PRINCIPI MODERNIH TELEKOMUNIKACIJA

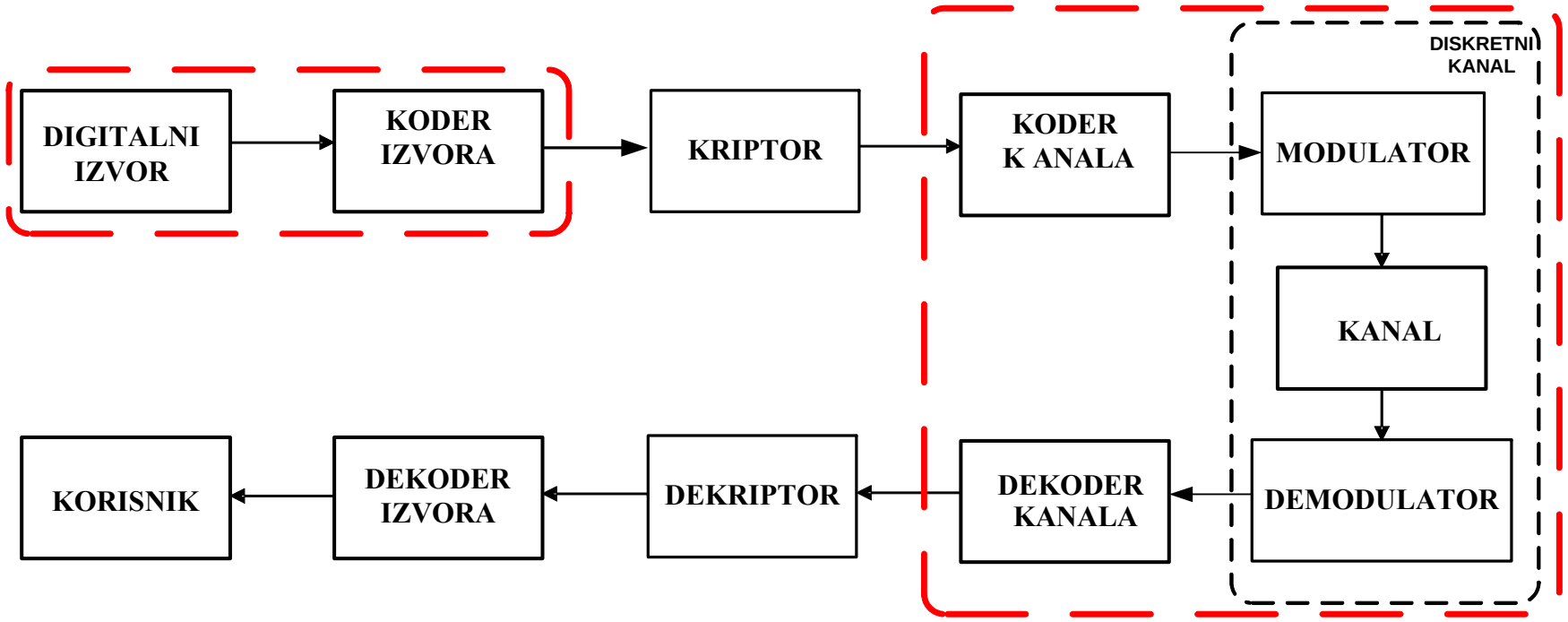
Elektrotehnički fakultet
Katedra za telekomunikacije
Beograd, 2019/2020.

-II-
Statistički kodovi (kompresija)

Predrag Ivaniš, Elektrotehnički fakultet
Katedra za Telekomunikacije, Beograd

Blok šema sistema sa stanovišta teorije informacija

* Smatra se da izvor emituje nekakve simbole –> *q*-nivoski digitalni signal može se opisati sa *q* mogućih amplituda.
* Ciljevi sistema – *efikasan*, *siguran* i *pouzdan* prenos podataka.



Koder izvora, šifrator, zaštitni koder

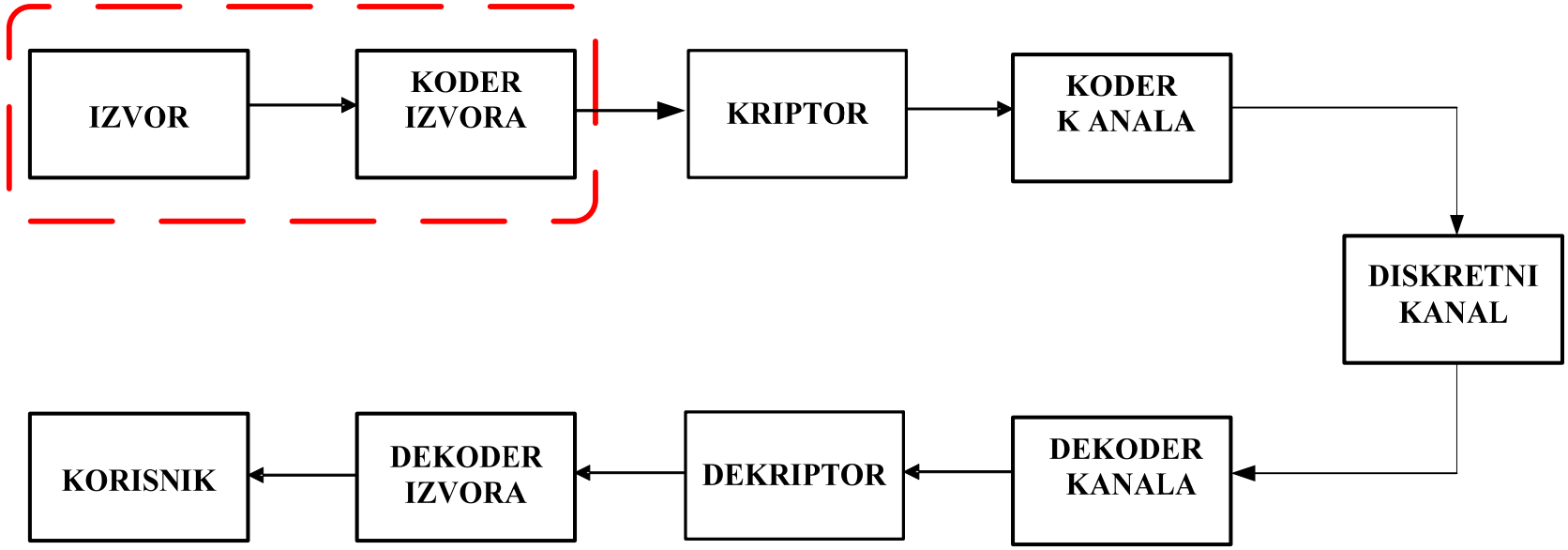
* ***Koder izvora*** **ovako digitalizovanu poruku pretvara u binarni oblik i ispuni neke dodatne zahteve:**
  - Cilj je svaku poruku predstaviti *minimalnim brojem binarnih simbola* a da informacija bude preneta. Koliko *informacija* zaista emituje izvor?
  - Dekoder izvora u idealnom slučaju obavlja inverznu funkciju.

* **Ovako dobijeni binarni niz se u sledećem bloku (*šifrator*) šifruje, što ima za cilj očuvanje tajnosti pri prenosu podataka.**

* ***Zaštitni koder*** **– ima cilj da što je moguće više smanji verovatnoću greške pri prenosu pojedinih simbola. Na ulazu i izlazu kodera pojavljuju se binarni simboli, dok transformaciju bitova u signale vrši modulator.**
  - Nakon zaštitnog kodera biti poruke su “oklopljeni” zaštitnim bitovima.

Blok šema telekomunikacionog sistema

* **Blok šema sa stanovišta teorije informacija**
  - **Zanima nas prenos informacija kroz telekomunikacioni kanal**
  - **Koliko prenos može biti efikasan, siguran i pouzdan?**
* **Posmatra se prenos na nivou bitova**
  - **U diskretni kanal ulaze nule i jedinice**
  - **Iz izvora izlaze nule i jedinice**



Diskretni izvor bez memorije

* **Opisuju se:**
  - Skupom mogućih poruka

$$S=\{s_1, s_2, ..., s_N\}$$

  - Verovatnoćama pojavljivanja pojedinih simbola iz ovog skupa

$$P(s_i),\ i=1,...N.$$

* **Primer:**
  - Izvor emituje šest simbola – **A, B, C, D, E, F**
  - Verovatnoće pojavljivanja

P(A)=0.5, P(B)=0.2, P(C)=0.1, P(D)=0.1, P(E)=0.07, P(F)=0.03

  - Primer sekvence

ADAABABCAEBAAACCBAFADABADABAEA

Efikasan prenos

* Neka sekvencu koju emituje diskretni izvor želimo da predstavimo u binarnom obliku
* ASCII kod – svaki simbol se predstavlja sa 7 bita
* Prethodni primer
  - Za prenos 30 slova iz prikazane sekvence potrebno je 30*7=210 binarnih simbola.
  - Koliko god ima slova potrebno je sedam puta više bita za njihov prenos.
* Da li se poruka može predstaviti manjim brojem binarnih simbola, a da se ne naruši informacija koja se prenosi?
  - Želimo da pravilno rekonstruišemo svih 30 slova na strani prijema.

ASCII tabela – za štampani tekst

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Predrag Ivaniš, Elektrotehnički fakultet
Katedra za Telekomunikacije, Beograd

Količina informacija

* **Informacija može imati više značenja**
  - sintaktički nivo – poruka nosi informacije ako na strani prijema postoji neizvesnost o tome koja će poruka biti primljena.
  - semantički nivo – zahteva se da korisnik razume značenje poruke (da je shvati)
  - pragmatički nivo – razmatra se vrednost informacija (korist koju izvlači korisnik)
* **Najjednostavniji način – pomoću logaritma**

$$Q(s_i)=\log(1/P(s_i))$$

* **Funkcija mora da zadovolji sledeće**
  - Količina informacija ne može biti negativna.
  - Ako je verovatnoća pojave simbola ravna jedinici, događaj je siguran i simbol ne nosi nikakvu informaciju prijemniku log(1)=0;
  - Ako su simboli nezavisni, količine informacija koju oni nose se sabiraju

$$Q(s_i s_k)=\log(1/P(s_i, s_k))= \log(1/[P(s_i)P(s_k)])= Q(s_i)+Q(s_k)$$

* **Ako je baza logaritma 2 jedinica je Šenon (*Claude Shannon*).**
* **Prethodni primer:**

$$Q(\mathrm{A})=\log_2(1/0.5)=\mathrm{ld}(2)=1[\mathrm{Sh}],\ Q(\mathrm{F})=\mathrm{ld}(1/0.03)=5.06[\mathrm{Sh}],\ \ldots$$

Entropija

*** Entropija predstavlja prosečnu "meru neizvesnosti (neopredeljenosti)" posmatrača o tome šta će izvor da emituje.**

- Emitovanjem pojedinih simbola izvor emituje u proseku tačno potrebnu količinu informacija i upravo potpuno razrešava ovu neizvesnost.

$$H(S) = \overline{Q(s_i)} = \sum_{i=1}^{q} P(s_i)Q(s_i) = \sum_{i=1}^{q} P(s_i)\mathrm{ld}\left(\frac{1}{P(s_i)}\right) = -\sum_{i=1}^{q} P(s_i)\,\mathrm{ld}\,P(s_i) \quad \left[\frac{\mathrm{Sh}}{\mathrm{simb}}\right].$$

* Primer

H(S)=P(A)Q(A)+P(B)Q(B)+P(C)Q(C)+P(D)Q(D)+P(E)Q(E)+P(F)Q(F)
=0.5*1[Sh]+0.2*2.32[Sh]+...+0.03*5.06[Sh]=2.05 [Sh/simb]

Claude Shannon

* Šenon je definisao količinu informacija u poruci, a srednja količina informacija po poruci je entropija. Ona predstavlja meru neizvesnosti o poruci koju će izvor emitovati.
* Imajući u vidu statističko kodovanje Šenon je pokazao vezu entropije i minimalnog broja signala po poruci potrebnog za predstavljanje informacija koje emituje izvor.
* Time se postavljaju granice do kojih je moguće obaviti kompresiju.

Claude Shannon - Father of the Information Age
https://www.youtube.com/watch?v=z2Whj_nL-x8

The Man Who Turned Paper Into Pixels
https://www.youtube.com/watch?v=Q8rVJZ-VDKQ

Tech Icons: Claude Shannon
https://www.youtube.com/watch?v=z7bVw7lMtUg

Ultimate Useless Machine
https://www.youtube.com/watch?v=vIlAmkmHX60

Hafmenov postupak

- Hafmenov kod koji odgovara izvoru koji:

- Emituje šest simbola
- Verovatnoće zadate u drugoj koloni tabele

- Postupak

- Poređati po opadajućim verovatnoćama
- Sažimati po dva simbola i dati im isti prefiks

s_i	$P(s_i)$	x_i	s_i	$P(s_i)$	x_i	s_i	$P(s_i)$	x_i	s_i	$P(s_i)$	x_i	s_i	$P(s_i)$	x_i
s_1	0.5	0	s_1	0.5	0	s_1	0.5	0	s_1	0.5	0	s_1	0.5	0
s_2	0.2	11	s_2	0.2	11	s_2	0.2	11	$s_3s_4s_5$ s_6	0.3*	10	s_2 $s_3s_4s_5$ s_6	0.5*	1
s_3	0.1	101	s_3	0.1	101	s_4s_5 s_6	**0.2***	100	s_2	0.2	11			
s_4	0.1	1000	s_4	**0.1**	1000	s_3	**0.1**	101						
s_5	**0.07**	10010	s_5 s_6	**0.1***	1001									
s_6	**0.03**	10011												

Srednja dužina kodne reči, efikasnost koda

- Srednja dužina kodne reči:

$$L_{sr} = 0.5*1 + 0.2*2 + 0.1*3 + 0.1*4 + 0.07*5 + 0.03*5 = 2.1 \ [b/s]$$

- Entropija izvora

$$H(s) = \sum_{i=1}^{6} P(s_i) ld \frac{1}{P(s_i)} = 2.0502 \ [Sh/simb]$$

-Efikasnost

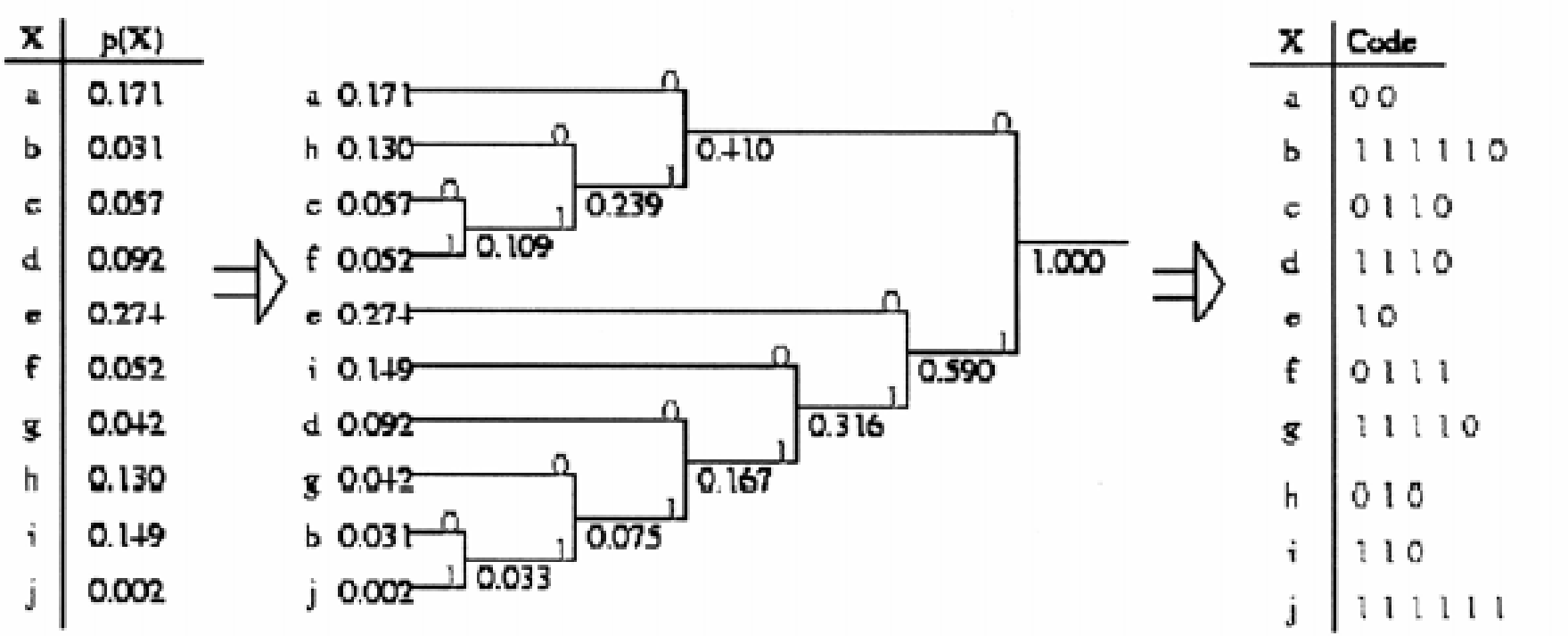
$$\eta = \frac{H(s)}{Lsr} \cdot 100\% = 97.63\%$$

- Stepen kompresije

$$\rho = \frac{\lceil ld(q) \rceil}{Lsr} = \frac{3}{2.1} = 1.4285$$

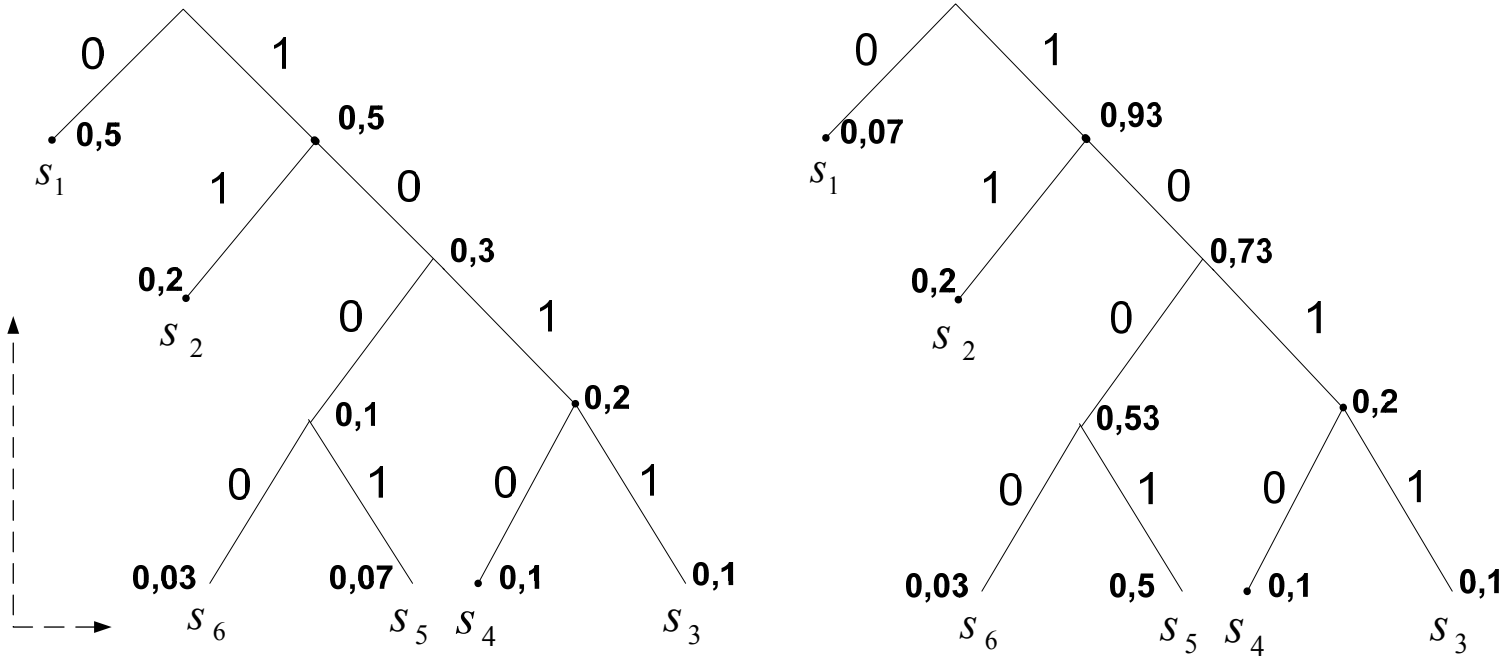
Predstava pomoću stabla, drugi primer

- **Deset simbola izvorne liste, verovatnoće su im bitno različite!**
- **Dužina kodne reči bitno zavisi od verovatnoće pojavljivanja simbola kome je reč pridružena.**



Sibling property

* **Ako je binarni prefiks Hafmenov kod, kodno stablo ima osobinu "izdanaka“ (sibling property).**
  - Kodno stablo ima ovu osobinu ako je svaki čvor (osim korena) povezan sa čvorom porekla, ili “roditeljskim” čvorom, koje se nalazi jedan hijerarhijski nivo iznad njega. Pritom je potrebno da se svi čvorovi odozdo nagore i sleva nadesno mogu sortirati po neopadajućim (ili u obrnutom poretku po nerastućim) verovatnoćama.
  - Primer jednog stabla (prikazano levo) koje ima osobinu izdanaka i jednog kodnog stabla (prikazano desno) koje nema osobinu izdanaka.



Robert Fano

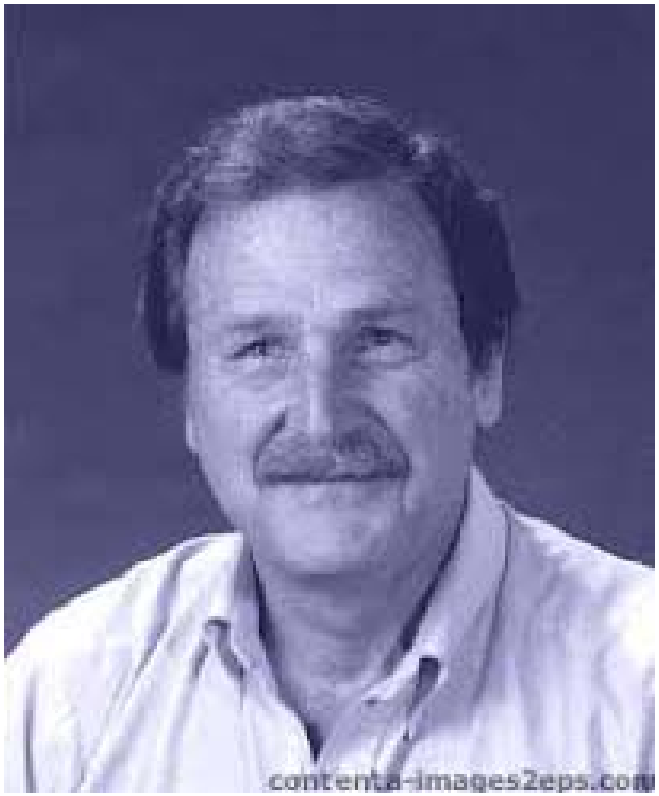
* Predstavu koda pomoću stabla je predložio Fano (*Robert Fano*), koji je u vreme nastanka Teorije informacija sarađivao sa Šenonom i već 1952. godine držao kurs (*Information Theory Course*) iz ove naučne discipline na doktorskim studijama na MIT (*Massachusetts Institute of Technology*).
* Šenon i Fano su predložili prvi praktičan algoritam statističkog kodovanja (Šenon-Fanoov postupak) i on je bio skoro optimalan.

Robert Fano – interview
https://www.youtube.com/watch?v=Rfdbk663L0I

Project MAC (1963), saradnici Marvin Minsky, John McCarthy (smislio programski jezik Lisp) **:**
- operating systems
- artificial intelligence
- theory of computation

David Huffman

* Hafmenov algoritam nastao je kao rezultat seminarskog rada Dejvida Hafmena koji je urađen u okviru ispita koji je polagao na pomenutom predmetu na MIT koji je polagao kod profesora Fanoa.
* Ovaj studentski rad je 1952. godine objavljen u formi naučnog rada.



Proširenje izvora, entropija proširenja

* **Ako se umesto pojednih simbola posmatraju sekvence od po 2, 3 ili više (*n*) sukcesivnih simbola, tada se kaže da se posmatra drugo, treće ili *n-to proširenje izvora*.**
  - Ono se obično obeležava sa S^n a broj njegovih simbola je upravo q^n.
  - Drugim rečima, *n*-to proširenje izvora je izvor čiji su simboli sekvence od po *n* simbola prvobitnog izvora.
* **Primer**
  - Originalni izvor emituje poruke iz skupa S={A, B, C} sa verovatnoćama
    P(A)=1/2, P(B)=1/4, P(C)=1/4;
  - Proširenje izvora "emituje" složene simbole iz sledećeg skupa:
    S^2={AA, AB, AC, BA, BB, BC, CA, CB, CC}.
  - Računa se na osnovu verovatnoća složenih simbola $\sigma_i=s_i s_k$, za izvore bez memorije važi $P(s_i,s_k)=P(s_i)P(s_k)$.
    H(S)=1/2*1+1/4*2+1/4*2=1.5 [Sh/simb]
    P(AA)=0.5*0.5=0.25,..., P(CC)=0.25*0.25=0.0625
    $H(S^2)$=0.25*ld(4)+...+0.0625*ld(16)=3 [Sh/simb]= 2H(S).

Prva Šenonova teorema

* **Ako u tekstu postoji suvišnost, ona se može otkloniti kompresijom.**
  - Koliki stepen kompresije se maksimalno može postići?
  - Kolika je minimalna dužina kodne reči a da se sačuva kompletna informacija (da kod bude nedestruktivan)?

* **Prva Šenonova teorema – dovoljnim proširivanjem reda izvora i njegovim kodiranjem može se postići proizvoljno visoka efikasnost:**

$$\lim_{n\to\infty} \frac{L_{sr,n}}{nH(s)} = 1$$

- Ovaj izraz važi i za izvore s memorijom!
- Kompresija može najviše ići do nivoa gde se svaki simbol u proseku predstavlja sa onoliko bita koliko iznosi entropija izvora.

Hafmenov kod primenjen na proširenje izvora

* **Posmatra se izvor koji emituje dva simbola sa sledećim verovatnoćama:**

s_i	s_1	s_2
$P(s_i)$	0.7	0.3

* **Potrebno je izvršiti binarno statističko kodovanje (po Hafmenovom postupku) izvora informacija, njegovog drugog i trećeg proširenja. Odrediti efikasnost svakog od postupaka i uporediti rezultate sa postavkom Prve Šenonove teoreme.**
  - Postupak statističkog kodovanja elemenata liste originalnog izvora je trivijalan:

s_i	$P(s_i)$	x_i
s_1	0.7	0
s_2	0.3	1

  - Entropija originalnog izvora, srednja dužina kodne reči i efikasnost

$$H(s) = 0.7ld\frac{1}{0.7} + 0.3ld\frac{1}{0.3} = 0.8813\ Sh/simb$$

$$L_{sr} = 0.7*1 + 0.3*1 = 1\ bit/simb$$

$$\eta = \frac{0.8813}{1}\cdot 100\% = 88.13\%$$

Hafmenov kod primenjen na proširenje izvora

* **Ako se umesto pojednih simbola posmatraju sekvence od po 2, 3 ili više (*n*) sukcesivnih simbola, tada se kaže da se posmatra drugo, treće ili *n-to proširenje izvora*.**
  - Ono se obično obeležava sa S^n a broj njegovih simbola je upravo q^n.
  - Drugim rečima, *n*-to proširenje izvora je izvor čiji su simboli sekvence od po *n* simbola prvobitnog izvora.
* **Postupak statističkog kodovanja elemenata liste II proširenja izvora:**

s_i	$P(s_i)$	x_i	s_i	$P(s_i)$	x_i	s_i	$P(s_i)$	x_i
$\sigma_1 = s_1 s_1$	0.49	1	σ_1	0.49	1	$\sigma_2\sigma_3\sigma_4$	0.51*	0
$\sigma_2 = s_1 s_2$	0.21	01	$\sigma_3\sigma_4$	**0.30***	00	σ_1	0.49	1
$\sigma_3 = s_2 s_1$	**0.21**	000	σ_2	**0.21**	01			
$\sigma_4 = s_2 s_2$	**0.09**	101						

  - Entropija II proširenja, srednja dužina kodne reči i efikasnost

$$H^2(s) = 2H(s) = 1.7626\ Sh/simb$$

$$L_{sr} = 0.49*1 + 0.21*2 + 0.21*3 + 0.09*3 = 1.81\ bit/simb$$

$$\eta = \frac{1.7626}{1.81} \cdot 100\% = 97.38\%$$

Prva Šenonova teorema – primer 2

* **Posmatrajmo binarni izvor bez memorije sa verovatnoćama pojavljivanja simbola P(0)=0.99 i P(1)=0.01.**
  - Originalni izvor ima entropiju H(S)=0.0808. On se može kodovati trivijalnim Hafmenovim kodom 0->0, 1->1, čime se jedan simbol izvorne liste predstavlja sa jednim bitom, pa je L_{sr}=1. Efikasnost je 8.08% a stepen kompresije 0%.
  - Drugo proširenje sastoji se od četiri simbola sa verovatnoćama P(00)=0.9801, P(01)=0.0099, P(10)=0.0099 i P(11)=10^{-4} i entropijom H(S)=0.1616. Srednja dužina kodne reči je L_{sr}=1.0299. Efikasnost je 15.7%.
  - Sa povećanjem reda proširenja efikasnost teži maksimalnoj vrednosti i postignuti stepen kompresije znatno raste.
  - U slučaju petog proširenja biće P(00000)=$(0.99)^5$=0.951 i ovom simbolu će biti dodeljena kodna reč '0', dok će preostale petobitne kombinacije iz izvorne liste imati znatno manje verovatnoće. To tnači da će pet nula iz izvorne liste biti zamenjene jednim binarnim simbolom iz kodne liste a da se ne izgubi informacija koja se prenosi!
  - Pri dovoljno velikom proširenju **100 simbola iz izvorne liste mogu se zameniti sa nešto više od osam simbola kodne liste** (i jedno i drugo su biti). Ova tvrdnja nije ništa drugo nego nešto drugačije interpretirana I Šenonova teorema!

Izvori sa memorijom *m*-tog reda

* **Verovatnoća emitovanja trenutnog simbola zavisi od verovatnoća prethodnih *m* emitovanih simbola:**
  - **Štampani tekst:**

    TELEKOMUNIKACIJE
  - **Praktično svi izvori slučajnog signala u prirodi.**

* **Entropija izvora s memorijom *m*-tog reda:**

$$H_m(\mathrm{S}) = \sum_{j=1}^{q}\sum_{i_1=1}^{q}\sum_{i_2=1}^{q}\cdots\sum_{i_m=1}^{q} P(s_{i_1}, s_{i_2}, \ldots, s_{i_m}, s_j)\,\mathrm{ld}\left(\frac{1}{P(s_j / s_{i_1}, s_{i_2}, \ldots, s_{i_m})}\right) =$$

$$= \sum_{S^{m+1}} P(s_{i_1}, s_{i_2}, \ldots, s_{i_m}, s_j)\,\mathrm{ld}\left(\frac{1}{P(s_j / s_{i_1}, s_{i_2}, \ldots, s_{i_m})}\right).$$

* **Entropija izvora s memorijom prvog reda:**

$$H_1(\mathrm{S}) = \sum_{j=1}^{q}\sum_{i=1}^{q} P(s_i, s_j)\,\mathrm{ld}\left(\frac{1}{P(s_j / s_i)}\right)$$

Primer izvora sa memorijom

* **Govor, odnosno štampani tekst je jedan od najvažnijih primera diskretnog niza s memorijom.**
* **Entropije za neke jezike date su u tabeli:**
  - **H_{max} – kad bi sva slova bila jednako verovatna**
  - **H_0 – bez memorije, poznate verovatnoće pojavljivanja slova**
  - **H_1 – verovatnoća pojave svakog slova zavisi samo od prethodnog**

Jezik (tekst)	Razmak	H_{max}	H_0	H_1	H_2
srpski	da	4,95	4,24	3,41	-
hrvatski	da	4,76	4,19	3,59	3,10
ruski	da	5,00	4,05	3,52	3,01
engleski	ne	4,70	4,14	3,56	3,30
engleski	da	4,76	4,03	3,32	3,10
francuski	da	4,76	3,95	3,17	2,83
nemački	da	4,76	4,04	3,42	2,82

Suvišnost (redundansa)

* **Suvišnost (redundansa)**

$$R = \frac{H_{\max} - H}{H_{\max}} \cdot 100 = \left(1 - \frac{H}{H_{\max}}\right) 100 [\%],$$

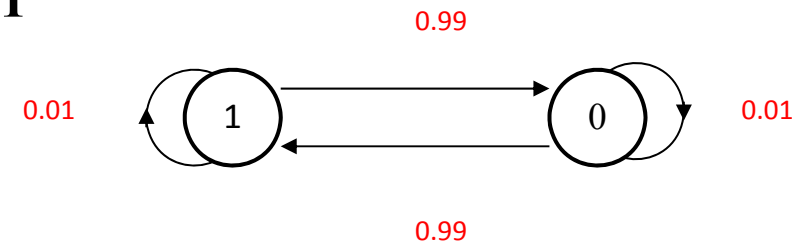
* **Procenjena entropija i suvišnost za neke jezike:**
  - **Veliki deo konstrukcija u svakom jeziku je predvidiv;**
  - **Nije neophodno preneti baš svako slovo da bi razumeli poruku:**

    T L K M N K C J E
  - **Suvišnost obično iznosi preko 70%**

Jezik	Entropija [Sh/simb]	Suvišnost [%]
engleski	1,30	72,7
ruski	1,37	72,6
francuski	1,40	70,6

Hafmenov algoritam za izvore s memorijom

* **Nema imamo izvor sa memorijom prvog reda, pri čemu su simboli jednakoverovatni a tranzicione verovatnoće su *P(0/1)=P(1/0)*=0.99 i *P(0/0)=P(1/1)*=0.01**



$$H(S) = 0.99 \times \mathrm{ld}\frac{1}{0.99} + 0,01 \times \mathrm{ld}\frac{1}{0.01} = 0.0808\ [Sh/simb]$$

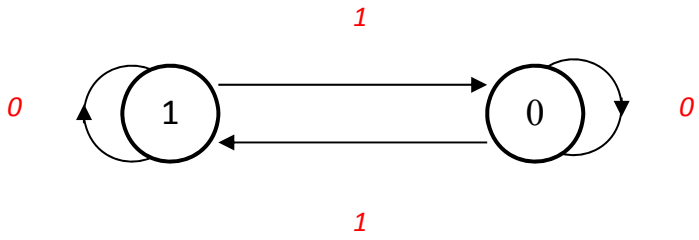
* **Primena Hafmenovog algoritma:**
  - Na originalni izvor -> Lsr=1 [bit/simb], efikasnost je 8.08%. Iako su simboli jednakoverovatni pa je entropija pridruženog izvora 1 [Sh/simb], efikasnost je mala zbog prisustva memorije.
  - Na II proširenje -> P(00)=0.5*0.01=0.005, P(01)=0.5*0.99=0.495, P(10)=0.5*0.99=0.495, P(11)=0.5*0.01=0.005 pa je entropija pridruženog izvora 1.0808 [Sh/simb] dok je sama entropija originalnog izvora H(S)=0.1616 [Sh/simb]. Sada je Lsr=1.488 [bit/simb], pa je efikasnost nešto veća i iznosi 10.86%.
  - **Iako je Hafmenov algoritam dizajniran za izvore bez memorije, proširenjem izvora implicitno se uzima u obzir prisustvo memorije!**

Deterministička binarna sekvenca

* **Odrediti entropiju sekvence**

‘ 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 ’

* **Očigledno je u pitanju izvor prvog reda pa je dijagram stanja i izraz za entropiju isti kao u prethodnom primeru**



$$H(S) = 0{,}5 * 0 * \mathrm{ld}\frac{1}{0} + 0{,}5 * 1 * \mathrm{ld}\frac{1}{1} + 0{,}5 * 1 * \mathrm{ld}\frac{1}{1} + 0{,}5 * 0 * \mathrm{ld}\frac{1}{0} = 0\,[Sh/simb]$$

* **Pošto je entropija jednaka nuli, jasno je da ovaj izvor ne emituje informacije!**
  - U pitanju je deterministički signal, on ne nosi informaciju!

Kodovi koji uzimaju u obzir memoriju izvora

* **Hafmenov algoritam ima nekoliko nedostataka:**
  - osetljivost na gubljenje sinhronizacije
  - potreba za poznavanjem statistike sekvence koja se prenosi
  - najozbiljniji nedostatak Hafmenovog postupka je što se izvor tretira kao izvor bez memorije.
* **Tačnije rečeno, ako je u pitanju izvor s memorijom, ne koriste se statističke zavisnosti koje postoje u sekvenci izvornih simbola.**
* **Efikasnost se može povećati proširivanjem izvora, ali se može desiti da se tek pri proširenju višeg reda dobija kod koji ima željenu efikasnost.**
  - Pri takvom proširenju broj kodnih reči može da bude veoma veliki.
  - Uzimanje u obzir statističke zavisnosti u sekvenci može se postići ako se verovatnoće simbola proširenog izvora ne računaju na osnovu prostih verovatnoća pojavljivanja (kao da je izvor bez memorije), već se dobijaju statističkom obradom poruke koju treba kodovati.
  - Da bi ove verovatnoće bile pouzdane za proširenja višeg reda potrebna je poruka velike dužine.

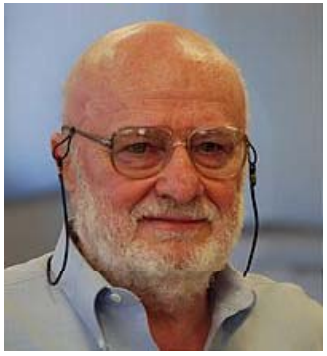
Univerzalni kod, kompleksnost

* **Ovo dovodi do ideje formiranja pojma *univerzalnog koda*. To je kod koji vrši kompresiju prenošene sekvence bez apriornog poznavanja njenih statističkih osobina (termin je uveo Andrej Nikolajevič Kolmogorov).**
  - Sekvenca može da bude i nestacionarna. U koderu se formira, na osnovu poznatog dela poruke, koji je već ušao u koder, model na osnovu koga se vrši kompresija.
  - U principu ovaj model je adaptivan, tj. stalno prati statističke osobine poruke koja se koduje. U slučaju izvora bez memorije može se približiti po performansama kompaktnom kodu, a u ostalim slučajevima vrši se adaptacija na tekuću statistiku sekvence.

* **S gledišta Teorije informacija može se smatrati da je kompleksnost neke sekvence simbola ravna potrebnom broju bita da se ona opiše. Ovako definisana kompleksnost bi bila jednaka količini informacija koju nosi ta sekvenca.**
  - Kolmogorov je uveo *računsku kompleksnost (comutational complexity)* za neku sekvencu i ona je ravna minimalnoj dužini programa, na nekom osnovnom programskom jeziku, nezavisno od korišćenog računara, kojom se može generisati (odštampati) ta sekvenca.

Kompleksnost, kodovi sa rečnikom

* **Ako je niz potpuno slučajan, tada dužina sekvence odgovara i dužini najkaćeg opisa sekvence.**
  * U slučaju stacionarnog diskretnog procesa bez memorije pokazuje se da srednja vrednost algoritamske kompleksnosti teži entropiji kada dužina sekvence neograničeno raste.
* **Lempel-Zivov algoritam se može shvatiti kao pokušaj da se napiše program kojim će dekoder (na prijemu) generisati sekvencu komprimovanu od strane kodera na predaji**
  * Lempel-Zivov (LZ) postupak kodovanja je jedna vrsta univerzalnog kodovanja u okviru koje ne postoji eksplicitni model. Osnove LZ kodovanja date su u radovima Abrahama Lempela i Jakoba Ziva 1977. godine i 1978. godine (LZ77, LZ78, LZW)



Lempel Zivov (LZ) algoritam

* **Dve faze:**
  - Prvo se formira rečnik na osnovu dela sekvence koju emituje izvor;
  - Kada je rečnik jednom formiran, on se koristi za kompresiju ostalog dela sekvence koju emituje izvor.

* **Obično se koristi za kompresiju teksta**
  * Na prvih 256 pozicija slova, brojevi i specijalni znaci (prošireni ASCII).
  * Poznavanje ovog (manjeg, standardnog i nezavisnog od statistike prenošene sekvence!) dela rečnika je potreban i dovoljan uslov da se rekonstruiše rečnik i izvrši dekompresija samo na osnovu presretnute sekvence!
  * Ukupna veličina rečnika je obično 2048 ili 4096 adresa, pa se svaki složeni simbol upisan u rečnik predstavlja kombinacijom od 11 ili 12 bita.

LZ, kompresija

abbaabbaaba bb**abb**abb ->0110242 366 (tj. 000 001 001 000 010 100 010 011 110 110)

```
set w = NIL
loop
        read a character K
        if wK exists in the dictionary
                w =wK
        else
                output the code for w
                add wK to the string table
                w=K
        endif
end loop
```

rečnik						
adresa	**sadržaj**	w	k	wk	?	out
0	a					
1	b	nil	a	a	+	
		a	b	ab	-	0
2	ab	b	b	bb	-	1
3	bb	b	a	ba	-	1
4	ba	a	a	aa	-	0
5	aa	a	b	ab	+	
		ab	b	abb	-	2
6	abb	b	a	ba	+	
		ba	a	baa	-	4
7	baa	a	b	ab	+	
		ab	a	aba	-	2

LZ vs. Hafmen

* **Neka je sekvenca koju treba komprimovati**

  abababababababababa…

* **ukupno:**
  - **dve podsekvence dužine 2 (ab,ba)**
  - **dve podsekvence dužine 3 (aba,bab)**
  - **dve podsekvence dužine 4 (abab,baba)**

rečnik	
adresa	**sadržaj**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab
7	baba
8	ababa
...	...

* Ako rečnik ima 16 adresa (sa 4 bita)
  -> na adr 14. i 15. će biti sekvence dužine 8
* Ako rečnik ima 4096 adresa (sa 12 bita)
  -> na adr 4094. i 4095. će biti sekvence dužine 2048!

* U realnosti rečnik ima ukupno 4096 pozicija, prvih 256 pozicija osnovni simboli (0-255) a na pozicijama (256-4095) izvedeni simboli.
* Naravno, statistička zavisnost je znatno manja nego u navedenom primeru ali je sasvim dovoljna da za štampani tekst radi bolje nego Hafmen.

Prednosti LZ algoritma

* **LZ postupci kompresije primenjeni na engleski tekst postižu kompresiju od oko 55%, tj. toliko smanjuju broj potrebnih bita, dok primena Hafmenovog postupka na istom uzorku daje kompresiju od oko 43%.**
  - Objašnjenje je jednostavno. LZ algoritmi uzimaju u obzir statističku zavisnost, dok Hafmenov postupak tretira tekst kao statistički nezavisan niz.
* **Osim ove osobine, koja je obično presudna za izbor statističkog koda, Lempel-Zivov algoritam ima bar još dve velike prednosti u odnosu na Hafmenov algoritam:**
  - Njegova programerska realizacija je znatno jednostavnija (kako u koderu, tako i u dekoderu). Da bi se u ovo uverio, čitaocu se predlaže da napiše program koji vrši Hafmenovo kodovanje izvora koji emituje proizvoljan broj simbola sa zadatim verovatnoćama pojavljivanja.
  - Kod Lempel-Zivovog postupka ne postoji problem sinhronizacije – svaki segment smešten na određenu memorijsku lokaciju predstavlja se kodnom reči fiksne dužine – $\log_2(M)$, pri čemu M označava veličinu rečnika.

Primena LZ i Hafmenovog algoritma

* LZ77 u kombinaciji sa Hafmenovim algoritmom čini osnovu praktično svih komercijalno dostupnih alata za kompresiju teksta (ZIP, WINZIP, ARJ, LHA, ...). LZW algoritam se koristi pri kompresiji slike i čini osnovu GIF formata.
* U novije vreme se pojavljuju algoritmi koji na optimalan način kombinuju LZ77 i Hafmenov algoritam – npr. DEFLATE algoritam, koga je smislio *Phil Katz* s namerom da unapredi ZIP postupak, standardizovan je 1996. godine a danas se koristi u GZIP postupku kompresije teksta i PNG formatu zapisa statične slike.
* Primenom ovakvih algoritama za neke vrste podataka praktično je dostignut maksimalno mogući stepen kompresije (dostignuta je granica određena entropijom).
* Moguć je i veći stepen kompresije statične i pokretne slike ili audio zapisa, ali su u pitanju destruktivne metode kojima se žrtvuje kvalitet nauštrb smanjenja veličine odgovarajućih fajlova – najpoznatiji ovakvi algoritmi su razne varijante JPEG i MPEG standarda.

Primene algoritama za kompresiju

PRIMENE:
- Nedestruktivna kompresija pisanog teksta ili slike obično se zasniva na LZ kodovima (LZ77, LZW) ili kombinaciji LZ/Hafmen.

Utility	*Format*	*Compression*
pkarc (DOS) arc (Unix, Mac, etc.)	.arc, .ark	LZW
arj (DOS)	.arj	LZ77 + hashing, secondary static Huffman
Compuserve GIF	.gif	LZW
gzip	.gz	LZ77 + hashing, secondary static Huffman
lha, lharc	.lha, .lhz	LZ77 + tries, secondary static Huffman
squeeze (DOS)	.sqz	LZ77 + hashing
pkzip (DOS) zip (Unix) WinZip (Windows)	.zip	LZ77 + hashing, secondary static Huffman
zoo (DOS/Mac/Unix)	.zoo	LHA
freeze (Unix)	.F	LZ77 + hashing, secondary adaptive Huffman
yabba (Unix)	.Y	LZ78 variant
compress (Unix)	.Z	LZW

JPEG:
1. do irreversible compression on colour channels
2. compute the *Discrete Cosine Transform* for
3. "reduce" the DCT output: more reduction
4. Huffman encode the reduced output

MPEG:
1. do irreversible compression on colour channels (not on shade channel)
2. for each block of 16x16 in a frame, try to find a "similar" block in a previous (*or future*)
3. store the differences between blocks instead of storing entire blocks
4. Huffman encode the whole thing

Run Length i diferencijalno kodovanje

* Jednostavan algoritam kompresije koji se koristi kada u nizu podataka koji emituje izvor postoji veliki broj simbola koji se uzastopno ponavljaju.
* Ovaj algoritam je prvi put primenjen za kompresuju radarske slike koja je generisana na meteorološkim stanicama, zatim se koristio pri prenosu televizijske slike [26], a od strane provajdera *Compuserve* je bio predložen u jednom od prvih formata za kompresiju slike.
* Zasniva se na principu da se dugi niz bita šalje kao jedan podatak, a osnovna prednost algoritma je izuzetna jednostavnost. Kodovanje je brzo, dekodovanje je izuzetno jednostavno, a kompresija se lako implementira (softverski ili hardverski). Ilustracije radi, sledećem nizu m koji emituje izvor odgovara sekvenca c na izlazu kodera

m = 000000000001111111111111111110000000111111111111111100000000

c = (11,0) (18,1) (7,0) (20,1) (8,0)

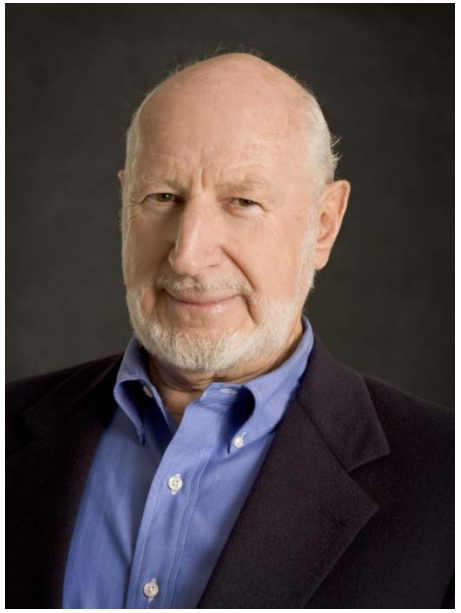
* Često se koristi i diferencijalni kod, kod koga se koduje razlika dva simbola koja emituje izvor $c_i = m_i - m_{i-1}$, čime se dobija

m =	17	17	18	19	19	20	20	20	19	19	18	17	16	16	16	18	17	17
c =	17	0	1	1	0	1	0	0	-1	0	-1	-1	-1	0	0	2	-1	0

dok se pri dekodovanju primenjuje pravilo $m_i = c_i + c_{i-1}$ za $i>1$, odnosno $m_1 = c_1$.

Aritmetički kod

* *Peter Elias* (profesor MIT, tvorac konvolucionih kodova) je početkom šezdesetih godina XX veka predložio da se blokovi ulaznih simbola koduju združeno stvarajući jedinstvenu sekvencu kodovanih simbola, potencijalno kraću od sekvence koju proizvodi simbolski kod.
* Pomenuti metod je prvi put dokumentovan u knjizi koju je pisao *Norman Abramson* (profesor na Stenfordu, smislio ALOHAnet) i kasnije unapređen u radovima gde se prvi put naziva *aritmetičkim kodom*. Naziv potiče od činjenice da se veći blok simbola, recimo 1 MB podataka, koduje jedinstvenim decimalnim brojem iz opsega [0,1).



Aritmetički kod

* Aritmetički kod je adaptacija igre pogađanja (*guessing game*) u kojoj se od učesnika traži da pogađa slovo po slovo unapred zadate rečenice, pritom zapisujući broj pokušaja koji je potreban da bi se slovo pogodilo.
* Ovo je ilustrovano sledećom tabelom, gde su u gornjoj vrsti napisana slova (prema redosledu njihovog pojavljivanja), a u donjoj broj pokušaja koji je bio potreban da se odgovarajuće slovo pogodi.

N	e	-	p	u	c	a	j	-	t	o	p	o	m	-	k	o	m	a	r	c	a
1	2	1	12	3	4	1	1	1	9	3	6	1	1	1	16	2	15	1	1	1	1

* Zbog semantičkih pravila jezika za pogađanje pojedinih slova potreban je različit broj pokušaja. Na primer, iako se slova “e” i “k” pojavljuju isti broj puta u posmatranoj rečenici (po jednom), prosečan igrač će znatno teže pogoditi slovo “k” kojim započinje nova reč.
* Ako bi se posmatrana slova kodovala brojem pokušaja potrebnim za njihovo pogađanje, izlazni skup bi zadržao isti ukupan broj simbola 31 (30 slova i znak za razmak), jer je to ujedno i maksimalan broj pokušaja pogađanja karaktera. S druge strane, potencijal za kompresiju “sekvence pokušaja” bi se povećao, jer bi dominirale vrednosti koje odgovaraju malom broju pokušaja.

Berouz-Viler transformacija, Bzip2

* *Berouz-Viler transformaciju* naziva se po prezimenima autora (*Michael Burrows*, *David Wheeler*) i predložena je 1983. godine (nije objavljena) od strane Vilera a zaokruženo rešenje je publikovano 1994. godine..
* Ova transformacija ne predstavlja kompresioni algoritam, već metod kojim se ulazni podaci rearanžiraju, tako da se poveća njihov kompresioni potencijal.

$$X=\begin{bmatrix} A & N & A & N & A & S \\ N & A & N & A & S & A \\ A & N & A & S & A & N \\ N & A & S & A & N & A \\ A & S & A & N & A & N \\ S & A & N & A & N & A \end{bmatrix} \Rightarrow Y=\begin{bmatrix} A & N & A & N & A & S \\ A & N & A & S & A & N \\ A & S & A & N & A & N \\ N & A & N & A & S & A \\ N & A & S & A & N & A \\ S & A & N & A & N & A \end{bmatrix}.$$

* Izlaz iz transformatora je sekvenca SNNAAA, koja ima veći stepen memorije nego originalna sekvenca, pa je pogodnija za kompresiju.

Berouz-Viler transformacija, Bzip2

* *Burrows, Michael; Wheeler, David J. (1994), A block sorting lossless data compression algorithm, Technical Report 124, Digital Equipment Corporation*
  - David Wheeler - University of Cambridge Computer Laboratory
  - Michael Burrows - University of Cambridge, Microsoft, Google

* Danas se mnogo koristi u bioinformatici, gde se vrši fragmentacija i sekvenciranje DNK (*next-generation sequencing*, *NGS*)

Literatura

[1] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal,* vol. 27, pp. 379-423, July 1948; pp. 623-656, October 1948.

[2] S. Lin, D. J. Costello, *Error Control Coding*, Second Edition, Prentice Hall, New Jersey, 2004.

[3] D. Drajić, P. Ivaniš, “*Uvod u teoriju informacija sa kodovanjem*”, treće izdanje, Akademska misao, Beograd, 2009.

[4] P. Ivaniš, “*Zbirka rešenih zadataka iz teorije informacija i kodovanja*”, Akademska misao, Beograd, 2013.

[5] D. J. Costello, Jr., J. Hagenauer, H. Imai, S. B. Wicker, “Applications of Error-Control Coding”, *IEEE Trans. Inform. Theory*. Vol 44 (1998), pp. 2531-2560

[6] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, John Wiley & Sons, Ltd, England, 2002.