható (ha a cél a legrövidebb kódszó hossz). Az előző gondolatmenet alapján egy *p* valószínűségű esemény tehát *m=1/p* féle lehetőség közüli

bekövetkezésként 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).

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. Ebben ez esetben nem egyész szám jön ki (2.58). Majd később látni fogjuk, hogy az ideálist megközelítő kódolással ez a szám megközelíthető (***átlagos*** kódszó hossz).

**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 egy másik kódszó után írt karakterek segítségével generálhatók.

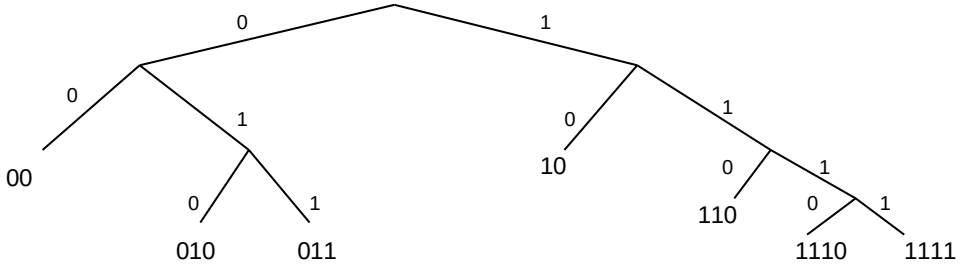
***Prefix tulajdonság***: egyik kódszó sem folytatása egy másiknak.
pl: a {01, 001, 100, 1100} kód prefix

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

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

A {10, 100, 1000, 10000} kód nem prefix, de megfejhető, mert a kódszavak határát jelzi az első karakter.

**Prefix tulajdonságú kód fagráf segítségével generálható.**
Pl. ***bináris prefix kódot bináris fával generálhatunk***. A ***gyökértől egy-egy levélig haladva megkapjuk a kódokat***. (Kódokat csak levelekhez rendelünk, belső pontokhoz nem!)




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. ***Az átlagos kódszóhosszt nevezik kód költségének***.

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)

**Az M számú karakterből álló üzenet kódjának átlagos hossza a következő gondolatmenettel számítható:**

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

$$p_i \cong \frac{n_{a_i}}{M}, \quad M \to \infty$$

- 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:*** $$\lim_{M\to\infty} \frac{Kódolt üzenet hossza}{M} = \sum_i p_i l_i$$

***ennek létezik alsó korlátja***, s ez bizonyíthatóan a következőkben definiált ***entrópia***.

Definíció: Ha egy X teljes eseményrendszer 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

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

Ezt úgy is értelmezhetjük, hogy a p*i* valószínőségű 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.)

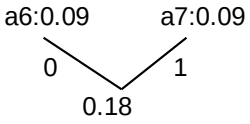
***Általános elv, hogy a ritkábban előforduló eseményt (karaktert) kódoljuk hosszabb kóddal.***

**Huffman kód**

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

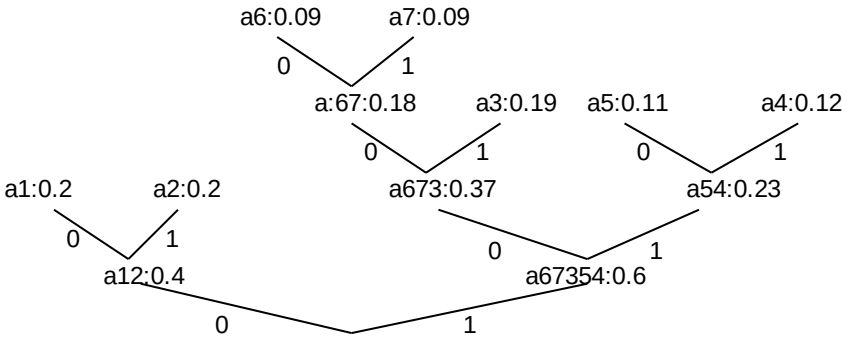
{a1:0.2 a2:0.2, a3:0.19, a4:0.12, a5:0.11,a6:0.09,a7:0.09} $\sum_i p_i = 1$ teljesül

a. ***Rendezzük valószínűségük szerint sorrenbe az eseményeket.***
(A példában már a megadáskor így rendeztük.)
b. ***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 első ponttól újra, 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 bit
Az entópia: 2.727 bit

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 ***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 a 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 (entrópia) 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.

**ZIP** kódolás fő jellemzői

- Egy vagy több file ***egyetlen file-ba becsomagolva***
- ***A file-ok a ZIP file-ban szétválasztva vannak benne***, egyenként törölhetők, módosíthatók, vagy új file-ok tehetők be.
- A program az egyes file-okat ***különböző módszerekkel tömöríti, Huiffmann kódolást is használ***
- A tömörítéssel együtt ***titkosítás is történhet***
- ***A hiba hatása korlátozódik*** arra az eredeti file-ra, melynek becsomagolt formájában a hiba előfordul.

## Run Length Encoding (RLE, Futam hossz kódolás)

- A fekete-fehér képet, ha sorokra bontjuk, sok egyforma képpont követi egymást. ehhez képest ritka a váltás.

- Az egyforma adatokból álló sorozat helyett a sorozat darabszámát és az elemet továbbítjuk. (pl:5w2b9w  5 fehér, 2 fekete, 9 fehér)

- Legegyszerűbb esetben a tömörítés csak egy soron belül történik.

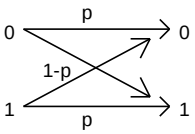
Pl. „A” betű képe esetén:

| | |
|---|---|
| 00000**11**000000000 | 5w2b9w |
| 0000**11011**0000000 | 4w2bw2b7w |
| 000**11**000**11**000000 | 3w2b3w2b6w |
| 00**111111111**00000 | 2w9b5w |
| 0**11**0000000**11**0000 | w2b7w2b4w |
| **11**00000000**11**000 | 2b9w2b3w |

Bonyolultabb algoritmus azt is kihasználja, hogy az egymás utáni sorok hasonlóak.

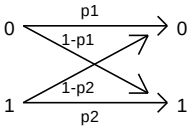
**Fix hosszúságú kódolás** (csatorna kódolás)

***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 (p>0.5)*, 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***.

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***.

- ***Redundancia:*** 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.

- 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 megengedetté 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 kódok között távolságot lehet értelmezni.
***Két kódszó Hamming távolsága H(a,b): az eltérő bitek (koordináták) száma***
Pl: H(101,010)=3

***A kódszó súlya W(a): az 1-es bitek száma a kódszóban***
W(10110)=3

Ennek segítségével könnyen kiszámítható két kódszó Hamming távolsága, a ***két kódszót bitenként EXOR (kizáró VAGY) kapcsolatba hozva, az eredmény súlya adja a Hamming távolságukat***. (Két bit EXOR kapcsolata (jele: $\oplus$) csak akkor ad 1-et, ha a bitek eltérők.)

Pl. legyen a két kód a: 10110, b: 01011 H(a,b)=W(a ⊕b)

```
  10110
⊕ 01011
---------
  11101      W(11101)=4
```

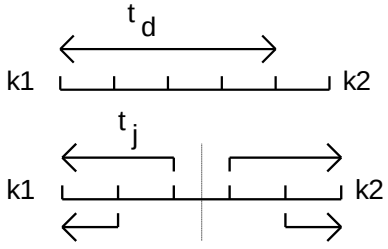
***Kód Hamming távolsága:*** ***a minimális Hamming távolság a kódszavak között***
***Hiba detektálása:*** ***a hibás kódszó nem eleme a kódtérnek (nem használják)***
***Hiba javítása:*** ***a kódtér legközelebbi elemére javítunk***

**Példák fix hosszúságú kódokra:**
***1 hibát detektál (jelez) a kétszer ismétlő kód*** (pl: 0000, 0101, 1010, 1111). Hiba van, ha a kód két fele nem megegyező. A kód 4 bites, tehát a kódtér (az összes lehetséges 4 bites kód) 16 elemű. Ebből csak 4-et használunk, 12-őt nem. A fenti példában 2 bit átviteléhez 4 bitet használunk (nagy a redundancia). Ennek ellenére ez a kód csak 1 hiba ***jelzésére*** képes.

***1 hibát javít a 3-szor ismétlő kód*** (pl: 000000,010101,101010,111111). Amelyik két kódrész egyező, az a helyes. A kód 6 bites, tehát a kódtér 64 elemű. Ebből csak 4-et használunk, 60-at nem A példában 2 bit átviteléhez 6 bitet használunk (nagy a redundancia), ennek ellenére ez a kód csak 1 hiba ***javítására*** képes.

***1 ill. páratlan hibát detektál a paritás kód.*** Az kódszót kiegészítik egy ***paritás bittel***, mely megegyezéstől függően párosra vagy páratlanra egészíti ki a többi bitben levő 1-esek számát. Pl. Három bites paritás kód generálása. Az eredeti kód 2 bites: a2a1: 00, 01, 10, 11. Ehhez hozzáteszünk egy páros paritás bitet (p): a2a1p: 000, 011, 101, 110
***Páratlan számú hiba hatására a kód paritása ellenkezőjére változik***, amit a vevő oldal a paritás ellenőrzésekor észrevesz. A 3 bites paritás kódnál kódtér 8 elemű, ebből 4-et (a felét) használjuk és minden 1 ill. páratlan számú átállítódásos hibát észlel. Ez kis redundancia (csak 1 plusz bitre van szükség) árán jó hiba jelzési arány.
**A Hamming távolság (H) és a hiba detektálás ill. hiba javítás közötti összefüggés *átállítódásos hibák esetére***:

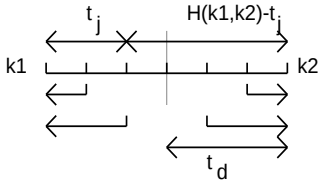
a. ***Maximum*** $t_d$ ***hiba detektálható***, ha $H \geq t_d + 1$




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

c. ***Maximum*** $t_j$ ***hiba javítható és maximum*** $t_d$ ***(*** $t_d > t_j$ ***) hiba detektálható*** ha

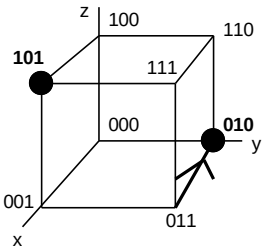
$H \geq t_d + t_j + 1$




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 $t_d > t_j$

***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 011 megy át a csatornán, akkor ahogy a mellékelt 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.

**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.*** A feltételezett ***1 hiba esetén a vett és számított paritásbitek összehasonlításával képzett bináris szám megadja 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 (***a bit pozíciók 1-től sorszámozódnak***). Mindebből következik, hogy ha *k* információs bitünk van, akkor ennek a kód összes *k+p* bitpozíciójára rá kell tudni mutatnia.

Pl. 3 paritás bit esetén az összes bit $2^3 - 1 = 7$ lehet (a 0-t le kell vonni, mert az a hibátlant jelez), így az információs bitek száma 7-3 = 4 lehet.

A paritás bitek a 2 egész számú hatványának megfelelő bitpozíciókon vannak:

**a7 a6 a5 p4 a3 p2 p1**

***Egy-egy paritás bit azon sorszámú információs bitek alapján számítandó (azok paritását egészíti ki párosra), amelyek indexének bináris megfelelőjét a megfelelő paritás bitek alá leírva, az adott paritás bit alatti szám 1.*** Az előbbi példánál maradva, az alábbi táblázatból kiderül, hogy p4 az a7,a6,a5, paritását, p2 az a7, a6, a3 paritását, p1 pedig a7, a5, a3 paritását egészíti ki párosra.

| | **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. A megváltozott paritás bitek helyett 1-et, a többi helyett 0-t írva, az így kapott bináris szám megadja a hibás bit pozícióját. A hibás bitet meginvertálva

megkapjuk az eredeti hibátlan bitet. (Csak 1 hiba esetén működik!) Az így előállított Hamming kódot H(7,4)-el jelölik H(összes bitek száma, információs bitek száma)

Pl. Az átküldendő információs bitek: 1011
A paritásbitekkel kiegészített teljes kódszó:

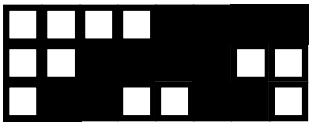
| | a7 | a6 | a5 | **p4** | a3 | **p2** | **p1** |
|---|---|---|---|---|---|---|---|
| Az eredeti kódszó: | 1 | 0 | 1 | **0** | 1 | **0** | **1** |
| Legyen a5 hibás, a vett kódszó: | 1 | 0 | **0** | **0** | 1 | **0** | **1** |
| A vett kódszó számított paritás bitjei: | | | | **1** | | **0** | **0** |
| 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 $101_2=5_{10}$ megadja a hibás bit pozícióját. (Alsó index-szel a számrendszert jelöltük.) A hibajavításhoz tehát ezt a bitet kell invertálni.
Hamming kódot használtak pl. az analóg TV-k teletext átvitelében.

**Néhány egyéb fontos kód**

**Pozíció kódok**
Pozició (helyzet) kódolására használják (pl: a forgó szinpad éppen hogy áll). ***Az egymás melleti pozíciók kódja között a Hamming távolság 1.*** Így a pozíció érzékelők (pl. foto érzékelők) a pozíció határ átmenetnél nem adnak hibásan "távoli" pozíciót jelentő kódot, ahogy az egynél több Hamming távolságú kód estén előfordulhatna. Többfajta pozíció kód létezik. Alább a Gray kódot mutatjuk be.

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



Tükrözéses módszerrel lehet kisebb bitszámú pozíció kódból nagyobbat készíteni.
Induljunk ki a 1 bites Gray kódból: 01
Folytassuk a kódok felírását fordított sorrendben (tükrözés):
01 10
A régi kódok felé írjunk 0-át, a tükrözöttek elé pedig 1-et:
0011
0110
Az egymás alatti számok adják a kódot, 00,01,11,10):

Készítsünk ebből ugyanezzel a szabállyal 3 bites Gray kódot:
0000**1111**
00**1111**00
01**10**0**11**0

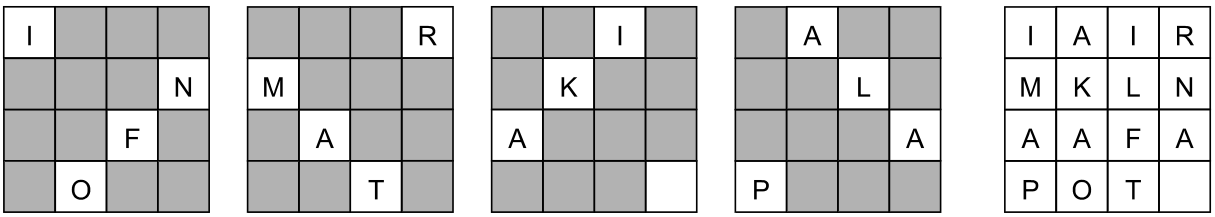
Látható, hogy így megkaptuk a fenti ábrának megfelelő Gray kódot.

## Titkosítás

A titkosítás célja az, hogy csak az tudja elolvasni az üzenetet, akinek szól.
Titkos az, amit nem tudunk elolvasni, például
- hieroglifák
- bármilyen idegen nyelv, amit nem ismerünk

A jó titkosírást a kulcs ismerete nélkül nagyon nehéz, vagy gyakorlatilag lehetetlen megfejteni.
Kulcs: szótár és/vagy algoritmus, amivel az üzenet titkosítható és megfejthető.
A klasszikus titkosírásokban kétféle módszert alkalmaztak:
- helyettesítés, ennek egyszerű esete a karakterkódok hexadecimális megadása.
- áthelyezés, amikor a szöveg karaktereit átrendezik.

Az áthelyezéses titkosítás (betűk szisztematikus összekeverése) klasszikus esete a perforált négyzetrács alkalmazása.
Készítsünk négyzetrácsot, melyen a négyzeteket úgy perforálják, hogy a négyzetet 90 fokonként forgatva mindig más helyen legyenek a lyukak:




A rácson lévő üres helyekre beírjuk az „INFORMATIKA ALAP” szöveg első 4 betűjét, majd 90 fokkal elforgatjuk a rácsot, és folytatjuk. Ezt négyszer lehet ismételni. A rács levétele után a jobb oldalon lévő négyzet látszik. A titkos üzenet: „IAIRMKLNAAFAPOT “.
A megfejtéshez négyzetbe rendezve le kell írni a titkosított szöveget, majd a rácsot ráhelyezve és forgatva az eredeti szöveg elolvasható.
A gyakorlatban sokkal nagyobb négyzetrácsot készítenek, amelyik nehezebben megfejthető. Ehhez hasonló elven működött a 2. világháborúban a német tengeralattjárókon alkalmazott Enigma titkosító – abban a szöveget háromszor egymás után titkosították, azaz az átrendezett szöveget egy másik kulccsal újra átrendezték. A szövetségesek a kódot sok-sok üzenetet elemezve megfejtették.
Természetesen a helyettesítés és áthelyezés kombinálható.

A számítógépes világban széleskörűen alkalmazzák a titkosítást. Kétféle alkalmazást különböztetnek meg:
- titkos kulcsú. Ennél a titkosításra használt kulcs alkalmazható a dekódolásra is. Ha valaki hozzájut a kulcshoz, az meg tudja fejteni az üzenetet.

A titkos kulcs megvan az üzenetküldőnél és a vevőnél is. Ha a kulcsot ellopják, vagy feltörik, az üzenet megfejthető.
- nyilvános kulcsú: a titkosításra más kulcs szolgál, mint a dekódolásra. A titkosításra szolgáló kulcs nyilvános lehet, így bárki küldhet titkos üzenetet, amit viszont csak a jogosított vevő – akinél a kulcs van – tud dekódolni.

A titkos kulcs csak a vevőnél van meg, az üzenetküldő(k) csak a nyilvános kulcsot kapják meg. Ha a nyilvános kulcsot ellopják, csak üzenetet küldeni tudnak, dekódolni nem.

## Hivatkozások

Titkosítás  Titkos kulcsú titkosítás  Nyilvános kulcsú titkosítás

**Számábrázolások**

Pozícionális számábrázolás, tetszőleges ***r*** számrendszerben, n számjegyű számra:

$$D = \sum_{i=0}^{n-1} D_i r^i \qquad D \in \{0, r-1\}$$

***Kettes számrendszerben***

$$D = \sum_{i=0}^{n-1} D_i 2^i \qquad D \in \{0,1\}$$

$111\ 1101\ 1110 = 1*2^{10}+1*2^9+1*2^8+1*2^7+1*2^6+0*2^5+1*2^4+1*2^3+1*2^2+1*2^1+0*2^0$

***Decimális-bináris konverzió***

A 2-vel osztás maradékai adják a bináris számjegyeket, a legkisebb helyiértéktől kezdve.

| Osztandó | Osztó | Hányados | Maradék |
|---|---|---|---|
| 2014 | :2 | 1007 | 0 |
| 1007 | :2 | 503 | 1 |
| 503 | :2 | 251 | 1 |
| 251 | :2 | 125 | 1 |
| 125 | :2 | 62 | 1 |
| 62 | :2 | 31 | 0 |
| 31 | :2 | 15 | 1 |
| 15 | :2 | 7 | 1 |
| 7 | :2 | 3 | 1 |
| 3 | :2 | 1 | 1 |
| 1 | :2 | 0 | 1 |

$2014 = 111\ 1101\ 1110_{BIN}$

***Hexadecimális (16-os)***

| decimális: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **hexadecimális:** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| bináris: | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

***Hexadecimmális-bináris konverzió***
Az egyes hexa számjegyeket a megfelelő 4 bites bináris számokkal helyettesítjük.

$A9BC_H = 1010\ 1001\ 1011\ 1100$

***Bináris- hexadecimális konverzió***
4-es csoportokat képzünk a legkisebb bittel kezdődően, ha a baloldalon kevesebb mint 4 bit marad, 0-kkal kiegészítjük. Ezután az így kapott 4 bites bináris számokat a hexadecimális megfelelőjükkel helyettesítjük.

10110111101111
0010 1101 1110 1111
2 D E F

***Előjeles számok ábrázolása***

| | ***előjel+ abszolut értékes*** | ***egyes komplemens*** | ***kettes komplemens*** |
|---|---|---|---|
| 3 | 011 | 011 | 011 |
| 2 | 010 | 010 | 010 |
| 1 | 001 | 001 | 001 |
| 0 | 000, 100 | 000, 111 | 000 |
| -1 | 101 | 110 | 111 |
| -2 | 110 | 101 | 110 |
| -3 | 111 | 100 | 101 |
| -4 | | | 100 |

***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.

**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.

**Aritmetikai műveletek egész számok között**

***Összeadás***

Összeadás szabályai (általában 2 operandus között):

0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 0, az átvitel 1

Példa:

| | |
|---|---|
| 01001001 | 73 |
| + 10011111 | +159 |
| = 11101000 | =232 |

***Szorzás***

2-es számrendszerben a 2-vel szorzásnál 1-el balra toljuk a számjegyeket.
0011*10 = 0110

N szorozva M, ahol $M = r_n*2^{n\cdots}+ r_1*2^1 + r_0*2^0$ $r_i \in \{0,1\}$ számmal.
$N*M = N*r_n*2^{n\cdots}+ N*r_1*2^1 + N*r_0*2^0$

Abban az estben, ha $r_i = 1$, akkor az N szám i-szer balra shifteltjét ($2^i$-szeresét) hozzáadjuk az eddigi szorzatösszeghez. Ha $r_i = 0$, akkor 0-át adunk hozzá (semmit nem csinálunk).

Binárisan:

| | |
|---|---|
| 1110 * 1101 | 14*13 = 182 |
| + **1110** | |
| + **00000** | |
| + **11100** | |
| +**1110000** | |
| **=10110110 = 128+32+16+4+2 = 182** | |

**Valós számok ábrázolása 2-es számrendszerben, fixpontos számábrázolás**

A tört részeket 2 negatív hatványaival adjuk meg:

$$N= r_n*2^{2\cdots}+ r_1*2^1 + r_0*2^0 \bullet + r_{-1}*2^{-1} + r_{-2}*2^{-1}$$

A tört rész helyét ponttal jelöltük:

$110.101_{BIN} = 4+2+1/2+1/8 = 6.625$

A lebegőpontos számárázolással itt nem foglalkozunk.