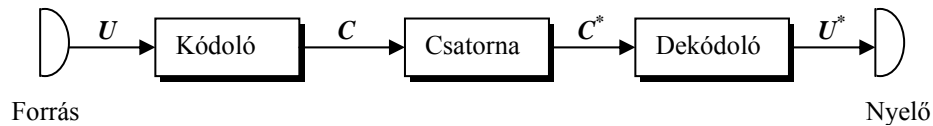


1. Forráskódolás

A legtöbb hírközlő rendszer alapvető feladata: információ átvitele üzenetek vagy adatok formájában az információ forrástól a célig, az információ nyelőig. Az információt elektromos jelek hordozzák, amelyeket a csatornán továbbítunk. Jelöljük a forrás üzenet készletét U -val. A csatorna nem ideális voltából adódóan és a zaj következtében azonban a vevőoldalon U -nak egy torzított változata U^* jelenik meg.

A cél az, hogy:

- a csatorna és a zaj torzító hatása minimális,
- a csatornán adott idő alatt továbbított üzenetek száma pedig maximális legyen.



Ez a két követelmény ellentmond egymásnak, mivel az üzenettovábbítás sebességének növelése következtében nőnek a torzítások és a hibák. Bizonyos formájú üzenetek azonban jobban alkalmasak egy adott csatornán történő továbbításra, mint mások, így ezek gyorsabban és kevesebb hibával továbbíthatók. Emiatt az U üzenet készletet célszerű lehet egy kódolóval módosítani és ezzel előállítani egy új, a csatornához jobban alkalmazkodó C kódkészletet. Ekkor persze szükség van egy dekódolóra, amely a torzított C^* -ből U^* -ot nyeri vissza.

1.1. Az információ és az entrópia

Az információelmélet az üzenetek előfordulásának relatív gyakoriságait (valószínűségeit) vizsgálja, függetlenül azok értelmétől. A fenti modellből kiindulva tételezzük fel, hogy az U üzenetkészlet minden tagja kifejezhető egy véges méretű X szimbólumkészlet, az un. szimbólum ábécé elemeinek bizonyos kombinációjával.

A továbbiakban feltételezzük, hogy az üzenet egymás utáni szimbólumokra történő lebontásakor az egymást követő szimbólumok függetlenek egymástól. Az ilyen információ forrásokat **memória-mentes forrásoknak** nevezzük. A továbbiakban ilyen forrásokkal foglalkozunk. (Memóriás forrásokra egy példa: az un. Markov-források)

Jelöljük X elemeit $X = \{x_1, x_2, x_3, \dots, x_M\}$ -mel, ahol M a szimbólumkészlet mérete és $p_i = \mathbf{P}\{x_i\}$ -vel az i -ik szimbólum előfordulásának **valószínűségét**! A $P = \{p_1, p_2, p_3, \dots, p_M\}$ **valószínűségeloszlásra** természetesen fenn áll:

$$p_i \geq 0 \quad i=1,2,3,\dots,M$$

$$\sum_{i=1}^M p_i = 1 \quad (1.)$$

Definíciók:

Az i -ik szimbólum **információtartalmának** mérésére előfordulásának valószínűségét használjuk.

$$I(x_i) = \lg \frac{1}{p_i} \geq 0 \quad [\text{bit/szimbólum}] \quad (2.)$$

(Itt ld a kettes alapú logaritmust jelöli.)

A (2.) összefüggés azt fejezi ki, hogy a kis valószínűségű esemény bekövetkezte nagy, míg a biztos esemény (mikor $p_i=1$) előfordulása zérus információ tartalommal bír.

Az információ mennyiségnek mint valószínűségi változónak a várható értékét **entrópiának** nevezzük

$$H(P) = \mathbf{E}\{I\} = \sum_{i=1}^M p_i I(x_i) = \sum_{i=1}^M p_i ld \frac{1}{p_i} \quad (3.)$$

Az elnevezés onnan ered, hogy ha p_i annak a valószínűsége, hogy egy rendszer az állapot i -ik állapotában van, akkor a (3.) kifejezés a statisztikus mechanikából és a termodinamikából ismert entrópiadefinícióval azonos. Az entrópia a statisztikus mechanikában a rendszer rendezetlenségének mértéke, az információelméletben pedig az üzenetforrással kapcsolatban értelmezett **bizonytalanságé**.

Az entrópiára általában igaz:

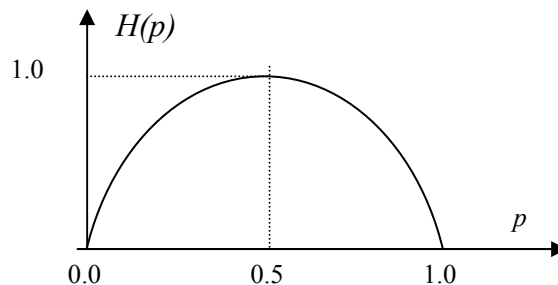
1. $H(P) \geq 0$
2. $H(P)$ akkor és csak akkor zérus, ha egy kivételével az összes valószínűség zérus, az az egy pedig egységnyi.
3. $H(P) \leq ld M$
4. $H(P) = ld M$ akkor és csak akkor, ha minden valószínűség egyforma ($1/M$).

Bináris forrásról akkor beszélünk, amikor a *szimbólumábécé* két szimbólumból $X = \{0,1\}$ áll. A valószínűségeloszlás ekkor egyetlen p paraméterrel adható meg:

$$P = \{p, (1-p)\}$$

A bináris forrás entrópiája:

$$H(p) = p ld \frac{1}{p} + (1-p) ld \frac{1}{1-p} \quad (4.)$$



Bináris forrás entrópiája az egyik szimbólum előfordulási valószínűségének függvényében

1.2. Változó szóhosszúságú kódok

Térjünk most rá arra a kérdésre, hogyan válasszunk kódot, vagyis az $\{x_1, x_2, x_3, \dots, x_M\}$ szimbólumábécéhez milyen szabály szerint rendeljük hozzá a $C = \{c_1, c_2, c_3, \dots, c_M\}$ kódábécét.

A kódábécé elemei bináris reprezentációjú számok lesznek, mivel napjaink digitális technológiájához ez a formátum illeszkedik a legjobban.

A legegyszerűbb kódolási ötlet az lehetne, hogy a szimbólumábécé elemeinek sorszámát választjuk kódnak. Például, egy 26 elemű abc esetén akkor különböztethetők meg a kódok, ha a kódszavak legalább 5 bitesek ($2^5 = 32 > 26$). Tekintetbe véve, hogy a szimbólumok előfordulási valószínűsége nem egyforma az üzenetekben, valamint azt, hogy kihasználatlan kódok is vannak, megállapíthatjuk, hogy ez a “kézenfekvő” ötlet egyáltalán nem gazdaságos.

Hatékonyabban járnánk el, ha elvetnénk az egyforma hosszúságú kódok alkalmazását: a nagyobb valószínűséggel előforduló elemekhez rövidebb kódokat, a ritkábban előfordulókhöz a hosszabb a kódokat rendelnénk hozzá.

Ha a kód szóhosszúsága változik, de nem akarjuk a továbbítandó adatok mennyiségét növelni (redundancia) elválasztó karakterek beszúrásával, vagy minden egyes kód hosszúságának megadásával, akkor olyan kódot kell konstruálnunk, melyből egyértelműen el lehet dönteni, hogy egy kódnak a továbbítás során mikor van vége és mikor kezdődik egy új kódszó. Nyilvánvaló, hogy ezt a döntést akkor tudjuk végrehajtani, ha kódszavaink olyanok, hogy egyik kódszó sem lehet folytatása egy másik kódszónak. Az ilyen kódokat nevezzük *prefix-mentes* kódoknak.

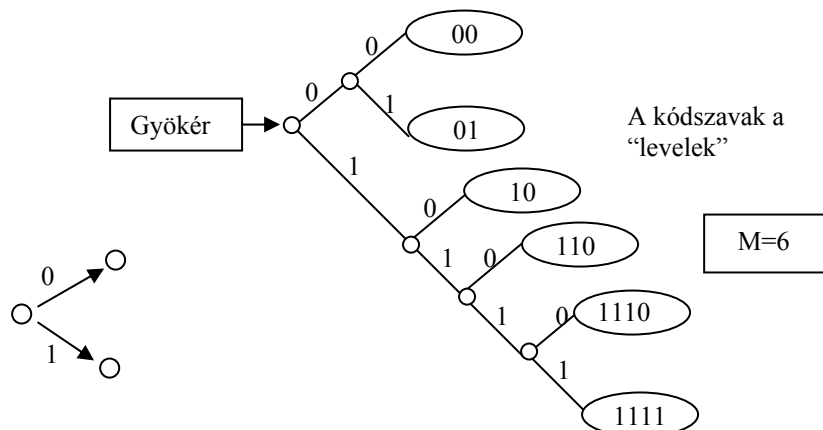
Itt jegyezzük meg, ha az i -ik kódszó hosszúságát n_i -nek választjuk, akkor az **egyértelmű dekódolhatóság** feltétele:

$$\sum_{i=1}^M 2^{-n_i} \leq 1 \quad (5.)$$

az un. **Kraft egyenlőtlenség** teljesülése (amit itt nem bizonyítunk).

1.3. Prefix-mentes kódok

Prefix-mentes kódokat egy bináris fa segítségével tudunk generálni, amely fa végső ágain (a “levelek” helyén) helyezkednek el a kódszavak.



Az ábrán egy nem teljes, bináris fa látható. A gyökértől elindulva, ha elágazási ponthoz érünk, akkor a kódszó egy bitjét generáljuk. A bit értéke attól függ, hogy felfelé vagy lefelé haladunk tovább. Az ábrán a felfelé haladáskor a "0" bitet, a lefelé haladáskor az "1" bitet választottuk (ami természetesen önkényes választás). Mivel levelek (kódszavak) csak a legvégső ágakon helyezkednek el, a generált kód biztosan prefix-mentes.

A dekódolás algoritmus is ezen bináris fa alapján működhet. A dekódolóban a levelek helyén a kódszavak helyett a kódszóhoz tartozó szimbólumot helyezzük el. Egy éppen beérkezett kódszó egymás után érkező bitjeit egy vasúti rendező pályaudvar váltó-vezérlő jeleként használva, eljutunk a dekódolt szimbólumhoz.

A kérdések ezek után a következők:

- Hogyan lehet prefix-mentes kódokat *szisztematikusan generálni* a P forráseloszlás ismeretében?
- Hogyan állítható elő a *legtömörebb* reprezentáció?
- Mi a *tömörítés elvi határa*?

1.4. A Shannon-Fano kódolás

Legyen adva az $X = \{x_1, x_2, x_3, \dots, x_M\}$ szimbólumkészlethez tartozó $P = \{p_1, p_2, p_3, \dots, p_M\}$ diszkrét forráseloszlás. Ebben az algoritmusban az egyes szimbólumokhoz tartozó kódszavak n_i hosszát a szimbólum információ tartalmának felfelé kerekített értékére választjuk:

$$n_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil \quad [\text{bit/szimbólum}] \quad (6.)$$

ahol $\lceil \cdot \rceil$ a felfelé kerekítés jele. (Ezért igaz az: $a \leq \lceil a \rceil < 1 + a$ egyenlőtlenség.)

Az így megállapított szóhosszúságú kódok dekódolhatók lesznek, mert a Kraft egyenlőtlenség értelmében:

$$\sum_{i=1}^M 2^{-n_i} = \sum_{i=1}^M 2^{-\lceil \log_2 \frac{1}{p_i} \rceil} \leq \sum_{i=1}^M 2^{-\log_2 \frac{1}{p_i}} = \sum_{i=1}^M 2^{\log_2 p_i} = \sum_{i=1}^M p_i = 1 \quad (7.)$$

azaz:

$$\sum_{i=1}^M 2^{-n_i} \leq 1$$

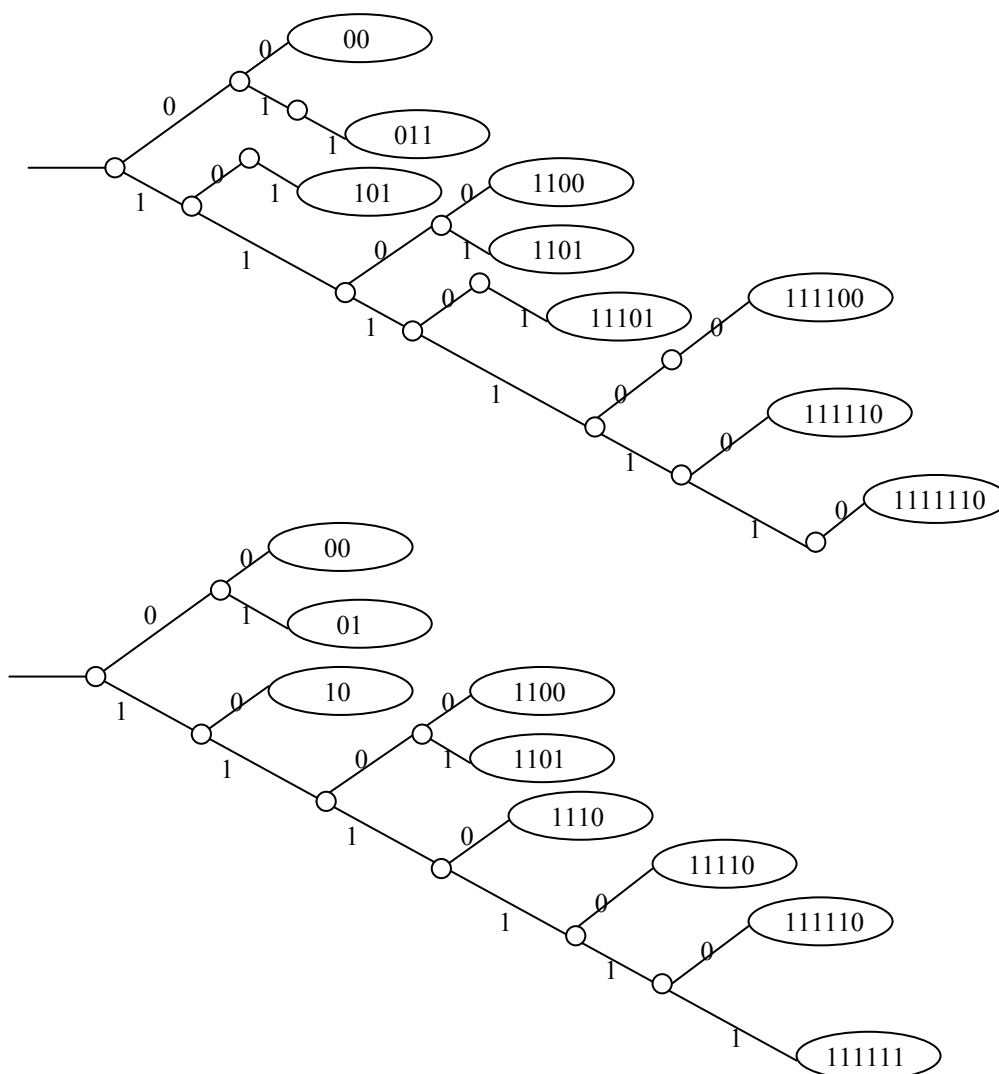
A Shannon-Fano algoritmus:

1. A szóhosszúságok meghatározása az $n_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$ képlettel.
2. Egy segédmenyiség kiszámítása: $F_{i+1} = F_i + p_i$, ahol $F_1 = 0$ $i = 1, 2, 3, \dots, M$
3. A kód generálása: $C_i = \lfloor 2^{n_i} F_i \rfloor$ ahol $\lfloor \cdot \rfloor$ a lefelé kerekítést jelöli.

A módszer alkalmazására tekintsünk egy numerikus példát $M = 9$ szimbólum esetére!

i	1.	2.	3.	4.	5.	6.	7.	8.	9.
p_i	0.49	0.14	0.14	0.07	0.07	0.04	0.02	0.02	0.01
n_i	2	3	3	4	4	5	6	6	7
F_i	0.00	0.49	0.63	0.77	0.84	0.91	0.95	0.97	0.99
C_i	0	3	5	12	13	29	60	62	126
C_i^{BIN}	00	011	101	1100	1101	11101	111100	111110	1111110

A táblázat kitöltése után rajzoljuk le a kódgenerálást leíró bináris fát (gráfot).
Látható, hogy a generált kód redundáns, ugyanis egyes ágvégek lerövidíthetők (lásd alsó ábra). Ez a példa is szemlélteti, hogy a Shannon-Fano eljárás bár szisztematikus, de nem az optimális (legrövidebb) kódot eredményezi.



Az **átlagos szóhosszúság** a szóhosszúság várható értéke:

$$L(P) = \mathbf{E}\{n_i\} = \sum_{i=1}^M p_i n_i \quad (8.)$$

Példánkban ez az érték, ha kiszámítjuk $L = 2.8900$ –nek adódik.

Felmerül a kérdés: lehet-e ez az érték kisebb (mint láttuk, igen.) és mi az elvi határ ?

Az átlagos szóhosszúságra vonatkozóan megadható egy **alsó** és **felső korlát** a (Shannon-féle) forráskódolási tétellel:

$$\boxed{H(P) \leq L < 1 + H(P)} \quad (9.)$$

Ez a tétel azt állítja, hogy egy adott forráseloszláshoz mindig létezik olyan veszteségmentes (dekódolható) kód, melyre a (9.) állítás igaz.

A Shannon-Fano algoritmusra a bizonyítás:

$$1. \quad L = \sum_{i=1}^M p_i \left\lceil \log_2 \frac{1}{p_i} \right\rceil \geq \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} = H(P) \quad (10.)$$

$$2. \quad L = \sum_{i=1}^M p_i \left\lceil \log_2 \frac{1}{p_i} \right\rceil < \sum_{i=1}^M p_i \left(1 + \log_2 \frac{1}{p_i} \right) = \sum_{i=1}^M p_i + \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} = 1 + H(P) \quad (11.)$$

Példánkban $H(P) = \sum_{i=1}^9 p_i \log_2 \frac{1}{p_i} = 2.3136$ adódik. ($2.3136 < 2.8900 < 3.3136$)

Az elvi határtól való távolságot az alább értelmezett hatásfokkal reprezentáljuk:

$$\eta(P) = \frac{H(P)}{L(P)} = 80.06\% \quad (12.)$$

1.5. Az *optimális (Huffman) kódolási eljárás*:

Ez az eljárás egy táblázatot használó szisztematikus módszer és abban az értelemben optimális, hogy a legkisebb átlagos szóhosszúságot eredményezi.

Az algoritmus lépései:

1. A táblázat $k=0$ oszlopába kiindulásul a *szimbólumok valószínűségeit* írjuk csökkenő sorrendben.
2. Ez után un. *állapot-redukciót* hajtunk végre olyan módon, hogy az oszlopban lévő két legkisebb értéket összeadjuk. (Egyforma értékek esetén célszerű a legalsót választani.)

3. Kitöltjük az új oszlopot, melyben az állapotok száma már eggyel kisebb. Ugyanakkor valamilyen módszerrel jelöljük, hogy melyik két állapotot vontuk össze.
4. A 2. és a 3. lépést $(M-1)$ -szer ismételjük, míg az utolsó lépésben az összevonás után egy állapotot kapunk. Ennek az egy állapotnak az értéke (1.) értelmében 1.00 kell hogy legyen. (Ez egyben ellenőrzési lehetőség is, hogy eddig jól számoltunk-e.) Ezt az állapotot tekintjük a gyökérnek.
5. A $k=0$ oszloptól kiindulva, majd a gyökér felé haladva, minden szimbólumhoz kiolvassuk a kódszavakat. A kódszó bináris karaktereit jobbról balra haladva kapjuk meg úgy, hogy amikor egy összevonási helyhez érünk, akkor generálunk kód karaktert. A karakter értéke attól függ, hogy az összevonási pontba alulról vagy felülről érkezünk. (Például alulról jöve 1-et, felülről érkezve 0-at választunk)

A módszer szemléltetésére tekintsünk ismét egy numerikus példát. Annak érdekében, hogy összehasonlíthassuk az eddig megismert két módszert, a forráseloszlás legyen ugyanaz, mint az előző példában.

x_i/k	0.	1.	2.	3.	4.	5.	6.	7.	8.
x_1	0.49 —	0.49 —	0.49 —	0.49 —	0.49 —	0.49 —	0.49 —	0.49 —	1.00
x_2	0.14 —	0.14 —	0.14 —	0.14 —	0.14 —	0.14 —	0.28 —	0.51 —	
x_3	0.14 —	0.14 —	0.14 —	0.14 —	0.14 —	0.14 —	0.23 —		
x_4	0.07 —	0.07 —	0.07 —	0.07 —	0.14 —	0.23 —			
x_5	0.07 —	0.07 —	0.07 —	0.07 —	0.09 —				
x_6	0.04 —	0.04 —	0.04 —	0.09 —					
x_7	0.02 —	0.02 —	0.05 —						
x_8	0.02 —	0.03 —							
x_9	0.01 —								

A szimbólumokhoz tartozó kódok és szóhosszúságok:

x_i	C_i	n_i
x_1	0	1
x_2	100	3
x_3	101	3
x_4	1100	4
x_5	1101	4
x_6	1110	4
x_7	11110	5
x_8	111110	6
x_9	111111	6

Az átlagos szóhosszúságot kiszámítva: $L(P) = \sum_{i=1}^9 p_i n_i = 2.3300$ -át kapunk, ami kisebb az előző példában kapott értéknél. A hatásfok:

$$\eta(P) = \frac{H(P)}{L(P)} = 99.33\%$$

1.6. Blokk-kódolás

A blokk-kódolás jelentőségét egy példa segítségével kívánjuk bemutatni. Legyen egy $X = \{0,1\}$ bináris forrásunk, melynek valószínűség-eloszlása $P_X = \{p, (1-p)\}$. A "0" szimbólum előfordulási valószínűsége legyen: $p = 0.9$ és $(1-p) = 0.1$ az "1" szimbólum valószínűsége. A forrásból egymás után kilépő szimbólumok legyenek egymástól függetlenek.

Ha a forrásszimbólumokat egyenként kódoljuk, egy bites kódszavakat kell minimálisan választani a függetlenség miatt. (Valójában nem hajtunk végre kódolást, mivel maguk a szimbólumok lehetnek a kódszavak.)

Ezzel a módszerrel az átlagos kódszó hosszúság: $L_1 = 1$ és az entrópia:

$$H_1(p) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} = 0.1368 + 0.3322 = 0.4690$$

-nek adódik. A " kódolás " hatásfoka: $\eta_1 = \frac{H_1}{L_1} = 46.90\%$

A $H_1 \leq L_1 < H_1 + 1$ feltétel természetesen teljesül, de láthatóan L_1 a felső határhoz van közelebb.

Felmerül a kérdés: lehetne-e valamilyen módon javítani a hatásfokon?

Ennek érdekében például a forrásból egymás után kilépő 3 szimbólumot foglaljuk össze egy ún. blokkba. A blokkok Y szimbólumábécéje $N = 8$ elemet fog tartalmazni:

$$Y = \{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8\} = \{ "000", "001", "010", "011", "100", "101", "110", "111" \}$$

Az Y forrás $P_Y = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$ eloszlását az X forrás eloszlásából tudjuk meghatározni. Mivel az X forrás elemei egymástól függetlenek, ezért:

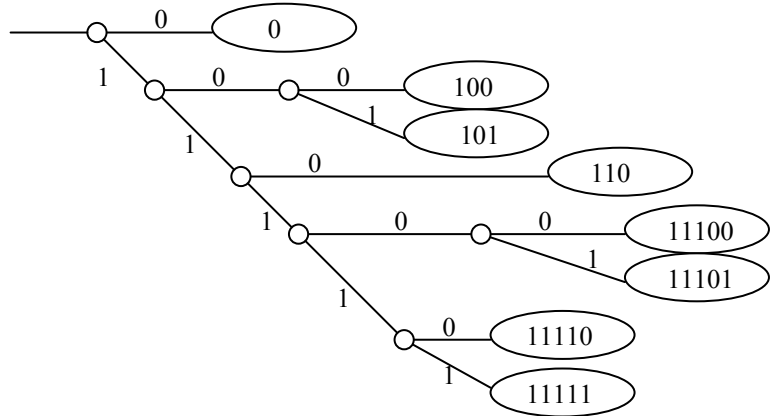
$$\begin{aligned} p_1 &= \Pr\{y = "000"\} &= p^3 &= 0.729 \\ p_2 = p_3 = p_5 &= \Pr\{y = \text{"két 0 és egy 1"}\} &= p^2(1-p) &= 0.081 \\ p_4 = p_6 = p_7 &= \Pr\{y = \text{"két 1 és egy 0"}\} &= p(1-p)^2 &= 0.009 \\ p_8 &= \Pr\{y = "111"\} &= (1-p)^3 &= 0.001 \end{aligned}$$

Huffman kódolást választva, a táblázat:

y_i	0.	1.	2.	3.	4.	5.	6.	7.
"000"	0.729 —	0.729 —	0.729 —	0.729 —	0.729 —	0.729 —	0.729 —	1.000
"001"	0.081 —	0.081 —	0.081 —	0.081 —	0.081 —	0.162 —	0.271 —	
"010"	0.081 —	0.081 —	0.081 —	0.081 —	0.081 —	0.109 —		
"100"	0.081 —	0.081 —	0.081 —	0.081 —	0.109 —			
"011"	0.009 —	0.009 —	0.018 —	0.028 —				
"101"	0.009 —	0.009 —	0.010 —					
"110"	0.009 —	0.010 —						
"111"	0.001 —							

A táblázatból kiolvasható kódok és szóhosszúságok:

y_i	c_i	n_i
"000"	0	1
"001"	100	3
"010"	101	3
"100"	110	3
"011"	11100	5
"101"	11101	5
"110"	11110	5
"111"	11111	5



A blokk-kódok átlagos szóhosszúsága $L_3(P) = \sum_{i=1}^8 p_i n_i$ alapján:

$$L_3 = 0.729 + 0.081 \cdot (3 + 3 + 3) + 0.009 \cdot (5 + 5 + 5) + 0.001 \cdot 5 = \\ = 0.729 + 0.729 + 0.135 + 0.005 = 1.598 \text{ bit/ szimbólum}$$

és az entrópiája:

$$H_3(P) = \sum_{i=1}^M p_i \log_2 \frac{1}{p_i} = 1.4070$$

adódik. A kód hatásfoka:

$$\eta_3 = \frac{H_3}{L_3} = 88.05 \%$$

Most az $L_3 = 1.598$ -as érték már sokkal közelebb van a H_3 -as elvi határhoz, mint az előző esetben. Meg lehetne-e még jobban közelíteni? Ha igen, mitől várhatjuk azt? A blokk $N=3$ választása meglehetősen önkényesnek tűnik.

A kérdéssel érdemes foglalkozni, mert a példa kedvéért mondjuk van egy 3000 szimbólumból álló X állományunk. Az első módszerrel 3000, a másodikkal csak 1598 bitért kell fizetnünk, amikor a drága csatornát egy szolgáltatótól igénybe vesszük.

1.7. Blokk-kódok entrópiája

Ha a memóriamentes X forrás elemei az $X = \{x_1, x_2, x_3, \dots, x_M\}$ szimbólumok, adott $P = \{p_1, p_2, p_3, \dots, p_M\}$ valószínűségi eloszlással és $H_1(P)$ entrópiával, akkor az N darab egymás utáni szimbólum összefogásával kapott blokk entrópiája:

$$H_N(P) = N H_1(P) \quad (13.)$$

Természetesen a blokk-kódra is igaz a (9.) tétel, mely szerint:

$$H_N(P) \leq L_N < 1 + H_N(P) \quad (14.)$$

amibe a (13.)-at behelyettesítve:

$$N H_1(P) \leq L_N < 1 + N H_1(P) \quad (15.)$$

Osszuk el (15.)-öt N -nel!

$$H_1(P) \leq \frac{L_N}{N} < \frac{1}{N} + H_1(P) \quad (16.)$$

A (16.) azt fejezi ki, hogy az *egy eredeti szimbólumra vett szóhosszúság* (L_N/N) N növelésével két olyan határ közé szorítható be, amelyek egymáshoz tetszőlegesen közel lehetnek.

A (13.) bizonyítását általánosabb esetre is érvényes összefüggés belátásával végezzük el a következő módon:

Legyen két, egymástól független $\mathbf{X}=\{x_1, x_2, x_3, \dots, x_M\}$ és $\mathbf{Y}=\{y_1, y_2, y_3, \dots, y_N\}$ szimbólum-ábécével rendelkező forrásunk. Az egyik mérete M a másiké N . A két forrás eloszlása rendre $\mathbf{R}=\{r_1, r_2, r_3, \dots, r_M\}$ és $\mathbf{Q}=\{q_1, q_2, q_3, \dots, q_N\}$.

A két forrás entrópiája:

$$H_X(R) = \sum_{i=1}^M r_i \log \frac{1}{r_i} \quad \text{és} \quad H_Y(Q) = \sum_{j=1}^N q_j \log \frac{1}{q_j} \quad (17.)$$

Egyesítsük a két forrásból származó x_i és y_j szimbólumokat ($i=1,2,\dots,M$ $j=1,2,\dots,N$) egyetlen $z_k = (x_i y_j)$ szimbólumba. Az új szimbólumok ábécéje $\mathbf{Z}=\{z_1, z_2, z_3, \dots, z_{MN}\}$, melynek mérete MN .

Az egyesített forrás eloszlása:

$$p_k = \mathbf{Pr}\{x = x_i \text{ és } y = y_j\} = \mathbf{Pr}\{x = x_i\} \mathbf{Pr}\{y = y_j\} = r_i q_j \quad (18.)$$

melynek entrópiája:

$$\begin{aligned} H_Z(P) &= \sum_{k=1}^{MN} p_k \log \frac{1}{p_k} = \sum_{i=1}^M \sum_{j=1}^N r_i q_j \log \frac{1}{r_i q_j} = \sum_{i=1}^M \sum_{j=1}^N r_i q_j \left(\log \frac{1}{r_i} + \log \frac{1}{q_j} \right) = \\ &= \left[\sum_{i=1}^M \sum_{j=1}^N r_i q_j \log \frac{1}{r_i} \right] + \left[\sum_{i=1}^M \sum_{j=1}^N r_i q_j \log \frac{1}{q_j} \right] = \sum_{j=1}^N q_j \sum_{i=1}^M r_i \log \frac{1}{r_i} + \sum_{i=1}^M r_i \sum_{j=1}^N q_j \log \frac{1}{q_j} = \\ &= \sum_{j=1}^N q_j H_X(R) + \sum_{i=1}^M r_i H_Y(Q) = H_X(R) \sum_{j=1}^N q_j + H_Y(Q) \sum_{i=1}^M r_i = H_X(R) + H_Y(Q) \end{aligned} \quad (19.)$$

Mivel: $\sum_{i=1}^M r_i = 1$ és $\sum_{j=1}^N q_j = 1$

Tehát végül a:

$$H_Z(P) = H_X(R) + H_Y(Q) \quad (20.)$$

eredményt kapjuk. **A független források entrópiája tehát összeadódik.**

Ha $X=Y$, akkor (20.) alapján:

$$H_Z(P) = 2H_X(R)$$

Innen már teljes indukcióval könnyű belátni a (13.) teljesülését, amit bizonyítani akartunk.

1.8. A bináris csatorna kapacitása.

Mint a bevezetőben említettük, a csatornában a zajok és torzítások miatt hiba következhet be, emiatt a vett üzenet nem egyezik meg a leadottal. Ha hiba van, akkor azt ki akarjuk javítani, tehát további (hibajavító) üzenetet kell küldenünk, ami azt jelenti, hogy a hibázó csatorna miatt a továbbítandó adatmennyiség megnövekszik.

Az alábbiakban azt a kérdést kívánjuk vizsgálni, hogy **elvileg** mennyivel kell megnövelni a továbbítandó adatok mennyiségét annak érdekében, hogy a hibátlan üzenet átjusson a vételi oldalra. Tételezzük fel, hogy a csatornában bináris (0,1) kódokat küldünk és a továbbítás során a *hibás átvitel valószínűsége*: p . Az eredeti üzenet hosszúsága legyen: K , míg a *lehető legrövidebb*, de teljes (az eredeti + a javító) üzenet hossza legyen: N . ($N \geq K$)

A csatorna kapacitását a:

$$C(p) = \frac{K}{N} \leq 1 \quad (21.)$$

hányadossal definiáljuk.

Arra vagyunk kíváncsiak, hogyan függ az átvitel hatékonysága (a csatorna kapacitása) a *hibás átvitel valószínűségétől*, a p -től.

Hangsúlyoznánk, hogy most nem egy gyakorlati megvalósítást, hanem az *elvi határt* keressük, ezért a körülményeket speciálisan fogjuk megválasztani. A kísérletünket képzeljük el úgy, mintha egy olyan számítógép előtt ülnénk melynek hibás a billentyűzete (hibázó beviteli csatorna), de a képernyőt (a vételi oldalt) jól látjuk.

Gépeljünk be ebbe a számítógépbe elsőnek az eredeti üzenetnek szánt $L_0 = K$ hosszú bináris sorozatot. A képernyőn megjelenik ez a sorozat, (de a néha hibázó billentyűzet miatt) a képernyőn némelyik karakter nem az lesz, amit küldtünk. Képezzünk most egy ugyancsak K hosszúságú hiba sorozatot oly módon, hogy azokba a pozíciókba ahol az átvitel jól sikerült 0-át, ahol meg nem sikerült, oda 1-et írunk.

Ebben a hibákat leíró sorozatban az 1-esek előfordulási valószínűsége p , a kiindulási feltételünknek megfelelően. Ha ez a valószínűség kicsi, akkor a hibának, mint forrásnak a $H_0(p)$ entrópiája is kicsi lesz. (Hangsúlyozzuk, hogy $H_0(p)$ nem az eredeti üzenetforrás, hanem a *csatornát jellemző hibaforrás* entrópiáját jelöli.)

A K hosszú (első) hibasorozat entrópiája (13.) értelmében:

$$H_1 = K H_0(p) \quad (22.)$$

Ezt a hibasorozatot érdemes forráskódolni, mivel az entrópiája kicsi. Ha a K elég nagy, blokkos kódolással az első kódolt hibáüzenet L_1 hosszúsága (15.) értelmében megközelítheti H_1 értékét:

$$L_1 = H_1 = K H_0(p) \quad (23.)$$

Másodszorra ezt a kódolt hibaüzenetet küldjük el, amit a vételi oldalon dekódolunk, majd a tartalomtól függően kijavítjuk a hibákat. A billentyűzet azonban nem javult meg, így ez a javító üzenet is hibás lehet, melynek hibáit újra látjuk a képernyőn. Képezzünk most egy második L_1 hosszúságú hibasorozatot, melynek H_2 entrópiája:

$$H_2 = L_1 H_0(p) \quad (24.)$$

Ezt a sorozatot is blokkosan kódoljuk és úgy továbbítjuk. A szükséges hosszúság, hasonlóan a fenti gondolatmenethez:

$$L_2 = H_2 = L_1 H_0(p) = K H_0^2(p) \quad (25.)$$

A hibák hibáinak javítási eljárását (elvileg végtelen sokszor) ismételve tudjuk az üzenetet hibamentesen átvinni. A teljes átvendő hosszúság:

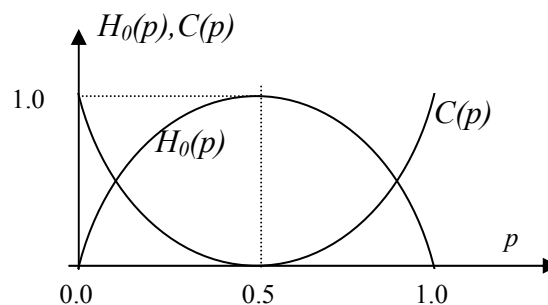
$$\begin{aligned} N &= L_0 + L_1 + L_2 + \dots = K + K H_0 + K H_0^2 + \dots = K(1 + H_0 + H_0^2 + \dots) = \\ &= K \frac{1}{1 - H_0(p)} \end{aligned} \quad (26.)$$

Ebből a csatorna kapacitása:

$$C(p) = \frac{K}{N} = 1 - H_0(p) = 1 - p \log \frac{1}{p} - (1 - p) \log \frac{1}{1 - p} \quad (27.)$$

Az ábrából láthatóan, ha $p = 1$ (a hiba valószínűsége a legnagyobb) az ugyanazt a csatornkapacitást eredményezi, mint a $p = 0$ -hoz tartozó (hibamentes) eset. Ez persze nem meglepő, mert a $p = 1$ azt jelenti, hogy a csatorna *szisztematikusan* minden bitet invertál és egy, a kimenetre helyezett inverterrel az eredeti bitfolyam hibamentesen visszaállítható.

A $p = 0.5$ eset láthatóan a legrosszabb. Ekkor ugyanis a kimenet (függetlenül a bemenettől) tetszőleges értéket vehet fel. (Nincs esély az átvitelre.)



Bináris csatorna kapacitása a csatorna hiba előfordulási valószínűségének függvényében