Margareta Klinčić

Dodatni zadaci

Zadatak-1: *(Kraftova nejednakost) Dan je skup simbola sa pripadnim vjerojatnostima:*

$$S = \begin{pmatrix} S_1 & S_2 & \cdots & S_m \\ p_1 & p_2 & \cdots & p_m \end{pmatrix}$$

Simboli su jednoznačno kodirani prefiksnim kodom. Ako je m = 6 i ako su duljine kodnih riječi (l_1, l_2, …, l_6) = (1, 1, 2, 3, 2, 3), odredite donju granicu za broj znakova abecede prefiksnog koda d.
Izvor: *Thomas M. Cover, Joy A. Thomas: Elements Of Information Theory, Second Edition*

Rješenje:
Za svaki prefiksni kod s abecedom od d simbola (baza koda) i duljinama kodnih riječi l_1, l_2, …, l_i, …, l_n *vrijedi Kraftova nejednakost:*

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

Nakon uvrštavanja dobivamo:

$$\frac{1}{d} + \frac{1}{d^2} + \frac{1}{d^3} \leq \frac{1}{2}$$

Vidimo da je najmanji broj koji zadovoljava danu nejednakost 3, tj. $d_{min} = 3$.

***

Zadatak-2: *(Shannon-Fano) Prema danoj tablici koja opisuje izvorište, kodirajte simbole Shannon-Fanoovim kodom te dekodirajte niz* '00011111011111000'.
Izvor: *vlastiti primjer*

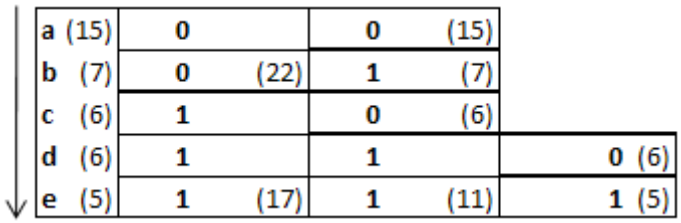
Simbol	*Frekvencija*
a	*15*
b	*7*
c	*6*
d	*6*
e	*5*

Rješenje:

Prema sljedećem algoritmu:

1. Poredamo simbole prema padajućoj vjerojatnosti.
2. Podijelimo dobivenu listu simbola na dva dijela tako da razlika suma vjerojatnosti pojedinih dijelova bude minimalna.
3. Prvoj polovici liste pridijelimo binarnu znamenku 0, a drugoj 1.
4. Rekurzivno primjenjujemo korake 2 i 3 na obje polovice.

dobivamo:



Sa slike očitavamo:

Simbol	p_i	***Kodna riječ***	l_i
a	15	00	2
b	7	01	2
c	6	10	2
d	6	110	3
e	5	111	3

Prosječna duljina kodne riječi je $L = \sum_{i=1}^{5} p_i l_i = 2.2821$.

Shannon-Fanoov kod spada u skupinu prefiksnih kodova koji imaju sljedeće svojstvo:
Niti jedna kraća kodna riječ ne pojavljuje se kao prefiks duže kodne riječi.
Postupak dekodiranja je, dakle, jednoznačan:

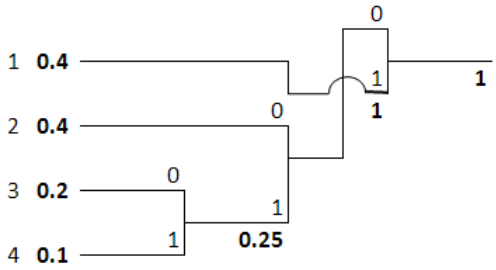
'00011111011111000' → '00 01 111 10 111 110 00' → 'abeceda'

***

Zadatak-3: *(Huffman) Izvorište generira četiri različita simbola iz skupa simbola {1, 2, 3, 4} s vjerojatnostima pojavljivanja (0.4, 0.35, 0.2, 0.05) respektivno. Kodirajte optimalnim binarnim kodom koristeći Huffmanov postupak kodiranja!*
Izvor: http://en.wikipedia.org/wiki/Huffman_coding

Rješenje:

Binarno stablo gradimo s lijeva na desno i to tako da simbole prvo poredamo prema vjerojatnosti pojavljivanja, a zatim odaberemo dva simbola sa najmanjom vjerojatnošću. Ta dva simbola spojimo tako da dobijemo jedan simbol sa vjerojatnošću pojavljivanja jednakoj sumi vjerojatnosti pojedinog simbola. Postupak se ponavlja sve dok ne preostane samo jedan simbol. Stablo tada čitamo unazad, tj. s desna na lijevo pridružujući granama stabla određeni bit.



Iz grafa sa slike čitamo:

Simbol	p_i	***Kodna riječ***	l_i
1	0.4	1	1
2	0.35	00	2
3	0.2	010	3
4	0.05	011	3

Obično se signal predstavljen sa četiri simbola kodira s 2 bita po simbolu. Entropija izvora ovdje je $H(X) = \sum_{i=1}^{4} p_i \log_2 p_i = 1.7394\ bit/simbol$. Ukoliko se za kodiranje koristi Huffman, prosječna duljina kodne riječi je $L = \sum_{i=1}^{4} p_i l_i = 1.85\ bit/simbol$ što se još uvijek razlikuje od teoretske granice $(\boldsymbol{H(X)} \leq L \leq H(X) + 1)$ iz razloga što su vjerojatnosti simbola različite od negativnih potencija broja dva. Npr., ako bi vjerojatnosti pojavljivanja simbola bile {0.5, 0.25, 0.125, 0.125} respektivno, prosječna duljina kodne riječi izjednačava se s donjom granicom. Simbole kodiramo na sljedeći način:

Simbol	p_i	***Kodna riječ***	l_i
1	0.5	0	1
2	0.25	10	2
3	0.125	110	3
4	0.125	111	3

Entropija i prosječna duljina kodne riječi tada su jednake i iznose $1.75\ bit/simbol$.

***

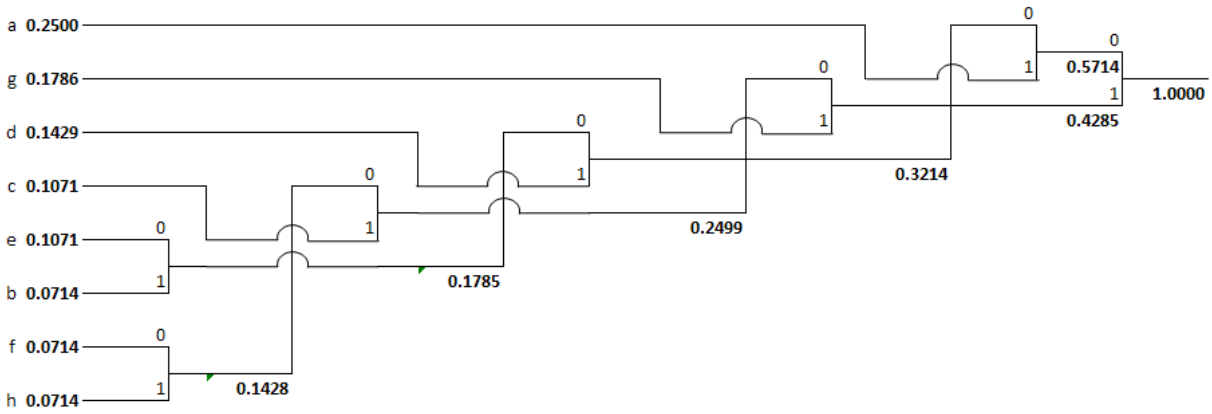
Zadatak-4: *(Huffman) Izvorište generira osam različitih simbola iz skupa simbola {a, b, c, d , e, f, g, h}. Generiran je niz* `'cadgghedefebaabcdaaafacdghgg'`. *Kodirajte optimalnim binarnim kodom koristeći Huffmanov postupak kodiranja te usporedite efikasnost kodiranja u odnosu na kodiranje ravnomjernim binarnim kodom!*
Izvor: *http://thalia.spec.gmu.edu/~pparis/classes/notes_101/node35.html (modficiran)*

Rješenje:

Računamo vjerojatnosti pojavljivanja simbola:

Simbol	p_i
a	7/28 = 0.2500
b	2/28 = 0.0714
c	3/28 = 0.1071
d	4/28 = 0.1429
e	3/28 = 0.1071
f	2/28 = 0.0714
g	5/28 = 0.1786
h	2/28 = 0.0714

Simbole kodiramo slično kao u prethodnom zadatku:



Iz grafa sa slike čitamo:

Simbol	p_i	***Kodna riječ***	l_i
a	0.2500	01	2
b	0.0714	0001	4
c	0.1071	101	3
d	0.1429	001	3
e	0.1071	0000	4
f	0.0714	1000	4
g	0.1786	11	2
h	0.0714	1001	4

Kodiramo li simbole ravnomjernim binarnim kodom dobivamo:

Simbol	p_i	***Kodna riječ***	l_i
a	0.2500	000	3
b	0.0714	001	3
c	0.1071	010	3

d	0.1429	011	3
e	0.1071	100	3
f	0.0714	101	3
g	0.1786	110	3
h	0.0714	111	3

Prosječna duljina Huffmanove kodne riječi je $L = \sum_{i=1}^{8} p_i l_i = 2.8929$, dok ta ista veličina u drugom slučaju iznosi 3. Za ilustraciju, promatramo li niz od 100 simbola koji kodiramo ravnomjernim binarnim kodom (kod s fiksnom duljinom kodne riječi) bit će nam potrebno $100 \times 3 = 300$ simbola. Kod Huffmanovog koda, za kodiranje zadanog niza od 100 simbola, potrebno nam je $25 \times 2 + 7 \times 4 + 11 \times 3 + 14 \times 3 + 11 \times 4 + 7 \times 4 + 18 \times 2 + 7 \times 4 = 289$ simbola. Imamo, dakle uštedu od $^{11}/_{300} = 3.6\%$. Ovisno o vrsti izvorišta i korištenom kodu, ušteda može biti znatno veća (za primjer vidjeti sljedeći zadatak). Nadalje, usporedimo li efikasnosti kodova $\mathcal{E}_B = {}^{H(X)}/_{L}$, za

Huffmanov dobivamo $\mathcal{E}_B = 0.9856 = 98.56\%$, a za ravnomjerni binarni kod $\mathcal{E}_B = 0.9504 = 95.04\%$.

***

Zadatak-5: *(Huffman) Izvorište generira simbole iz skupa od osam različitih simbola {a, b, c, d, e, f, g, h} s vjerojatnostima pojavljivanja (3, 2, 2, 1, 1, 1, 1, 1). Simboli se kodiraju Huffmanovim kodom. Tablica kodiranja nalazi se u nastavku. Odredite postotak uštede u odnosu na kodiranje istog skupa simbola s istim vjerojatnostima kodom fiksne duljine – ASCII 8-bitnim kodom.*
Izvor: *vlastiti primjer*

Simbol	p_i	***Kodna riječ***	l_i
a	3	10	2
b	2	001	3
c	2	010	3
d	1	011	3
e	1	110	3
f	1	111	3
g	1	0000	4
h	1	0001	4

Rješenje:

Kodiramo li simbole navedenim Huffmanovim kodom, za proizvoljan niz od 12 simbola potrebno nam je $3 \times 2 + 2 \times 2 \times 3 + 3 \times 1 \times 3 + 2 \times 1 \times 4 = 35$ bita. Za kodiranje tog istog niza znakova ASCII 8-bitnim kodom potrebno nam je $12 \times 8 = 96$bita. Upotrebom Huffmanovog koda postižemo uštedu od $^{96-35}/_{96} = 63.54\%$.

***

Zadatak-6: *(Lempel-Ziv-Welch) Uz polazni rječnik naveden u nastavku kodirajte poruku* 'TOBEORNOTTOBEORTOBEORNOT#' *LZW algoritmom.*
Izvor: *http://en.wikipedia.org/wiki/LZW*

D[0]=#
D[1]=A
D[2]=B
D[3]=C
…
D[26]=Z

Rješenje:

Razlog učestalog ponavljanja slova i parova slova u ovom primjeru jest taj da bi ipak postigli određenu razinu kompresije na ovako kratkom nizu. Kada se LZW algoritam koristi za kodiranje i kompresiju pravih podataka, ponavljanje je rjeđe, tako da početni dijelovi poruke nemaju velik udio u komprimiranju. Međutim, daljnjim kodiranjem razina kompresije raste sa duljinom poruke.
Napomena: **T** – radna riječ
o – novi simbol

Ulaz	***Izlaz***	***Proširenje rječnika***
TOBEORNOTTOBEORTOBEORNOT#	20	D[27] = TO
OBEORNOTTOBEORTOBEORNOT#	15	D[28] = OB
BEORNOTTOBEORTOBEORNOT#	2	D[29] = BE
EORNOTTOBEORTOBEORNOT#	5	D[30] = EO
ORNOTTOBEORTOBEORNOT#	15	D[31] = OR
RNOTTOBEORTOBEORNOT#	18	D[32] = RN
NOTTOBEORTOBEORNOT#	14	D[33] = NO
OTTOBEORTOBEORNOT#	15	D[34] = OT
TTOBEORTOBEORNOT#	20	D[35] = TT
TOBEORTOBEORNOT#	27	D[36] = TOB
BEORTOBEORNOT#	29	D[37] = BEO
ORTOBEORNOT#	31	D[38] = ORT
TOBEORNOT#	36	D[39] = TOBE
EORNOT#	30	D[40] = EOR
RNOT#	32	D[41] = RNO
OT#	34	D[42] = OT#
#	0	

Kodirana poruka glasi: 20 15 2 5 15 18 14 15 20 27 29 31 36 30 32 34 0.

***

Zadatak-7: *(Lempel-Ziv-Welch) Uz polazni rječnik naveden u nastavku dekodirajte kodiranu poruku '*`20 15 2 5 15 18 14 15 20 27 29 31 36 30 32 34 0`*' LZW algoritmom.*
Izvor: *http://en.wikipedia.org/wiki/LZW*

D[0]=#
D[1]=A
D[2]=B
D[3]=C
…
D[26]=Z

Rješenje:

Sve što nam je potrebno za dekodiranje je polazni rječnik, proširenja ionako rekonstruiramo sami prolazeći kroz postupak dekodiranja. Dekodiranje je kod LZW algoritma vrlo slično kodiranju.

Ulaz	***Izlaz***	***Proširenje rječnika***
20 15 2 5 15 18 14 15 20 27 29 31 36 30 32 34 0	T	
15 2 5 15 18 14 15 20 27 29 31 36 30 32 34 0	O	D[27] = TO
2 5 15 18 14 15 20 27 29 31 36 30 32 34 0	B	D[28] = OB
5 15 18 14 15 20 27 29 31 36 30 32 34 0	E	D[29] = BE
15 18 14 15 20 27 29 31 36 30 32 34 0	O	D[30] = EO
18 14 15 20 27 29 31 36 30 32 34 0	R	D[31] = OR
14 15 20 27 29 31 36 30 32 34 0	N	D[32] = RN
15 20 27 29 31 36 30 32 34 0	O	D[33] = NO
20 27 29 31 36 30 32 34 0	T	D[34] = OT
27 29 31 36 30 32 34 0	TO	D[35] = TT
29 31 36 30 32 34 0	BE	D[36] = TOB
31 36 30 32 34 0	OR	D[37] = BEO
36 30 32 34 0	TOB	D[38] = ORT
30 32 34 0	EO	D[39] = TOBE
32 34 0	RN	D[40] = EOR
34 0	OT	D[41] = RNO
0	#	

Poslana poruka je: '`TOBEORNOTTOBEORTOBEORNOT#`'.

***

Zadatak-8: *(Lempel-Ziv '77) Koristeći algoritam LZ77 kodirajte poruku* '`abrakadabrakadabrak`'. *Maksimalna duljina posmičnog prozora je 7, a prozora za kodiranje 6 simbola.*
Izvor: *http://www.binaryessence.com/dct/en000138.htm (modificiran)*

Rješenje:

Algoritam nosi ime *LZ77* prema dvojici autora (Abraham Lempel i Jacob Ziv) te godini objavljivanja (1977.). LZ77 postiže kompresiju tako da za dio poruke koji je potrebno kodirati postavi pokazivač na podatke koji su već prošli kroz koder, odnosno dekoder uz uvjet da se ti podaci u određenoj mjeri poklapaju. U tu svrhu koriste se varijable *pomakUnazad*, *duljina* i *sljedeciSimbol*. *PomakUnazad* nam govori koliko se trebamo vratiti unazad da bismo pronašli niz koji se poklapa sa trenutnim nizom koji želimo kodirati, *duljina* koliko je taj niz dugačak, a *sljedeciSimbol* koji simbol dolazi nakon tog niza. Posmični prozor i prozor za kodiranje daju nam ograničenja koliko daleko unazad, odnosno unaprijed možemo gledati. Ako ne možemo pronaći podudaran niz, *pomakUnazad* i *duljinu* postavljamo na 0, a *sljedeciSimbol* postaje simbol na ulazu.

***Napomena:** (x, y, z)
x – pomakUnazad
y – duljina
z – sljedeciSimbol

Kodiranjem poruke '`abrakadabrakadabrak`' algoritmom LZ77 uz zadane parametre dobivamo:
`(0,0,a) (0,0,b) (0,0,r) (3,1,k) (2,1,d) (7,6,d) (7,5,kraj).`

***

Zadatak-9: *(Lempel-Ziv '77) Dekodirajte kodiranu poruku* `'(0,0,a) (0,0,b) (0,0,r) (3,1,k) (2,1,d) (7,6,d) (7,5,kraj)'` *koristeći algoritam LZ77.*
Izvor: *vlastiti primjer*

Rješenje:

Značenje varijabli isto je kao u prethodnom zadatku i sve što trebamo učiniti je pravilno ih pročitati. Tako dobivamo:

Ulaz	***Izlaz***
(0,0,a) (0,0,b) (0,0,r) (3,1,k) (2,1,d) (7,6,d) (7,5,kraj)	a
(0,0,b) (0,0,r) (3,1,k) (2,1,d) (7,6,d) (7,5,kraj)	ab
(0,0,r) (3,1,k) (2,1,d) (7,6,d) (7,5,kraj)	abr
(3,1,k) (2,1,d) (7,6,d) (7,5,kraj)	abr ak
(2,1,d) (7,6,d) (7,5,kraj)	abrak ad
(7,6,d) (7,5,kraj)	abrakad abrakad
(7,5,kraj)	abrakadabrakad abrak
	abrakadabrakadabrak

Poslana poruka je '`abrakadabrakadabrak`'.

***

Zadatak-10: *(Aritmetičko kodiranje) Dan je skup simbola X = {a, e, i, o, u, !} s vjerojatnostima pojavljivanja (0.2, 0.3, 0.1, 0.2, 0.1, 0.1) respektivno. Optimalno kodirajte poruku* `'eaii!'` *koristeći aritmetičko kodiranje.*
Izvor: *http://www.stanford.edu/class/ee398a/handouts/papers/WittenACM87ArithmCoding.pdf*

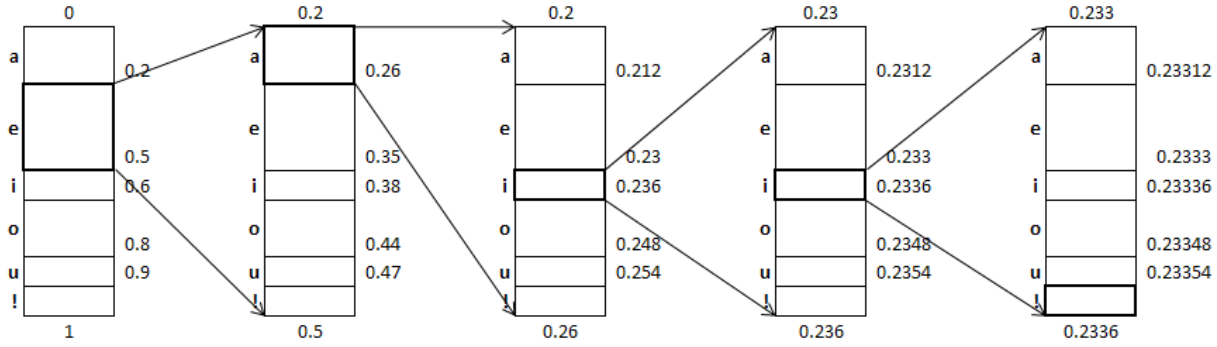
Rješenje:

Simbol: xi	***p(xi)***	***[Ds, Gs)***
a	0.2	[0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
!	0.1	[0.9, 1)

Inicijalno, interval kodera je [0, 1). Nakon prepoznavanja prvog simbola poruke (e) ograničavamo interval na [0.2, 0.5). Nadalje, nakon nailaska drugog simbola (a), novi interval sužujemo na njegovu prvu petinu, [0.2, 0.26), itd.. Formalno, intervale računamo na sljedeći način:

$$D' = D + (G - D)D_S$$
$$G' = D + (G - D)G_S$$

Primijenimo li gore navedeni postupak, dobivamo:



Dakle, rješenje je bilo koji broj iz intervala [0.23354, 0.2336).

***

Zadatak-11: *(Aritmetičko kodiranje) Dan je skup simbola X = {a, e, i, o, u, !} s vjerojatnostima pojavljivanja (0.2, 0.3, 0.1, 0.2, 0.1, 0.1) respektivno. Ako znate da simbol ! predstavlja kraj poruke, dekodirajte poruku kodiranu aritmetičkim kodom:* '`0.23354321`'.
Izvor: *vlastiti primjer*

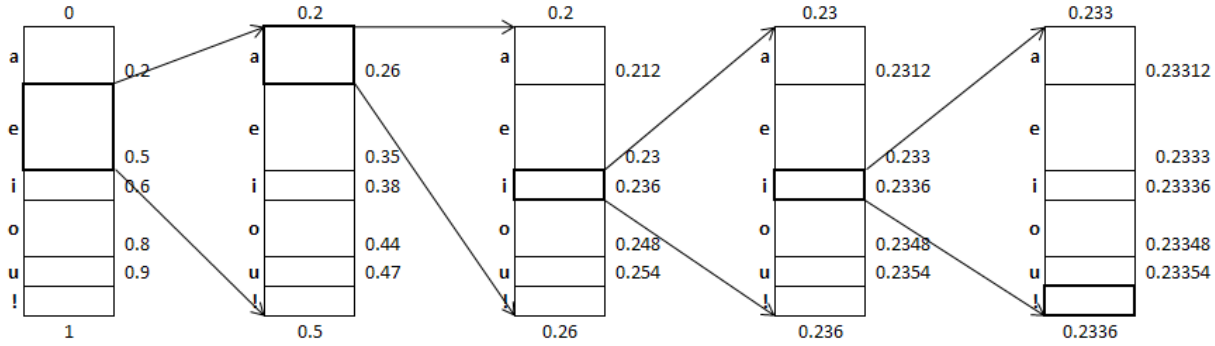
Rješenje:

Simbol: xi	***p(xi)***	***[Ds, Gs)***
a	0.2	[0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
!	0.1	[0.9, 1)

Pristupamo postupku dekodiranja koji se provodi tako da za svaki simbol računamo pripadni interval te provjeravamo kojem intervalu kodirana poruka pripada. S dekodiranjem stanemo kada naiđemo na oznaku kraja poruke (!). Naime, da bi dekoder mogao jednoznačno dekodirati poruku, uz bilo koji broj iz intervala koji je odredio koder, dekoder mora unaprijed znati i duljinu originalne poruke ili pak oznaku kraja poruke. Pripadni intervali računaju se na sljedeći način:

$$D' = D + (G - D)D_S$$
$$G' = D + (G - D)G_S$$

Tako dobivamo:



Poslana poruka je '`eaii!`'.

***