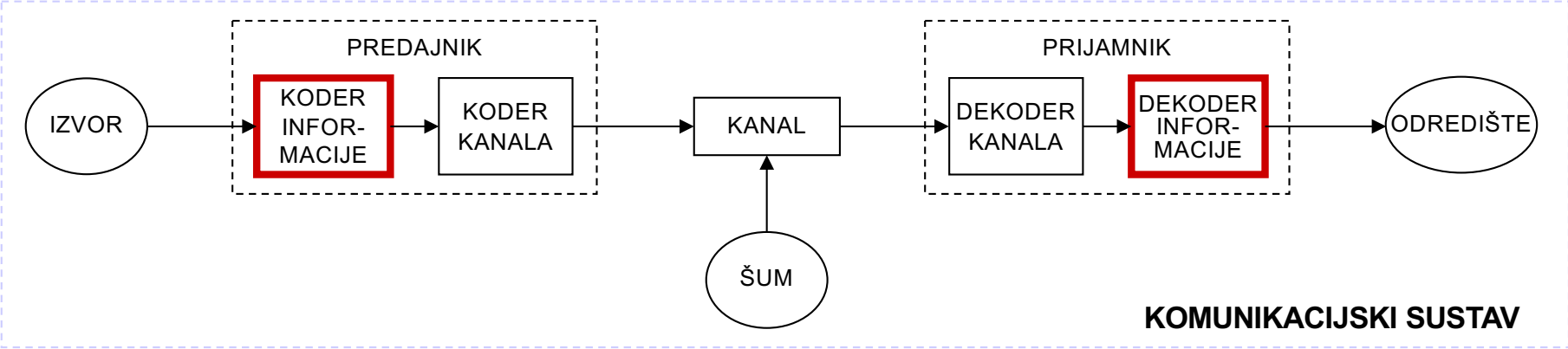
Zavod za telekomunikacije

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



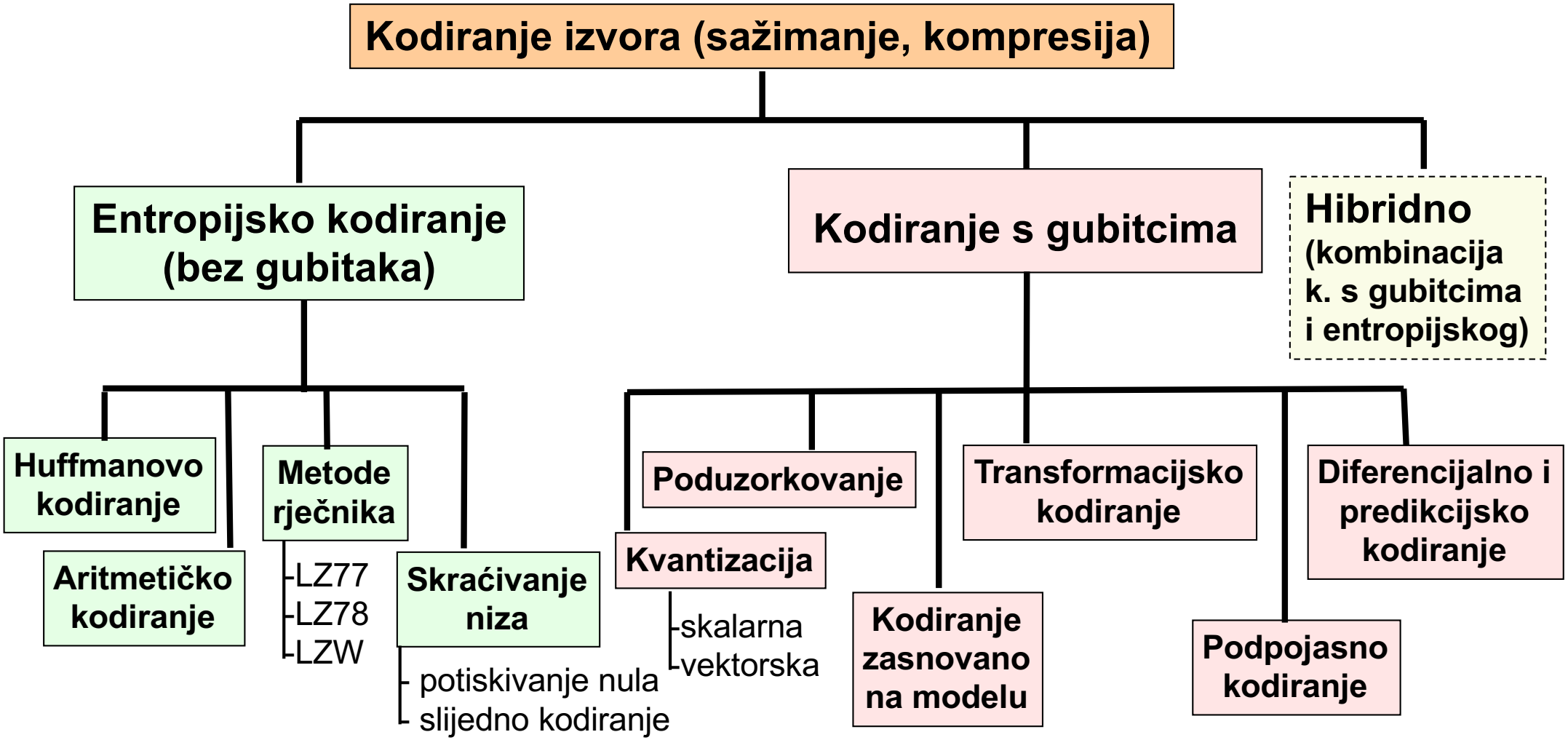
Entropijsko kodiranje

- 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 bez 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

Matematički model izvora informacije

DEFINICIJA

- Izvor informacije promatramo kao stohastički proces, tj. niz slučajnih varijabli: $X_1 , X_2 , \ldots X_t, \ldots, X_n$
  - $A = \{\alpha_1, \ldots, \alpha_m\}$ – abeceda izvora
  - Poruku od n znakova moguće je zapisati kao niz od n članova $x = (a_1, a_2, \ldots, a_n)$; $a_i \in A$, $i = 1, \ldots, n$
    - $x \in A^n$, $A^n = \{(a_1, a_2, \ldots, a_n); a_i \in A, i = 1, \ldots, n\}$
    - skup A^n (Kartezijev produkt skupa A) je skup svih poruka od n članova abecede A; njihov broj je m^n
- Diskretni izvor (A, P) definiramo pomoću abecede A i razdiobe vjerojatnosti na skupu A^n:

$$P(x) = P(X_1 = a_1, X_2 = a_2, \ldots, X_n = a_n) , (a_1, a_2, \ldots, a_n) \in A^n$$

$$\sum_{x \in A^n} P(x) = 1$$

Osnovne karakteristike izvora

- svaki simbol kojeg izvor generira ima određen stupanj neodređenosti (nepredvidivosti)
  - veća nepredvidivost slijeda znači i prijenos veće količine informacije po simbolu
  - ako izvor nametne određenu strukturu slijedu simbola, nepredvidivost se smanjuje, a s njom i količina informacije po simbolu
- izvor bez memorije
  - promatrano u različitim vremenskim trenucima nema korelacije između simbola na izlazu izvora
- izvori s memorijom
  - u bilo kojem trenutku simbol na izlazu izvora ovisi o jednom ili više simbola koji su generirani prije njega
  - primjer: markovljevi izvori

Stacionarni izvor

DEFINICIJA

- 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:

AEAAEEAAAEEEEAAAAAEEEEEAAAAAEEEEEE...

Ergodički izvor

DEFINICIJA

- 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 izvora)

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

Entropija diskretnog izvora informacije

- za $n \geq 1$ moguće je definirati veličinu

$$H_n = \frac{1}{n} H(X_1, X_2, \ldots, X_n) = -\frac{1}{n} \sum_{x \in A^n} P(a_1, \ldots, a_n) \log P(a_1, \ldots, a_n)$$

- a za $n \geq 2$

$$h_n = H\left(X_n \middle| (X_1, X_2, \ldots, X_{n-1})\right) = -\sum_{x \in A^n} P(a_1, \ldots, a_n) \log P\left(a_n \middle| (a_1, \ldots, a_{n-1})\right)$$

  - u oba izraza vrijedi $x = (a_1, a_2, \ldots, a_n) \in A^n$
- moguće interpretacije:
  - veličina H_n je srednja vlastita informacija simbola u n-članoj poruci
  - h_n je uvjetna entropija n-tog simbola ako je poznato prethodnih n-1 simbola

Entropija diskretnog izvora informacije (2)

- Teorem:
  - ako je niz h_n (n = 2, 3, ...) konvergentan,
  - ako je niz H_n (n = 1, 2, ...) konvergentan,
  - tada vrijedi:

  - veličinu $H_n \geq 0$ nazivamo **entropija** zadanog diskretnog stacionarnog **izvora informacije**
    - prosječna informacija koju "nosi" svaki pojedini simbol poslan iz zadanog izvora informacije

Entropija izvora bez memorije

- ako diskretni stacionarni izvor emitira simbole iz A s jednakom vjerojatnošću, tj. $P(\alpha_i) = p_i \geq 0$ u bilo kojem trenutku t_k tada vrijedi

$$\sum_{i=1}^{m} p_i = 1$$

$$P(a_1, a_2, \ldots, a_n) = P(a_1) \cdot P(a_2) \cdot \ldots \cdot P(a_n)$$

- nadalje vrijedi: $$P\left(a_n \middle| (a_1, a_2, \ldots, a_{n-1})\right) = \frac{P(a_1, a_2, \ldots, a_n)}{P(a_1, a_2, \ldots, a_{n-1})} = P(a_n)$$
- ovakav se izvor naziva izvor bez memorije
  - slanje simbola u sadašnjem trenutku stohastički je neovisno o prethodno poslanim simbolima
- entropija izvora bez memorije

$$H_n = \frac{1}{n}\left[H(X_1) + \ldots H(X_n)\right] = H(X_1) = -\sum_{i=1}^{m} p_i \log(p_i)$$

  - također vrijedi: $h_n = H_n$

Izvori s memorijom – primjer: markovljevi izvori

Markovljevi lanci

- niz diskretnih slučajnih varijabli X_0, X_1, X_2, ... naziva se stohastički lanac
- svaka varijabla opisuje stanje nekog sustava u trenucima t_0, t_1, t_2, ...
- pretpostavka: S = {1, 2, 3, ...} je skup svih stanja u kojima se lanac može nalaziti
  - ovaj skup može biti konačan ili beskonačan
  - za modeliranje većine izvora pretpostavka je da je skup S konačan
- definicija markovljevog lanca
  - lanac X_0, X_1, X_2, ... je **markovljev** ako za sve izbore stanja i_1, ..., i_n vrijedi $$P\left(X_{n+1}=i_{n+1}\middle|X_n=i_n,\ldots,X_0=i_0\right)=P\left(X_{n+1}=i_{n+1}\middle|X_n=i_n\right)$$
  - t_n predstavlja sadašnjost
  - stanje u budućnosti ovisi samo o sadašnjem stanju, ali ne i o načinu na koji je slučajni proces došao u to stanje

Matrica prijelaznih vjerojatnosti

- ako vrijedi: $p_{ij} = P\left(X_{n+1} = j \middle| X_n = i\right) = P\left(X_1 = j \middle| X_0 = i\right)$
  - lanac je **homogen**, tj. prijelazne vjerojatnosti ovise samo o stanjima *i* i *j*, a ne o trenutku prijelaza
- matrica prijelaznih vjerojatnosti daje vjerojatnosti prijelaza iz jednog stanja u drugo, u jednom koraku markovljevog lanca
- ako je skup stanja $S = \{1, 2, ..., N\}$, tada je matrica prijelaznih vjerojatnosti matrica dimenzije $N \times N$

$$\Pi = \left[p_{ij}\right] = \begin{bmatrix} P(1|1) & P(2|1) & \dots & P(N|1) \\ P(1|2) & P(2|2) & \dots & P(N|2) \\ \vdots & \vdots & \ddots & \vdots \\ P(1|N) & P(2|N) & \dots & P(N|N) \end{bmatrix}$$

$$P\left(j|i\right) \geq 0$$

$$\sum_{j=1}^{N} P\left(j|i\right) = 1, \forall i = 1, ..., N$$

stohastička matrica

Svojstva matrice prijelaznih vjerojatnosti

- vjerojatnost da se izvor nalazi u nekom određenom stanju varira tijekom vremena
- vjerojatnost da sustav pređe iz stanja *i* u stanje *j* u *m* koraka

$$p_{ij}(m) = P\left(X_{n+m} = j \middle| X_n = i\right)$$

- pri čemu vrijedi: $p_{ij}(1) = p_{ij} = P(j|i)$ i $\Pi(1) = \Pi$
- vjerojatnosti $p_{ij}(m)$ zadovoljavaju Chapman-Kolmogorovljeve jednadžbe

$$p_{ij}(m) = \sum_k p_{ik}(r) p_{kj}(m-r), \forall r = 1, 2, \ldots, m-1$$

- odnosno u matričnom obliku $\Pi(m) = \Pi(r)\Pi(m-r)$

$$\Pi(m) = \Pi^m$$

Stanje izvora u nekom trenutku

- označimo vjerojatnost da se izvor u trenutku t_n nalazi u stanju i kao $p_i(n) := P(X_n = i)$
- tada je razdiobu tih vjerojatnosti moguće opisati vektorom

$$\mathbf{p}(n) = \left[p_1(n), p_2(n), ..., p_N(n) \right] \qquad \sum_{i=1}^{N} p_i(n) = 1, \forall n \in \mathbf{N}$$

- ako je $\mathbf{p}(0)$ **vektor početnih vjerojatnosti**, tada se stanje izvora u trenutku t_n može opisati kao

$$\mathbf{p}(n) = \mathbf{p}(0)\boldsymbol{\Pi}^n$$

- a vezu između dva uzastopna vremenska trenutka opisuje izraz

$$\mathbf{p}(n) = \mathbf{p}(n-1)\boldsymbol{\Pi}$$

Stacionarne vjerojatnosti

- ponekad je važno poznavati ponašanje izvora nakon duljeg vremenskog razdoblja
- teorem: ako postoji broj n takav da su svi elementi matrice Π^n strogo pozitivni (to znači da se u *n* koraka može iz svakog stanja preći u bilo koje drugo stanje), tada za svaki *j* postoji (i ne ovisi o *i*)

$$\pi_j = \lim_{n\to\infty} p_{ij}(n)$$

- koji ne ovisi o *i*
  - π_j su stacionarne vjerojatnosti
    - predstavljaju vjerojatnosti da će izvor u nekom dalekom trenutku (kad nestane utjecaj početnog stanja) generirati simbol *j*
  - markovljev lanac za kojeg postoji gore navedeni limes naziva se ergodični ili regularan

Određivanje stacionarnih vjerojatnosti

- početne jednadžbe: $\mathbf{p}(n)=\mathbf{p}(n-1)\boldsymbol{\Pi}$

  $$p_j(n)=\sum_k p_k(n-1)\,p_{kj}$$

- ako postoje stacionarne vjerojatnosti, onda vrijedi:

$$\pi_j=\lim_{n\to\infty} p_j,\quad \pi_j=\sum_k \pi_k p_{kj},\forall j,\quad \sum_k \pi_k=1$$

- ili u matričnom obliku: $\boldsymbol{\Pi}^{\mathrm{T}}\boldsymbol{\pi}=\boldsymbol{\pi}$
- primjer:
  - zadana je matrica

$$\boldsymbol{\Pi}=\begin{bmatrix}0,25 & 0,5 & 0,25\\ 0,5 & 0,0 & 0,5\\ 0,0 & 0,25 & 0,75\end{bmatrix}$$

$$\boldsymbol{\Pi}^{\mathrm{T}}\begin{bmatrix}\pi_1\\ \pi_2\\ \pi_3\end{bmatrix}=\begin{bmatrix}\pi_1\\ \pi_2\\ \pi_3\end{bmatrix}$$

$$\begin{bmatrix}0,25 & 0,5 & 0,0\\ 0,5 & 0,0 & 0,25\\ 0,25 & 0,5 & 0,75\end{bmatrix}\begin{bmatrix}\pi_1\\ \pi_2\\ \pi_3\end{bmatrix}=\begin{bmatrix}\pi_1\\ \pi_2\\ \pi_3\end{bmatrix}$$

$$\begin{bmatrix}\pi_1\\ \pi_2\\ \pi_3\end{bmatrix}=\begin{bmatrix}2/13\\ 3/13\\ 8/13\end{bmatrix}$$

Markovljev izvor – primjer izvora s memorijom

- markovljev izvor (A, P) modeliramo markovljevim lancom
  - pretpostavka: vjerojatnost slanja simbola a_n u trenutku t_n ovisi samo o prethodno slanom simbolu a_{n-1}
  - proces generiranja informacije potpuno je određen ako su poznate
  - a) razdioba početnih vjerojatnosti $p_i = P(X_i = \alpha_i) \geq 0$
    - $i = 1, 2, ..., N$, $\sum_{i=1}^{N} p_i = 1$
  - b) matrica Π prijelaznih vjerojatnosti
    - čiji su elementi $p_{ij} = P(X_k = \alpha_j \mid X_{k+1} = \alpha_i) \geq 0$; $i, j = 1, 2, ..., N$, $\sum_{j=1}^{N} p_{ij} = 1$
  - ovako opisan izvor naziva se **jednostavni markovljev izvor** informacije

Entropija jednostavnog markovljevog izvora

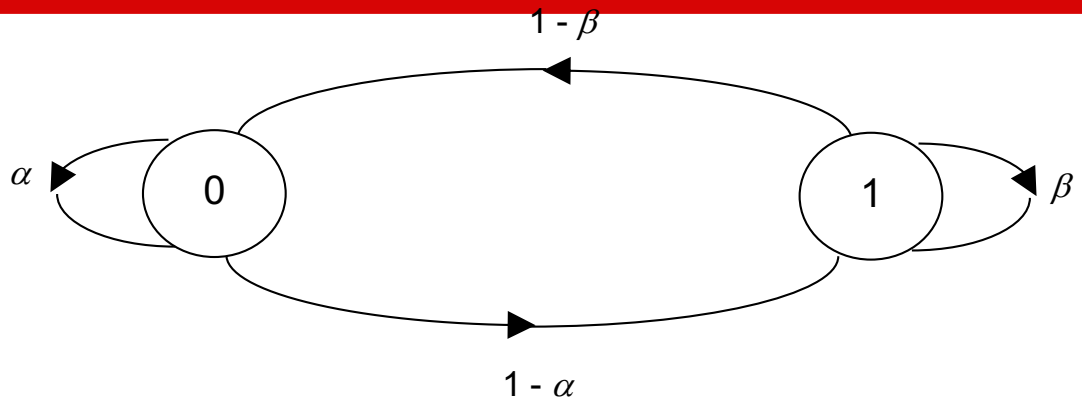
- uzmimo $n > 2$ i napišimo $P\left(a_n \left| (a_1,\ldots,a_{n-1}) \right.\right) = \frac{P\left(a_1,\ldots,a_{n-1},a_n\right)}{P\left(a_1,\ldots,a_{n-1}\right)}$
- pa za jednostavan markovljev izvor vrijedi

$$P\left(a_n \left| (a_1,\ldots,a_{n-1}) \right.\right) = P\left(a_n \left| a_{n-1} \right.\right), (a_1,\ldots,a_n) \in A^n$$

- izvor ima memoriju prvog reda
  - izvor bez memorije ima memoriju nultog reda
- entropija jednostavnog markovljevog izvora

$$H = \lim_{n\to\infty}\left(h_n\right) = -\sum_{i=1}^{N} p_i \sum_{j=1}^{N} p_{ij} \log\left(p_{ij}\right)$$

Markovljev izvor s dva stanja



- zadane prijelazne vjerojatnosti:
  - $\alpha = P(X_1 = 0|X_0 = 0)$, $\beta = P(X_1 = 1|X_0 = 1)$
  - $0 < \alpha, \beta < 1$
- vrijedi:
  - $1 - \alpha = P(X_1 = 1|X_0 = 0)$, $1 - \beta = P(X_1 = 0|X_0 = 1)$
  - $0 < \alpha, \beta < 1$
- stacionarne vjerojatnosti: $\pi_1 = \frac{1-\beta}{2-\alpha-\beta}, \pi_2 = \frac{1-\alpha}{2-\alpha-\beta}$

$$\Pi = \begin{bmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{bmatrix}$$

Prikaz markovljevog izvora grafom

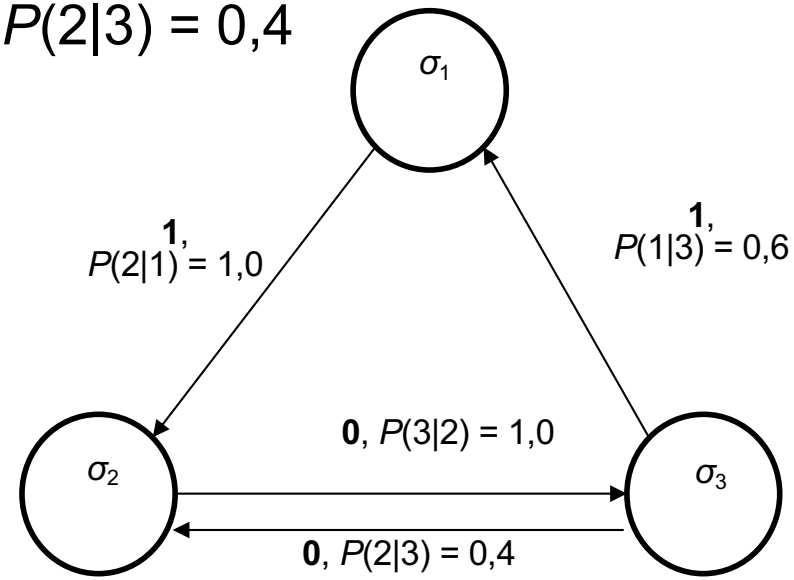
- markovljev izvor određen je:
  - abecedom izvora, *A*,
  - skupom stanja, Σ,
  - skupom prijelaza između stanja,
  - skupom oznaka pridruženih prijelazima
    - te su oznake simboli iz abecede izvora
  - s dva skupa vjerojatnosti
    - prvi skup se odnosi na početnu razdiobu vjerojatnosti definiranu nad skupom stanja
    - drugi skup je skup vjerojatnosti pridruženih prijelazima između stanja – za svaki par stanja σ_i i σ_j definirana je vjerojatnost prijelaza iz stanja σ_i u stanje σ_j, označena kao

$$P\left(j\middle|i\right)=P\left(X_{n+1}=j\middle|X_n=i\right)$$

  - stanja se u grafu prikazuju čvorovima, a prijelazi između stanja granama kojima su pridružene oznake

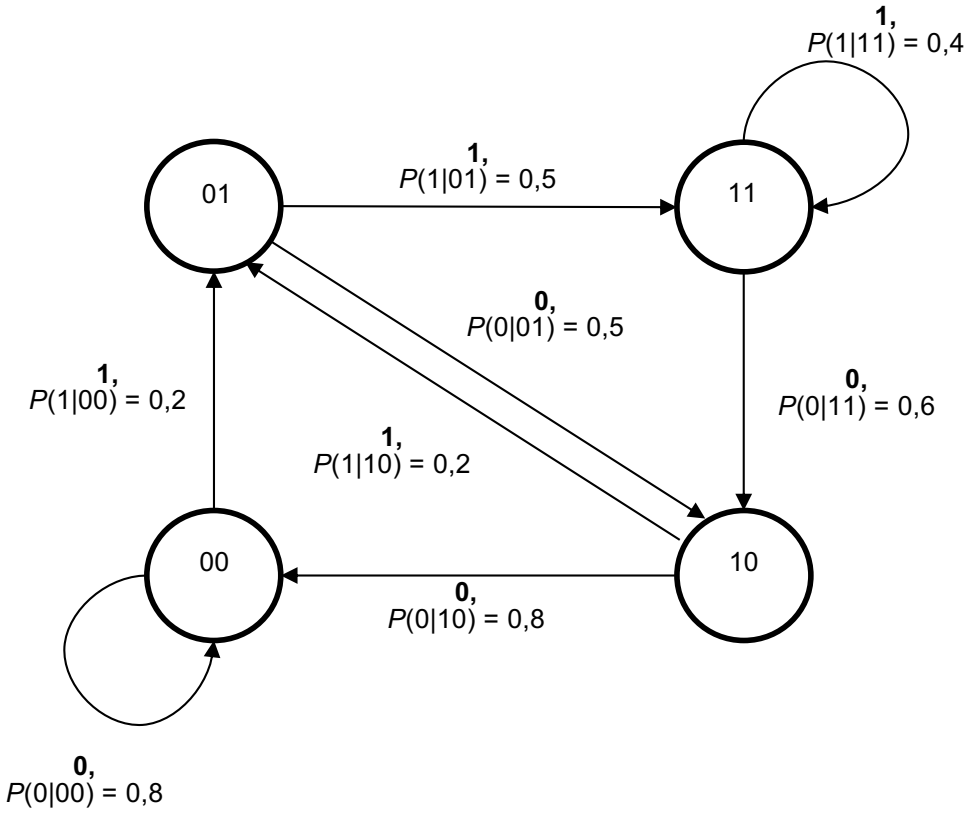
Primjer 1: graf markovljevog izvora

- razmotrimo markovljev izvor definiran abecedom {0, 1} i skupom stanja $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$
- definirana su četiri prijelaza između stanja
  - $\sigma_1 \rightarrow \sigma_2$, s oznakom 1 i vjerojatnošću $P(2|1) = 1{,}0$
  - $\sigma_2 \rightarrow \sigma_3$, s oznakom 0 i vjerojatnošću $P(3|2) = 1{,}0$
  - $\sigma_3 \rightarrow \sigma_1$, s oznakom 1 i vjerojatnošću $P(1|3) = 0{,}6$
  - $\sigma_3 \rightarrow \sigma_2$, s oznakom 0 i vjerojatnošću $P(2|3) = 0{,}4$
- razdioba **početnih vjerojatnosti**:
  - $P(\sigma_1) = P(\sigma_2) = P(\sigma_3) = 1/3$



Primjer 2: pamćenje dva zadnja simbola

- Vjerojatnost pojave simbola ovisi o dva prethodna simbola „zapamćena” u trenutnom stanju izvora



- Stacionarne vjerojatnosti
  - $P(00) = 0,4$
  - $P(01) = 0,3$
  - $P(10) = 0,2$
  - $P(11) = 0,1$

Kodiranje

Kodiranje

- Dodjela kodnih riječi simbolima poruke

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

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

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

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

Prosječna duljina kodne riječi

DEFINICIJA

- 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:

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

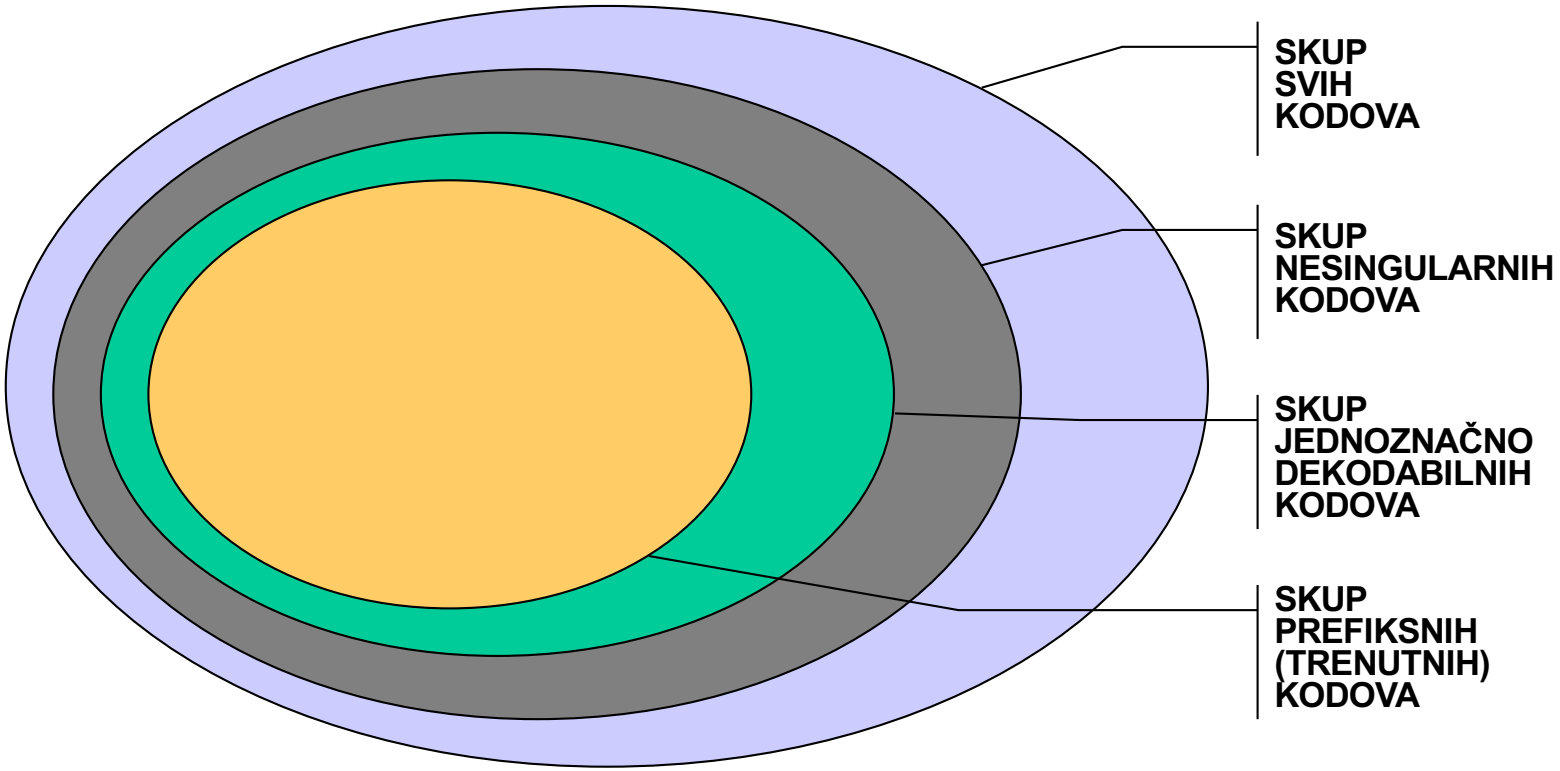
Primjer kodiranja 2

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



Nesingularni kodovi

DEFINICIJA

- 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

DEFINICIJA

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

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

- 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

Prefiksni (trenutni) kodovi

DEFINICIJA

- 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

Kraftova nejednakost

TEOREM

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

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

  i obrnuto, za bilo koji skup duljina kodnih riječi l_i 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

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

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

   - 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: $\varepsilon = \dfrac{H(X)}{L}$

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

SF kodiranje: kada je kod optimalan?

- Kada su vjerojatnosi simbola raspoređene prema:

$$p(x_i) = 2^{-l_i}$$

- Gdje je l_i duljina odgovarajuće kodne riječi
- Npr:

$$p(x_1) = 2^{-2} = \frac{1}{4} \qquad p(x_4) = 2^{-3} = \frac{1}{8}$$

$$p(x_2) = 2^{-2} = \frac{1}{4} \qquad p(x_5) = 2^{-4} = \frac{1}{16}$$

$$p(x_3) = 2^{-2} = \frac{1}{4} \qquad p(x_6) = 2^{-4} = \frac{1}{16}$$

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 – tvrdnja 2?

(2) U optimalnom kodu, dva simbola s najmanjim vjerojatnostima imaju kodne riječi jednake duljine (vrijedi za prefiksni kod)

- Zamislimo da nemaju!
  - kodne riječi A i B
  - A ima *k* bitova više od B
  - prefiksni kod –> B sigurno nije prefiks od A
  - možemo ukloniti višak bitova (*k*) iz riječi A (A i B će ostati različiti)
  - uvjet: optimalni kod – ostale riječi sigurno nisu duže od skraćenog A
    - slijedi: A sigurno nije niti prefiks neke druge kodne riječi
  - dakle: moguće je napraviti kraći kod koji je optimalan
    - tvrdnja 2 vrijedi!

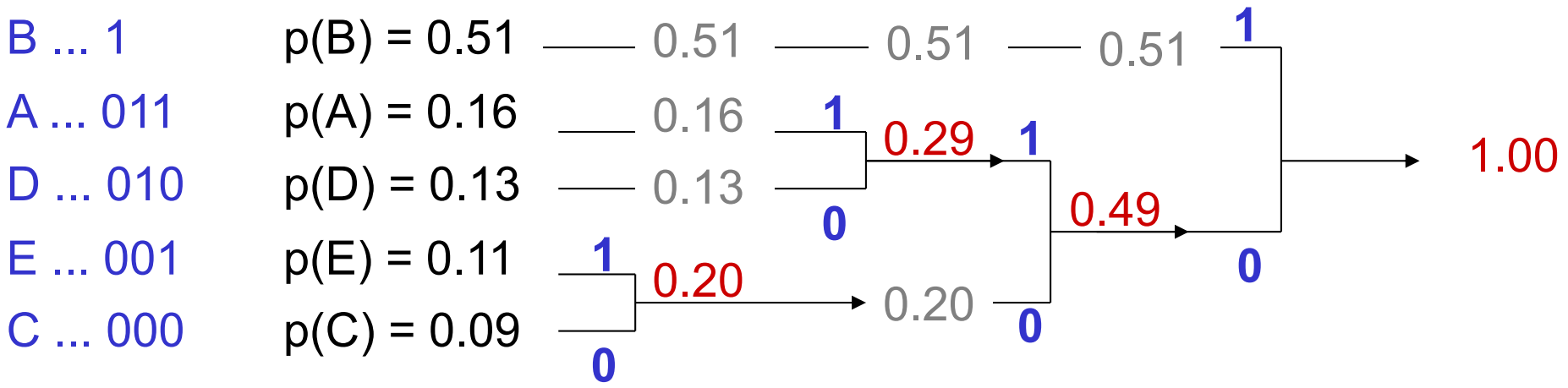
Huffmanovo kodiranje: postupak

- 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 $2^2 \leq 5 \leq 2^3$).
- Entropija: **1.96 bit/simbol**



- Prosječna duljina dobivenog koda u našem slučaju je:

$$L = \sum_{x \in X} p_x \, l_x$$ = 3 x (0.09+0.11+0.13+16) + 0.51 = **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
- dekoder mora poznavati tablicu tj. stablo (frekvencije simbola)!

Zavod za telekomunikacije

Huffman i Shanon-Fano

Primjer "Hello world"

Primjer

- kodiramo frazu “HELLO WORLD”
- npr. ASCII?
  - 72 69 76 76 79 32 87 79 82 76 68
  - u bitovima:
    01001000010001010100110001001100010011110010000001010111010011111010100100100110001000100
    - 88 bitova za 11 znakova!
- kompresija!

Primjer

- “HELLO WORLD”

SIMBOL	POJAVLJIVANJA	$p(x_i)$
H	1	0.091
E	1	0.091
L	3	0.273
O	2	0.182
(razmak)	1	0.091
W	1	0.091
R	1	0.091
D	1	0.091

Shannon-Fano

SIMBOL	$p(x_i)$								$C(x_i)$
L	0.273	*0.546*	0	*0.273*	0				00
O	0.182		0		1	0			010
(razmak)	0.091		0	*0.273*	1	1			011
W	0.091	*0.455*	1		0	*0.182*	0	0	1000
R	0.091		1	*0.273*	0		0	1	1001
D	0.091		1		0	*0.091*	1		101
H	0.091		1	*0.182*	1	0			110
E	0.091		1		1	1			111

Shannon – Fano vs ASCII

Simbol	p(xi)	C(xi)
L	0.273	00
O	0.182	010
E	0.091	111
H	0.091	110
razmak	0.091	011
W	0.091	1000
R	0.091	1001
D	0.091	101

- “HELLO WORLD”
  - 110 111 00 00 010 011 1000 010 1001 00 101
    H E L L O _ W O R L D
  - 32 bita naspram 88 bitova

Shannon-Fano

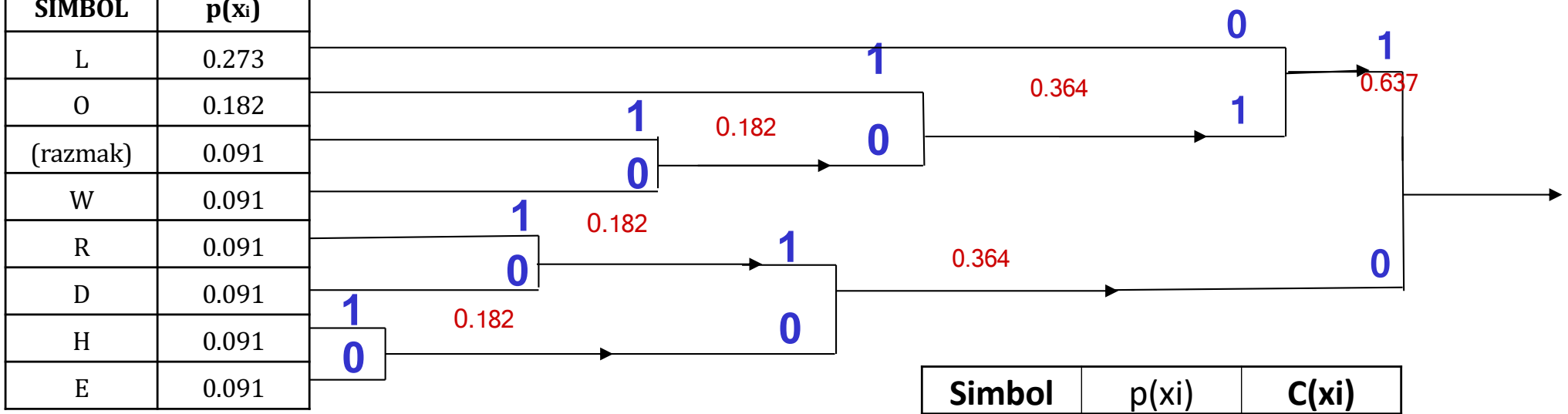
Simbol	p(xi)	**C(xi)**
L	0.273	00
O	0.182	010
E	0.091	111
H	0.091	110
razmak	0.091	011
W	0.091	1000
R	0.091	1001
D	0.091	101

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i = -(6 \cdot (0.091 \cdot \log 0.091) + 0.273 \cdot \log 0.273 + 0.182 \cdot \log 0.182 = 2.85 \left[\text{bit/simbol}\right]$$

$$L = \sum_{i=1}^{8} p_i l_i = 0.273 \cdot 2 + 4 \cdot (0.091 \cdot 3) + 2 \cdot (0.091 \cdot 4) + 0.182 \cdot 3 = 2.912 \left[\text{bit/simbol}\right]$$

Huffman

SIMBOL	$p(x_i)$
L	0.273
O	0.182
(razmak)	0.091
W	0.091
R	0.091
D	0.091
H	0.091
E	0.091



Simbol	p(xi)	C(xi)
L	0.273	10
O	0.182	111
E	0.091	000
H	0.091	001
razmak	0.091	1101
W	0.091	1100
R	0.091	011
D	0.091	010

Huffman

Simbol	p(xi)	C(xi)
L	0.273	10
O	0.182	111
E	0.091	000
H	0.091	001
razmak	0.091	1101
W	0.091	1100
R	0.091	011
D	0.091	010

- "HELLO WORLD"
  - SF: 110 111 00 00 010 011 1000 010 1001 00 101
    H E L L O _ W O R L D
  - H: 001 000 10 10 111 1101 1100 111 011 10 010

Huffman

Simbol	p(xi)	C(xi)
L	0.273	10
O	0.182	111
E	0.091	000
H	0.091	001
razmak	0.091	1101
W	0.091	1100
R	0.091	011
D	0.091	010

$$L = \sum_{i=1}^{8} p_i l_i = 0.273 \cdot 2 + 4 \cdot (0.091 \cdot 3) + 2 \cdot (0.091 \cdot 4) + 0.182 \cdot 3 = 2.912 \left[\text{bit/simbol}\right]$$

Primjer lošeg koda

Simbol	Vjerojatnost	Kodna riječ
a_1	0.95	1
a_2	0.02	01
a_3	0.03	00

- Entropija: 0.335 bit/simbol
- Prosječna duljina:1.05 bit/simbol: 213% više od entropije!!
- Problem: a_1 se često ponavlja – Huffman "nema rješenje"
- Ipak: kod je optimalan

Prošireni Huffmanov kod

PROŠIRENI 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

- Ideja: kombinacija više simbola
  - ukupno n^m simbola
  - n=3, m=2 -> 9 simbola
- Prošireni kod: L=1.222
  - dijelimo s m=2 = 0.611 bit/simbol: 72% više od entropije.
- Bolje je kodirati duže sekvence (blokove), ali tada broj kodnih riječi raste eksponencijalno
  - kompleksnije

Huffmanovo kodiranje: primjene

- Česta primjena unutar složenijih algoritama
- Primjeri:
  - standardi za telefaks (T.4, T.6)
  - standard za nepomičnu sliku JPEG
  - DEFLATE (ZIP)
  - mp3
- Jedan od razloga
  - “bolja” kodiranja - patenti

Aritmetičko kodiranje

- Autori Pasco & Rissanen (nezavisno), 1976. godine
- Djelomično princip proširenog Huffman kodiranja
- Algoritam uzima kao ulaz cijele nizove simbola (“poruke”) i preslikava ih na realne brojeve, ovisno o (**poznatim**!) statističkim svojstvima
- Patenti (IBM)

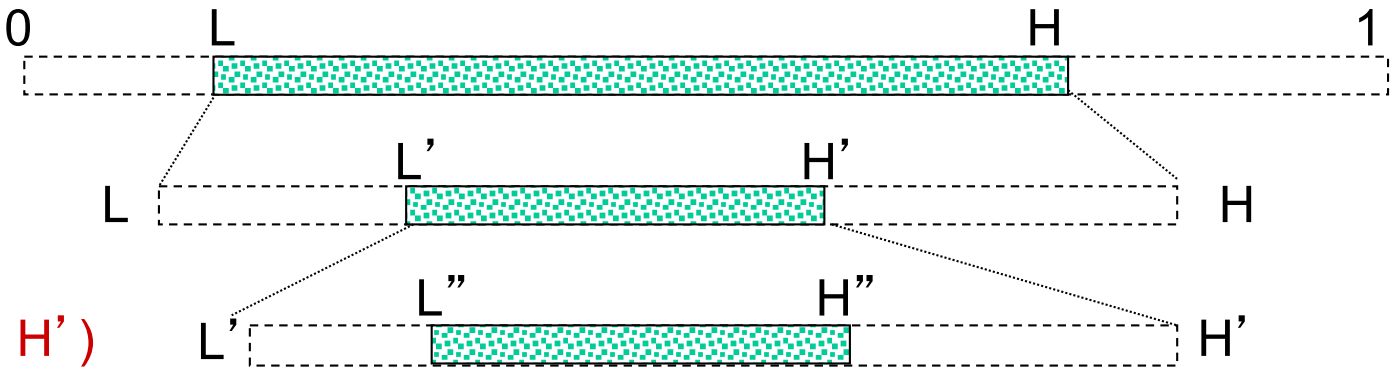
Aritmetičko kodiranje: postupak

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

početni interval [0, 1)

podinterval [L, H)

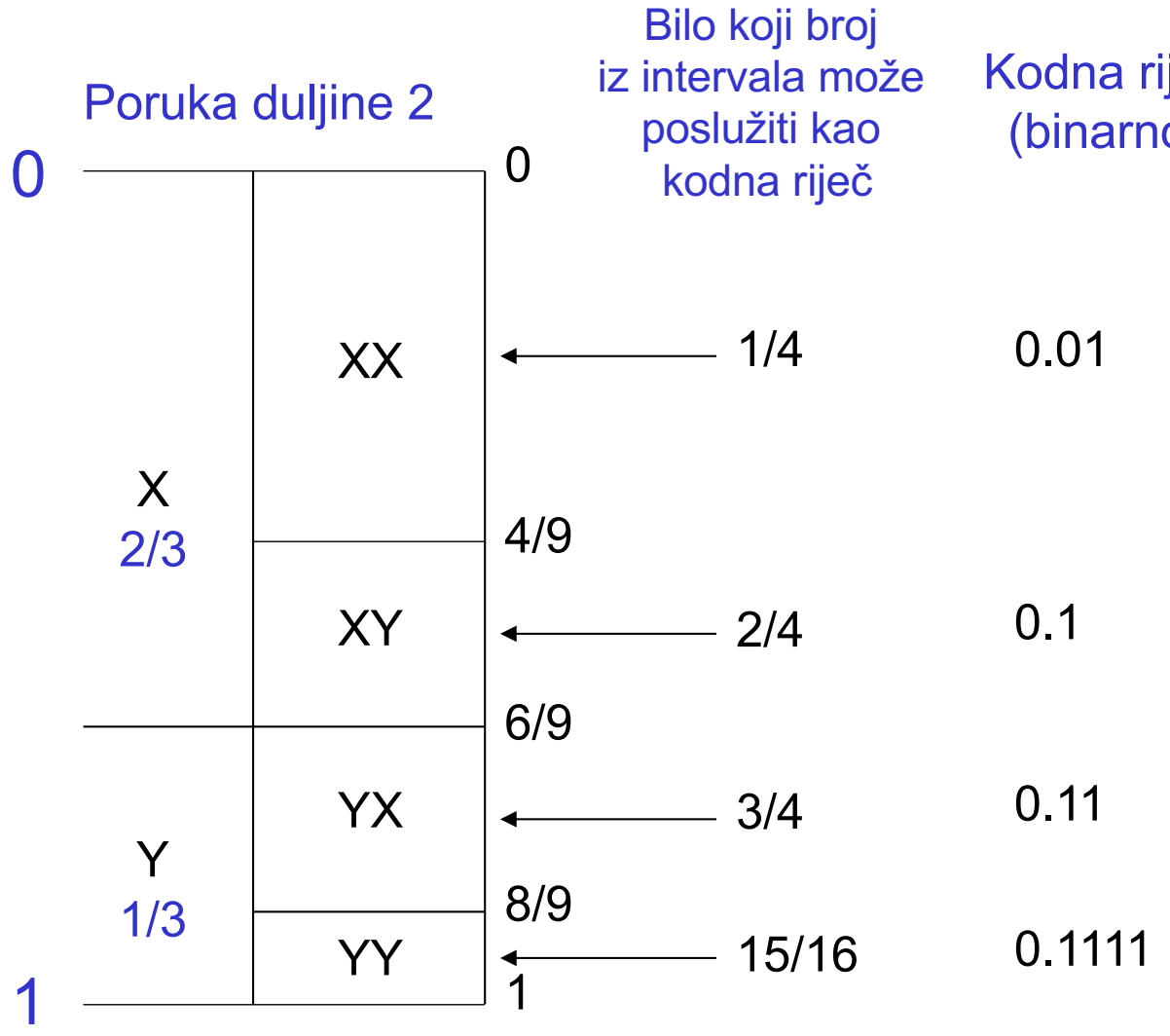
novi podinterval [L', H')



Aritmetičko kodiranje: primjer (1)

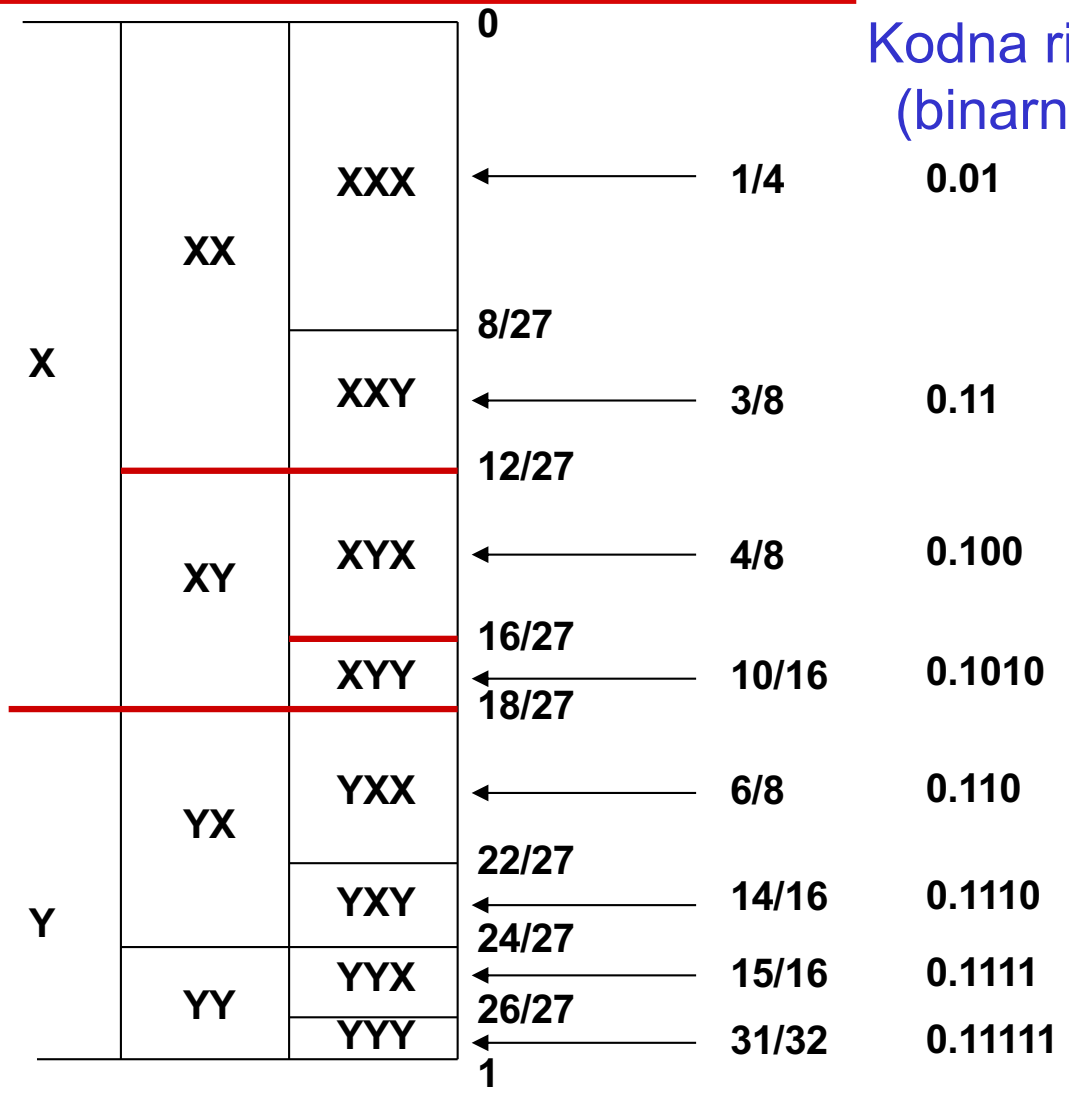
- M=2
- simboli: X, Y
  p(X) = 2/3
  p(Y) = 1/3

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



Postupak dekodiranja

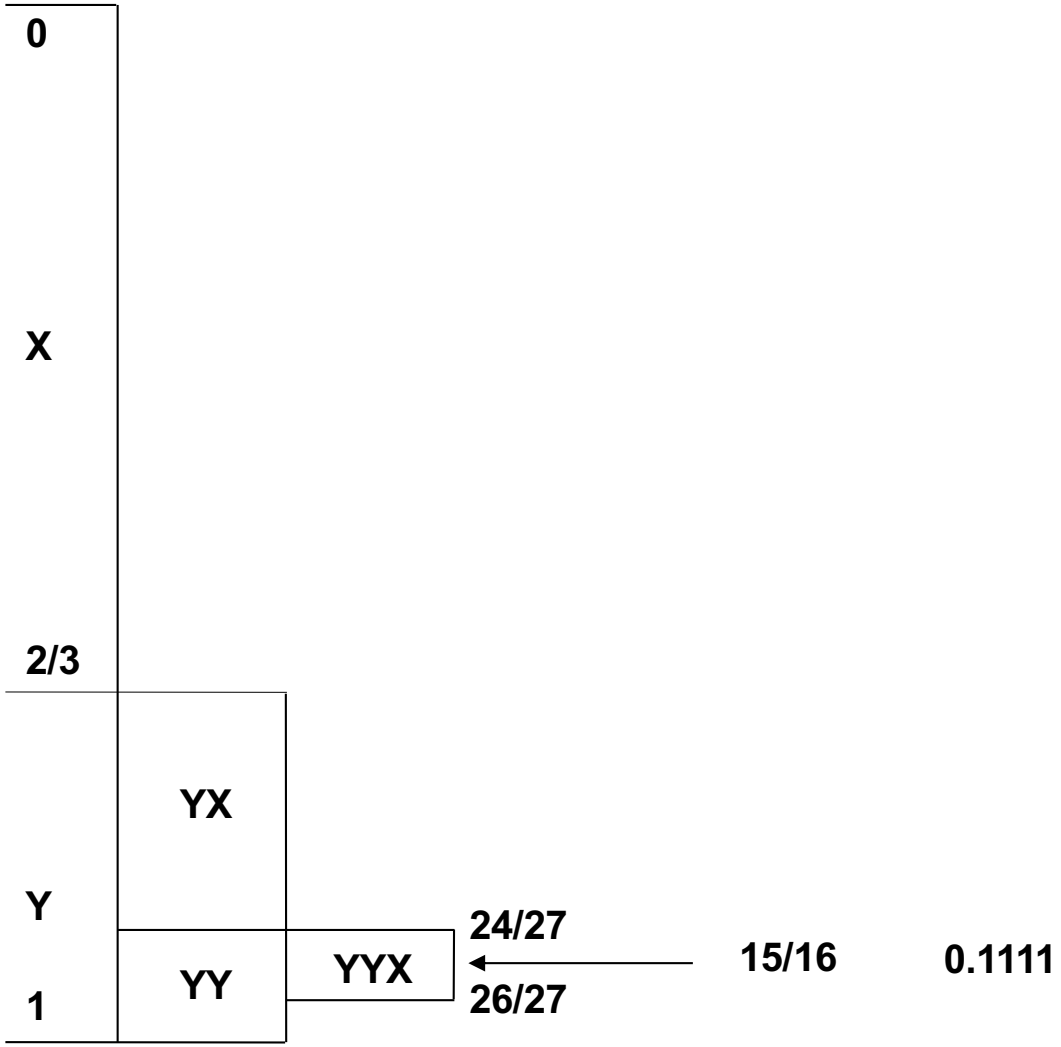
1. Podijeli početni interval [0, 1) u podintervale po vjerojatnostima pojavljivanja simbola
2. Uzmi primljeni kod kao realni broj
3. Pronađi podinterval u kojem se nalazi broj (kod)
4. Zapiši simbol koji odgovara tom podintervalu
5. Podijeli taj podinterval u n novih podintervala, proporcionalno vjerojatnostima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

Dekodiranje: primjer

- primjer za poruku duljine 3

- *M*=3
- 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:

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

- P(x) – duljina promatranog podintervala!
- Na ovakav način dobiva se uvijek prefiksni kod

Primjer – praktičniji postupak (zbirka!)

- Definiramo granice D (donja) i G (gornja)
- Prvi simbol -> D=0, G=1
- Za svaki sljedeći simbol računamo prema:

$$D' = D + (G - D)D_s$$
$$G' = D + (G - D)G_s$$

- D_s i G_s – donja i gornja granica pojedinog simbola

- Konačno: kada završimo sa svim simbolima poruke <D_s, G_s> predstavlja konačni interval

Implementacija

- Do sada opisani algoritam praktično 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 s cijelim brojevima
  - Koristi prikaz s 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 pomiču 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

$D' = D + (G\text{-}D)\cdot D_s$

$G' = D + (G\text{-}D)\cdot G_s$

- D' i G' – nove granice podintervala
- G-D - duljina promatranog podintervala
- Ds i Gs – donja i gornja granica kumulativnog podskupa za simbol

	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 može biti statičan, no najčešće je prilagodljiv

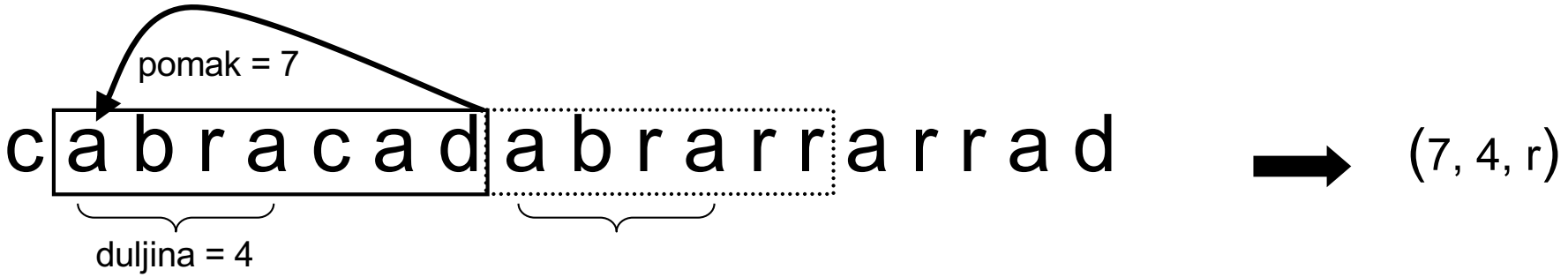
Metode s prilagodljivim rječnikom

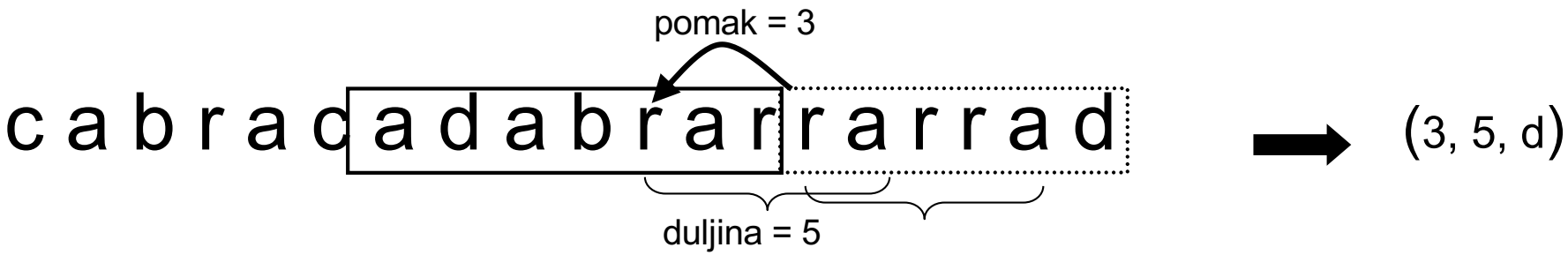
- Koder i dekoder dinamički grade rječnik
  - LZ77: Rječnik je posmični prozor
  - LZ78: riječi se grade dodavanjem slova na postojeće riječi (u početku rječnik je prazan)
  - Lempel-Ziv-Welch (LZW) algoritam
    - izvorni algoritam smislili Ziv i Lempel (1977 - LZ77, 1978 - LZ78), a Welch ga je doradio 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 posmičnom prozoru (*search buffer*) se traže simboli iz "sljedećeg" prozora za kodiranje (*look-ahead buffer*)
- 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





LZ77 – primjer (1/2)

- Koristeći algoritam LZ77 kodirajte poruku *aacaacabcabaaac** uzimajući pri tome da maksimalna duljina posmičnog prozora (PP) iznosi 4 simbola, a prozora za kodiranje (PZK) 6 simbola. Simbol "*" označava kraj poruke.

LZ77 – primjer (2/2)

- PP=4, PZK=6
- Postupak rješavanja:

		izlaz(x,y,z)
1. korak	a a c a a c a b c a b a a a c *	(0,0,a)
2. korak	a a c a a c a b c a b a a a c *	(1,1,c)
3. korak	a a c a a c a b c a b a a a c *	(3,4,b)
4. korak	a a c a a c a b c a b a a a c *	(3,3,a)
5. korak	a a c a a c a b c a b a a c *	(1,2,c)
6. korak	a a c a a c a b c a b a a a c *	(0,0,*)

- Kodirana poruka glasi:

(0,0,a) (1,1,c) (3,4,b) (3,3,a) (1,2,c) (0,0,*)

Algoritmi LZ78 i LZW (1/2)

- Umjesto posmičnog prozora, zasebna memorija za rječnik
  - Rječnik je poredana lista riječi (nizova simbola)
  - Riječ se dohvaća pomoću indeksa (rednog broja)
- Prednost nad LZ77?
  - LZ77 gradi rječnik a LZ78 radi s gotovim riječnikom
    - (koji napravi prilikom kodiranja)
  - LZ78 i LZW mogu dekodirati bilo koji simbol u nizu, LZ77 mora početi od početka niza!

Algoritmi LZ78 i LZW (2/2)

- 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
    - https://www.youtube.com/watch?v=3Bg8rN1R_kw#t=0m49s
- LZW
  - Kao LZ78 ali počinje s gotovim rječnikom od *n* simbola
    - Tipično 256 simbola u koje su uključena sva slova i znakovi (+ 4k)
    - https://www.youtube.com/watch?v=rZ-JRCPv_O8#t=0m34s

Algoritam LZW

- Algoritam kodiranja:

```
1.  RadnaRiječ = slijedeći simbol sa ulaza
2.  WHILE (ima još simbola na ulazu) DO
3.    NoviSimbol = slijedeći simbol sa ulaza
4.    IF RadnaRiječ+NoviSimbol postoji u rječniku THEN
5.       RadnaRiječ = RadnaRiječ+NoviSimbol
6.    ELSE
7.       IZLAZ: kod za RadnaRiječ
8.       dodaj RadnaRiječ+NoviSimbol u rječnik
9.       RadnaRiječ = NoviSimbol
10.   END IF
11. END WHILE
12. IZLAZ: kod za RadnaRiječ
```

https://www.youtube.com/watch?v=rZ-JRCPv_O8#t=0m34s

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

Primjer – LZW (1/5)

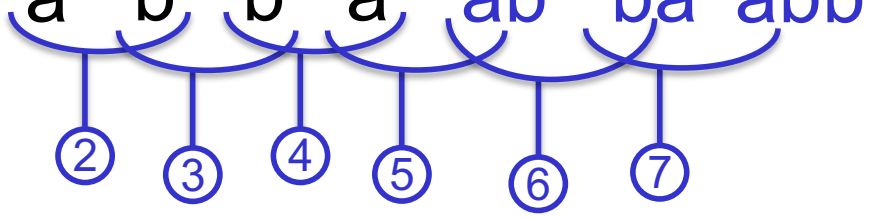
- Uzimajući polazni rječnik D gdje je D[0] = a i D[1] = b dekodirajte kodiranu poruku 0 1 1 0 2 4 6 kodiranu algoritmom LZW.

Primjer – LZW (2/5)

- dekodiramo prvi simbol koji je već poznat u rječniku, on ujedno postaje i radna riječ
- dekodiramo slijedeći simbol koji je poznat u rječniku i njega kombiniramo sa simbolom iz radne riječi
  - ako kombinacija ne postoji u rječniku zapisujemo ju u rječnik
  - ako kombinacija već postoji u rječniku kombiniramo ju sa sljedećim dekodiranim simbolom
- postupak ponavljamo dok ne dekodiramo cijelu poruku

Primjer – LZW (3/5)

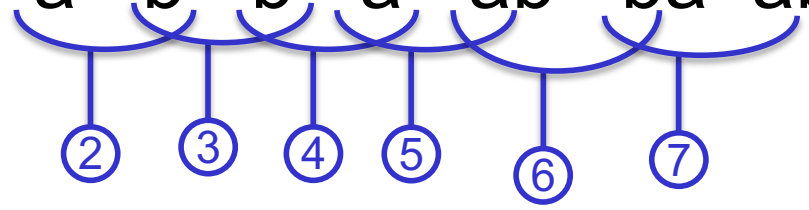
- Ulaz: 0 1 1 0 2 4 6
- Izlaz: a b b a ab ba abb



Rječnik
D[0] = a
D[1] = b
D[2] = ab
D[3] = bb
D[4] = ba
D[5] = aa
D[6] = abb
D[7] = baa

Primjer – LZW (4/5) – provjera kodiranjem

- Ulaz: a b b a ab ba abb



- Izlaz: 0 1 1 0 2 4 6

Rječnik
D[0] = a
D[1] = b
D[2] = ab
D[3] = bb
D[4] = ba
D[5] = aa
D[6] = abb
D[7] = baa

Primjer – LZW (5/5)

- Ulazna poruka: 0 1 1 0 2 4 6
- Dekodirana poruka: a b b a ab ba abb

Metode rječnika: primjene

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

Metode skraćivanja niza

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

  894000000000000000000000000000000

  → 894f32

  zastavica (flag) koja označuje nule

  broj ponavljanja

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

  ABCCCCCCCCC DEFFFABC...

  8 okteta 3 okteta

  ABC!8 DEFFFABC...

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

- Primjena: prva generacija telefaksa, unutar JPEG-a

Zavod za telekomunikacije

Teorija informacije

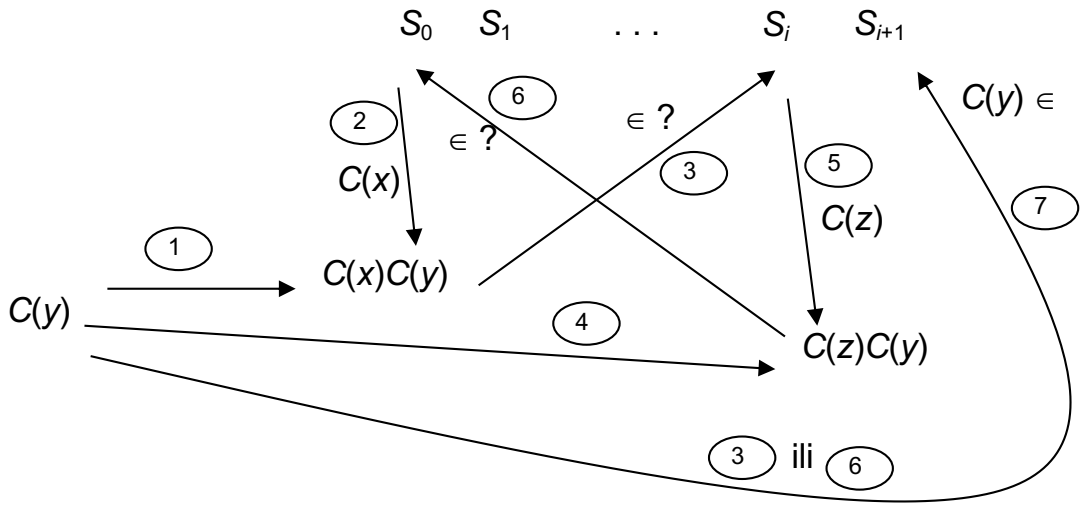
Entropijsko kodiranje – Sardinas-Pattersonov test

Sardinas-Pattersonov test

- skup pravila za određivanje jednoznačne dekodabilnosti koda:
  - neka je zadan kôd K čiji je skup polaznih kodnih riječi označen kao kup S_0
  - iz skupa S_i (i = 0, 1, ...) treba stvoriti skup S_{i+1}, pravilom:
    - kodna riječ $C(y)$ se dodaje u skup S_{i+1} ako i samo ako postoji kodna riječ $C(x)$ iz skupa S_0 takva da je kodna riječ $C(x)C(y)$ element skupa S_i ili
      - primjer: $C(y) = 0$, $C(x) = 11 \rightarrow C(x)C(y) = 110$
    - ako postoji kodna riječ $C(z)$ iz S_i takva da je $C(z)C(y)$ kodna riječ iz skupa S_0
      - napomena: u prvom koraku testa vrijedi $S_i = S_0$
  - kod je **jednoznačno dekodabilan** ako niti jedan skup S_i ($i \geq 1$) ne sadrži kodne riječi iz S_0 (osnovni uvjet!)

Sardinas-Pattersonov test (2)

- dodatak na prethodno pravilo:
  - ako je $S_{i+1} = \{\}$ ili za svaki sljedeći S_{i+1} vrijedi $S_{i+1} = S_i$, tada je jasno da u konačnom broju iteracija skup S_{i+1} neće sadržavati elemente iz skupa S_0, što također znači da je kod jednoznačno dekodabilan
    - primjer: ako skupovi S_1, S_2 i S_3 ne sadrže elemente iz S_0, a $S_4 = \{\}$ ili vrijedi $S_3 = S_4 = S_5 = ...$, kod je jednoznačno dekodabilan



Primjer

- zadano je pet kodova, *A*, *B*, *C*, *D* i *E*

Simboli	Kôd *A*	Kôd *B*	Kôd *C*	Kôd *D*	Kôd *E*
x_1	000	0	0	0	0
x_2	001	01	10	10	10
x_3	010	011	110	110	1100
x_4	011	0111	1110	1110	1101
x_5	100	01111	11110	1011	1110
x_6	101	011111	111110	1101	1111

- potrebno je odrediti da li su a) jednoznačno dekodabilni, b) prefiksni i c) nesingularni

Kôd *A*

- sve kodne riječi koda *A* su jednake duljine
- a) provjera jednoznačne dekodabilnosti
- polazni skup S_0 = {000, 001, 010, 011, 100, 101}
- potrebno je kreirati skup S_1 tako da svaka kodna riječ $C(y)$ iz tog skupa čini s nekom riječi $C(x)$ iz skupa S_0 kodnu riječ $C(x)C(y)$ koja također pripada skupu S_0
  - vrijedi S_1 = {}
  - skup je prazan jer su sve riječi koda *A* jednake duljine
  - sukladno pravilima Sardinas-Pattersonovog testa zaključak je da je kôd A **jednoznačno dekodabilan**

Kôd *A* (2)

- b) provjera prefiksnosti
  - pregledom kodnih riječi: niti jedna kodna riječ nije prefiks druge kodne riječi → kôd *A* je prefiksni
  - pomoću Kraftove nejednakosti
    - za kôd A vrijedi: $l_i = 3$, $\forall\, i \in \{1, ..., 6\}$

$$\sum_{i=1}^{n} d^{-l_i} = \sum_{i=1}^{6} 2^{-l_i} = 6 \cdot 2^{-3} = \frac{6}{8} \leq 1$$

  - dakle, kôd A je **prefiksni**
- c) provjera nesingularnosti
  - za sve simbole koda *A* vrijedi $\forall x_i, x_j, i \in \{1,\ldots,6\}, i \neq j \Rightarrow C(x_i) \neq C(x_j)$
    - za različite simbole x_i kôd *A* daje različite riječi, dakle, kôd *A* je **nesingularan**

Kôd *B*

- a) provjera jednoznačne dekodabilnosti
- $S_0 = \{0, 01, 011, 0111, 01111, 011111\}$
- $S_1 = \{1, 11, 111, 1111, 11111\}$
  - na primjer, u prvom koraku $S_0 = S_0$, $C(y) = 1$ i $C(x) = 0$ tvore $C(x)C(y) = 01$
- u drugom koraku tražimo S_2 takav da $C(y) \in S_2$
  - zajedno s $C(x)$ iz S_0 tvori kodnu riječ $C(x)C(y)$ iz S_1, ili
  - zajedno s $C(z)$ iz S_1 tvori kodnu riječ $C(z)C(y)$ iz S_0
  - vrijedi $S_2 = \{\}$
  - dakle, S_1 i S_2 ne sadržavaju kodne riječi iz $S_0 \rightarrow$ kôd *B* je **jednoznačno dekodabilan**

Kôd *B* (2)

- b) provjera prefiksnosti
  - pregledom kodnih riječi: svaka kodna riječ je prefiks sljedeće kodne riječi (0 – 01, 01 – 011, ...) → kôd *B* **nije prefiksni**
  - provjera: da li je s kodnim riječima čije su duljine jednake duljinama kodnih riječi koda *B* moguće konstruirati prefiksna kod?
    - $l_i = i,\ \forall i \in \{1,...,6\}$ $\sum_{i=1}^{6} 2^{-i} = 0,9844 \leq 1$
    - Kraftova nejednakost je zadovoljena, s kodnim riječima takvih duljina moguće je sastaviti prefiksni kod
- c) provjera nesingularnosti
  - kôd *B* je **nesingularan** jer različitim simbolima pridružuje različite kodne riječi

Kôd *C*

- a) provjera jednoznačne dekodabilnosti
- S_0 = {0, 10, 110, 1110, 11110, 111110}
- S_1 = {}
  - nema kodnih riječi koje zajedno s kodnim riječima iz S_0 tvore kodne riječi iz S_0
  - dakle, kôd *C* je **jednoznačno dekodabilan**
  - b) niti jedna kodna riječ nije prefiks druge kodne riječi → kôd *C* je **prefiksni**
  - sukladno kodu *B*, zadovoljena je Kraftova nejednakost zbog identičnih duljina kodnih riječi
- c) kôd *C* je **nesingularan** jer različitim simbolima pridružuje različite kodne riječi

Kôd *D*

- a) $S_0 = \{0, 10, 110, 1110, 1011, 1101\}$
- prvi korak:
  - tražimo kodne riječi koje zajedno s kodnim riječima iz S_0 tvore opet kodne riječi iz S_0: $S_1 = \{1,11\}$
    - skup S_1 ne sadrži simbole iz skupa S_0
- drugi korak:
  - tražimo kodne riječi koje zajedno s kodnim riječima iz S_0 tvore kodne riječi iz S_1: {}
  - ili kodne riječi koje zajedno s kodnim riječima iz S_1 tvore kodne riječi iz S_0: {0, 10, 01, 110, 011, 101} = S_2
  - skup S_2 sadrži simbole iz skupa S_0 (potcrtani) → kôd *D* **nije jednoznačno dekodabilan**

Kôd *D* (2)

- primjer sljedova simbola koji će kreirati jednake poruke:
  - $x_2x_3 \rightarrow 10110$ i $x_5x_1 \rightarrow 10110$
- b) kodna riječ 110 je prefiks kodnoj riječi 1101
  - kôd *D* **nije prefiksni kôd**
- nije zadovoljena Kraftova nejednakost

$$\sum_{i=1}^{6} 2^{-l_i} = 1,0625 > 1$$

- sa zadanim duljinama kodnih riječi nije moguće kreirati prefiksni kod!
- c) kôd *D* je **nesingularan**

Kôd *E*

- a) S_0 = {0, 10, 1100, 1101, 1110, 1111}
- pri korak:
  - tražimo kodne riječi koje zajedno s kodnim riječima iz S_0 tvore opet kodne riječi iz S_0: S_1 = {}
    - kôd *E* je **jednoznačno dekodabilan**
    - b) niti jedna kodna riječ nije prefiks druge kodne riječi → kôd *D* je **prefiksni kôd**
- Kraftova nejednakost je zadovoljena $\sum_{i=1}^{6} 2^{-l_i} = 1 \leq 1$
- c) kôd *E* je **nesingularan**