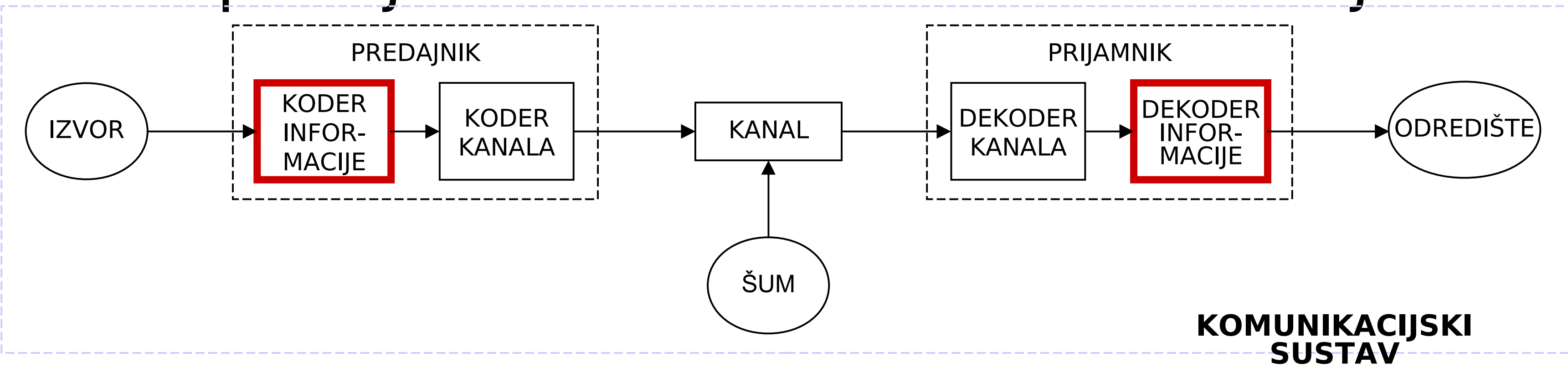
Teorija informacije

Entropijsko kodiranje

Kodiranje i kompresija

- Kodiranje: dodjela kodnih riječi simbolima poruke
- Kompresija: kodiranje koje smanjuje broj bitova potreban za izražavanje poruke
- U jasnom kontekstu, koristimo ove pojmove kao sinonime
- Kompresija se vrši u koderu informacije



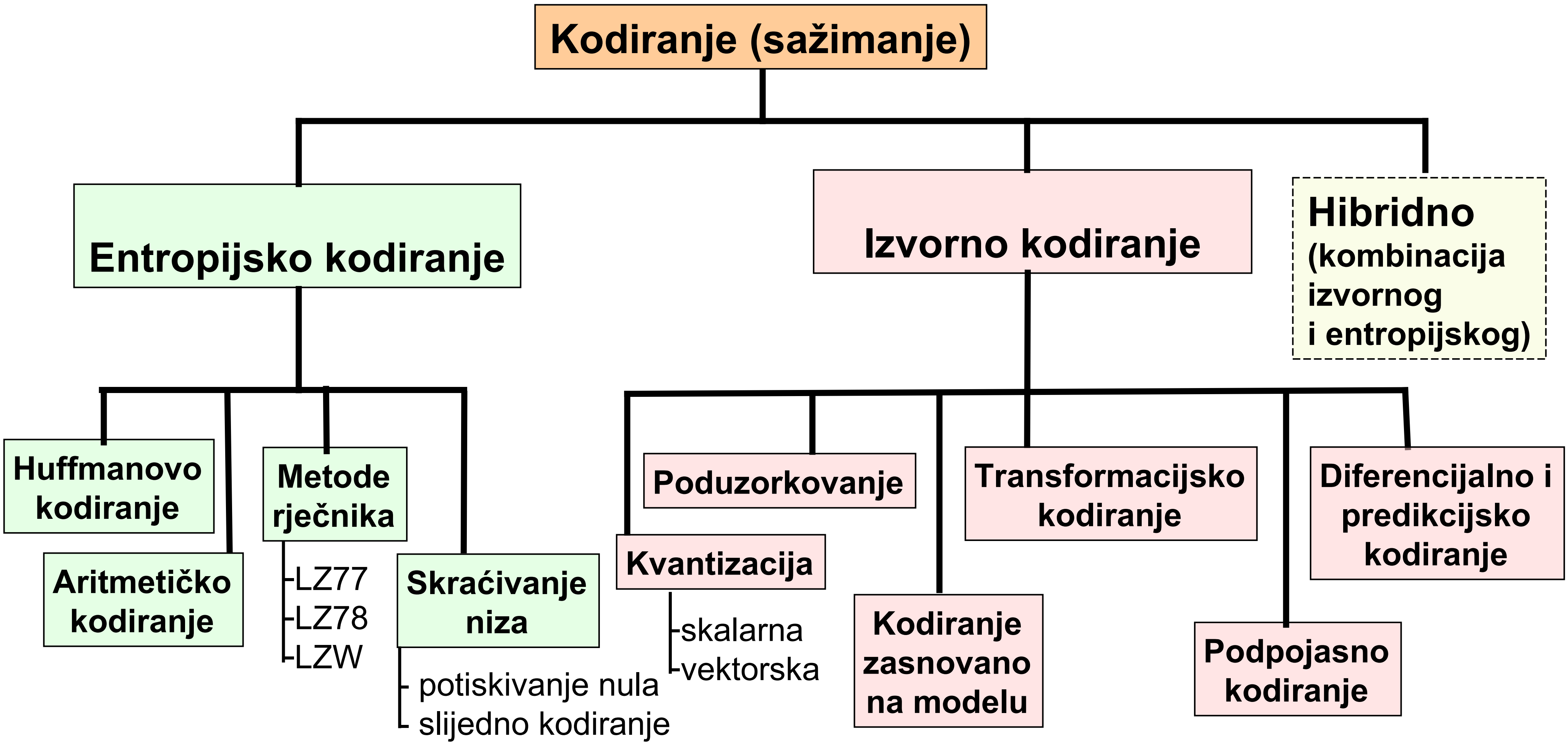
Entropijsko kodiranje

- Uvod u kodiranje i kompresiju
  - Definicije, podjela metoda kompresije
  - Uvod u entropijsko kodiranje
- Karakteristike izvora informacije
  - Stacionarni izvor, ergodički izvor, izvori s memorijom (Markovljevi)
- Vrste kodova i njihova svojstva
  - Singularni, nesingularni, jednoznačno dekodabilni, prefiksni kodovi
- Optimalno kodiranje
- Metode entropijskog kodiranja
  - Shannon-Fanoovo kodiranje
  - Huffmanovo kodiranje
  - Aritmetičko kodiranje
  - Metode rječnika (LZ77, LZ78, LZW)
  - Metode skraćivanja niza (potiskivanje nula, slijedno kodiranje)

Osnovna svojstva kompresije

- Kompresija bez gubitaka
  - Komprimirani podaci mogu se dekomprimiranjem rekonstruirati bez gubitka informacije (*reverzibilno*)
  - Primjene: npr. tekst, medicinske slike, satelitske snimke
- Kompresija s gubicima
  - Cilj je ili dobiti najbolju vjernost rekonstruiranih podataka za zadanu brzinu (bit/s) ili postići najmanju brzinu za zadanu granicu vjernosti
  - Primjene: npr. govor, slika, video
- Važan parametar je omjer kompresije
  - Omjer veličine komprimiranih i originalnih podataka, npr. 1:10

Klasifikacija postupaka kodiranja



Uvod u entropijsko kodiranje

- Osnovna ideja: skraćeno zapisati višestruko ili često ponavljane simbole ili nizove simbola
- Zajedničko svim metodama entropijskog kodiranja:
  - temelje se direktno na teoriji informacije
  - kodiranje <u>bez</u> gubitaka
  - omjer kompresije ovisi samo o statističkim svojstvima izvora informacije
  - poruka se promatra isključivo kao niz niz slučajnih vrijednosti, ne uzimaju se u obzir svojstva medija (za razliku od izvornog kodiranja)

Karakteristike izvora informacije

- Izvor informacije promatramo kao stohastički proces, tj. niz slučajnih varijabli:

$$X_1, X_2, \ldots, X_n$$

- Izvor u potpunosti opisan raspodjelom združenih vjerojatnosti pojavljivanja varijabli:

$$P\left\{\left(X_1, X_2, \ldots, X_n\right) = \left(x_1, x_2, \ldots, x_n\right)\right\} = p\left(x_1, x_2, \ldots, x_n\right)$$

- Općenito, moguća zavisnost među varijablama

DEFINICIJA

Stacionarni izvor

- Statistička svojstva se ne mijenjaju s vremenom

$$P\{(X_1, X_2, \ldots, X_n) = (x_1, x_2, \ldots, x_n)\} = P\{(X_{1+l}, X_{2+l}, \ldots, X_{n+l}) = (x_1, x_2, \ldots, x_n)\},$$
$$\forall\, l, (x_1, x_2, \ldots, x_n) \in X^n, n > 0$$

- Trivijalan primjer stacionarnog izvora:

AEAEAEAEAEAEAEAE.....

- Trivijalan primjer nestacionarnog izvora:

AEAAEEAAAEEEAAAAEEEEAAAAAEEEEE...

DEFINICIJA

Ergodički izvor

- Izvor kao skup svih mogućih proizvedenih nizova
  - Prosjek po skupu: prosjek pojavljivanja simbola na nekom mjestu u nizu, gledano među svim nizovima
  - Prosjek po vremenu: učestalost pojavljivanja simbola unutar pojedinog niza
- Ergodičnost: prosjek po skupu = prosjek po vremenu
- Svaki proizvedeni niz ima ista svojstva i ona se ne mijenjaju u vremenu
- Za entropijsko kodiranje promatramo ergodičke izvore (aproksimacija stvarnih

Ergodičnost izvora - primjer

- Izvor počinje 1/3 sa A, 1/3 B i 1/3 E
  - Ako počne sa A ili B ponavlja ih izmjenično
  - Ako počne sa E, ponavlja samo E
  - Skup mogućih nizova:

    Niz 1: ABABABABABABAB...

    Niz 2: BABABABABABABA...

    Niz 3: EEEEEEEEEEEEEE...

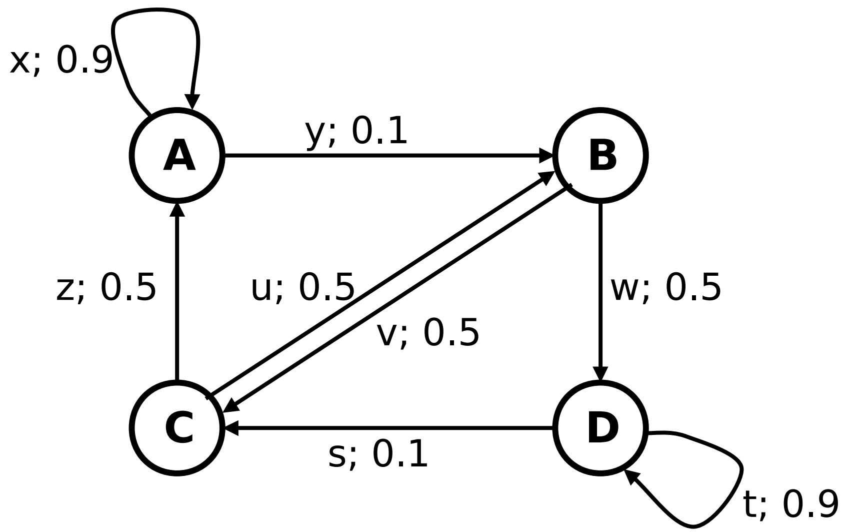
Simbol	Prosjek po vremenu za niz 1	Prosjek po vremenu za niz 2	Prosjek po vremenu za niz 3	Prosjek po skupu
A	½	½	0	⅓
B	½	½	0	⅓
E	0	0	1	⅓

Izvori s memorijom

- Vjerojatnost pojavljivanja simbola je ovisna o jednom ili više prethodnih simbola
- Neki nizovi simbola vjerojatniji od drugih
- Većina prirodnih izvora su izvori s memorijom
  - Npr. iz slova u tekstu, zvuk govora, slika

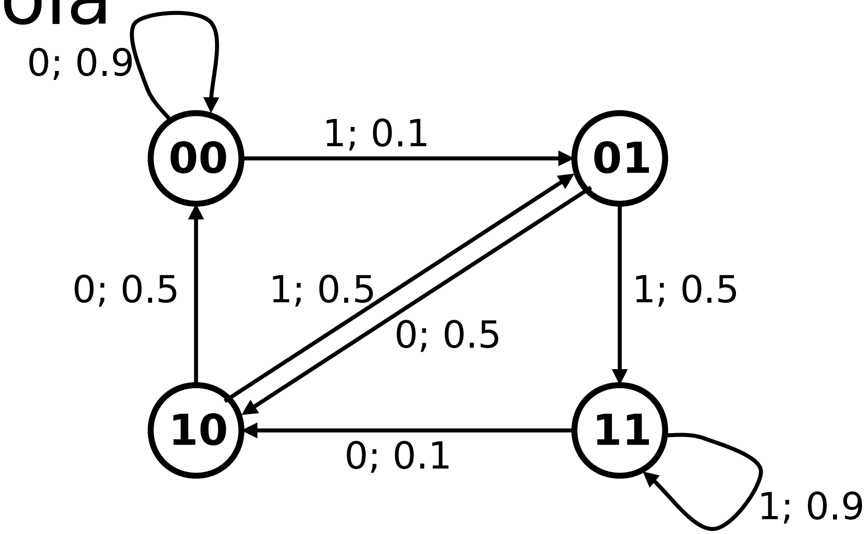
Markovljevi informacijski izvori

- Izvori s memorijom često se mogu opisati pomoću Markovljevih
- Stanja, vjerojatnosti prijelaza
- Pri prijelazu stanja generira se simbol



Primjer Markovljevog izvora

- Binarni Markovljev izvor s memorijom od dva simbola



- Tipičan izlaz:

000000000000000001111111111111111110000111111111100000111111 111111....

DEFINICIJA

Kodiranje

- Dodjela kodnih riječi simbolima poruke

$$X = \{ x_1, x_2, \ldots, x_i, \ldots, x_n \}$$

$$x_i \xrightarrow{KODIRANJE} C(x_i)$$

$$C(x_i) \in D^*, \; D = \{ a_1, a_2, \ldots, a_d \},$$

- Kodiranje sa svojstvom sažimanja: kompresija
- U praksi gotovo uvijek binarna abeceda
  - *d* = 2, *D* = {0,1}
  - Izlaz kodera: struja bitova (engl. *bitstream*)

DEFINICIJA

Prosječna duljina kodne riječi

- Duljina pojedine kodne riječi: $l(x_i)$, skraćeno l_i
  - broj simbola koji čine tu kodnu riječ
- Prosječna duljina kodne riječi (prosječna duljina koda):

$$L = \sum_{i=1}^{n} p(x_i) l(x_i) = \sum_{i=1}^{n} p_i l_i$$

- Za dugačku poruku od N simbola, očekivana duljina kodirane poruke je NL
- L [bit/simbol] je mjera efikasnosti koda

Primjer kodiranja 1

SIMBOL (x_i)	VJEROJATNOST POJAVLJIVANJA $p(x_i) = p_i$	KODNA RIJEČ (C_i)	DULJINA KODNE RIJEČI (l_i)
1	1/2	0	1
2	1/4	10	2
3	1/8	110	3
4	1/8	111	3

- Prosječna duljina kodne riječi:

$$L = \sum_{i=1}^{n} p_i l_i = 0.5 \cdot 1 + 0.25 \cdot 2 + 0.125 \cdot 3 + 0.125 \cdot 3 = 1.75 \left[bit / simbol \right] = H(X)$$

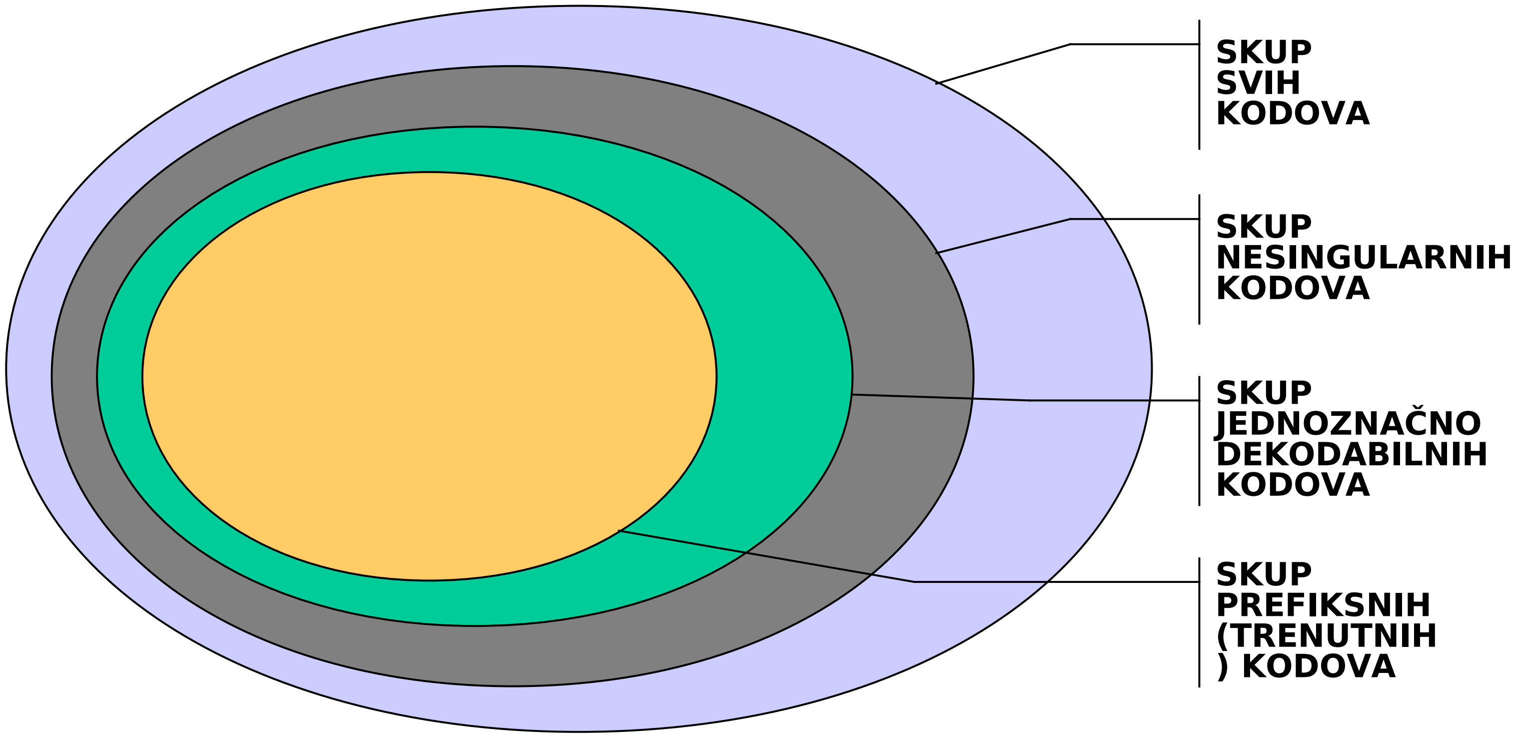
Primjer kodiranja 2

SIMBOL (x_i)	VJEROJATNOST POJAVLJIVANJA $p(x_i) = p_i$	KODNA RIJEČ (C_i)	DULJINA KODNE RIJEČI (l_i)
1	1/3	0	1
2	1/3	10	2
3	1/3	11	2

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i = -\log \frac{1}{3} = 1.58 \left[\text{bit/simbol}\right],$$

$$L = \sum_{i=1}^{n} p_i l_i = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 2 + \frac{1}{3} \cdot 2 = 1.66 \left[\text{bit/simbol}\right].$$

Vrste kodova



DEFINICIJA

Nesingularni kodovi

- Kod je nesingularan ako svakom simbolu dodjeljuje drugačiju kodnu riječ

$$x_i \neq x_j \Rightarrow C(x_i) \neq C(x_j)$$

- To nije garancija jednoznačnosti
- Primjer:
  - Simboli A, B, C; kod: C(A) = 0, C(B) = 01 i C(C) = 1
  - “ABC” → “0011”
  - “0011” → ?

Jednoznačno dekodabilni kodovi

$$x \xrightarrow{KOD} C(x)$$

$$x_1 x_2 \ldots x_n \xrightarrow{PROŠIRENI\ KOD} C(x_1 x_2 \ldots x_n) = C(x_1) C(x_2) \ldots C(x_n)$$

- Kod jednoznačno dekodabilan ako je proširenje nesingularno
  - Različite poruke → različite kodirane poruke
- Primjer:
  - Simboli A, B, C; kod: C(A) = 0, C(B) = 01 i C(C) = 011
  - “ABC” → ““001011” → “ABC”
  - “001...” → ?
- Ne može se trenutno dekodirati

DEFINICIJA

Prefiksni (trenutni) kodovi

- Prefiksni kod je kod u kojem niti jedna kodna riječ nije prefiks neke druge kodne riječi
- Svaka kodna riječ se može trenutno dekodirati, bez znanja iduće kodne riječi
- U prethodnom primjeru, problem je upravo u tome što su kodne riječi jedna drugoj prefiks

Vrste kodova: primjer

SIMBOL (x_i)	VRSTA KODA			
	SINGULARNI	NESINGULARNI	JEDINSTVENO DEKODABILNI	PREFIKSNI
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111
"1234" →	0000	00100110	100011110	010110111
Dekodirano	?	?	1234	1234
Prvih 6 simbola	?	?	? (123 ili 124)	123

TEOREM

Kraftova nejednakost

- Za svaki prefiksni kod sa abecedom od d simbola i duljinama kodnih riječi $l_1, l_2, \ldots, l_n$ vrijedi:

$$\sum_{i=i}^{n} d^{-l_i} \leq 1$$

  i obrnuto, za bilo koji skup duljina kodnih riječi li koje zadovoljavaju ovu nejednakost, postoji prefiksni kod s takvim duljinama kodnih riječi.
- Određuje minimalne duljine kodnih riječi potrebne za prefiksni kod

Kraftova nejednakost – primjeri

1. Prethodni primjer koda {0, 10, 110, 111}
   - Binarna abeceda, D=2

$$\sum_{i=1}^{n} 2^{-l_i} \le 1$$

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1$$

   - Nema kraćeg koda

2. Tražimo kod za tri simbola

$2^{-1} + 2^{-2} + 2^{-2} = 1$ =>mora postojati pref. kod duljina 1, 2, 2

Optimalni kodovi (1/2)

- Općenito, više kodova zadovoljava K.N.; koji je optimalan?
  - npr: {0, 10, 110, 111}, {111, 0, 10, 110}...
- Optimalan kod: prefiksni kod sa najmanjom mogućom prosječnom duljinom kodne riječi

$$\min \left\{ L = \sum_{i=1}^{n} p_i l_i \right\} \text{uz uvjet } \sum_{i=1}^{n} d^{l_i} \leq 1$$

Optimalni kodovi (2/2)

- Minimum se dobiva za:

$$l_i^* = -\log_d p_i \Rightarrow L = -\sum_{i=1}^{n} p_i \log_d p_i = H(X)$$

- Ali l_i moraju biti cijeli brojevi, pa se ne može uvijek postići L=H: $L \geq H(X)$

- Za optimalni kod, prosječna duljina kodne riječi je unutar jednog bita od entropije: $H(X) \leq L < H(X) + 1$
- Efikasnost koda: $\eta = \frac{H(X)}{L}$

Metode entropijskog kodiranja

- Shannon-Fanoovo kodiranje
- Huffmanovo kodiranje
  - optimalno kodiranje
  - binarno stablo
  - kraći zapis čestih znakova
- Aritmetičko kodiranje
  - poopćenje Huffmanovog kodiranja
  - cijela poruka se pretvara u jednu kodnu riječ
- Metode rječnika
  - isti rječnik kodnih riječi na strani pošiljatelja i primatelja
  - dinamička konstrukcija rječnika
  - Lempel-Ziv (LZ77, LZ78), Lempel-Ziv-Welch (LZW)
- Metode skraćivanja niza
  - potiskivanje nula, slijedno kodiranje

Shannon-Fanoovo kodiranje

- Jedna je od prvih metoda kodiranja utemeljenih na teoriji informacije
- Ne daje uvijek optimalan kod
  - Vrlo rijetko se koristi
- Zasniva se na željenim svojstvima kôda:
  - Niti jedna kodna riječ ne smije biti prefiks neke druge kodne riječi;
  - Želimo da se u kodiranim porukama simboli 0 i 1 pojavljuju s podjednakom vjerojatnošću.

Shannon-Fanoovo kodiranje: postupak

- Posložiti simbole po padajućim vjerojatnostima
- Podjela simbola u grupe
- Dodjela znamenke 0 jednoj, a 1 drugoj grupi
- Postupak se ponavlja dok se grupe ne svedu na 1 simbol

Shannon-Fanoovo kodiranje: primjer

x_i	$p(x_i)$	KORAK 1	KORAK 2	KORAK 3	KORAK 4	KODNA RIJEČ	DULJINA KODNE RIJEČI
x_1	0.25	0	0			00	2
x_2	0.25	0	1			01	2
x_3	0.125	1	0	0		100	3
x_4	0.125	1	0	1		101	3
x_5	0.0625	1	1	0	0	1100	4
x_6	0.0625	1	1	0	1	1101	4
x_7	0.0625	1	1	1	0	1110	4
x_8	0.0625	1	1	1	1	1111	4
Prosječna duljina kodne riječi:							2.75

Huffmanovo kodiranje

- D. A. Huffman,1952. godine
- Kodira pojedinačne simbole kodnim riječima promjenjive duljine, ovisno o (poznatim!) vjerojatnostima njihova pojavljivanja
- Temelji se na dvije jednostavne činjenice:
  (1) U optimalnom kodu, simboli s većom vjerojatnošću pojavljivanja imaju kraće kodne riječi od onih s manjom vjerojatnošću
  (2) U optimalnom kodu, dva simbola s najmanjim vjerojatnostima imaju kodne riječi jednake duljine (vrijedi za prefiksni kod)
- Ishod: sažetiji zapis (npr. tipičan tekst se sažima za 45%)

Huffmanovo kodiranje: postupak

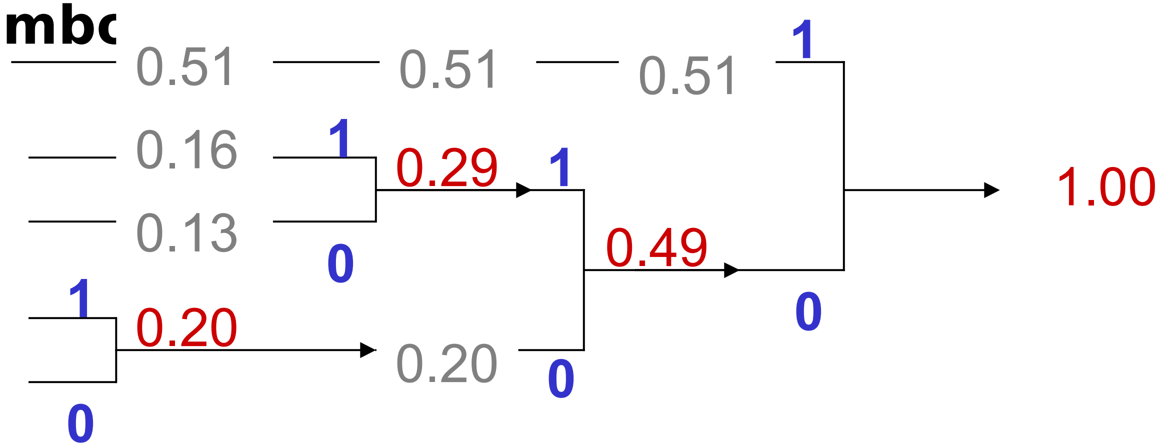
- Algoritam stvaranja koda:
  1. Sortiraj simbole po padajućim vjerojatnostima
  2. Pronađi dva simbola s najmanjim vjerojatnostima
  3. Jednom od njih dodijeli simbol “0”, drugom “1”
  4. Kombiniraj ta dva simbola u jedan nadsimbol (nadsimbol je novi simbol čija je vjerojatnost pojavljivanja jednaka zbroju vjerojatnosti pojavljivanja dvaju simbola od kojih je nastao) i zapiši ih kao dvije grane binarnog stabla, a nadsimbol kao račvanje iznad njih
  5. Ponavljaj 1-4 dok ne dobiješ samo jedan nadsimbol
  6. Povratkom kroz stablo očitaj kodove

- Podatkovna struktura algoritma je binarno stablo
- Algoritam dekodiranja koristi isti postupak za gradnju stabla
  - Dekoder mora znati vjerojatnosti pojavljivanja simbola

Huffmanovo kodiranje: primjer

- Skup simbola {A, B, C, D, E} s vjerojatnostima pojavljivanja   p(A) = 0.16, p(B) = 0.51, p(C) = 0.09, p(D) = 0.13, p(E) = 0.11
- Za uniformni kod, prosječna duljina koda je **3 bit/simbol** (jer je $2^2 < 5 < 2^3$).
- Entropija: **1.96 bit/simbo**

B ... 1	p(B) = 0.51
A ... 011	p(A) = 0.16
D ... 010	p(D) = 0.13
E ... 001	p(E) = 0.11
C ... 000	p(C) = 0.09



- Prosječna duljina dobivenog koda u našem slučaju je:

$$L=\sum_{x \in X} p_x \, l_x \quad = 3 \times (0.09+0.11+0.13+16) + 0.51 = \mathbf{1.98\ bit/simbol}$$

Huffmanovo kodiranje: svojstva

- kodiranje je idealno ako su vjerojatnosti 1/2, 1/4, ... ,$1/2^n$
- u stvarnim slučajevima to obično nije slučaj, te rezultat ovisi o vjerojatnostima pojavljivanja simbola
- prednosti:
  - jednostavan za izvedbu
  - vrlo dobro kodiranje za „dobre“ vjerojatnosti pojavljivanja simbola
- nedostaci:
  - vjerojatnosti pojavljivanja simbola moraju biti poznate; ovise o primjeni (tekst, slika)
  - za “loše raspoređene” vjerojatnosti pojavljivanja dobiju se izrazito loši kodovi

Primjer lošeg koda i prošireni Huffmanov kod

Simbol	Vjerojatnost	Kodna riječ
a_1	0.95	0
a_2	0.02	10
a_3	0.03	11

PROSIRENI KOD

Simbol	Vjerojatnost	Kodna riječ
a_1a_1	0.9025	0
a_1a_2	0.0190	111
a_1a_3	0.0285	100
a_2a_1	0.0190	1101
a_2a_2	0.0004	110011
a_2a_3	0.0006	110001
a_3a_1	0.0285	101
a_3a_2	0.0006	110010
a_3a_3	0.0009	110000

- Entropija: 0.335 bit/simbol
- Prosječna duljina:1.05 bit/simbol: 213% više od entropije!!
- Prošireni kod: 1.222 / 2 = 0.611 bit/simbol: 72% više od entropije.
- Bolje je kodirati duže sekvence, ali tada broj kodnih riječi raste eksponencijalno

Huffmanovo kodiranje: primjene

- Česta primjena unutar složenijih algoritama

- Primjeri:
  - standardi za telefaks (T.4, T.6)
  - standard za nepomičnu sliku JPEG

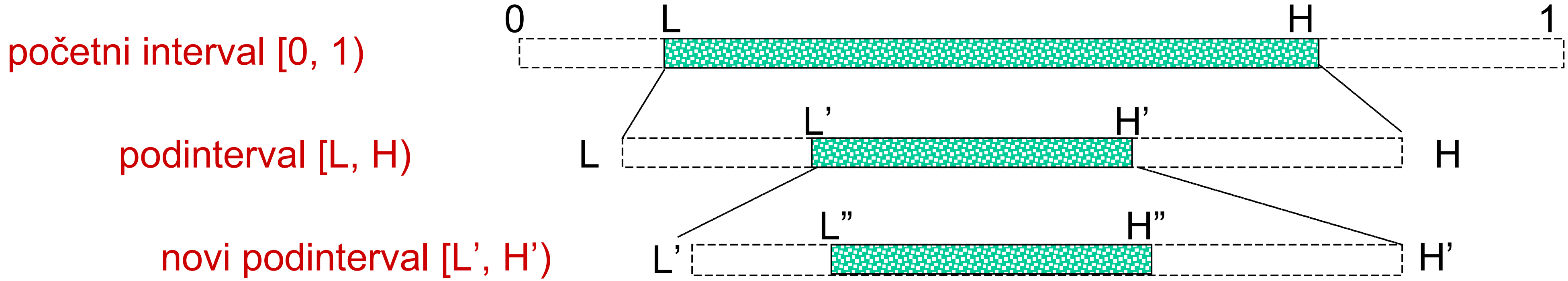
Aritmetičko kodiranje

- Autori Pasco & Rissanen (nezavisno), 1976. godine

- Algoritam uzima kao ulaz cijele nizove simbola ("poruke") i preslikava ih na realne brojeve, ovisno o (poznatim!) statističkim svojstvima

Aritmetičko kodiranje: postupak

1. Podijeli interval [0, 1) u n podintervala koji odgovaraju simbolima iz abecede; duljina svakog podintervala proporcionalna vjerojatnosti odgovarajućeg simbola
2. Iz promatranog skupa podintervala, odaberi podinterval koji odgovara sljedećem simbolu u poruci
3. Podijeli taj podinterval u n novih podintervala, proporcionalno vjerojatnostima pojavljivanja simbola iz abecede; tako nastaje novi skup podintervala koji promatramo
4. Ponavljaj korake 2 i 3 dok cijela poruka nije kodirana
5. Konačni kod za čitavu poruku je jedan broj iz intervala u binarnom obliku



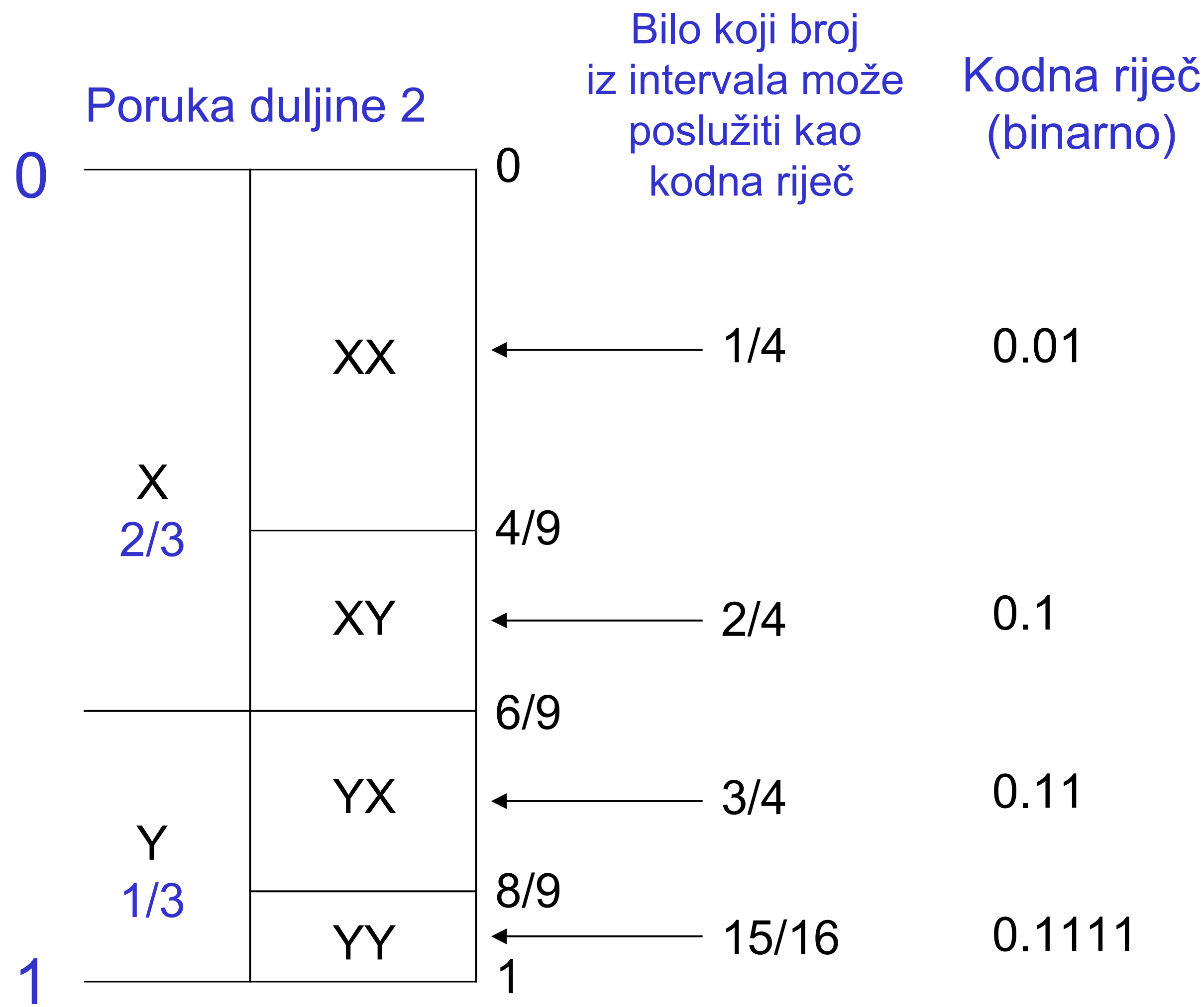
Aritmetičko kodiranje: primjer (1)

- M=2
- simboli: X, Y

  p(X) = 2/3

  p(Y) = 1/3

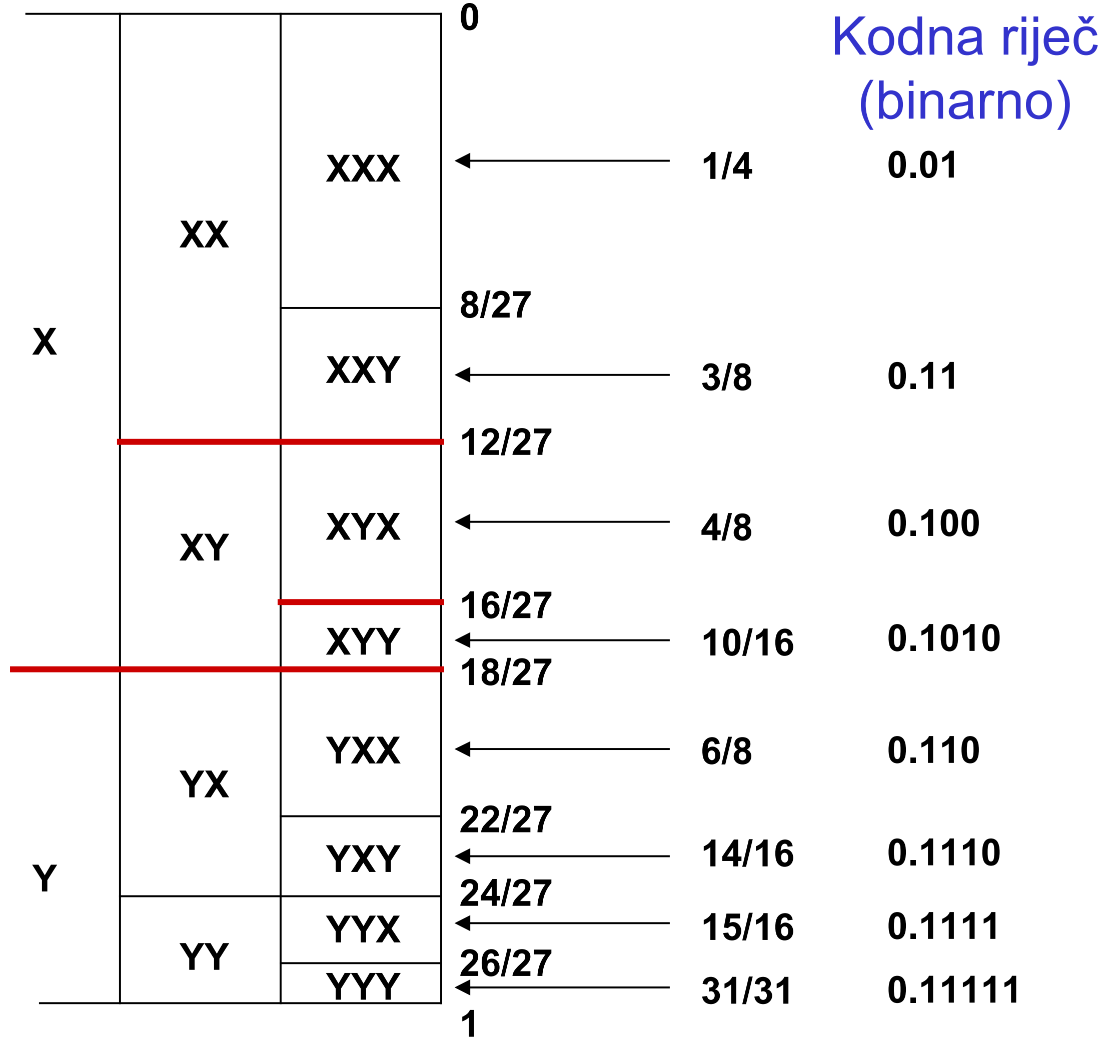
- poruka duljine 2 (moguće poruke XX, XY, YX, YY) kodira se onim brojem bita dovoljnim za jedinstveno određivanje intervala (binarni razlomak!)



Aritmetičko kodiranje: primjer (2)

- primjer za poruku duljine 3

- M=2
- simboli:
  X, Y
  $p(X) = 2/3$
  $p(Y) = 1/3$



Postupak dekodiranja

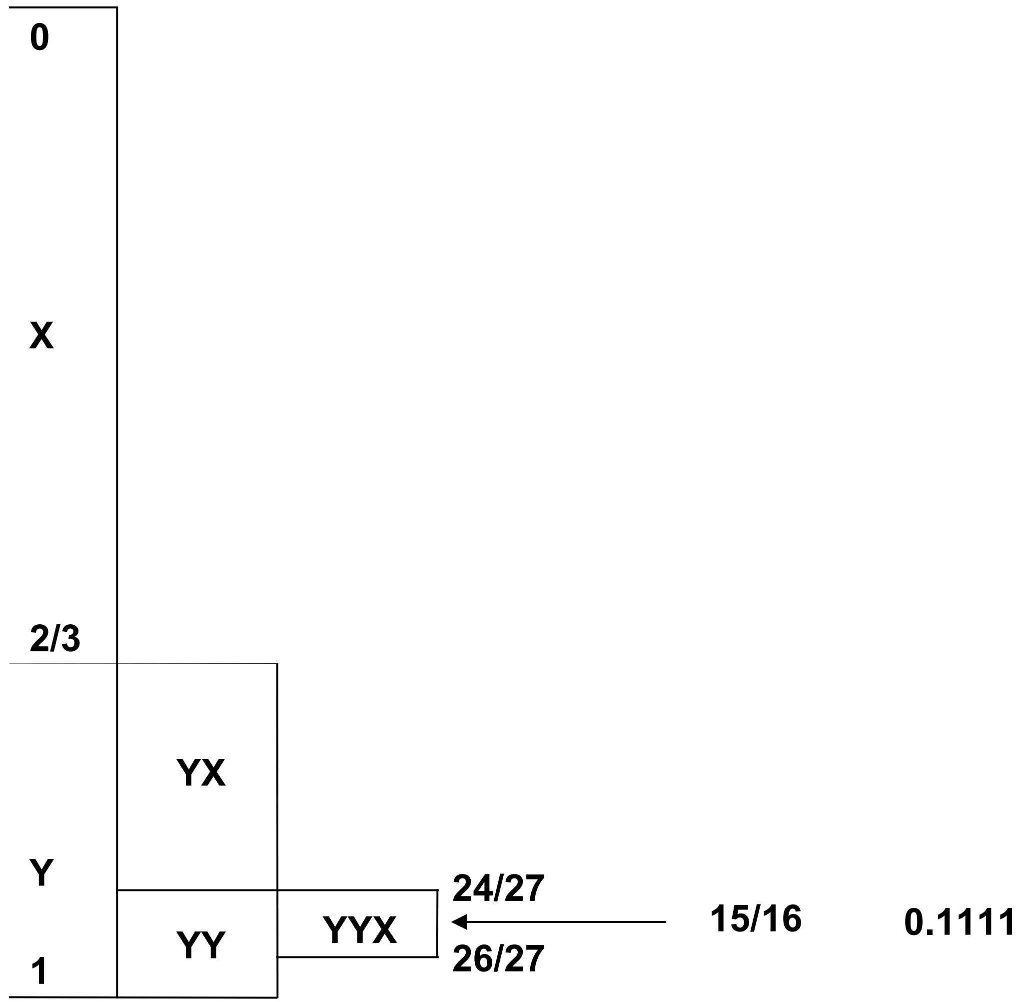
1. Podijeli početni interval [0, 1) u podintervale po vjerojatnostima pojavljivanja simbola
2. Uzmi primljeni kod kao realni broj
3. Pronađi podinterval u kojem se nalazi broj (kod)
4. Zapiši simbol koji odgovara tom podintervalu
5. Podijeli taj podinterval u n novih podintervala, proporcionalno vjerojatnostima pojavljivanja simbola iz abecede; tako nastaje novi skup podintervala koji promatramo

Dekodiranje: primjer

- primjer za poruku duljine 3

- *M*=2
- simboli:
  X, Y
  $p(X) = 2/3$
  $p(Y) = 1/3$

- Primljeni kod 1111 tj. 15/16



Odabir koda

- Kojim brojem iz podintervala kodirati poruku?
- Može se uzeti bilo koja vrijednost iz podintervala
- Dovoljan broj znamenki:

$$l(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil + 1 \text{ [bit]}$$

- Na ovakav način dobiva se uvijek prefiksni kod

Implementacija

- Do sada opisani algoritam neupotrebljiv
  - Neprihvatljivo čekanje do kraja poruke
  - Algoritam podrazumijeva beskonačnu preciznost realnih brojeva – na računalu prikaz s pomičnim zarezom
  - Operacije s realnim brojevima su skupe
- Potreban je algoritam koji:
  - Koristi operacije sa cijelim brojevima
  - Koristi prikaz sa fiksnim brojem bitova
  - Proizvodi simbole koda tokom postupka kodiranja, a ne na kraju

Aritmetičko kodiranje: praktičan postupak

- Osnovni postupak podjele na podintervale je isti
- Koristi se fiksni broj znamenki za prikaz intervala
- Kada je prva znamenka u prikazu gornje i donje granice ista, interval se *renormalizira*:
  - Prvih n znamenki se šalje na izlaz kodera
  - Znamenke se pomiću ulijevo za jedno mjesto
  - Desno se dodaje znamenka: 0 na donju, 1 na gornju granicu intervala (ako su znamenke binarne)

Renormalizacija: primjer

x	p(x)
RAZMAK	1/10
A	1/10
B	1/10
E	1/10
G	1/10
I	1/10
L	2/10
S	1/10
T	1/10

	GORNJA GRANICA	DONJA GRANICA	DULJINA INTERVALA	KUMULATIVNI IZLAZ
Početno stanje	99999	00000	100000	
Kodiraj B (0.2-0.3)	29999	20000		
Renormalizacija, izlaz: 2	99999	00000	100000	.2
Kodiraj I (0.5-0.6)	59999	50000		.2
Renormalizacija, izlaz: 5	99999	00000	100000	.25
Kodiraj L (0.6-0.8)	79999	60000	20000	.25
Kodiraj L (0.6-0.8)	75999	72000		.25
Renormalizacija, izlaz: 7	59999	20000	40000	.257
Kodiraj RAZMAK (0.0-0.1)	23999	20000		.257
Renormalizacija, izlaz: 2	39999	00000	40000	.2572
Kodiraj G (0.4-0.5)	19999	16000		.2572
Renormalizacija, izlaz: 1	99999	60000	40000	.25721
Kodiraj A (0.1-0.2)	67999	64000		.25721
Renormalizacija, izlaz: 6	79999	40000	40000	.257216
Kodiraj T (0.9-1.0)	79999	76000		.257216
Renormalizacija, izlaz: 7	99999	60000	40000	.2572167
Kodiraj E (0.3-0.4)	75999	72000		.2572167
Renormalizacija, izlaz: 7	59999	20000	40000	.25721677
Kodiraj S (0.8-0.9)	55999	52000		.25721677
Renormalizacija, izlaz: 5	59999	20000		.257216775
Renormalizacija, izlaz: 2				.2572167752
Renormalizacija, izlaz: 0				.25721677520

Usporedba aritmetičko - Huffman

Huffman	**Aritmetičko kodiranje**
Kodira svaki simbol posebno	Kodira cijelu poruku jednim kodom: realni broj 0 - 1
Minimalno 1 bit/simbol	Moguće < 1 bit/simbol
Duljina poruke nije važna	Teoretski optimalno za dugačke poruke
Kodiranje niza simbola moguće samo proširenim Huffman kodom	Uvijek se kodira cijela poruka
Jednostavno za računanje	Zahtjevnije za računanje

Aritmetičko kodiranje: primjene

- Primjena kao komponente u raznim standardima i za razne vrste medija
- Dokumenti
  - JBIG (Joint Bi-level Image Processing Group)
- Slika
  - JPEG
- Sintetički sadržaji/animacija
  - MPEG-4 FBA (Face and Body Animation)

Metode rječnika

- Algoritmi kodiranja metodama rječnika uzimaju kao ulaz nizove simbola (“riječi”) promjenjive duljine i kodiraju ih kodnim riječima stalne duljine iz rječnika
- Ne trebaju znati vjerojatnosti pojavljivanja simbola, nazivaju se i *univerzalni koderi*
- Koder i dekoder moraju imati isti rječnik
- Rječnik moze biti statičan, no najčešće je prilagodljiv

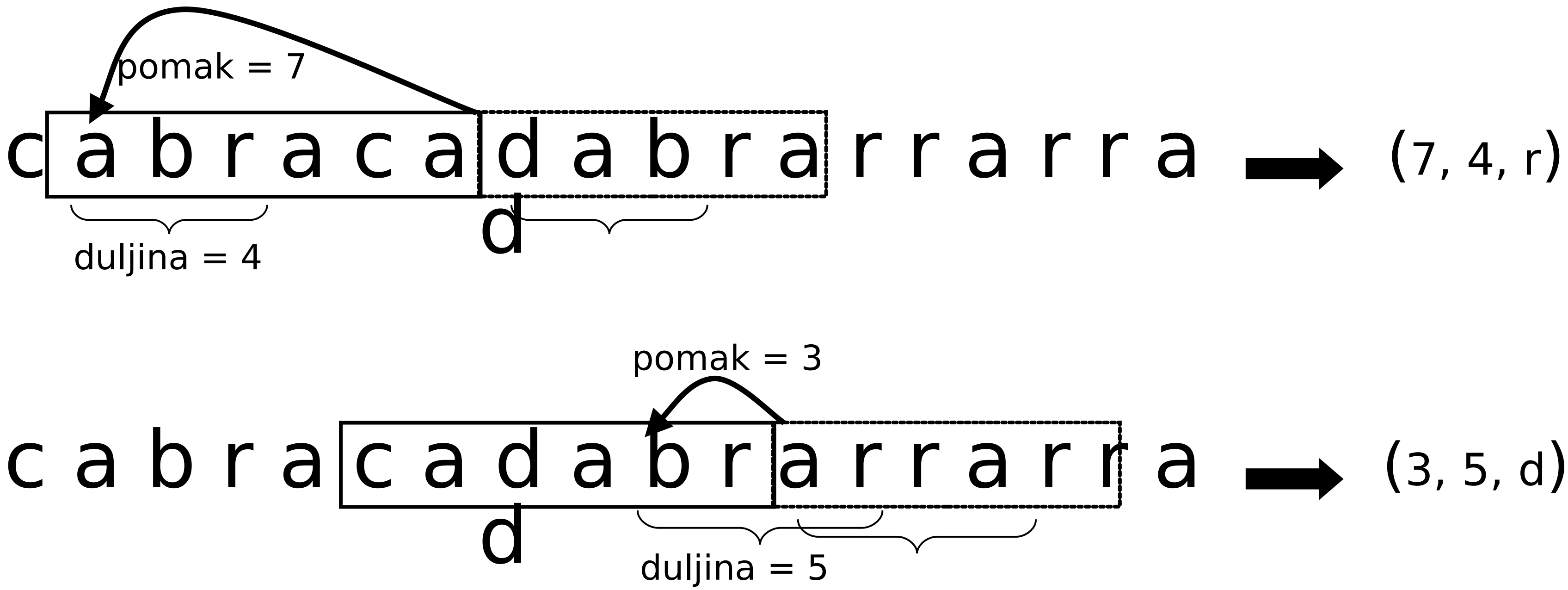
Metode s prilagodljivim rječnikom

- Koder i dekoder dinamički grade rječnik
  - LZ77: Rječnik je posmični prozor
  - LZ78: riječi se grade dodavanjem slova na postojeće riječi (u početku rječnik je prazan)
  - Lempel-Ziv-Welch (LZW) algoritam
    - izvorni algoritam smislili Ziv i Lempel (1977 - LZ77, 1978 - LZ78), a Welch ga je doradio i poboljšao 1984 (zato LZW)
    - algoritam relativno jednostavan, iako složeniji od Huffmanovog
    - izvorni LZW algoritam koristi rječnik s 4K riječi, s tim da su prvih 256 riječi standardni ASCII kodovi

Algoritam LZ77

- Rječnik je posmični prozor od N zadnjih simbola
- U svakom koraku traži se u rječniku najduži niz simbola jednak nadolazećim simbolima, te se kodira kao uređena trojka (*pomak*, *duljina*, *sljedeći_simbol*)
- Nedostatak: “kratka” memorija

LZ77: primjer kodiranja



Algoritmi LZ78 i LZW

- Umjesto posmičnog prozora, zasebna memorija za rječnik
  - Rječnik je poredana lista riječi (nizova simbola)
  - Riječ se dovaća pomoću indeksa (rednog broja)
- LZ78
  - Rječnik u početku prazan
  - U svakom koraku šalje se (*indeks*, *idući simbol*)
    - Indeks pokazuje na najdulju riječ u rječniku jednaku nadolazećem nizu simbola
    - Rječnik se nadopunjava novim riječima tijekom kodiranja

LZW algoritam

- Algoritam kodiranja:

```
RadnaRiječ = slijedeći simbol sa ulaza
WHILE (ima još simbola na ulazu) DO
   NoviSimbol = slijedeći simbol sa ulaza
   IF RadnaRiječ+NoviSimbol postoji u rječniku THEN
      RadnaRiječ = RadnaRiječ+NoviSimbol
   ELSE
      IZLAZ: kod za RadnaRiječ
      dodaj RadnaRiječ+NoviSimbol u rječnik
      RadnaRiječ = NoviSimbol
   END IF
END WHILE
IZLAZ: kod za RadnaRiječ
```

Kodiranje algoritmom LZW: primjer

Sadržaj rječnika na početku:

kodna riječ	znak
(1)	A
(2)	B
(3)	C

Niz znakova koje treba kodirati:

Mjesto	1	2	3	4	5	6	7	8	9
Simbol	A	B	B	A	B	A	B	A	C

LZW:

korak	mjesto	sadržaj rječnika		izlaz iz kodera
1.	1	(4)	A B	(1)
2.	2	(5)	B B	(2)
3.	3	(6)	B A	(2)
4.	4	(7)	A B A	(4)
5.	6	(8)	A B A C	(7)
6.	9			(3)

LZW kodiranje: primjer dekodiranja

KORAK	ULAZ DEKODERA	DEKODIRANI SIMBOLI	SADRŽAJ RJEČNIKA
1	(1)	A	
2	(2)	B	(4) AB
3	(2)	B	(5) BB
4	(4)	AB	(6) BA
5	(7)	ABA	(7) ABA
6	(3)	C	

Metode rječnika: primjene

- LZW
  - UNIX compress
  - GIF
  - Modem V.24 bis
- LZ77
  - ZIP

Metode skraćivanja niza

- potiskivanje ponavljanja (engl. *repetition supression*)
- primjer - potiskivanje nula:

zastavica (*flag*) koja označava nule

894000000000000000000000000000000000 → 894f32

broj ponavljanja

- slijedno kodiranje (engl. *run-length encoding*)
- algoritam kodiranja temelji se na kraćem zapisu ponavljanih simbola pomoću specijalnog znaka (!)
- primjer: ABCCCCCCCCDEFFFABC...

ABCCCCCCCC DEFFFABC...

8 okteta 3 okteta

ABC!8 DEFFFABC...

3 okteta 3 okteta ← "isplati" se za 4+ znakova

- Primjena: prva generacija telefaksa, unutar JPEG-a