Információs rendszerek elméleti alapjai

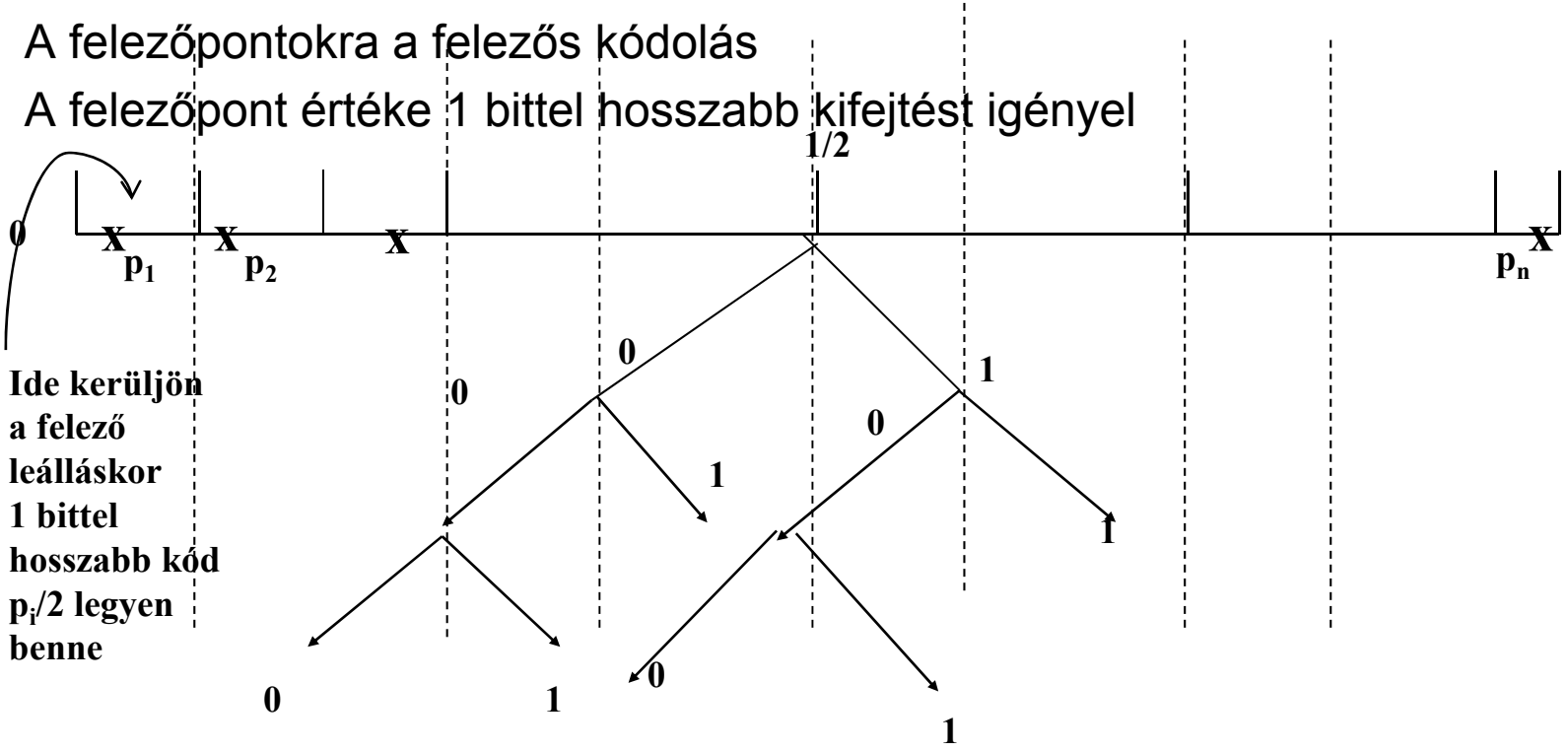
Információelmélet

Gilbert-Moore

Szemléltetése hasonló a Shannon kódhoz

A felezőpontokra a felezős kódolás

A felezőpont értéke 1 bittel hosszabb kifejtést igényel



Az intervallum felezőpontjára felezéses kódfa, leáll ha egyedüli marad,

$p_1, \ldots, p_n$ nincsenek rendezve

$Q_1, \ldots, Q_n$ az intervallumok felezőpontja

A kódját (mint Shannon - Fano kódnál) felezésekkel építjük fel.

$$\sum_{i=1}^{n} p_i l_i < H + 2$$

Gilbert-Moore

A forrás kimeneti jelei tetszőleges sorrendbe rendezhetők

pl. alfabetikus, nem a val.szg.-ek szerint

u_i kódszó, $length(u_i) = l_i$

Legyen $2^{1-l_i} \leq p(u_i) < 2^{2-l_i}$

$$\alpha_1 = \frac{1}{2} p(u_1),$$

$$\alpha_2 = p(u_1) + \frac{1}{2} p(u_2),$$

$$\vdots$$

$$\alpha_i = p(u_1) + p(u_2) + \ldots + p(u_{i-1}) + \frac{1}{2} p(u_i)$$

$$\alpha_1 \leq \alpha_2 \leq \ldots \leq 1$$

Az u_i üzenetre vonatkozó kód az α_i bináris kifejtése l_i hosszban

Példa Gilbert-Moore kódra

Example 2.15

Take the first three letters of the alphabet a, b and c. With the given probabilities of occurring for the English language (see Figure 2.2) we find:

symbol	probability	l_i	α_i	code ($r = 2$)
u_1	0.064	5	0.032	00001
u_2	0.013	8	0.071	00010010
u_3	0.022	7	0.088	00010111

△

Huffman kód

Optimális kód:

Rekurzív felépítés az eloszlás elemszáma szerint

$(q_1,\ldots,q_s)$ – ismerjük az eloszlást

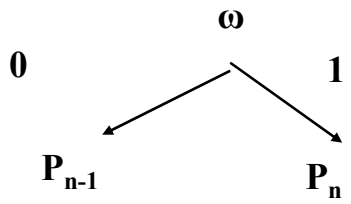
$(p_1,\ldots,p_s,p_{s+1})$ – re, legyen p_s, p_{s+1} a két legkisebb

p_s, p_{s+1} kódja legyen azonos hosszú,

$p_s : \omega 0, p_{s+1} : \omega 1$, alakú.

A $(p_1,\ldots,p_s + p_{s+1} = q)$ eloszlásra készítjük el a Huffman kódot,

legyen q kódja ω



Huffman kód

Example 2.13

symbol	probability				code ($r = 2$)
u_1	0.4	0.4	0.4	0.4	1
u_2	0.3	0.3	0.3	0.3 (0)	00
u_3	0.1	0.1	0.2 (0)	0.3 (1)	011
u_4	0.1	0.1 (0)	0.1 (1)		0100
u_5	0.06 (0)	0.1 (1)			01010
u_6	0.04 (1)				01011

A két legkisebb val.szg-ű forráskód szimbólumot összevonjuk, ezzel az üzenet *abc*-t egy szimbólummal csökkentettük. Újra rendezzük, majd megint a két legkisebb val.szg-ű forráskód szimbólumot összevonjuk. Ezt ismételjük rekurzív módon, addig, amíg az üzenet *abc* két szimbólumra redukálódik, ezeket a szimbólumokat a bináris kód *0* illetve *1* szimbólumával jelöljük meg. Ezek után a rekurzív eljárást megfordítjuk, visszafelé menve *0* illetve *1* szimbólumokat konkatenálunk a kódszavakhoz, azokon a pontokon, ahol előbb két kódszót összevontunk.

Huffman kód

Állítás: A prefix kódok között a Huffman kód optimális

Bizonyítás: indukcióval

$$\underbrace{\sum_{i=1}^{n} l(\omega_i')p_i}_{\text{Bármely más prefix kód}} \geq \underbrace{\sum_{i=1}^{n} l(\omega_i)p_i}_{\text{Huffman-kód}}$$

$$\sum_{i=1}^{n} l(\omega_i)p_i < H + 1$$

Vegyünk tetszőleges

$\omega_1, \ldots, \omega_n$, prefix - mentes kódot

$(p_1, \ldots, p_{n-1}, p_n)$ - hez p_{n-1}, p_n, legyen a két legkisebb érték

Ha nem a leghosszab kódok tartoznak hozzájuk,

akkor cseréljük fel. Ezzel a kódhossz várható értéke nem nő

Átrendezhetjük a leghosszabb kódokat(párban vannak), hogy

p_{n-1}, p_n kódja $\omega 0, \omega 1$ legyen.

Összevonva $p_n + p_{n+1} = q$ - t, a feladatot eggyel rövidebb eloszlásra

vezettük vissza.

Huffman kód

Egy szimbólumra jutó kódhossz várható értéke

$\frac{1}{\mathrm{N}}$ - re közelíti a H - t.

Ha gyakoriságok vannak : $x_i - n_i -$ szer fordul elő

$$\sum_{i=1}^{n} n_i = N$$

$$min \sum_{i=1}^{n} n_i l(\omega_i) \Leftrightarrow min \sum_{i=1}^{n} \underbrace{\frac{n_i}{N}}_{\text{eloszlás}} l(\omega_i)$$

A $\xi_1,\ldots,\xi_n$ együttes eloszlásához a Huffman kóddal,

vagy a Shannon - Fano kóddal elérhető

$H(\xi_1,\ldots,\xi_n)+1$ várható értékű kódhossz.

Mivel $H(\xi_1,\ldots,\xi_n) = N(H+\delta)$,

az egy szimbólumra jutó kódhossz,

$$\frac{1}{N}\left(H(\xi_1,\ldots,\xi_n)+1\right) \le (H+\delta)+\frac{1}{N} \sim H+\frac{1}{N}$$

Nevezetes kódok

Tetszőleges közel tudunk menni a C /H sebességgel, ha nagyon sok hosszú kódsorozatot kódolok egyben.

Nagyon nagy kódok, valamilyen tipikus halmazon való viselkedéshez hasonlóan működnek.

Univerzális forráskódolás

Adaptív kód:

Nincs előre adott kódszó rendszer, amit konstans kezdeti költségű kódként át kell küldeni. (mint pl. Shannon-Fano. Huffman kód)

Univerzális forráskódolás

(Győrfi,L.: Információ és kódelmélet)

- Az eddig vizsgált kódok alkalmazásakor az adó és a vevő között átvitelre kerülő bitek két csoportot alkotnak.
- Először átvisszük a blokk-kódot leíró információt.
- Ez egy állandó költséget
- jelent, független az üzenet tényleges hosszától. Majd következnek az üzenet kódszavai.

Univerzális forráskódolás

Elméleti vizsgálataink során azzal a feltételezéssel éltünk, hogy a továbbítandó üzenetünk végtelen hosszú.

Ily módon, a kódok aszimptotikus viselkedését tekintve, az állandó költség fajlagosan nullához tart, tehát elhanyagolható.

A gyakorlatban azonban véges forrásokkal van dolgunk.

Univerzális forráskódolás

- Ebben az esetben az állandó költség akár nagyobb is lehet, mint az üzenet kódszavainak összhosszúsága.
- Ezt elkerülendő , jó lenne, ha rendelkezésünkre állna egy olyan technika, amelynek nincs állandó költsége, de aszimptotikusan ugyanolyan jó tömörítési arányt ér el, mint a blokk-kódok.
- Az állandó költség abból adódik, hogy a kódot a forráson előzetesen elvégzett statisztikai vizsgálatok (a forrásszimbólumok gyakorisága) alapján hozzuk létre, tehát ezek az adatok szükségesek a kód leírásához.

Univerzális forráskódolás

- Ehelyett járjunk el úgy, hogy menet közben gyűjtünk információt a forrásszimbólumokról, vagyis az aktuális szimbólumot az ezt megelőző szimbólumok alapján kódoljuk.
- Az ilyen kódokat **adaptív kódok**nak nevezzük, alkalmazásuk során nincs állandó költség. (Létezik pl. adaptív Huffman-kód). A most tárgyalásra kerülő Lempel–Ziv-kódok is ebbe a családba tartoznak.

Univerzális forráskódolás

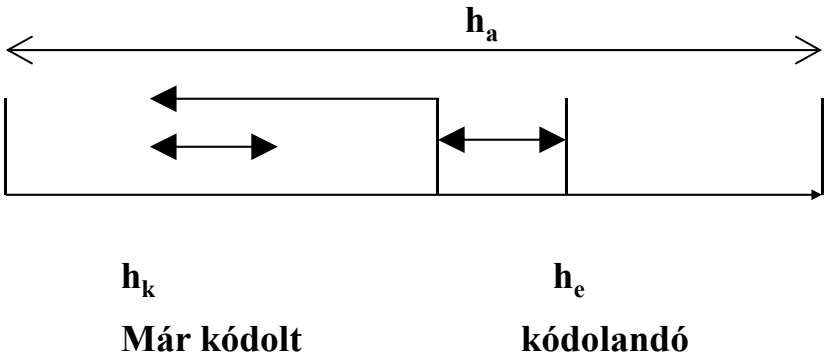
Az első LZ-algoritmus az 1977-ben publikált LZ77.

Lempel-Ziv kódok

LZ77 –algoritmus

Csúszóablakban nézi a forrásszimbólumokat

$h_a = h_k + h_e$



Az LZ77 algoritmus (ld. Győrfi)

A kódoló a forrásszimbólumok sorozatát egy h_a hosszú csúszó ablakon keresztül vizsgálja.

Az ablak két részből áll:

- egy kereső pufferből, amely a legutóbb kódolt h_k darab forrásszimbólumot tartalmazza,
- és egy előretekintő pufferből, amely a következő h_e darab kódolandó szimbólumot tartalmazza ($h_a = h_k + h_e$).

A kódoló a kereső pufferben megkeresi az előre tekintő puffer első szimbólumával megegyező szimbólumokat. Ehhez egy hátrafelé haladó mutatót használ.

$$\lceil \log h_k \rceil + \lceil \log h_e \rceil + \lceil \log |X| \rceil \quad (*)$$

Az LZ77 algoritmus (ld. Győrfi)

Megnézi, hogy a megtalált pozíciókkal kezdődően, a kereső pufferben lévő szimbólumok milyen hosszan egyeznek meg az előretekintő puffer szimbólumaival, és a találatok közül azt választja ki, amelytől kezdve a leghosszabb az egyezés.

A kódoló ezután elküld egy $<t\ h\ c>$ hármast, ahol t a kereső pufferben megtalált szimbólum távolsága az előretekintő puffertől (*offset*), h a kereső- és az előre- tekintő puffer egyező szimbólumainak legnagyobb hosszúsága, c pedig az első, az előretekintő pufferben lévő nem egyező karakter kódszava.

Azért küldjük el az első nem egyező karakter kódját is, hogy kezeljük azt az esetet, amikor az előre tekintő puffer szimbólumait nem találjuk meg a kereső pufferben. Ilyenkor t és h értéke 0. Egy hármas kódolásához állandó hosszúságú kód használatával (*)bit szükséges, ahol $|X|$ a forrásábécé mérete.

Az egyező szimbólumok hosszúságának átviteléhez nem log h_k, hanem log h_e bit szükséges.

Ennek oka, hogy az egyezés hossza meghaladhatja a kereső puffer hosszát,vagyis az egyezőrész átlóghat az előretekintő pufferbe.

$\lceil log\ h_k \rceil + \lceil log\ h_e \rceil + \lceil log|X| \rceil \quad (*)$

LZ77

- A LZ77 egy rendkívül egyszerű adaptív algoritmus, amely nem igényel előzetes ismeretet vagy feltevést a forrásról.
- Megmutatható, hogy az eljárás hatékonysága aszimptotikusan (h_k, $h_e \to \infty$) megközelíti az optimális algoritmusét, amely előzetesen ismeri a forrás eloszlást, azaz *stacionárius* és *ergodikus* forrás esetén az átlagos kód szóhossz konvergál *H*-hoz (bináris *abc*), ha (h_k, $h_e \to \infty$).
- Bár ez aszimptotikusan igaz, a gyakorlatban az LZ77 számos továbbfejlesztése ismeretes, amelyek célja a hatékonyság növelése.
- A népszerű PKZIP és ARJ tömörítőkben a hármasokat nem fix, hanem változó hosszúságú kóddal kódolják.
- Egy másik variáció változtatható méretű kereső és előretekintő ablakot használ.

LZ77

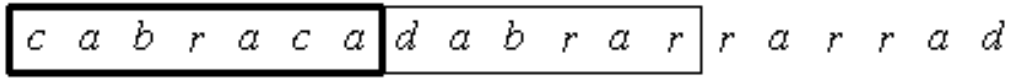
- Az LZ77 legegyszerűbb módosítása annak kiküszöbölése, amikor egyetlen karaktert kódolunk egy hármassal. Ez egy jelző bittel oldható meg. Ezzel jelezzük, hogy nem egy hármast, hanem csak egy kódszót küldünk át.
- Az LZ77 alkalmazása során a forrásszimbólumok legutóbb kódolt sorozatát használjuk, így azzal a feltételezéssel élünk, hogy a minták egymáshoz közeli intervallumokban visszatérnek (a mozgó ablakon belül). Szélsőséges esetben, ha az ismétlődés hossza éppen eggyel hosszabb a kereső puffer méreténél, nem tudunk tömöríteni.
- Az LZ-algoritmus következő , 1978-as verziójánál (LZ78) ezt a problémát egy másfajta, adaptív szótár alapú rendszerrel oldják fel.
- http://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossless/lz77/example.htm
- http://www.stringology.org/DataCompression/lz77/index_en.html

LZ77

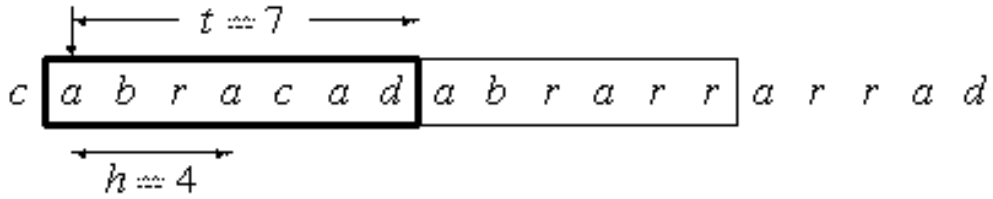
1.10. példa. Legyen a bementünk a következő:

$$\ldots cabracadabrarrarrad \ldots$$

$h_a := 13$, $h_k := 7$, $h_e := 6$. Tegyük fel, hogy az első néhány karaktert már kódoltuk. Ekkor:

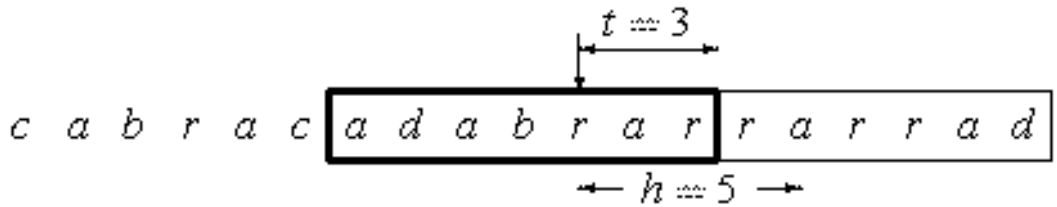


Látható, hogy az előretekintő puffer első karaktere, d, nem található meg a keresőpufferben. Átküldjük a $\langle 0, 0, f(d) \rangle$ hármast, ahol $f(d)$ a d karakter kódját jelöli. Az ablakot eggyel jobbra mozgatjuk, így:



LZ77

A mutatót a keresőpufferben hátrafelé mozgatva, az előretekintő puffer első szimbólumával (a) egyező karaktert $t = 2$ távolságra találjuk meg. Ekkor az egyezés hosszúsága $h = 1$. Tovább haladva a mutatóval $t = 4$-nél szintén egy 1 hosszú egyezés adódik. Végül $t = 7$-nél találjuk meg a legjobb választást $h = 4$ hosszal. Tehát az *abrar* karaktereket a $\langle 7, 4, f(r) \rangle$ hármassal kódoljuk, és az ablakot 5 pozícióval jobbra toljuk, így:



Az első egyezést $t = 1$-nél $h = 1$ hosszan találjuk. A második egyezés $t = 3$-nál van, hossza első ránézésre $h = 3$. Azonban az egyezés az előretekintő pufferbe is átnyúlik, ezért $h = 5$. Az átküldendő hármas: $\langle 3, 5, f(d) \rangle$. A dekódolás során ez az „átlógás” nem okoz gondot, mert az első három karaktert könnyen megkapjuk a már előzőleg dekódolt karakterekből, a maradék kettőt pedig az előbbi lépés során megkapott 3 karakter segítségével nyerjük.

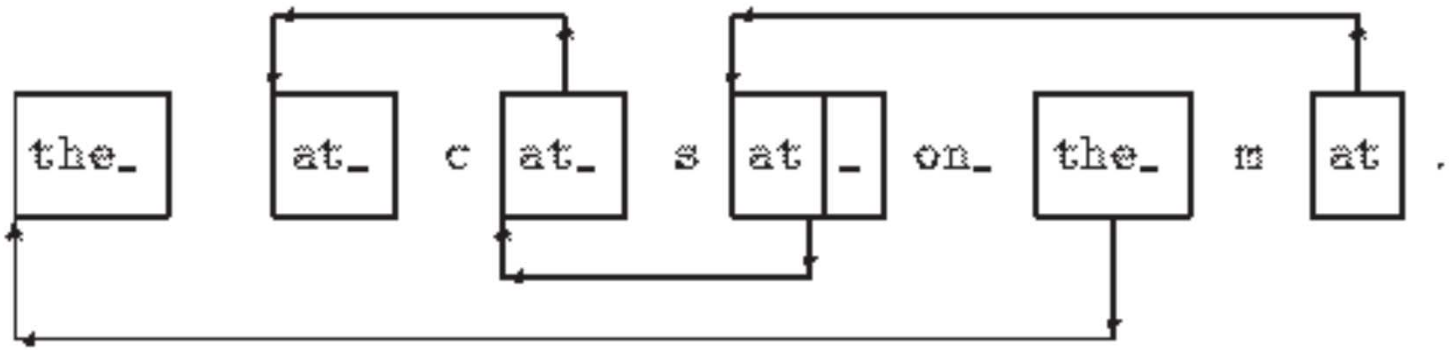
The_fat_cat_sat_on_the_mat.

(0,0,t) (0,0,h) (0,0,e) (0, 0,_) (0, 0,f) (0, 0, a) (0, 0, t) (0, 0, _) (0, 0, c);

(4, 3, s) (4, 3, o);

(0, 0, n) (0, 0, _);

(l9, 4, m) (ll,2,.)


the_
at_
c
at_
s
at
_
on_
the_
m
at
.

LZ78

Az LZ78 algoritmus

A kódoló és a dekódoló is szótárt épít az előzőleg elő fordult sorozatokból. A kódoló megkeresi a forrásszimbólumok aktuális pozíciójától kezdődő leghosszabb egyezést a szótárban. Átküld egy <*i*, *c*> párt, ahol *i* az egyező karaktersorozat szótárbeli indexét jelöli, *c* pedig az első nem egyező karakter kódja, majd felveszi a szótárba az *i* indexű egyező karaktersorozat és a *c* karakter konkatenációjaként kapott stringet („karakter füzért”) (a következő szabad indexet adja neki). Ha nem talál egyező karaktersorozatot a szótárban, akkor a <*0*, *c*> párost küldi át, *c* itt is az első nem egyező karakter kódja, amely ebben az esetben természetesen az első feldolgozandó szimbólum.

LZ78

a kódoló kimenete	szótár index	szótár bejegyzés
$\langle 0, f(d)\rangle$	1	d
$\langle 0, f(a)\rangle$	2	a
$\langle 0, f(b)\rangle$	3	b
$\langle 3, f(a)\rangle$	4	ba
$\langle 0, f(c)\rangle$	5	c
$\langle 1, f(a)\rangle$	6	da
$\langle 3, f(b)\rangle$	7	bb
$\langle 2, f(c)\rangle$	8	ac
$\langle 6, f(b)\rangle$	9	dab

a kódoló kimenete	szótár index	szótár bejegyzés
$\langle 4, f(c)\rangle$	10	bac
$\langle 9, f(b)\rangle$	11	$dabb$
$\langle 8, f(d)\rangle$	12	acd
$\langle 0, f(e)\rangle$	13	e
$\langle 13, f(c)\rangle$	14	ec
$\langle 1, f(e)\rangle$	15	de
$\langle 14, f(d)\rangle$	16	ecd
$\langle 13, f(e)\rangle$	17	ee

1.10. ábra. Az 1.11. példa LZ78 kódolásának menete.

1.11. példa. Kódoljuk a következő sorozatot az LZ78 algoritmussal:

dabbacdabbacdabbacdabbacdeecdeecdee

Kezdetben a szótár üres, ezért az első 3 szimbólumot egyenként felvesszük a szótárba, és a 0 indexszel átküldjük: $\{0, f(d)\}, \{0, f(a)\}, \{0, f(b)\}$. A negyedik szimbólum a b, amely szerepel a szótárban, a következővel együtt (ba) viszont már nem, ezért átküldjük a $\{3, f(a)\}$ párost, amelyből a 3 jelöli a b indexét, $f(a)$ pedig a következő karakter, vagyis az a kódját. A ba sorozatot felvesszük a szótárba, indexe 4 lesz. Így folytatjuk az eljárást, az eredményt az 1.10. ábrán látható táblázatban foglaltuk össze. Látható, hogy a szótárbeli bejegyzések egyre hosszabbak, és ha a bemeneti sorozat ismétlődik, akkor előbb-utóbb az egész sztring szerepelni fog a szótárban.

LZ78

- Megmutatható, hogy az LZ78 egy betűre jutó átlagos kódszó hossza konvergál $H(\mathbb{X})/\log s$ -hez minden stacionárius és ergodikus forrásra.
- $\mathbb{X}=X_1, X_2, \ldots, X_n$ ergodikus, stacionárius
- s a kódábécé elemszáma
- Az LZ78 algoritmus egyik hibája, hogy a szótár folyamatosan, korlát nélkül növekszik.
- A gyakorlatban egy bizonyos határon túl gátat szabunk a növekedésnek: vagy rendszeresen eltávolítjuk a felesleges vagy ritkán használt bejegyzéseket, vagy egy idő után fix szótárasként működik tovább az eljárás.

Az LZW algoritmus

- Terry Welch az LZ78 módosításával egy olyan technikát dolgozott ki, amellyel megtakarítható az <*i c*> párból a *c* karakterkód átküldése. Ez az ún. LZW algoritmus.
- A kódoló tehát csak szótárbeli indexeket küld át. Ehhez szükséges, hogy a szótárban már a kiinduló állapotban is szerepeljen az összes egybetűs szimbólum a forrásábécéből. A kódolás során az aktuális pozíciótól kezdve addig olvassuk be a forrásszimbólumokat a *p* pufferbe, amíg a sorozat szerepel a szótárban.
- Ha az *a* karakter az első olyan, amelyre *pa* nincs benne a szótárban (az egymás után írással a konkatenációt jelöltük), akkor átküldjük a *p* sorozat indexét, a *pa* sorozatot felvesszük a szótárba és az *a* karaktertől kezdve folytatjuk az eljárást.

1.12. példa. Kódoljuk az LZW algoritmussal az előző sorozatunkat:

$$dabbacdabbacdabbacdabbacdeecdeecdee$$

A forrásábécé $X = \{a, b, c, d, e\}$, kezdetben ez az 5 bejegyzés szerepel a szótárban. A kódoló először veszi a d karaktert. Ez benne van a szótárban, így hozzáilleszti a következő, az a karaktert. A da sorozat már nem szerepel a szótárban, ezért átküldi a d indexét, vagyis a 4-et, felveszi a szótárba a da sorozatot a 6. helyre, és megy tovább az a-val kezdve. Az a szerepel a szótárban, így hozzáveszi a b-t. ab nincs bent, tehát átküldi a indexét, a 2-t, felveszi ab-t, és folytatja az eljárást b-től, stb. Az 1.11. ábrán látható táblázat tartalmazza a kódolás végeztével a szótárban található indexeket és karaktersorozatokat. A kódoló kimenete a következő:

$$4, 1, 2, 2, 1, 3, 6, 8, 10, 12, 9, 11, 7, 16, 4, 5, 5, 11, 21, 23, 5$$

index	bejegyzés
1	*a*
2	*b*
3	*c*
4	*d*
5	*e*
6	*da*
7	*ab*
8	*bb*
9	*ba*
10	*ac*
11	*cd*
12	*dab*
13	*bba*

index	bejegyzés
14	*acd*
15	*dabb*
16	*bac*
17	*cda*
18	*abb*
19	*bacd*
20	*de*
21	*ee*
22	*ec*
23	*cde*
24	*eec*
25	*cdee*

1.11. ábra. Az 1.12. példa szótára.

- A Unix COMPRESS parancsa és a G I F (Graphics Interchange Format) képtömörítő eljárás is az LZW algoritmust használja,

Matematikai kitérő –

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

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

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

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

Matematikai kitérő –

BIZONYÍTÁS:

a) Tegyük fel, hogy $p(x_k) > p(x_j)$ és $|f(x_k)| > |f(x_j)|$. Ekkor x_j és x_k kódját felcserélve egy új f^* kódot vezethetünk be, amelyre

$$\begin{aligned}\sum_{i=1}^{n} p(x_i)|f(x_i)| - \sum_{i=1}^{n} p(x_i)|f^*(x_i)| = \\ = p(x_k)|f(x_k)| + p(x_j)|f(x_j)| - p(x_k)|f(x_j)| - p(x_j)|f(x_k)| = \\ = p(x_k)(|f(x_k)| - |f(x_j)|) - p(x_j)(|f(x_k)| - |f(x_j)|) = \\ = (p(x_k) - p(x_j))(|f(x_k)| - |f(x_j)|) > 0,\end{aligned}$$

tehát f nem lehet optimális.

b) Az állítás egyszerűen belátható, ha arra gondolunk, hogy $|f(x_{n-1})| < |f(x_n)|$ esetén az $|f(x_n)|$ utolsó bitjét levágva az optimálisnál kisebb átlagos kódszóhosszú kódot kapnánk, ami még mindig prefix tulajdonságú. Valóban, mivel az eredeti kódszó minden más kódszónál legalább eggyel hosszabb volt, a csonkítással kapott új kódszó az eredeti kód prefix tulajdonsága miatt nem azonos semelyik másikkal, és ugyanezen okok miatt nem is folytatása semmilyen más kódszónak.

c) Az előző gondolatmenetből világos, hogy ha létezik olyan $f(x_i)$ kódszó, hogy $f(x_i)$ és $f(x_n)$ csak az utolsó bitben különböznek, akkor az a) és b) miatt $|f(x_i)| = |f(x_{n-1})| = |f(x_n)|$. Így ha $i \neq n-1$, akkor x_i és x_{n-1} kódját felcserélve c) teljesül, és a kód optimális marad. ■

Hivatkozás

http://www.tankonyvtar.hu/en/tartalom/tamop425/0046_informacio_es_kodelmelet/ch04s07.html

Stacionárius és ergodikus fogalom matematikai meghatározása

Az **információforrást** $\mathbb{X}$ betűvel jelöljük, és az $X_1, X_2, \ldots$ valószínűségi változók végtelen sorozatával modellezzük, azaz a forrás az i-edik időpillanatban az X_i jelet bocsátja ki. Az X_i valószínűségi változók mindegyike ugyanabból a véges $\mathcal{X} = \{x_1, \ldots, x_n\}$ halmazból, a forrásábécéből veszi az értékét. Az $\mathbb{X}$ forrást statisztikai tulajdonságaival jellemezzük, vagyis adottnak vesszük, ha minden véges dimenziós eloszlását ismerjük. Egy $\mathbb{X}$ forrást emlékezetnélkülinek vagy memóriamentesnek mondunk, ha az $X_1, X_2, \ldots$ valószínűségi változók függetlenek. Az $\mathbb{X}$ forrás stacionárius, ha $X_1, X_2, \ldots$ stacionárius sztochasztikus folyamat, vagyis ha az $X_1, X_2, \ldots, X_n$ és az $X_{k+1}, X_{k+2}, \ldots, X_{k+n}$ valószínűségi változók együttes eloszlása megegyezik minden pozitív n-re és k-ra, tehát a véges dimenziós eloszlások az „időeltolásra" invariánsak. Az $\mathbb{X}$ stacionárius forrás ergodikus, ha tetszőleges $f(x_1, \ldots, x_k)$ függvényre az $\frac{1}{n}\sum_{i=1}^{n} f(X_i, X_{i+1}, \ldots, X_{i+k-1})$ 1-valószínűséggel konvergál az $\mathbf{E} f(X_1, \ldots, X_k)$ várható értékhez, amennyiben az véges.