4.6.2 Prilagodljivo binarno zaporedno kodiranje

Metodo prilagodljivega binarnega zaporednega kodiranja (angl Binary Adaptive Sequential Coding, BASC) sta razvila Moffat in Anh [26] in je namenjena za zapis nenegativnih celoštevilskih vrednosti. Metoda omogoča sprotno prilagajanje potrebnega števila bitov b glede na predhodnji simbol. Začetno vrednost b vstavi uporabnik. Za trenutno celo število x_i izračunamo število bitov b', potrebnih za predstavitev x_i. Obravnavamo dve možnosti:

- $b' \leq b$: kodirnik v tem primeru zapiše 0, ki ji sledi binarna predstavitev števila x_i, zapisana kot $b-$bitno število.
- $b' > b$: kodirnik zapiše $(b' - b)$ enic, ki jim sledi ničla, tej pa $b' - 1$ manj pomembnih bitov števila x_i (najpomembnejši bit x_i mora biti 1, zato ga lahko izpustimo).

Po vsakem koraku postavimo $b' = b$. Oglejmo si primer. Imejmo zaporedje celih nenegativnih števil (15,6,2,3,0,0,0,0,4,5,1,7,8) in naj bo $b = 5$. Prvo število 15 lahko zapišemo s 4-rimi biti, zato je $b' = 4$, kar je manj kot b. Kodirnik zapiše 0, ki ji sledi binarni zapis števila 15 z $b = 5$ biti. Dobimo torej 0|01111, b pa dobi vrednost 4. Naslednje število je 6, $b' = 3$ in dobimo 0|0110 ter postavimo $b = 3$. Zatem kodiramo število 2 kot 0|010 in postavimo $b = 2$. Število 3 zatem zakodiramo kot 0|11, b pa ostane 2. Prvo izmed štirih ničel zakodiramo kot 0|00 in postavimo $b = 0$. Preostale ničle zakodiramo samo s po enim bitom 0. Naslednje število je 4, zato je $b' = 3 > b$. Kodirnik zapiše $b' - b = 3$ enice, ki ji sledi 0, tej pa dva najmanj pomembna bita števila 4, to je 00. Dobimo torej 111|0|00, b pa postavimo na 3. Naslednje število je 5, ki ga lahko zapišemo z $b = 3$ biti. Kodirnik torej zapiše 0|101 ter tako nadaljuje do konca.

4.6.3 Interpolativno kodiranje

Interpolativno kodiranje je predlagal Moffat s sodelavci [27, 28]. Tudi ta metoda priredi kode s spremenljivo dolžino posameznim simbolom, a ta dolžina je dinamična in je namesto od verjetnosti simbola odvisna od celotnega sporočila. Takšnemu pristopu pravimo tudi holističen[5] [16].

Kodiranje ne poteka zaporedno od začetka proti koncu ampak, v splošnem, po posebnem, vnaprej določenem, vrstnem redu. Koda simbola, ki ga kodiramo, je zato mnogo bolj odvisna od njegovega položaja v zaporedju

[5] holism je grška beseda, ki pomeni popolnost, celota.

kot od same vrednosti. Posledica tega je, da lahko včasih enolično uganemo kodiran simbol (ali celo daljše njih), zato takšen simbol kodiramo z nič biti.

Metodo razložimo na primeru. Sporočilo S naj ima 12 znakov: $S = c, b, b, a, a, a, c, a, a, c, c, c$. Abeceda S ima samo tri znake, $\Sigma(S) = \{a, b, c\}$. Vsakemu izmed znakov priredimo zaporedno celo število, prvi znak pa dobi vrednost 1. Naj velja: $a = 1$, $b = 2$ in $c = 3$. Dobimo $N = 3, 2, 2, 1, 1, 1, 3, 1, 1, 3, 3, 3$. Iz N tvorimo polje kumulativnih vrednosti monotono rastočega zaporedja $Q = 3, 5, 7, 8, 9, 10, 13, 14, 15, 18, 21, 24$. Metoda rekurzivno deli polje Q, zato potrebujemo spremenljivki L in H, ki določata spodnjo in zgornjo mejo zaporedja. Položaj simbola m, ki ga kodiramo, je najlažje izbrati na sredini kot

$$m = \left\lfloor \frac{L + H}{2} \right\rfloor . \tag{4.15}$$

Če imamo znani vrednosti L in H, lahko enostavno zakodiramo vrednost $Q[m]$. Najprej določimo interval vrednosti $[r_L, r_H]$, ki jih lahko ima vrednost $Q[m]$. Razmišljajmo takole: vrednost elementa v $Q[L]$ nam je znana. Ker je Q monotono naraščajoče zaporedje, dobimo najmanjšo možno vrednost v $Q[m]$ tako, da prištejemo toliko enic h $Q[L]$, kot je razdalja med $m - L$. Spodnjo vrednost izračunamo torej kot:

$$r_L = Q[L] + (m - L). \tag{4.16}$$

S podobnim razmislekom določimo tudi zgornjo mejo r_H. Največjo vrednost elementa $Q[m]$ dobimo, če se vse vrednosti od $Q[H]$ zmanjšujejo za 1. Tako dobimo zgornjo mejo kot:

$$r_H = Q[H] - (H - m). \tag{4.17}$$

S podatkom o intervalu lahko nato zakodiramo dejansko vrednost $Q[m] \in [r_L, r_H]$. Najprej določimo število bitov, potrebnih za kodiranje razlike vseh vrednosti v intervalu $[r_L, r_H]$, kot:

$$b = \lceil \log_2(r_H - r_L + 1) \rceil . \tag{4.18}$$

Vrednost $Q[m]$ nato zakodiramo z b biti. Najnižja možna vrednost je seveda L, ki je kodirana kot 0 z b-biti. Za prikaz delovanja postopka smo vhodno polje Q podali v razpredelnici 4.24, opremljeni tudi z indeksom polja i.

Razpredelnica 4.24: Vrednosti v polju Q

i	1	2	3	4	5	6	7	8	9	10	11	12
$Q[i]$	3	5	7	8	9	10	13	14	15	18	21	24

Na začetku je $L = 1$ in $H = 12$. Z enačbo 4.15 izračunamo $m = 6$, z enačbama 4.16 in 4.17 pa $r_L = 3 + (6 - 1) = 8$ in $r_H = 18$, kar nam da interval vseh možnih vrednosti za $Q[m] \in [8, 18]$. Teh vrednosti je 11, ki jih moremo zakodirati s štirimi biti (enačba 4.18). Vrednost $8_{(10)}$ bomo tako kodirali kot $0000_{(2)}$, $9_{(10)}$ kot $0001_{(2)}$, $10_{(10)}$ kot $0010_{(2)}$ vse do vrednosti $18_{(10)}$, ki jo zakodiramo kot $1010_{(2)}$. V našem primeru je $Q[6] = 10_{(10)}$, torej je iskana koda $0010_{(2)}$. Postopek kodiranja za naš primer prikažemo v razpredelnici 4.25. Vidimo, da v treh primerih kode nismo določili. V primeru, ko sta $L = 3$ in $H = 6$, sta vrednosti $Q[3] = 7$ in $Q[6] = 10$. Ker je Q monotono naraščajoč, vemo, da se na mestih $Q[4]$ in $Q[5]$ lahko nahajata samo 8 in 9. To formalno ugotovimo tako, da preverimo, ali je $Q[H] - Q[L] = H - L$. Takšno kontrolo bo lahko opravil tudi dekodirnik in samodejno vstavil potrebne vrednosti. Enak primer je tudi v primeru, ko sta $H = 9$ in $L = 7$. Nekoliko drugače pa moramo razmišljati, ko je $L = 9$ in $H = 12$. Razlika med $Q[12] = 24$ in $Q[9] = 15$ je 9, kar je mnogokratnik največje vrednosti, ki smo jo priredili elementom abecede (spomnimo, c smo kodirali z vrednostjo 3). Edina možnost da bomo od vrednosti elementa $Q[9] = 15$ dosegli vrednost v $Q[12] = 24$ je, da na prosti mesti $Q[10]$ in $Q[11]$ vpišemo vrednosti tako, da na ustrezni mesti v Q dvakrat prištejemo 3. Tudi ta primer lahko enostavno zaznamo. Če velja, da je $Q[H] - Q[L] = 3 \times (H - L)$, vpišemo na manjkajoča mesta vrednosti, povečane za 3, kar enolično realizira tudi dekodirnik. Kot vidimo, smo v našem primeru uspeli 12 števil zakodirati s 14-timi biti, kar je vsekakor zelo vzpodbudno.

Samo kodiranje lahko realiziramo na nekoliko učinkovitejši način. Ena izmed možnosti je kodiranje FELICS [29]. FELICS (angl. Fast, Efficient, Lossless Image Compression System) so razvili za brezizgubno stiskanje sivinskih slik. Vrednost piksla, ki ga kodiramo, določimo s pomočjo dveh že kodiranih pikslov. Eden izmed njiju ima manjšo vrednost (v našem primeru

Razpredelnica 4.25: Prikaz stiskanja z interpolativnim kodiranje

L	H	m	$[r_L, r_H]$	$Q[m]$	koda
1	12	6	$[8, 18]$	10	0010
1	6	3	$[5, 7]$	7	10
1	3	2	$[4, 6]$	5	01
3	6	/	/	/	/
6	12	9	$[13, 21]$	15	0010
6	9	7	$[11, 13]$	13	10
7	9	/	/	/	/
9	12	/	/	/	/

je ta označena z r_L), drugi pa večjo (r_H). Kodo, ki jo priredimo kodiranemu pikslu, določimo ob predpostavki, da bo barva kodiranega piksla zelo verjetno blizu aritmetične sredine obeh vrednosti. Zato takšnim vrednostim priredi krajše kode. FELICS obravnava tudi primere, ko je vrednost kodiranega piksla izven območja $[r_L, r_H]$, a to v našem primeru ni možno. Kode FELICS določimo na naslednji način. Najprej izračunamo število bitov za tako imenovane kratke kode:

$$k = \lfloor \log_2 (r_H - r_L + 1) \rfloor, \tag{4.19}$$

Nato izračunamo števili a in b

$$a = 2^{k+1} - (r_H - r_L + 1), \quad b = 2(r_H - r_L + 1 - 2^k), \tag{4.20}$$

kjer a določa število kratkih kod, določenih kot $2^k - 1, 2^k - 2, \cdots$, b pa število daljših kod, ki so $(k + 1)$ bitna števila.

Vzemimo primer iz razpredelnice 4.25, ko je $r_L = 13$, $r_H = 21$ in $H - L + 1 = 9$. Potem je $k = 3$, število krajših kod je $a = 2^4 - 9 = 7$, število daljših kod pa $b = 2(9 - 2^3) = 2$. Nato lahko določimo dejanske kode za a: $8-1 = 111$, $8-2 = 110$, do $8-7 = 001$. Kodi za b pa sta 0000, 0001. Krajše kode so razporejene na sredini področja, daljše pa na obeh krajiščih. b je sod, zato lahko množico števil vedno razdelimo v dve enako veliki podmnožici. Razpredelnica 4.26 kaže vse kode FELICS za interval $[13, 21]$. Ugotovimo, da je kode možno enolično dekodirati. V tem primeru vidimo, da prve tri

Razpredelnica 4.26: Kode FELICS

vrednost v $Q[m]$	koda
13	0001
14	001
15	010
16	011
17	100
18	101
19	110
20	111
21	0000

ničle označujejo kodiranje s daljšimi kodami, če pa naletimo na enico prej, smo uporabili kodiranje s krajšimi kodami. V našem primeru je kodirana vrednost $Q[9] = 15$, ki bi jo v načinu FELICS kodirali kot 010 in prihranili en bit. En bit bi prihranili tudi v primeru, ko na intervalu $[4, 6]$ kodiramo vrednost $Q[2] = 5$, ki bi jo kodirali z bitom 1.

Dekodiranje. Za uspešno dekodiranje potrebujemo začetne vrednosti, in sicer spodnjo L in zgornjo H vrednost, dolžino polja Q in seveda zaporedje bitov B. Inicializirano polje prikazuje razpredelnica 4.27, kjer sta že vstavljeni vrednosti $Q[L = 1] = 3$ in $Q[H = 12] = 24$. Za primer dekodiranja predpostavimo, da kodiranja nismo opravili po postopku FELICS, ampak so v stisnjeni datoteki zapisani biti iz razpredelnice 4.25, in sicer $B = 00101001001010$.

Razpredelnica 4.27: Inicializacija polja Q za dekodiranje

i	1	2	3	4	5	6	7	8	9	10	11	12
$Q[i]$	3											24

Po enačbi 4.15 najprej izračunamo položaj elementa $m = 6$ v polju Q, ki ga bomo dekodirali. Z uporabo enačb 4.16 in 4.17 izračunamo $r_L = 8$ in $r_H = 18$, nato pa določimo število bitov b, ki jih bomo prebrali iz B. Po enačbi 4.18 dobimo $b = 4$ in preberemo bite 0010. Bite pretvorimo v desetiški zapis ter vrednost prištejemo r_L. Dobimo $Q[m = 6] = 10$. Stanje kaže razpredelnica 4.28.

Razpredelnica 4.28: Dekodiranje elementa na položaju $m = 6$.

i	1	2	3	4	5	6	7	8	9	10	11	12
$Q[i]$	3					10						24

Dekodiranje rekurzivno nadaljujemo na enak način, kot je potekalo kodiranje. Postavimo $L = 1$ in $H = 6$, izračunamo $m = 3$, $r_L = 5$ in $r_H = 7$ ter $b = 2$. Preberemo bita 10, dekodiramo v $2_{(10)}$, rezultat prištejemo k r_L in dobimo $Q[3] = 7$. V ponovnem rekurzivnem klicu imamo $L = 1$ in $H = 3$, $m = 2$, $r_L = 4$, $r_H = 6$ ter $b = 2$. Ponovno preberemo 2 bita, to sta 01, in dekodiramo $Q[2] = 5$. Naslednji rekurzivni klic je z $L = 3$ in $H = 6$. Ker pa velja, da je $(Q[6] = 10 - Q[3] = 7) = 3$ enako $H - L = 3$, ne preberemo nobenega bita iz vhodnega zaporedja, saj vemo, da lahko vrednosti med L in H vpišemo tako, da inkrementiramo vrednost iz $Q[L]$ in jo vpisujemo na položaje od $L + 1$ do $H - 1$. S tem smo zapolnili levo polovico polja Q, dekodiranje desne polovice pa pričnemo z $L = 6$ in $H = 12$. Izračunamo $m = 9$, $r_L = 13$, $r_H = 21$ in dobimo $b = 4$. Preberemo naslednje štiri bite 0010, jih dekodiramo in dobimo $Q[9] = 15$. Nadaljujemo z $L = 6$ in $H = 9$, dobimo $m = 7$, $r_L = 11$, $r_H = 13$ in $b = 2$, zato včitamo dva bita 10 in dobimo $Q[7] = 13$. Trenutno stanje vidimo v razpredelnici 4.29

Razpredelnica 4.29: Vmesno stanje dekodiranja.

i	1	2	3	4	5	6	7	8	9	10	11	12
$Q[i]$	3	5	7	6	7	10	13		15			24

V naslednjem rekurzivnem klicu sta $L = 7$ in $H = 9$. Ker je $Q[9] - Q[7] = H - L$, natanko vemo, da je $Q[8] = 14$. V naslednjem rekurzivnem klicu sta $L = 9$ in $H = 12$. Tokrat pa velja, da je $Q[12] - Q[9] = 9$, kar je $3 \times (H - L)$. Zato napolnimo mesti $Q[10]$ in $Q[11]$ tako, da h $Q[L] = 15$ prištevamo 3. Dobimo $Q[10] = 18$ in $Q[11] = 21$. S tem se rekurzivnimi klici končajo, zaporedje pa smo dekodirali.

4.7 Vprašanja

1. Vhodni niz je datoteka ASCII s 1215 znaki. Z algoritmom stiskanja smo datoteko predstavili s 1142 biti. Izračunajte razmerje stiskanja,